Computer and Information Science; Vol. 16, No. 2; 2023
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Automation-Based User Input Sql Injection Detectionand Prevention Framework

Fredrick Ochieng Okello[1], Dennis Kaburu[2], & Ndia G. John[3]

[1] Faculty of Computing and Informatics, Mount Kenya University, Thika, Kenya. E-mail: Okello.alfredochieng2007@gmail.com

[2] Faculty of Computer, Jomo Kenya University of Agriculture and Technology, Thika, Kenya. E-mail: dennis.kaburu@gmail.com

[3] Faculty of Information Technology, Muranga University, Muranga, Kenya. E-mail: ndiajg@gmail.com

Correspondence: Fredrick Ochieng Okello, Faculty of Computing and Informatics, Mount Kenya University, Thika, Kenya. E-mail: Okello.alfredochieng2007@gmail.com

Received: March 6, 2023 Accepted: April 30, 2023 Online Published: May 2, 2023

doi:10.5539/cis.v16n2p51 URL: https://doi.org/10.5539/cis.v16n2p51

Abstract

Autodect framework protects management information systems (MIS) and databases from user input SQL injection attacks. This framework overcomes intrusion or penetration into the system by automatically detecting and preventing attacks from the user input end. The attack intentions is also known since it is linked to a proxy database, which has a normal and abnormal code vector profiles that helps to gather information about the intent as well as knowing the areas of interest while conducting the attack. The information about the attack is forwarded to Autodect knowledge base (database), meaning that any successive attacks from the proxy database will be compared to the existing attack pattern logs in the knowledge base, in future this knowledge base-driven database will help organizations to analyze trends of attackers, profile them and deter them. The research evaluated the existing security frameworks used to prevent user input SQL injection; analysis was also done on the factors that lead to the detection of SQL injection. This knowledge-based framework is able to predict the end goal of any injected attack vector. (Known and unknownsignatures). Experiments were conducted on true and simulation websites and open-source datasets to analyze the performance and a comparison drawn between the Autodect framework and otherexisting tools. The research showed that Autodect framework has an accuracy level of 0.98. The research found a gap that all existing tools and frameworks never came up with a standard datasets for sql injection, neither do we have a universally accepted standard data set.

Keywords: AIDS, database security, knowledge base, SIDS, SQL, injection, PHP

1. Introduction

The discipline of information security management attempts to ensure that an organization's information resources are adequately protected against attacks. SQLI vulnerabilities are common in academic settings. According to the (Microsoft Build, 2022) research, SQLI vulnerabilities exist in thirty-five percent of educational institutions, 32 percent of government institutions, and 21% of all organizations allowing hostile actors tomodify and replace queries that an application sends to its databases. This is particularly dangerous for the industry since it has the ability to expose highly personal information that attackers can use to impersonate people. The Autodect framework has aided in addressing the challenges of system vulnerabilities in educational institutions. According to (Erick & Wang, 2022), the risk posed to academic institutions goes beyond financial loss. These educational institutions store many sensitive data(information), such as student personal records, sensitive research data, and valuableintellectual property. Information loss or breach could endanger the individuals involved and significantly ruin a university's image. It is particularly dangerous for the industry since it can expose highly personal information that attackers can use to impersonate people. By protecting the safety of our information systems against attacks, MIS security management is the wayto go.

SQL injection is a method of attacking a data-driven application that entails inserting malicious SQL commands into an entry fieldfor execution, such as to dump database contents to the attacker. (Venkatesan & Devi et al, 2018) SQL injection necessitates a security hole in the application program, such as when a user input is

incorrectly checked for string literal or escape characters used in SQL statements or when user input is not strongly typed and executed unexpectedly.

SQL attacks are commonly employed to target websites but may also be used to attack SQL databases.

SQL injection is one of the top five web assaults, according to the OWASP study (Fitsum & Tsegay, et al 2022). According to them, the attacker must also recognize and identify the system's weaknesses.

With a well-coordinated sql injection attack,an attacker can create a false identity, tamper with existing data, and cause repudiation issues such as voiding transactions or changing balances. The attacker can take advantage of the system's entire exposure by destroyingor otherwise making data unavailable.

The Autodect framework has aided in preventing SQL injection attacks by automatically identifying and stopping them,but most importantly,by directing the attacker to a proxy database, where the information will be automatically collected and saved in the Autodect knowledge base.

This knowledge base will guide the analysis of future attacks and trends. As a result, this approach will ensure that MIS vulnerabilities are continually vetted.

The Structured Query Language Injection (SQLI) attack is the most serious of the injection attacks because it undermines the basic security services of secrecy, authentication, authorization, and integrity. SQLI attacks occur when malicious SQL instructions are injected into queries and forms to gain access to a database or manipulate its data (e.g., send the database contents to the attacker, modify or delete thedatabase content, etc.) In fact, most web applications today require a back end database to store user input and/or retrieve information that they have chosen (Ines & Habib et al, 2020)

Because assaults are specific to the online application, the language used, site features, database structure, security methods, or safeguards put in place, the SQL injection detection problem persists. Because of the individualized character of our requests, most SIDS do not identify new SQL injection threats, according to (Anastasia & Vladimir, 2019).

