**MACHINE LEARNING APPROACH TO PREVENT SQL INJECTION ATTACK**

**by**

**MD. TOUHIDUL ISLAM SHAWAN**

**ID: 213 – 44 – 222**

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Software Engineering

Supervised By

**Nusrat Jahan**

Assistant Professor

Department of SWE

Daffodil International University



**DAFFODIL INTERNATIONAL UNIVERSITY**

**Dhaka, Bangladesh**

**JULY 2023**

**APPROVAL**

This Thesis titled “**MACHINE LEARNING APPROACH TO PREVENT SQL INJECTION ATTACK**”, submitted by Md. Touhidul Islam Shawan to the Department of Software Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of M.Sc. in Software Engineering and approved as to its style and contents. The presentation has been held on 31th May 2023.

**Board of Examiners**

**(Name)** [**Font-12, Bold**] **Chairman**

**Designation**

Department of CSE [**Font-12**]

Faculty of Science & Information Technology

Daffodil International University

**(Name) Internal Examiner**

**Designation**

Department of CSE

Faculty of Science & Information Technology

Daffodil International University

**(Name) External Examiner**

**Designation**

Department of -------

Jahangirnagar University

**Declaration**

I hereby declare that, this thesis has been done by me under the supervision of **Nusrat Jahan, Assistant Professor, Department of SWE** Daffodil International University. I also declare that neither this thesis nor any part of this thesis has been submitted elsewhere for award of any degree or diploma.

**Supervised by:**

**Nusrat Jahan**

Assistant Professor

Department of SWE

Daffodil International University

**Submitted by:**

**Md. Touhidul Islam Shawan**

ID: 213-44-222

Department of SWE

Daffodil International University

**ACKNOWLEDGEMENT**

First I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes me possible to complete the final year thesis successfully.

I really grateful and wish my profound indebtedness to **Nusrat Jahan**, **Assistant Professor**, Department of SWE Daffodil International University, Dhaka. Deep Knowledge & keen interest of my supervisor in the field of “*Cyber Security and Machine Learning*” to carry out this thesis. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this thesis.

I would like to express my heartiest gratitude to **Dr. Imran Mahmud** Head**,** Department of SWE, for his kind help to finish my thesis and also to other faculty member and the staff of SWE department of Daffodil International University.

I would like to thank my entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

**Abstract**

SQL injection attacks represent critical danger to the security of web applications that depend on database systems. Traditional prevention methods, such as input validation, rule based, parameterized queries have limitations in effectively prevent this type of attack. In this paper, I propose a machine learning approach to prevent SQL injection attacks and see how can machine learning approach prevent SQL injection effectively. By training a machine learning model on a labeled dataset comprising both legitimate and malicious queries, my system can effectively distinguish between safe and potentially harmful SQL queries. My approach utilizes a combination of supervised learning algorithms and natural language processing techniques to accurately detect and prevent SQL injection attempts in real-time. I present the design and implementation of my system, including data preprocessing, feature extractions, model training, model evaluation. During the data preprocessing stage, I sanitize an normalize the dataset queries to ensure consistent and accurate analysis. Feature extraction transform the queries into meaningful representation that capture relevant information for classification. This step may involve examining the presence of specific keywords or characters, parsing query structure. I employ various algorithms, such as logistic regression, decision tree, random forest tree, support vector machine, naïve bayes, to train a model that can accurately classify queries as either safe or malicious. The model learns from the labeled dataset, utilizing the extracted feature to identify patterns associated with SQL injection attack. Experimental results demonstrate the effectiveness of my approach in accurately identifying and blocking SQL injection attacks while maintaining a low false positive rate.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **CONTENTS** | **PAGE** |
| Approval | i |
| Board of examiners | i |
| Declaration | ii |
| Acknowledgements | iii |
| Abstract | iv |
| **CHAPTER** |  |
| **CHAPTER 1: INTRODUCTION** | **1-5** |
| 1.1 Introduction | 1 |
| 1.2 Motivation | 4 |
| 1.3 Rationale of the Study | 4 |
| 1.4 Research Question | 5 |
| 1.5 Expected Outcome | 5 |
| **CHAPTER 2: BACKGROUND** | **5-9** |
| 2.1 Terminologies | 5-7 |
| 2.2 Background Literature | 7-9 |
| 2.3 Problem Statement | 9 |
| **CHAPTER 3: METHODOLOGY** | **10-22** |
| 3.1 Data Collection | 10 |
| 3.2 Data Preprocessing | 11 |
| 3.3 Feature Extraction | 12-13 |
| 3.4 Model Training | 13-15 |
| 3.5 Model Evaluation | 16-17 |
| 3.5.1 Accuracy | 17 |
| 3.5.2 Precision | 17 |
| 3.5.3 Recall | 17 |
| 3.5.4 F1-score | 17 |
| 3.5.5 TF-IDF Vectorizer Classification | 18 |
| 3.5.6 Count Vectorizer Classification | 19 |
| 3.6 Manual Cross Checking | 20-22 |
| **CHAPTER 4: EXPERIMENT RESULT AND DISCUSSION** | **22-25** |
| 4.1 Experimental Setup | 22-25 |
| **CHAPTER 5: CONCLUSION** | **25** |
| **REFERENCES** | **26-29** |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **FIGURES** | **PAGE NO** |
| Figure 3.1: Show 10 data from my dataset | 10 |
| Figure 3.2: Data preprocessing steps | 11 |
| Figure 3.4.1: Process of Training Models | 13 |
| Figure 3.4.2: How logistic regression works | 15 |
| Figure 3.5.5: Measurement of different algorithms using TF-IDF vectorizer | 18 |
| Figure 3.5.6: Measurement of different algorithms using Count vectorizer | 19 |
| Figure 4.1: Experiment website | 23 |
| Figure 4.2: On medium security attacker view all the information from user tables | 24 |
| Figure 4.3: On high security attacker gain password hash of all user | 24 |

**LIST OF TABLES**

|  |  |
| --- | --- |
| **TABLES** | **PAGE NO** |
| Table 3.5: Confusion matrix structure | 16 |
| Table 3.5.6: Chosen Model from two different vectorization method | 19 |
| Table 3.6.1 Count Vectorizer → Logistic Regression && Support Vector Machine Model | 20 |
| Table 3.6.2 TF-IDF Vectorizer → Logistic Regression && Naive Bayes Model | 21 |
| Table 3.6.2 Final Model using TF-IDF vectorizer | 22 |
| Table 4.1 SQL injection attack results on target website | 23 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Introduction**

SQL injection is common and severe security vulnerability in web applications, where an attacker can manipulate user input to inject malicious SQL queries. This can lead to unauthorized access, data breaches, or even complete compromise of the underlying database. Traditional methods of detecting SQL injection attacks rely on pattern matching a rule-based approaches, input validations [1], which may have limitations in identifying complex and evolving attack patterns.