Few web application firewalls and techniques can report correct SQLIA for their deployed services, according to (Haifeng &, Jianning, et al., 2020). Traditional tactics focus on blocking suspicious SQL requests but do not identify whether the attack is damaging or not. All of the current approaches for preventing SQLI have flaws, such as the absence of standard datasets, the need for admin interaction, and some being particular to certain programming languages. The autodect architecture solves these flaws using a four-layer strategy, which includes detection and prevention and adding a knowledge base and a proxy database.

1.1 Types of SQL Injection Attacks

1.1.1 Tautology-Based Attacks

Tautology-based attacks work by injecting code into one or more conditional SQL statement searches in order to make the SQLcommand evaluate as true conditions such as (1=1) or (- -) (Zainal & Manal, 2017). This method is extensively used to acquire database access by bypassing authentication on web pages. WHERE employee ID='1' OR 1=1 AND employeepassword=1200 FROM employee

1.1.2 Piggy Based Query

The attacker aims to inject additional queriesto retrieve, change, or add data in a piggy-backed query attack, according to (Ines &, Omar, et al, 2020). This database breach attack employsa query statement to the original query. The first query is original in this strategy, butfuture searches are infected. Because the attacker can use it to inject nearly any sort ofSQL, this attack poses a significant concern.

Example SELECT pass FROM userTable WITH user Id="user1" AND password="0";drop userTable

1.1.3 Logically Incorrect

An error-based SQL Injection attack iscarried out by putting erroneous data into a query, causing the database to make a mistake, according to (Anandha & Adhil, et al 2021). This isusually performed by compelling thedatabase to perform a potentially error-proneaction.

Uses the error messages supplied by thedatabase as a result of an erroneous query.These database error messages frequently offer useful information that allows attackers to determine the application and database schema vulnerability variables. Example WHERE User ID="11" AND Password = "1200" AND convert SELECT *FROM userTable (Char, no)

1.1.4 Union Injection Attack

The attacker utilizes the UNION operator to hitch a malicious query to the initial query, according to (Anandha & Adhil, et al 2021). The malicious query's results are used to connect the results of the first query, allowing the attacker to obtain thevalues of columns from additional tables. In this approach, the attacker sends the SQL query with the union of both correct anderroneous fields after providing incorrect data with a few correct columns. As a result,the data set from the database is retrieved correctly formatted.

According to the report, an attacker may inject the string "UNION SELECT Card No from Credit where acc no =1200" into the login field (international journal ofengineering applied sciences and technology,2016).

SELECT account information FROM clientsSELECT Card No FROM Credit WHERE login=UNION WHERE ACCOUNTNUMBER IS 1200 AND PASS IS AND PIN

IS SELECT card inf FROM clients WHERE login=UNION SELECT acc inf FROM clients WHERE login=UNION.

The login in the first sentence is null, so the query is invalid, however the other query retrieves the results.

1.1.5 Hexadecimal Attack

2016 (Santosh Kumar) SQL injection with hex code is a code injection technique that exploits a security flaw in the database layer of an application.

The vulnerability occurs when user input is either not strongly typed and then executed unexpectedly or when user input needs to be better checked for the specified input string and then inserted in SQL statements. According to (Santosh Kumar, 2016), hex code SQL injection attacks are also known as insertion attacks.

An attacker can inject harmful code into an application in various methods; hex codeis just one of them.

1.1.6 Binary Attack

Blind SQL injection is a sort of SQLI attack that creates true or false questions to thedatabase to ascertain the solution provided by the application's response, according to (S.S. Anandha Krishnan1; AdhilN Sabu, et al 2021).

Blind SQL injection occurs when an attackersends a SQL query to the database, leading the application to deliver a result. As a result,the conclusion is determined by whether the question is true or untrue. Depending on the decision, the information in the HTTPresponse will be changed or left unchanged. The attacker will determine if the message was received as true or false.

1.1.7 Time-based

It is another type of blind inferential SQLattack, according to ((Ines & Omar, et al ,2020) This method sends a SQL query to the database, which causes it to wait (for a specified number of seconds) before responding. As a result, the attacker can make some educated guessesabout whether anything is real or false basedon the time it took to react. As a result, an HTTP response will be generated immediately or after some time. The attacker makes decisions independently of the database.

1.2 Existing Tools to Deal with SQL Injection

1.2.1 Green SQL

Green SQL is an open-source databasefirewall that prevents SQL injection attacks on databases. The logic is based on a risk-scoring matrix that evaluates SQL statements as well as prevents recognized administrative procedures like DROP, CREATE, and so on.

It functions as a proxy and includes support for my SQL.

1.2.2 Green SQL Architecture

The supplied programconnects to the green SQL server instead of the MYSQL server. For MYSQL connections, Green SQL acts as a reverse proxy. Green SQL will gradually analyze SQL queries and forward them to the MySQL server in the backend.

If the query is deemed illegal, it is checked against a white list, and if it is discovered, it is forwarded to the actual MYSQL server. Ifthe data is invalid, the program will receive an empty result.

Green SQL might produce false positive andfalse negative errors during execution. As a result, some acceptable searches may be banned, while invalid queries may be permitted.

1.2.3 Microsoft Defender for SQL

It comprises two phases, one of which includes capabilities for detecting and mitigating potential database vulnerabilities and the other for monitoring aberrant activitythat could suggest threats to your databases.

Potential database vulnerabilities are discovered, tracked, and remedied using a vulnerability assessment service. Assessmentscans provide an overview of your SQL machine's security state and details.

An advanced threat prevention service monitors your SQL servers for risks including SQL injection, brute-force attacks,and privileged abuse. This service sends out security notifications that are actionable.

2. Literature Review

2.1 Related works (Frameworks)

The research focused on previous frameworksgeared toward the mitigation of sql injection in vulnerable databases andmanagement information systems. A closer look was taken at these frameworks' functionality and success rate to determine their shortcomings. The AUTODECT framework's contribution factored in all these shortcomings. These frameworks are discussed as below.

2.1.1 The Auto Black Box SQL Injection Vulnerability Test (SQL)Framework

The framework will employ two detection models: signature and anomaly-based. A mix of the two will be used to ensure near-perfect detection of user input SQL injection.The Auto Black Box SQL Injection Vulnerability Test Proposes (SQL). In SQLIA, this automates an SQLIV evaluation. A recent study has also shown that current SQLIVS has to be upgraded to reduce human vulnerability inspection expenses and the danger of being attacked due to inaccurate false-negative and false-positive results.

(Fitsum & Gizachew , et al, 2022) SQL injection attacks and thecreation of a complete framework employinga hybrid and machine learning method This framework is used to assess the efficacy of strategies that address specific difficulties in the context of the attacks.

2.1.2 DIAVA Framework

The bidirectional network traffic of sql operations and the multilevel regular expression model were investigated. DIAVA is a traffic-based system for detecting SQL injection attacks and analyzing spilled data for vulnerabilities. According to (Haifeng ,& Jianning et al, 2020), DIAVA can send tenants proactive warnings.

This framework is a novel traffic-based SQLIA detection and vulnerability analysis framework. DIAVA is meant to accordingly identify successful SQLIA among allsuspects, the capacity of the attacks and vulnerability of leaked data is evaluated by the framework based on its GPU-based dictionary attack analysis engine. DIAVA framework suffers from shortcomings; it has a two-layer approach, the front end that collects network traffic and the backend that identifies SQLIA and evaluates it. This framework needs to be clarified on how the analysis will be done, and neither have they stated how they obtain datasets for signature comparison; there is no proposed algorithm to model the matching patterns, which may capture the network traffic. The framework also lacks standard datasets or a frequently updated knowledge base for future comparisons.

2.1.3 SSF Framework

According to (Israr Ali, Syed Hosea, MansurIbrahim, 2018), The primary thought of the SQL injection shield framework (SSF) structure isto make a profile for web applications that can present to the normal behavior of users regarding sql queries they submit to the database.

This framework utilizes database logs to gather true blue questions, given that the logsare interruptions-free. SSF also uses misuseprocedures to catch any change in the structureof the query.

The framework uses the followingmethodology.

1. Attempting to identify SQL injection through checking peculiar SQL query structure.
2. Utilizes information condition amongthings that are more averse to changes. Abuse procedures as well as anomalies or blended approach or both. Consequently, other interruption identification strategies were utilized either independently or together.

The framework has the capacity to distinguish interlopes by identifying practices that contrast with the typical conduct of a part in the database. (Karmal et al, 2018) represented an upgraded model that can likewise distinguish gatecrashers indatabases with no parts connectedwith every client.

The weakness of the SSF is that it only works well in java and xml platforms; this way its work is limited across

a broad spectrum of platforms. This framework can only detect union attacks in sql injection, leaving others out; the framework, too, is dependent on user intervention as well as it does not prevent; it only detects. Lastly, SSF is meant to collect anomalies from the system loss, but no clear mechanisms can be achieved.

2.1.4 Automated Security Testing Framework for Detecting SQL injection Vulnerabilitiesin Web Applications

This framework uses a security tester to identify the appropriate test case before starting to exploit sql vulnerabilities in web-based applications during the testing phase. In identifying test cases of web applications and analyzing the test results of an attack, the framework captures critical issues affecting the security testing effectiveness (Nor F. A, Aziza A.M, 2018). The framework is separated into three stages

i. Test case generator, which is used to generate test cases by application of permutation technique for automatic generation.
ii. Attack generator for automating the injection attack process based on input generated in the first stage.
iii. Response generator for analysis and determination of whether an attack is successfulor not.

The Autodect framework has a four-layer approach: detection, prevention, knowledge base, and a proxy database. The shortcomings of this framework are that ithas no prevention mechanisms spelled out, it only deals with the detection, which then becomes an end in itself. It lacks a central depository where attacks or vulnerabilities are reported (knowledgebase).