We are living in the 21st century where every second we are interacting with web applications. Day by day these web applications are becoming more advanced than previous time. In the previous time web application served only a static page with all information. All the information was public [2]. Anyone could see that information when they visited website. But present day every modern web application provides different kind of services. Present day web applications are becoming the key factor for any kind of business, in almost every sectors. Web applications manage their business process that includes several services like supply chain management, customer relationship management, employee management. User can buy goods, do shopping, get medical help, entertainment, communicate with anyone in the world and more through web applications.

Now days modern web applications are dynamic. Every, users have their own space in the applications. Every, web applications follow some common architecture like login system, shopping cart, payment system etc. [3]. User can create an account on the applications. This process contains authentication mechanism. It checks if the user is properly authorized or not. For this mechanism work applications must need a database that can store user login credentials information and check with that information for validation. This is how the basic login system works. Database also store user data like username, email, password, images. Ecommerce applications database store username, password, product information, user payment credentials, order history, chat messages and more. As we use web applications, we exchange some kind of data with the applications. These data become huge and more valuable as we use the applications.

Modern web applications mainly consist of three components front-end, database, back-end. Database is the central to the application. It stores all the information of that applications. User mainly interact with front-end. Front-end accepts request from the user then send it to the back-end. Back-end then fetch data from database if requested data exists in the database. Then it sends back data to the front-end and front-end display to the user. This is how most of the web applications works.

In recent years SQL injections have become one of the most critical types of attack to web-based system. According to Open Web Application Security Project’s (OWASP) [1] top ten vulnerabilities for 21 it ranked on 3rd position from Injection category. SQL injection attacks are occurred by entering malicious characters into input field of web applications. In the early days of internet most of the web applications served static content and those were not target of SQL injection attack. The first large scale of SQL injection attack was occurred in February 2002 on Guess.com [2]. By this attack attacker retrieve more than 200,000 customers names, credit card number, and expirations dates [2]. Sony’s Play station network was attacked in April 2011 by SQL injection. From this attack more than 77 million accounts were compromised that include 12 million of credit cards were stolen. Information like user accounts, password, address, credit card spending records was leaked, that indirectly caused Sony to lose up to 170 million US dollars [4]. Russian hacker known as Rasputin used SQL injection vulnerability to gain access as superuser to the database server and successfully compromised the system [4]. A huge amount of sensitive information was stolen from this attack from more than 20 universities, government agencies in the United States and United Kingdom [4]. From these real examples it shows that how dangerous a SQL injection attack can be. So what SQL injection is actually. SQL injection is a web vulnerability that violets the security of web application and allows an attacker to interfere with the queries that an application makes to its database. SQL injection attack allows an intruder to view data that they are not normally able to retrieve. Data that the applications itself is able to access and might include data belonging to others users. By and large, an attacker ready to alter this information that can set off persistent changes to application’s way of behaving or content [3]. In some case an attacker can take this attack to the next label by escalating the attack to compromise the underlying server or other back-end infrastructure, or perform denial-of-service attack [1] [3].

How the SQL injection attack works. According to OWASP [1] an application is helpless against assault when client provided information isn’t as expected validated or filtered or sanitized by the application and it straightforwardly utilize dynamic queries or non-parameterized calls without context-aware escaping, hostile information is used inside object-relational mapping seek parameters to extract extra, sensitive information, SQL contains structure and malicious data in dynamic queries, commands and stored procedures. The main method for sending a malicious query is variable that might come inform of a GET request, a POST request, HTTP Cookies or the like, which is a technique for building modern dynamic web applications using GET request, POST request [1]. According to Ines Jemal [3] because of sensitive impact from SQL injection attack several works have addressed. They try only to detect the SQLI once attack occurred. Some of works try to prevent it before occurring. Although, many works, techniques are proposed to fight against SQL injection attack, none of these have addressed the full scope of the attack. Therefore, there is actually no solutions that can prevent or detect all the different types of the SQL injection attack on web applications. So many of researchers trying to get benefit from the machine learning approach to propose more sophisticated solutions.

This research aims to investigate the effectiveness of machine learning techniques in detecting SQL injection attacks.

* 1. **Motivation**

The motivation behind conducting this research stems from the pressing need to enhance the security of web applications. I want to know how effectively a machine learning approach can prevent SQL injection attack. SQL injection attacks continue to pose a significant threat, and traditional preventive measures have shown limitations in effectively mitigating this type of attacks. Manual code auditing, input validations can be tedious, error-prone and testing to keep up with in enormous and complex applications [5]. As web applications become increasingly large, it becomes more challenging to identify and address all potential vulnerabilities manually. The evolving nature of SQL injection attacks necessitates a proactive and adaptive defense mechanism. Attacker consistently develop new techniques and evasion strategies to bypass traditional preventive measures. Therefore, a dynamic and intelligent approach is required to keep up with these evolving attack vectors [5]. Machine learning offers a promising answer for address these difficulties. So why not machine learning approach to see how effectively can detect and prevent SQL injection attacks from traditional preventive methods.

* 1. **Rationale of the Study**

In recent years SQL injections attack have become one of the most critical types of attack to web-based systems. According to Open Web Application Security Project’s (OWASP) [1] top ten vulnerabilities for 21 it ranked on 3rd position in Injection category. SQL injection attacks are occurred by entering malicious characters into input field of web applications. Although there are some prevention methods like input validation, parameterized queries, rule-based methods, but SQL injections attack are happening day by day. Day by day this type of attack is evolving. Those method have limitation to prevent evolving attacks. Machine learning solutions give a promising result for detection and prevention on this evolving attacks. So I want to see how effectively machine learning approach can detect and prevent SQL injection attack rather from traditional method.

* 1. **Research Question**

Can a machine learning approach effectively detect and prevent SQL Injection Attacks in Web Applications?

* 1. **Expected Outcome**

The expected outcome of this study to build an effective and reliable machine learning mechanism that significantly reduce the risk of successful SQL injection attacks on web applications to prove that machine learning approach can effectively detect and prevent SQL injection attacks.

**CHAPTER 2**

**BACKGROUND**

* 1. **Terminologies**

SQL Injection: SQL injection attack a type of attack where attacker manipulate user input to inject malicious query into a database. It can lead unauthorized access, data breaches, data manipulation or destruction of the database [1].

Machine Learning: A machine learning is a field of Artificial Intelligence that focuses on the development of algorithms and models that learn from data and make predictions or give a decision without being explicitly programmed [6].

Supervised Learning: Supervised learning is a machine learning approach where a model is trained on labeled data, where the desired output or class labels are known. The model learns from the input-output pairs to make predictions or classifications on new, unseen data [7].

Natural Language Processing: Natural Language Processing in short NLP a branch of artificial intelligence that gives computer power to understand data like text or spoken words in such a way that human beings can. In this study NLP is used to analyze and understand the syntax and semantics of SQL queries. It helps our machine learning models to understand SQL queries data [8].

Feature Extraction: Computer only understand two values 1 or 0. It does not understand any other values. So how can we teach machine learning models other types of data like text, image, videos etc. Now feature extraction is a process to transform these types of raw data into numerical features that can be process while preserving the information in original dataset. In my case feature extractions is used to transform SQL queries into useful features like query structure, keywords, others characteristics that can help to identify if the query is safe or malicious [9].

Input Validation: Input validation is a preventive measure that is used in SQL injection prevention to ensure that user input is safe and does not contain any malicious code. It involves validating and sanitizing user-given data before executing as database queries [1].

Parameterized Queries: One of the traditional techniques used to prevent SQL injection attacks by separating the query logic from user input values [1].

Labeled Dataset: A dataset where each data point in this study SQL queries, is associated with predefined label indicating whether the query is legitimate or malicious. This labeled dataset is used to train the machine learning model [1].

UNION-based Attack: UNION-based attack techniques use to combine the results of two or more individual queries into a distinct result set within a single row. This way an attacker can manipulate queries to get back multiple tables information from the database by using SQL UNION operator [7].

Time-based Attack: A SQL injection attack technique that depend on sending an SQL query to the database which forces the database to wait for a specified amount of time before responding [7].

Error-based Attack: It also a SQL injection attack technique that enables attacker to exploit error output from the database to manipulate its data. By generating error with this type of attack attacker can easily find out about valuable information about database like database structure, table information, data type etc. [7].

TF-IDF Vectorizer: A vectorization technique used in natural language processing (NLP) to convert textual data into its relevant numerical data since machine learning algorithm can not understand textual data. Full form of TF-IDF vectorizer is Term Frequency-Inverse Document Frequency Vectorizer. We can calculate TF-IDF Vectorizer [10] with this formula:

TF = number of repetition of word in sentence / total word in sentence

IDF = log (total number of sentence / no of sentence contain the word)

TF-IDF vectorizer = TF \* IDF

Count Vectorizer: It is also a vectorization technique to convert textual data to its relevant numerical value. It converts text document into a matrix of token counts. It counts the occurrence of each word in the documentation and creates a feature vector. It does not consider the importance or rarity of words, only their frequencies [11].

* 1. **Background Literature**

Main contributions of this paper are presenting an overview of the SQL injection attack. Different types of attack goals sources are discussed and described. Discussion about a classification of different SQLI attack detection and prevention countermeasures. Comparison between different proposed SQL injection attack countermeasures [3].

In this paper they proposed a lightweight machine learning approach to prevent SQL injection attack with help of embedding of word and Convolutional Neural Network (CNN). They have proposed a system that decode and denoise http request and sue word2vec to generate word embedding from those decoded characters, then train CNN, Multi-layer perceptron classifier [4].

This paper discussed about some flaws of other techniques that use to prevent SQL injection attack. They proposed a machine learning model with Support vector machine (SVM) to detect and prevent SQL injection attack [12].

The contributions of this paper are providing a dataset with feature hashing to train a machine learning model called Support vector machine (SVM) to predict accurately SQLIA to prevent attacks [13]. But in my paper, this SVM model failed in manual cross-checking step to predict accurately of malicious SQL queries.

This paper proposed a machine learning based heuristic algorithm to prevent SQL injection attack. They use a dataset with only 616 SQL statement. With this dataset they trained 23 different machine learning algorithms and chose best of 5 from them. Then they developed a Graphical User Interface based on those 5 best classifiers. They got accuracy of 93.8% from models [14].

The purpose of this study to investigate the potential of machine leaning algorithm to detect SQL injection attack on Cloud SaaS based applications. They trained some machine learning algorithms with some dataset and made predictions based on accuracy rate of those models [15].

Purpose of this paper to investigate a machine learning model that can improve the efficiency of the SQL injection attack detection. Author trained some machine leaning algorithm like artificial neural network, support vector machine random forest, logistic regression but this paper does not show the model evaluation to compare those models based on accuracy, precision, recall, f1-score evaluation matrix [6].

This research paper build dataset by detecting malicious logs files of web applications. Then it trained some machine learning algorithms with that dataset. Main purpose of this paper is to find out most suitable dataset [7].

This paper mainly focused on one machine learning model called CNN- BiLSTM. Discussed about CNN-BiLSTM in depth. Make a comparison with others machine learning algorithms based on accuracy, precision recall f1-score to satisfy proposed model CNN-BiLSTM [16].

This research paper proposed using multi-phase algorithm framework to prevent SQL injection attack with improved machine learning and deep learning to enhance database security [5].

* 1. **Problem Statement**

Different research study proposed different machine learning model. They make comparison between different models. But they all have some gap to demonstrate the effectiveness of machine learning algorithm to detect and prevent SQL injection attack. This research is to investigate that effectiveness of machine learning algorithm.

**CHAPTER 3**

**METHODOLOGY**

To evaluate the effectiveness of machine learning in detecting SQL injection attacks, the fellow methodology will be employed:

**3.1** **Data Collection**

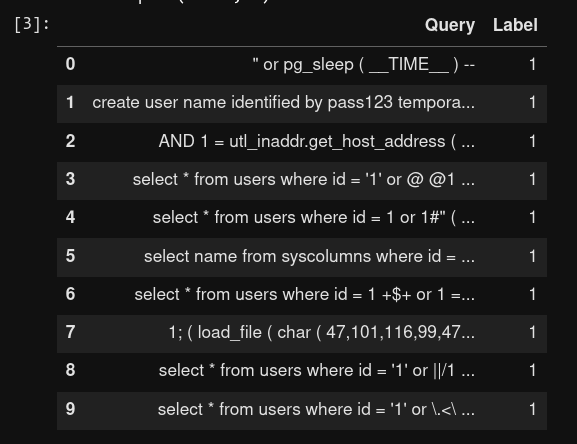
A dataset has been collected containing a combination of legitimate SQL queries and SQL injection attack queries from various source like Kaggle, GitHub. The dataset includes a variety of attack techniques, such as UNION-based attacks, time-based attack and error-based attack [3]. My collected dataset has two columns shown in Figure 1. One labeled as Query that contains all the SQL queries combination of legitimate and malicious queries. Another labeled as Label which contain only two values 1 and 0. Here 1 indicates that relative query is malicious and 0 indicates that relative query is legitimate.