The framework is based on test case generation, but there is no explicit, well-defined algorithm for test case generation. Test casegeneration per each case may lead to memoryoverload, CPU congestionand overloaded buffer. The frame fails to address the issue of standard datasets. The framework relies heavily on OWASP adaptation.

2.1.5 SQL Injection Detection and ExploitationFramework for Penetration Testing

(Muhammad &, Naqi , May 2019)The framework presents a novel semi-automated SQL injection detection and Exploitation (IDE) solution using constructive methods by combining machine learning and advanced python computationIDE notifying the user if the target databasehas sql injection vulnerability and can beexploited for security testing. The framework uses a machine learning process,the IDE system, to create a profile dictionary of various injection and exploitation examples. This profile dictionary is used to identify the sql injection Vulnerability and associated exploits. The framework banks hope on the fact that despite many executable sql injection tools, almost all of them provide a limited functionality set.

The framework uses a multi-model classification module that considers the time-variant property of functionality andbehavior features of SQL injection from the system level.

The shortcomings of this framework are that using the time-variant property means only one type of sql inference query attack can be detected. Since the system will bewaiting to flag down requests based on time delay, an attacker or attack willdelay based on the number of trials / this framework lacks a prevention unit, it only detects, and later the signatures are taken to the link detection signatures and existingsignatures in the dictionary. Most importantly, it is not able to generate standarddatasets. According to (Muhammad & Naqi, May 2019) static analysis reports many false negatives and positives. The IDE used is only best in whitebox pen testing as all the variables and targetinformation should be in hand to simulate theattack; this way, the framework is unsuitablefor black box pen testing.

2.2 Signature-based

Signature-based detection is commonly employed for malware detection over harmful files, according to (Muhammad Amirulluqman et al, 2019). The signature-based detection methodology performs well against previously identified assaults. Itchecks the test log files to the list of query requests and classifies them if they match.

Signature-based intrusion detection system (SIDS) provides a potential answer to the problem of online application security, according to (Nancy &, Syed , 2018). However, the system's success is heavily dependent on the quality of the signatures developed to detect attacks. According to their findings, a poor signature set can significantly increase the rate of false alarms making it nearly impossible to use the method. When using signatures, a security expert should be able to test thedetector against the identified defects by looking at the structure of the signatures.

Another issue in signature-based security is that a security expert establishes these signatures; therefore, the water tightness of the signatures is dependent on the security expert's skill, know-how, and perceived aim. The

attack domain and the accuracy with which the attack patterns are captured and implemented. According to (Nancy &Syed, 2018), both limited knowledge of the attack area and errors in the implementation of detection algorithms would generate a low-quality signature set, resulting in SIDS inefficiency.

Attackers create a list of a grey area vulnerability to exploit to attack a website, and the domain of an attack vector changes greatly depending on the cluster of flaws. In other words, SQL-based attack vectors will use SQL components and related commands, while XSS-based attack vectors will largely use HTML and Javascript language elements.

Regardless of the developer's knowledge, a well-defined rule set may play an equally important role in achieving flawless SIDS. According to (Massicote et al, 2015), detection coverage is one of the most important testing characteristics. The technology may allow an attack vector to breach a web application if the signature is incomplete or the signature pattern is not applied correctly. There is also the issue of the frequent trade-off between selecting a signature's sensitivity and its specificity. A generic signature can be modeled to provide broader detection coverage and deal with variations of attack vectors (Nancy Agarwal, Syed Zeeshan Hussain, 2018). However, it must damage genuine request traffic by wronglycategorizing attacks.

2.3 Anomaly-based

Anomaly detection and ReconstructionSchemas (AD&RS) must also be emphasized, according to (Malik & Ali, et al, 2019), so that Intrusion detection systems can perform with some Modules, such as Anomaly detection Module, Query reconstruction Module, Query delegation Module, and so on. Signature-based profiles can also be designed to accommodate all conceivable outcomes in terms of all valid possible inquiries and their profile-generating path. Databases contain signature-based profiles. When a user runs a query, it istranslated to a specified signature base type/shape and then examined/compared to previously stored searches. The request will be refused or allowed depending on whether the profile is normal or atypical.

According to, the implicit profile of normal functioning provides information about therange of allowable correlation responses (Anastasia & Vladimir, 2022). In designing a classification system that allows us to identify points within this zone from points outside, the challenge of finding anomalies will be solved. A one-classclassifier can be made in a number of ways.

2.4 Machine Learning

SQL injection attacks emphasizes the characteristics of the targeted online application, the programming language used to create it, the database structure, and the database that feeds it, according to (Anastasia & Vladimir , 2022). It is, therefore, not unexpected that intrusion protection systems sometimes notice new SQL injection threats. There have been attempts to lessen the individualized nature of queries, which should make detecting SQLinjection attacks easier, although this needs to address the problem in principle.