Figure 3.1: Show 10 data from my dataset

**3.2 Data Preprocessing**

I have collected around 30907 data of SQL queries contain both legitimate and SQL injection attack queries. I have preprocessed data to further use. I have checked dataset for duplication, missing value, null value. Luckily there are no duplication, missing value or null value in my data set. I split dataset into two parts. 70% of data are used for training my models and 30% of data are used to test my trained models.

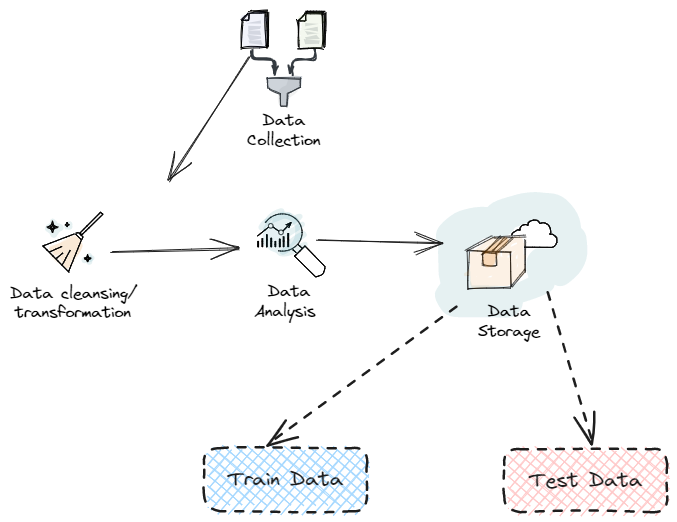


Figure 3.2: Data preprocessing steps

**3.3 Feature Extraction**

Relevant features extracted from my dataset’s column Query to represent the characteristics of both legitimate and SQL injections attack queries. These features include:

* Query structure: Query structure talk about the structure of the SQL query like how many commands are there in one query statement. Usually queries generated by the applications consist of one request only such as SELECT. But when an attacker injects malicious query in the original SQL statement, it brings out having one statement with having more than one requests [14].
* Keyword: Check for if there is any abnormal keyword or commands are in the queries. Generally, when an application reads users input and insert into a SQL query statement, it generally uses SELECT, INSERT, DELETE keyword. But if the statement contains abnormal keyword like DROP, COMMIT, ROLLBACK, REVOKE, GRANT, COMMIT, CREATE, DECLARE etc. then it may indicate that malicious actions is occurring. Having special keyword like keyword that use to know about SQL server version also indicate for having malicious activity [14].
* Syntactic patterns: In user input value having one or more SQL comment character that pass in the statement is not a common thing. User does not give any comment string trough input field. But attacker uses this facility to mess with database. Usually, attacker user this to include a comment character after their malicious queries to let the system neglect the rest of the query managed by those applications Also, for number of semicolons can manipulate the SQL statement. SQL query or statements terminate with one semicolon Now when an attacker execute malicious commands in the middle of the statement, attacker places a semicolon and then comments out the rest of the remaining statement. Which give result in having multiple semicolons [14].
* Specific attack payload: Having always true conditions on statement in the safe SQL statement is rare but attacker this is the common term used by attacker. This method uses to make sure the condition always satisfied after the OR operator [14]. Example: ‘ OR 1=1— . This query state will give true value because 1=1 is true always.

I have used two popular vectorize method of natural language processing (NLP) to convert my SQL query text data to its relevant numerical data. One is TF-IDF Vectorizer and another is Count Vectorizer. Both text classification method gives different model a promising accuracy, but in manual cross check models trained with count vectorizer classification’s converted data does not perform very well. It gives too many false positive. So, I have chosen models those are trained with TF-IDF vectorizer classification’s converted data.

**3.4 Model Training**

Various machine learning algorithm have used to train classification models using the extracted features. This includes popular algorithm such as Logistic Regression, Decision Tree, Random Forest Tree, Support Vector Machine, Naïve Bayes. Have saved a model based on accuracy for further use so I do not need to train my models again and again.

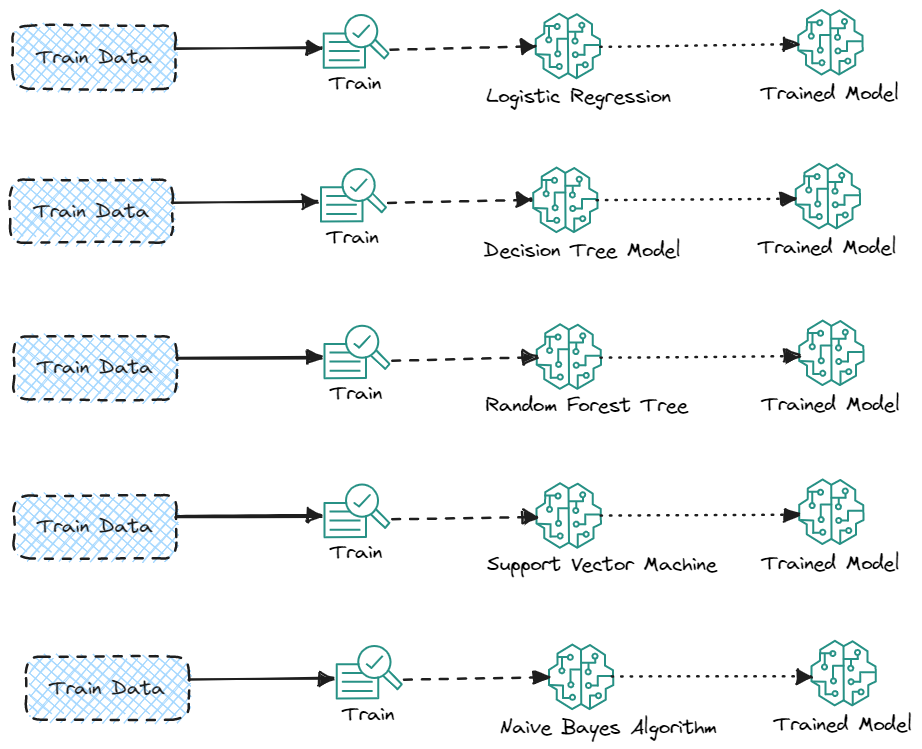


Figure 3.4.1: Process of Training Models

I have used both vectorizer technique to convert our textual data into their relevant numerical value. For both converted data I have trained those 5 machine learning algorithms to train model. Then measure the model’s performance and chose the best of them. Measurement chart can be seen on the Model evaluation section. My final chosen model is Logistic Regression model with TF-IDF Vectorization classification’s data

Classification methods [11] are critical components of machine learning and application development. Approximately 70% of data science challenges are categorization difficulties. There are other classification issues that may be solved, however logistic regression is a popular and effective regression approach for addressing the binary classification problem. Logistic regression may be used to solve a variety of classification issues, including spam identification, diabetes prediction, whether a buyer will purchase a product or not, and whether a user would click on a particular advertisement link or not in our study case if the SQL query is malicious or not. Logistic regression is a straightforward and widely used machine learning approach for two-class categorization. It is a statistical approach for classifying binary classes. Deep learning makes use of its core notions.

This diagram (Figure 3.4.2) shows how a logistic regression works. In logistic regression, we generally try to find out the probability that a data point belongs too a particular class, given as its input variables. In binary classification like my study case where my goal is to classify an input value SQL query into one of two classes if the query is malicious or not.

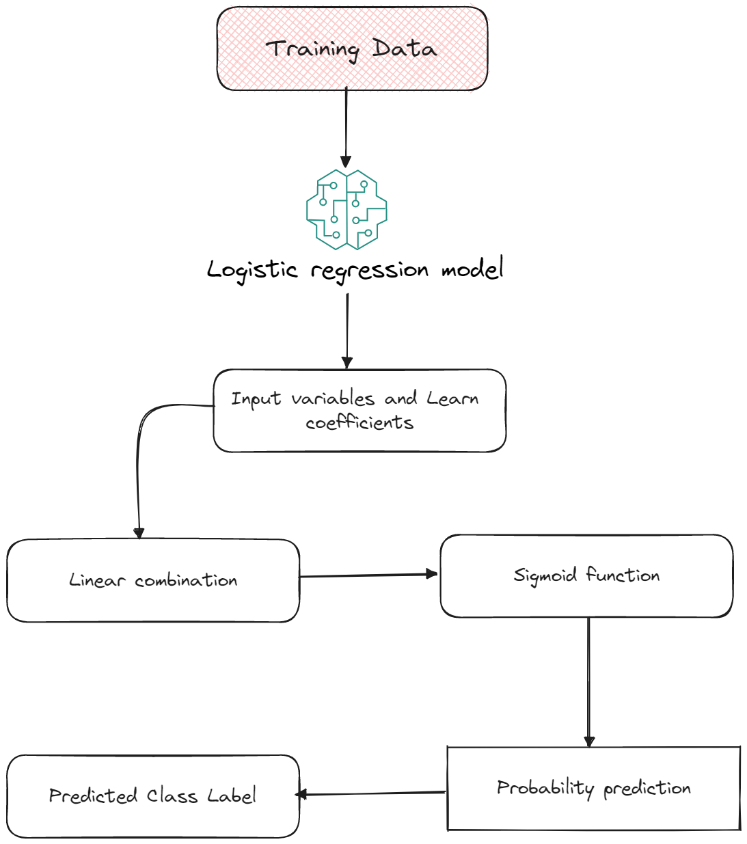


Figure 3.4.2: How logistic regression works**3.5 Model Evaluation**

The trained models have evaluated using standard metrics such as accuracy, precision, recall, f1-score, cross-validations techniques [14] to ensure robustness of the results.

I have used confusion matrix to evaluate our trained models. A confusion matrix is kind of table that provides summary of the performance of a classification model on a set of test data. Confusion matrix helps to visualize the number of

1. True positive (TP)
2. True negative (TN)
3. False positive (FP)
4. False negative (FN)

Confusion matrix has a tabular structure with two rows and two columns for binary classification problems, representing the actual and predicted class labels

Table 3.5: Confusion matrix structure

|  |  |  |
| --- | --- | --- |
| Actual Class | Predicted Class | |
| Positive | Negative |
| Positive | TP | FN |
| Negative | FP | TN |

From the table we can see that

* When predicted and actual value is positive then it is true positive
* When actual value is positive but our model gives negative then it false negative
* When actual value is negative but our model gives positive then it also false negative
* When actual value is negative and predicted value also negative than it is true negative.

By using this confusion matrix chart, we can measure our trained model to find out best model for my study. Accuracy, precision, recall f1-scorer these are common model evaluation metrics use in machine learning to assess the performance of classification models. These metrics provide insights into different aspects of the model’s performance.

**3.5.1 Accuracy**

Accuracy [14] for model evaluation measures the overall correctness of the model’s predictions by calculating the ratio of correctly classified instances to the total number of instances in the dataset.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

**3.5.2 Precision**

Precision [14] tells us about model’s ability to avoid false positives.

Precision = TP / (TP + FP)

**3.5.3 Recall**

Recall [14] known as sensitivity or true positive rate, measure the proportion of correctly predicted positive instances out of the actual positive instances. It works on model’s ability to identify all positive instances

Recall = TP / (TP + FN)

**3.5.4 F1-score**

Harmonic mean of precision and recall, a balanced evaluation metric that considers both precision and recall [14].

F1-score = 2 \* (Precision \* Recall) / (Precision + Recall)

**3.5.5 TF-IDF vectorizer classification**

Figure 3.5.5: Measurement of different algorithms using TF-IDF vectorizer

From the chart I can easily identify that logistic regression and naïve bayes algorithms giving me best result where other algorithms giving me low accuracy rate. Between logistic regression and naïve bayes logistic regression giving me best accuracy rate which 97.24% where naïve bayes giving 96.83%. Since logistic regression giving us highest accuracy rate, we chose logistic regression model from using TF-IDF Vectorization method

**3.5.6 Count Vectorizer Classification**

Figure 3.5.6: Measurement of different algorithms using Count vectorizer

From the chart I can easily identify that logistic regression, support vector machine and naïve bayes algorithms giving me best result where other algorithms giving me low accuracy rate. Logistic regression model giving me 98.90% accuracy, naïve bayes giving me 97.16% accuracy rate and support vector machine giving me 99.21% accuracy rate which is the highest accuracy rate.