Machine learning algorithms are one of the most important tools for identifying various types of attacks. Such methods have long been used to detect SQL injection attacks and other assaults that are comparable to them. The ability to recognize implicit relationships in data motivates the use of machine learning. It should presumably be able to identify known and unknown assaults. Consider some research studies on the subject. The parameters chosen to compose the area of interest and to determine the proper characteristic of real-world occurrences in this area varies across all machine learning-based methods. The rate at which symbols, classes of symbols, and keywords of SQL language were considered distinguishing feature of malicious requests. It's critical to teach the neural network recognizer to run a synthetic test that includes both conventional SQL queries and attacks. The implemented approach was unique in that a neural network recognizer was installed individually on the secured site.

2.5 Autodect Framework

The framework will employ a machinelearning-based model for the detection of abnormal and normal signatures. A mix of the two will be used to ensure near-perfect detection of user input SQL injection.

2.5.1 Goals of the Autodect Framework

i. Analyze the structure of SQL query requests based on heuristics
ii. Come up with a knowledge base that will check allowable patterns of SQL statements
iii. Come up with a white list of common SQL injection commands andstrings
iv. Create a proxy database serverthat will raise a red flag of a possiblesql injection command.
v. Prevent and stop SQL injectionattacks to a database using a proxy server and a knowledge base.

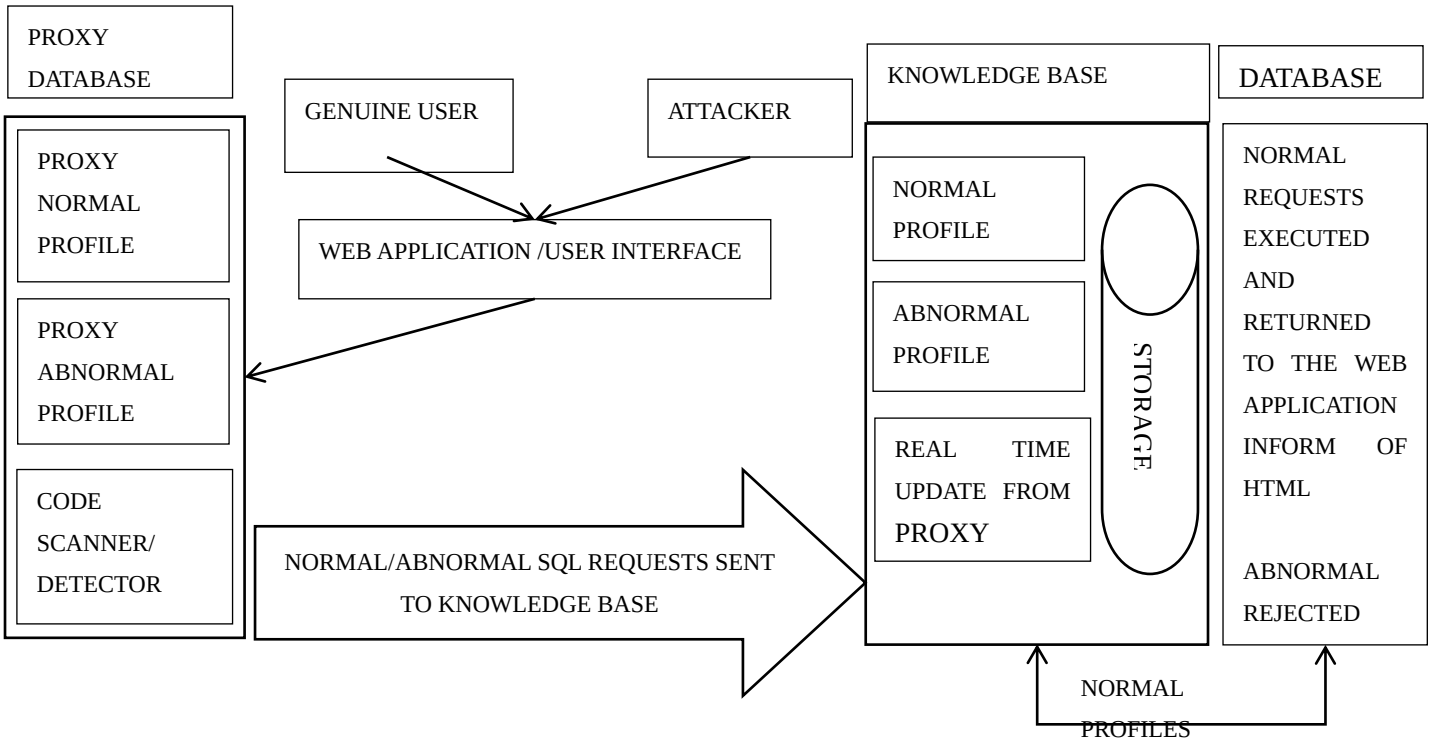

Figure 2. Autodect Framework Layout

2.5.2 Explanation of the Autodect Framework Layout

A genuine user or an attacker gains access through a web application (user interface) i.e a login page that requests a user name and a password. Both users will be allowed access to the proxy database. The scanner /detector unit of the proxy database will check the signatures entered by both (Genuine user and attacker), to determine whether they are abnormal or normal code profile. After this the detector unit sends the profiles to the knowledge base. If normal it will be stored in the normal profile database of the knowledge base and if abnormal it will be stored at the abnormal profile database of the knowledge base. There will always be real time updates from the proxy database to the knowledge base. Normal profiles (genuine signatures) will move to the database, executions carried out and results returned to the webpage.