Now I get different results by using two different vectorization methods for same dataset and using same machine learning algorithms. Now how can I choose the best model that will give accurate prediction. I can make decision after doing manual cross checking with these trained model

Table 3.5.6: Chosen Model from two different vectorization method

|  |  |  |  |
| --- | --- | --- | --- |
| TF-IDF Vectorization | | Count Vectorization | |
| Logistic regression - 97.24% | Naïve bayes - 96.83% | Logistic regression - 98.90% | Support vector machine - 99.21% |

**3.6 Manual Cross Checking**

I have written 7 SQL queries in a file those are tested on real website. In these 7 queries 5 of them are malicious queries that occurred successful SQL injection attack on website and 2 of theme are safe query. I have manually tested these queries with all these (mentioned on Table 3.5.6) model to see if they can predict if the queries are malicious or not. Here is the result

Table 3.6.1 Count Vectorizer → Logistic Regression && Support Vector Machine Model

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Actual Value | Predicted Value | Detected? |
| 'UNION SELECT user, password FROM users# | 1 | 0 | False |
| 1 or 1=1 UNION SELECT user, password FROM users# | 1 | 1 | True |
| 1'UNION SELECT user, password FROM users# | 1 | 0 | False |
| 1 or sleep(4)# | 1 | 1 | True |
| 1' benchmark(10000000,MD5(1))# | 1 | 0 | False |
| 5 -- | 0 | 0 | True |
| SELECT username, password FROM users | 0 | 0 | True |

Here from Table 3.6, we can see that using count vectorizer method logistic regression and support vector machine model gives 3 false negative (FN), 2 true negative (TN), 2 true positive (TP). So, using count vectorizer logistic regression and support vector machine model marked total 2 queries as malicious and 5 queries as safe but actually there is 5 malicious queries and 2 safe queries out of total 7 queries. But these models failed to identify malicious queries efficiently on manual checking. They both failed on manual cross checking.

Now check another two mentioned models (Table 3.5.6) using TF-IDF vectorizer logistic regression and naïve bayes model

Table 3.6.2 TF-IDF Vectorizer → Logistic Regression && Naive Bayes Model

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Actual Value | Predicted Value | Detected? |
| 'UNION SELECT user, password FROM users# | 1 | 1 | True |
| 1 or 1=1 UNION SELECT user, password FROM users# | 1 | 1 | True |
| 1'UNION SELECT user, password FROM users# | 1 | 1 | True |
| 1 or sleep(4)# | 1 | 1 | True |
| 1' benchmark(10000000,MD5(1))# | 1 | 1 | True |
| 5 -- | 0 | 0 | True |
| SELECT username, password FROM users | 0 | 0 | True |

Now from Table 3.6.2 we manually cross checking for using TF-IDF vectorization method logistic regression and naïve bayes [17] model performance. From the table we can see that both logistic and naïve bayes model give 5 true positive (TP) and 2 true negative (TN) means that both models successfully identify those 5 malicious queries and 2 safe queries. Since logistic regression model has highest accuracy between logistic regression and naïve bayes model, Logistic regression is now our final model

Table 3.6.2 Final Model using TF-IDF vectorizer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | Accuracy | Precision | Recall | F1-score |
| Logistic Regression | 97.24% | 93.76% | 98.85% | 96.25% |

**CHAPTER 4**

**EXPERIMENT RESULTS AND DISCUSSION**

**4.1 Experimental Setup**

I have setup Damn Vulnerable Web Application (DVWA) [18] website with database in my local machine that replicate real word web applications. This website has 3 security options.

1. Low – have no security at all, malicious SQL query will execute and attacker can access to the database
2. Medium – check for input validation on user inputs
3. High – have rule based security before sending SQL queries to database

In Figure 4.1 we can see the experimental website to carry out our SQL injection attack test. In website we can see there is an input field that except id number and if the id match, then it shows person name and surname information. I will use this input field to attack SQL injection with enabling those 3-security label low, medium, high one by one and check how can I bypass those security and do successful SQL injection attack.

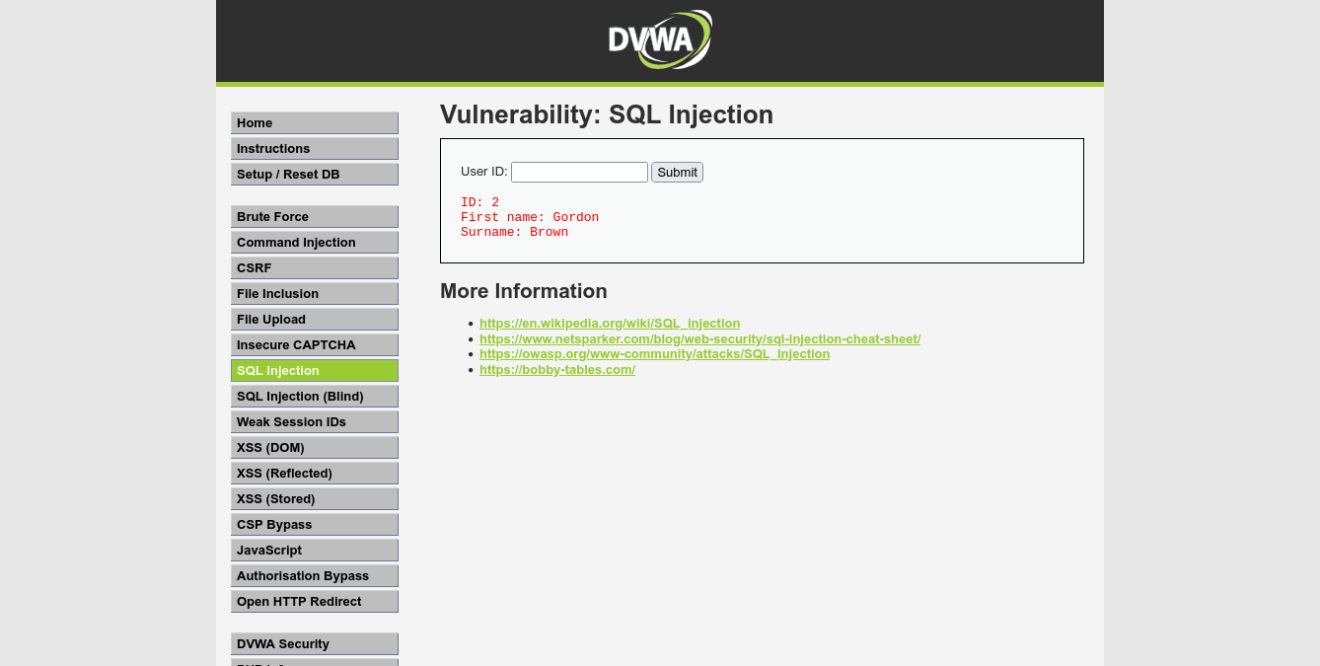


Figure 4.1: Experimental website

Table 4.1 SQL injection attack results on target website

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Label | Payload | Low | Medium | High |
| Malicious | 'UNION SELECT user, password FROM users# | 1 | 0 | 1 |
| Malicious | 1 or 1=1 UNION SELECT user, password FROM users# | 0 | 1 | 0 |
| Malicious | 1'UNION SELECT user, password FROM users# | 1 | 0 | 1 |
| Malicious | 1 or sleep(4)# | 1 | 1 | 0 |
| Malicious | 1' benchmark(10000000,MD5(1))# | 1 | 0 | 1 |
| Safe | 5 -- | 0 | 0 | 0 |
| Safe | SELECT username, password FROM users | 0 | 0 | 0 |

Here 5 queries are malicious queries and 2 are safe queries.

1 = Failed to detect malicious query

0 = Detect malicious query

From the Table 4.1 we can see that although we have traditional SQL injection prevention mechanism but failed to detect some malicious queries and successfully bypass and execute queries. So, attacker gain access to the database.

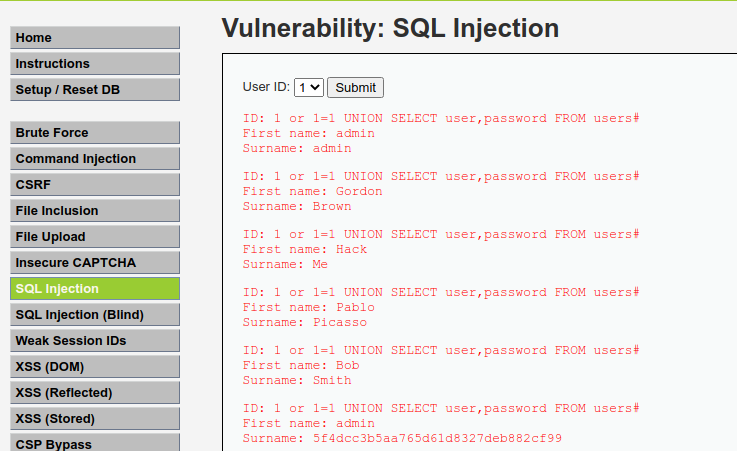


Figure 4.2: On medium security attacker view all the information from user tables

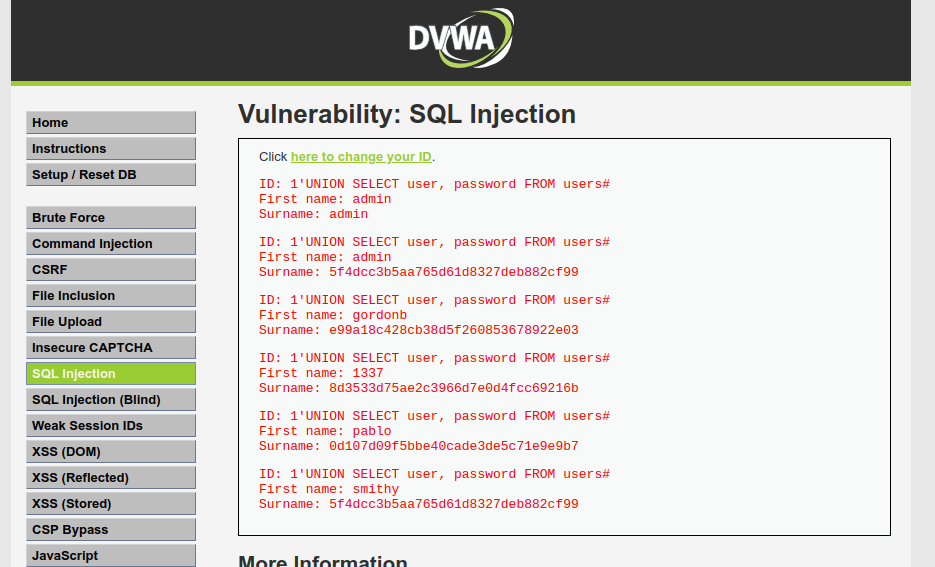


Figure 4.3: On high security attacker gain password hash of all user

For these 7 custom queries that contains both legitimate queries and malicious queries our security of web applications with input validation, rule-based validation has failed to detect some malicious query. But our machine learning model [Table 3.6.2] successfully detect all 5 malicious queries. So, I can say that machine learning model is showing a promising result to detect malicious queries efficiently. This is the purpose of this study.

**CHAPTER 5**

**CONCLUSION**

This research study to determine whether machine learning can effectively detect SQL injection attacks and potentially outperform rule-based methods, input validation. As we can see from the previous chapter that our machine learning performs better than traditional rule-based methods, input validations to detect SQL injection queries. The result will provide insights into the applicability and limitations of machine learning techniques in enhancing the security of web applications against SQL injection vulnerabilities.