3. Methodology

3.1.1 Input and Output Variables

The various sorts of sql injection queries were used as the study's input variables. Tautology-based, piggy-based, logically flawed, union injection, hexadecimal attack, binary, and time-based attacks were among the few.

The results of the input query, such as the database name, passwords, amount of time required, URL, and whether access is gainedto the web server apps, were our output variable.

3.1.2 Experimental Evaluation Techniques

Depending on the goals, different assessment approaches were used, such as quantitativetesting/experiments on websites, databases,and datasets. User testing for usabilityQuestionnaires surveys and case studies.

On the three real websites, the two simulationwebsites, the two open source datasets, and 100 code injection trials of tautology-based attacks, 80 trials of piggy-based attacks, 60 trials of logically incorrect attacks, and 150 union injection assaults will be conducted.

3.2 Parameters Evaluated

We took into account many variables, including runtime, pre-processingtime, disk space (overhead), memory, the accuracy of discoveries, and how accurate anomaly estimates, false positive and false negative rates, user satisfaction, and usability.

3.2.1 Evaluation Metrics

A set of measures were used to assess the Autodect framework. Some of these measures measured how well the

suggested framework performs overall, while others evaluated particular aspects. We began by looking at a binary classification issue with only two classes.

The two terms listed below were utilized to properly analyze the framework's performance.

True positive (TP) refers to the number of inputs that the framework correctly predicts as being positive, whereas false positive (FP)refers to the number of inputs that the framework wrongly predicts as beingpositive when they are negative.

The term "true negative" (TN) describes the proportion of inputs that the framework accurately predicts as being negative but which are positive.

		PREDICTION	
		POSITIVE	NEGATIVE
REAL	POSITIVE	TP	FN
	NEGATIVE	FP	TN

Figure 3.1. Evaluation Metrics Mattrix

3.3 Evaluation Formulas

The following formulas were adopted tohelp in the framework analysis.

3.3.1 Precision and Recall (P & R)

While the recall is the ratio of a correct positive prediction to all positive examples (TP +FN), precision is the ratio of correct positivepredictions to all predicted positives (TP + FP).

$$Precision = \left(\frac{TP}{TP + FP}\right)$$

$$Recall = \left(\frac{TP}{TP + FN}\right)$$

3.3.2 Accuracy

This made it easier to calculate how accurately the framework predicted theinputs, or the proportion of correctly predicted inputs to all inputs.

$$Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN}\right)$$

3.3.3 F1- SCORE (F1)

Additionally, these measurements can be coupled to calculate the harmonic mean ofrecall and precision.

$$F_{\beta} = (1 + \beta^2).\frac{T.P}{\beta^2.P + R}$$

4. Discussion of Findings

4.1 Response Rate

The researcher conducted 100 code injection attacks on five open-source datasets using tautology-based attacks, 80 piggy-based attacks, 60 logically incorrect attacks, and 150 union injection attacks.

The injected codes were also tested using well-known sql injection tools like Nmap (a free security scanner), Hping (an active network security tool), Green SQL, Microsoft defender for Mysql, Code scan labs, Metaspoilt (a popular tool for performing pen tests), sql map, and Nessus vulnerability scanner. All of these technologies fell short of the autodect framework's standard for detection and prevention precision. Some of the tools could either only detect or could only prevent. Other techniques, such as Hping, could only scan ports and could not find any other vulnerabilities.

Through email notification on the Google forms online survey, expert validation was carried out. Compared to the interviews, which had a 100% response rate, this had a 90% response rate. Because there were so few

interviewees—only 30 members of the general staff and 20 members of the IT department—everyone agreed to participate.

4.2 Research Presentation

The best value of the parameter and hyper parameter has a significant impact on the performance, therefore developing a deep learning model necessitates many experiments. Five instances were tested in order to choose the machine learning modelthat performed the best. The generalized experimentation possibilities are as follows:

Table 4.1 Case 1. Setting the dense layer (neuron) to 256

OptimizationAlgorithm	Adam
Learning rate	0.01
Activation fun (lastlayer)	Sigmoid
Loss function	Binary crossentropy
Epoch	10
Bach size	32
Dataset splitting	30%-70%
Dense layer neuron	256
OUTPUT	
Execution time	562.090
Training accuracy	0.977
Validation accuracy	0.952
Testing accuracy	0.966

Table 4.2. Case 2: Configuring activation function (Softmax), number of epochs (20), and Dense layer neuron (64)

OptimizationAlgorithm	Adam
Learning rate	0.01
Activation fun (lastlayer)	Softmax
Loss function	Binary crossentropy
Epoch	10
Bach size	32
Dataset splitting	20%-80%
Dense layer neuron	64
OUTPUT	
Execution time	56.201
Training accuracy	0.230
Validation accuracy	0.421
Testing accuracy	0.334

Table 4.3. Case 3: Setting optimization algorithm (RMSprop), Activation Function (Sigmoid),Epoch (10).

Loss function	Binary crossentropy
Epoch	10
Bach size	32
Dataset splitting	20%-80%
Dense layer neuron	64
OUTPUT	
Execution time	592.301
Training accuracy	0.930
Validation accuracy	0.961
Testing accuracy	0.958

Table 4.4 Case 4: Setting data splitting 90%-10%(training and testing)

OptimizationAlgorithm	RMSprop
Learning rate	0.01
Activation fun (lastlayer)	Sigmoid

Table 4.5 Case 5: Setting the data splitting (90%-10%)with an epoch size of 20.

OptimizationAlgorithm	Adam
Learning rate	0.01
Activation fun (lastlayer)	Sigmoid
Loss function	Binary crossentropy
Epoch	20
Bach size	32
Dataset splitting	15%-85%
Dense layer neuron	64
OUTPUT	
Execution time	998.301
Training accuracy	0.960
Validation accuracy	0.971
Testing accuracy	0.977

4.4 Experiment Results Discussion

The Google Colab notebooks cloud editor, tensoflow, pycharm, and Jupiter note book python API were used for all of the tests, and the GPU accelerator was turned on. Because itprovides free RAM and GPU, two essential resources for deep learning research. The studytested five situations with various parameters, hyper parameter values, and cases, as was described in the hyper parameter selection sub-section. The execution times varyeven if some of the situations have almostidentical training, validation, and testing

Table 4.6. Evaluation of the existing security frameworks used to preventuser input SQL injection

Attack attempts	**TOOLS**								
Total: 150	Nmap (Nmapfree security scanner)	Hping(Active network security tool)	Green SQL	Microsoft defender for MySQL	Code scan labs	SQL map	Metaspoi lt Framework(Pen testing security)	Nessus vulnerability scanner	AUTODECT FRAMEWORK
Successful	99/150	82/150	112/150	87/150	105/150	101/150	88/150	76/150	131/150
Not successful	51/15	68/150	38/150	63/150	45/150	49/150	62/150	74/150	19/150

4.5 Evaluation of Factors that Leads to sql Injection Detection

A pair of data: a parameter and a susceptible page should be taken into account for a successful SQLI attack. As an illustration, consider "username and password" as parameters and "lndex.php" as the vulnerable page. It is important to identify any potential attacks for each of these factors. Next, a series of scripts might be constructed to automatically submit the created lists and create an attack list and a benign list for the unsecured web application. Use that set to do a penetration test as well. A penetration test simulates an attack by a hostile person to evaluate the security of a computer system or network (Atefeh & Suhaimi, 2017). This comprises a system of constant watchfulness on all potential weak spots. These vulnerabilities may be brought on by unknown hardware and software failures, flaws in operational processes, technical defenses, or inappropriate system configuration. A potential attacker will conduct this study, which may involve actively exploiting weaknesses. A penetration test's goal is to evaluate the viability and potential consequences of an assault. In actuality, it is a component of a thorough security examination. The following section discusses the factors that influence user input sql injection detection.

4.5.1 Un usual/ un Expected Behavior

The research found out that user input sql injection attacks can cause web applications to behave in an un usual

or unexpected way. A penetration test from vulnerable websites proved that all the three simulation websites and two data sets returned errors. It was also noted that the two datasets displayed data that shouldn't be visible. The Autodect framework detected, prevented and flagged these unusual behavior.

4.5.2 Suspicious SQL Statement

If an attacker successfully injects malicious sql code into the web application, chances are that it may result in suspicious sql statement being executed. These statements may include un usual characters such as apostrophes or semicolons and to some extent they may include SQL key words that shouldn't be presented in the code. For lour testing and authenticity, we injected the following four statements, each representing four diferent kind of sql attacks.

4.5.2.1 Error Based sql Injection

'AND (SELECT COUNT(*)FROM users)=15; As their databases did not contain exactly 15 users, the three simulation websites issued error messages, in two of the three simulation websites, the message disclosed the database structure information. Autodect framework detected 97% of the error messages and prevented them from returning valuable messages concerning database structure.

4.5.2.2 Basic sql Injection

Since all the three simulation websites had login pages with user name and password fields. For penetration testing we entered the following code.

'OR 1=1;-

This caused the SQL query to return true in one of the websites, allowing us to bypass the login and gain access to the system. The Autodect framework was able to detect and prevent 95 cases in 100 trials summing up to 95%.

4.5.2.3 Union Based sql Injection

The researcher injected a union query into the original statement to return additional data that the researcher was not authorized to view. The following Union statement was injected.

UNION SELECT credit_card_number, null, null FROM users;--

We were able to get additional credit card numbers from the users table. The Autodect framework detected 107 trials out of 112. This averaged to 95.53%

4.5.2.4 Blind sql Attack

Our research also used a Boolean based or time based technique to extract data from the database . The following code was inserted .