# **References**

|  |  |
| --- | --- |
| [1] | "Open Worldwide Application Security Project," [Online]. Available: https://owasp.org/. [Accessed 16 May 2023]. |
| [2] | M. Alenezi, M. Nadeem and R. Asif, "SQL injection attacks countermeasures assessments," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 21, no. 2, pp. 1121-1131, February 2021. |
| [3] | I. Jemal, O. Cheikhrouhou, H. Hamam and A. Mahfoudhi, "Sql injection attack detection and prevention techniques using machine learning," *International Journal of Applied Engineering Research,* vol. 15, no. 6, pp. 569-580, 2020. |
| [4] | D. Chen, Q. Yan, C. Wu and J. Zhao, "Sql injection attack detection and prevention techniques using deep learning," *Journal of Physics: Conference Series,* vol. 1757, no. 1, p. 012055, 2021. |
| [5] | A. A. Abadulla , A. Badii and F. Stahl, "Multi-Phase Algorithmic Framework to Prevent SQL Injection Attacks using Improved Machine learning and Deep learning to Enhance Database security in Real-time," in *15th International Conference on Security of Information and Networks (SIN)*, Sousse, 2022. |
| [6] | Y. Abdulmalik, "An improved SQL injection attack detection model using machine learning techniques," in *International Journal of Innovative Computing*, 2021. |
| [7] | M. A. Azman, M. F. Marhusin and R. Sulaiman, "Machine learning-based technique to detect SQL injection attack," *Journal of Computer Science,* vol. 17, no. 3, pp. 296-303, 2021. |
| [8] | P. Roy, R. Kumar and P. Rani, "SQL Injection Attack Detection by Machine Learning Classifier," *In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC),* pp. 394-400, 2022. |
| [9] | K. R. Jothi, N. Pandey, P. Beriwal and A. Amarajan, "An efficient SQL injection detection system using deep learning," in *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2021. |
| [10] | T. Muhammad and H. Ghafory, "SQL Injection Attack Detection Using Machine Learning Algorithm," *Mesopotamian journal of cybersecurity,* pp. 5-17, 2022. |
| [11] | V. Sharma and S. Kumar, "Comparative Study of Machine Learning Algorithms for Prediction of SQL Injections," *In Computer Vision and Robotics: Proceedings of CVR,* pp. 455-466, 2023. |
| [12] | T. Latchoumi, M. S. Reddy and K. Balamurugan, "Applied Machine Learning Predictive Analytics to SQL Injection Attack," *European Journal of Molecular & Clinical Medicine,* vol. 07, no. 02, 2020. |
| [13] | S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, 2017. |
| [14] | M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," in *International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, Ras Al Khaimah, 2019. |
| [15] | D. Tripathy, R. Gohil and T. Halabi, "Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning," in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance*, Baltimore, MD, USA. |
| [16] | N. Gandhi, J. Patel, R. Sisodiya, N. Doshi and S. Mishra, "A CNN-BiLSTM based Approach for Detection of SQL Injection Attacks," in *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, Dubai, 2021. |
| [17] | H. Herman, I. Riadi and Y. Kurniawan, "Vulnerability Detection With K-Nearest Neighbor and Naïve Bayes Method Using Machine Learning," *International Journal of Artificial Intelligence Research,* vol. 7, no. 1, 2023. |
| [18] | d. "GitHub - digininja/DVWA: Damn Vulnerable Web Application (DVWA)," Github, 29 Februrary 2008. [Online]. Available: https://github.com/digininja/DVWA. [Accessed 12 March 2023]. |
| [19] | S. Mishra, "SQL injection detection using machine learning," 2019. |
| [20] | M. Alghawazi, D. Alghazzawi and S. Alarifi, "Detection of sql injection attack using machine learning techniques: a systematic literature review," *Journal of Cybersecurity and Privacy,* vol. 2, no. 4, pp. 764-777, 2022. |
| [21] | J. Hu, W. Zhao and Y. Cui, "A survey on SQL injection attacks, detection and prevention," in *In Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, 2020. |
| [22] | A. Falor, M. Hirani, H. Vedant, P. Mehta and D. Krishnan, "A deep learning approach for detection of sql injection attacks using convolutional neural networks," in *In Proceedings of Data Analytics and Management: ICDAM 2021*, Springer Singapore, 2022. |
| [23] | M. N. Kavitha, V. Vennila, G. Padmapriya and A. R. Kannan, "Prevention Of Sql Injection Attack Using Unsupervised Machine Learning Approach," vol. 12, 2021. |
| [24] | M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh and R. Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks," *Information Security Journal: A Global Perspective,* pp. 1-14, 2021. |
| [25] | D. E. Nofal and A. A. Amer, "SQL Injection Attacks Detection and Prevention Based on Neuro—Fuzzy Technique," *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges,* pp. 93-112, 2021. |
| [26] | B. A. Pham and V. H. Subburaj, "An experimental setup for detecting SQLi attacks using machine learning algorithms," *In Journal of The Colloquium for Information Systems Security Education,* vol. 8, no. 1, pp. 5-5, 2020. |
| [27] | Q. Li, W. Li, J. Wang and M. Cheng, "A SQL injection detection method based on adaptive deep forest," *IEEE Access, 7,* pp. 145385-145394, 2019. |
| [28] | M. M. Hassan, R. B. Ahmad and T. Ghosh, "SQL injection vulnerability detection using deep learning: a feature-based approach," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI),* vol. 9, no. 3, pp. 702-718, 2021. |
| [29] | A.-M. M. H. Ali and M. N. Jasim, "Review of SQL injection attacks: Detection, to enhance the security of the website from client-side attacks," *International Journal of Nonlinear Analysis and Applications,* vol. 13, no. 1, pp. 3773-3782, 2022. |
| [30] | P. Tang, W. Qiu, Z. Huang, H. Lian and G. Liu, "SQL injection behavior mining based deep learning," *In Advanced Data Mining and Applications: 14th International Conference, ADMA 2018,* vol. 14, pp. 445-454, 2018. |
| [31] | H. Zhang, B. Zhao, H. Yuan, J. Zhao, X. Yan and F. Li, "SQL injection detection based on deep belief network," in *In Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019. |
| [32] | L. Erdődi, Å. Å. Sommervoll, F. Zennaro and F. M. Zennaro, "Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents," *Journal of Information Security and Applications,* vol. 61, p. 102903, 2021. |
| [33] | D. Mehta, H. Suhagiya, H. Gandhi, M. Jha, P. Kanani and A. Kore, "SQLIML: A Comprehensive Analysis for SQL Injection Detection Using Multiple Supervised and Unsupervised Learning Schemes," vol. 4, no. 3, p. 281, 2023. |
| [34] | F. K. Alarfaj and N. A. Khan, "Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks," *Applied Sciences,* vol. 13, no. 7, p. 4365, 2023. |
| [35] | B. M. Thombare and D. R. Soni, "Prevention of SQL Injection Attack by Using Black Box Testing," *In 23rd International Conference on Distributed Computing and Networking,* pp. 266-272, 2022. |
| [36] | K. Baptista, E. M. Bernardino and A. M. Bernardino, "Detecting SQL injection vulnerabilities using artificial bee colony and ant colony optimization," *In Information Systems and Technologies: WorldCIST 2022,* vol. 3, pp. 273-283, 2022. |
| [37] | A. Hadabi, E. Elsamani, A. Abdallah and R. Elhabob, "An Efficient Model to Detect and Prevent SQL Injection Attack," *Journal of Karary University for Engineering and Science,* 2022. |
| [38] | Y. Guan, J. He, T. Li, H. Zhao and B. Ma, "SSQLi: A Black-Box Adversarial Attack Method for SQL Injection Based on Reinforcement Learning," vol. 15, no. 4, p. 133, 2023. |
| [39] | S. Lodha and A. Gundawar, "SQL Injection and Its Detection Using Machine Learning Algorithms and BERT," *In Cognitive Computing and Cyber Physical Systems: Third EAI International Conference, IC4S 2022, Virtual Even,* pp. 3-16, 2023. |
| [40] | D. Lu, J. Fei and L. Liu, "A Semantic Learning-Based SQL Injection Attack Detection Technology. Electronics," vol. 12, no. 6, p. 1344, 2023. |
| [41] | A. H. Farhan and R. F. Hasan, "Detection SQL Injection Attacks Against Web Application by Using K-Nearest Neighbors with Principal Component Analysis," *In Proceedings of Data Analytics and Management: ICDAM,* pp. 631-641, 2023. |
| [42] | K. Gaur, M. Diwakar, P. Singh, T. Sachdeva and N. K. Pandey, "SQL Injection Attacks and Prevention," *In 2023 6th International Conference on Information Systems and Computer Networks (ISCON),* pp. 1-4, 2023. |
| [43] | F. Ochieng, D. Kaburu and N. G. John, "Automation-Based User Input Sql Injection Detection and Prevention Framework," *Computer and Information Science,* vol. 16, no. 2, 2023. |
| [44] | S. Pillai and A. Sharma, "Hybrid unsupervised web-attack detection and classification–A deep learning approach," *Computer Standards & Interfaces,* p. 103738, 2023. |