' AND (SELECT ASCII(SUBSTRING((SELECT TOP 1 name FROM users),1,1)))>100;

This statement extracted the first character of the 'name' field in the 'users' table using the ASCII function. Because the ASCII value was greater than 100, it returned true, allowing the researcher to extract the data one character at a time. The Autodect framework was able to detect and prevent 143 trials out of 149, which averaged to 95.97%

4.5.3 Abnormal Traffic Patterns

Sql injection attacks can generate a lot of traffic to a web application. If the traffic pattern suddenly increases it could be a sign that an attack is taking place. Autodect framework was able to detect and prevent 98 trials out of 100. Totaling to around 98%.

4.5.4 Log Analysis

Web application logs can provide valuable information about sql injection attacks . Example if a log shows that a user entered sql command in a form filed it could indicate that an attacks is underway. Autodect framework is able to analyze the system logs and generate a report

5. Conclusion and Future Work

In this study, we employed a four layer approach of a detection unit, a proxy database, knowledge base and prevention unit. we first outlined the different categories of SQLIAs. Then we looked into SQL injection detection and prevention methods being used both past and present. Then, we contrasted these methods based on how well they prevented SQLIA. We also contrasted these methods' deployment requirements, which inconveniences institutions. The findings suggest that some present strategies should be improved in order to better defend against

SQLI assaults. Future studies should separate techniques that have been used as tools and assess their efficacy, efficiency, stability, flexibility, and performance to map out their strengths and weaknesses. Some further works should also be done by coming up with a clear algorithm to develop sql injection universally accepted datasets. Our framework depended on our knowledge base to come up with a formation of standard data sets

References

Atefeh, T., & Suhaimi, I. (2017). A Framework for Evaluation of Sql Injection Detection and Prevention Tools. *International Conference on Computational Intelligence, Communication Systems and Networks, ABU Technical Review 2018(274).*

Awang, N. F., & Manaf, A. A. (2015). Automated Security Testing Framework for Detecting Sql Injection Vulnerability In Web Application. Communications in Computer and Information Science, 160-171. https://doi.org/10.1007/978-3-319-23276-8_14

Azman, M. A., Marhusin, M. F., & Sulaiman, R. (2021). Machine learning-based technique to detect SQL injection attack. *Journal of Computer Science, 17*(3), 296-303. https://doi.org/10.3844/jcssp.2021.296.303

Balasubramanian (2018). Prevention of sql injection attacks in web browsers. Application Development and Design (PP. 1240-1274). https://doi.org/10.4018/978-1-5225-3422-8.ch052

Fitsum, G., & Tsegay, M., et al, (Feb 8th, 2022). attacks on SQL injection and developing a comprehensive framework, using a hybrid and machine learning approach. *Asian Journal.*

Gurina, A., & Eliseev, V. (2022). Quality criteria and method of synthesis for adversarial attack-resistant classifiers. *Machine Learning and Knowledge Extraction, 4*(2), 519-541. https://doi.org/10.3390/make4020024

Ines, J., & Habib, H., et al, (2020). *SQL Injection Attack Detection and Prevention Techniques Using Machine Learning.*

Israr, A., Syed, H. A., & Mansour, I. (2016). Intrusion Detection Framework for sql Injection. *Asian Journal For Engineering Sciences Technology, 6.*

Khaled, E., & Yasser, F. et al, (2017). A survey of sql injection attacks detection and prevention. a journal of computer and communication.

Malik, R. A., & Muhammad, S. A. M. et al (August 2019). A Novel Intrusion Detection and Prevention Model for SQL Injection Attacks. *2019 IJCSNS International Journal of Computer Science and Network Security, 19*(8).

Muhammad, A. et al, (2019). Machine Learning Based Technique to Detect Sql Injection Attack. *Computer Science Journal*, 296-303.

Muhammad, A., & Naqi K. (May 2019). *SQL Injection Detection and Exploitation Framework for Penetration Testing*. Doctoral thesis London Metropolitan University.

Nancy, A., & Syed, Z. (2018). identification of flaws in the design of signatures for intrusion detection systems. *Asian Journal.*

S.S., A. K. (2021, June 30). SQL Injection Detection Using Machine Learning. *Revista Gestão Inovação E Tecnologias, 11*(3), 300-310. https://doi.org/10.47059/revistageintec.v11i3.1939

Scheel, C., Mecham, J., Zuccarello, V., & Mattes, R. (2018). An evaluation of the interrater and intra-rater reliability of OccuPro's functional capacity evaluation. *Work, 60*(3), 465-473. https://doi.org/10.3233/WOR-182754

Shachi, M., Shourav, N. S., Ahmed, A. S. S., Brishty, A. A., & Sakib, N. (2021, June 21). A Survey on Detection and Prevention of SQL and NoSQL Injection Attack on Server-side Applications. *International Journal of Computer Applications, 183*(10), 1-7. https://doi.org/10.5120/ijca2021921396

Venkatesan, R., Devi, D. R., Keerthana, R., & Kumar, A. A. (2018). A novel approach for detecting DDoS attack in H-IDS using association rule. 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA). https://doi.org/10.1109/icscan.2018.8541174

Copyrights