

SCAMM: Detection and Prevention of SQL Injection Attacks Using a Machine Learning Approach

by

Auninda Alam - 19241021
Marjan Tahreen - 19241020
Md Moin Alam - 17101060
Shahnewaz Ali Mohammad - 19241014
Shohag Rana - 20141033

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
BRAC University
January 2021

© 2020. BRAC University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis which is submitted contains our original research while pursuing a degree at BRAC University.
2. The thesis paper does not include any materials that has been previously published by a third party, with an exception where it has been clearly and appropriately cited with complete and accurate citation.
3. All the key sources of support have been acknowledged by us.

Student's Full Name & Signature:



Auninda Alam
19241021



Md Moin Alam
17101060



Marjan Tahreen
19241020



Shahnewaz Ali Mohammad
19241014



Shohag Rana
20141033

Approval

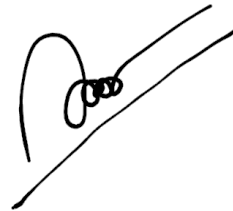
The thesis/project titled “SCAMM: Detection and Prevention of SQL Injection Attacks Using a Machine Learning Approach” submitted by

1. Auninda Alam - 19241021
2. Marjan Tahreen - 19241020
3. Md Moin Alam - 17101060
4. Shahnewaz Ali Mohammad - 19241014
5. Shohag Rana - 20141033

In Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 11, 2021.

Examining Committee:

Supervisor:
(Member)



Dr. Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)



Md. Golam Rabiul Alam
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Mahbubul Alam Majumdar
Professor and Dean
Department of Computer Science and Engineering
BRAC University

Abstract

Importance of cyber-security in protecting our valuable data and information is huge in this era of technology. Since numerous amounts of cyber-attacks take place every day, the development of a more secured system so that it can predict and stop cyber-attacks from happening, has been our concern for years. This research paper is focused on developing such a means that will be able to detect and prevent SQL Injection Attack successfully. SQL Injection attack is a type of cyber-attack that uses malicious SQL queries for internal data manipulation and retrieving hidden information from the back-end database that were not intended to be displayed. SQL Injection Attack even makes a database vulnerable to other kinds of attacks. Since most of the organizations use a SQL based back end database to store data, all of their data is exposed to a simple form of attack if they are not properly defended. The aim of this research is to develop a model by finding out the best machine learning algorithm to predict and prevent SQL Injection Attack. A brief explanation of our workplan, our experimentation and the results of our experiments are discussed in this paper.

Keywords: *Machine Learning; SQL Injection; SCAMM; Naive Bayes; KNN; Neural Network Classifier; Random Forest; Logistic Regression;*

Dedication

First of all, this research work is devoted to our parents, without whom we would not have been able to do this. It is due to their meaningful life lessons and sacrifices which they made for us, helped us to work hard and achieve our goals.

It is also dedicated to our honorable supervisor without whom this research would not have been possible. Our honorable supervisor believed in us, motivated and encouraged us to get the work done. Without his valuable guidelines and teachings, this research would not have been achievable.

Finally, we dedicate our research to BRAC University. BRAC University is a renowned University in our country. The university is always inspiring the students to become global citizens. The university gave us the platform we needed in order to share our knowledge and research.

Acknowledgement

The very first person to be stated is our honorable thesis supervisor, Dr. Muhammad Iqbal Hossain. He is the most significant person in our research. He has guided, supported and motivated us until the end. As a result of his continued support, we were able to perform this research. Next, we would like to thank other faculty members and all senior brothers and sisters for leading us down the right direction and providing us with unwavering guidance and constant motivation through the process of studying and writing this article. Without them, this fruitful journey would not have been feasible.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Dedication	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	1
1 Introduction	2
1.1 Motivation	3
1.2 Problem Statement	3
1.3 Objective and contributions	4
1.4 Thesis Structure	4
2 Background	6
2.1 Literature Review	6
2.2 Algorithms	10
2.2.1 Random Forest	10
2.2.2 KNN	10
2.2.3 Naïve Bayes Algorithm	11
2.2.4 Logistic Regression	11
2.2.5 Neural Network Classifier	12
3 Proposed model	13
3.1 Dataset description	13
3.1.1 Data preprocessing	13
3.2 Model description (Workflow)	14
4 Experimentation and Result Analysis	16
4.1 Experimentation	16
4.2 Result Analysis	18
5 Conclusion	20

List of Figures

2.1	Random Forest Algorithm	10
2.2	KNN Algorithm	11
2.3	A logistic function	12
2.4	A simple Neural Network	12
3.1	Workflow of our proposed model.	14
4.1	Result of Logistic Regression	16
4.2	Result of Neural Network Classifier	17
4.3	Result of Random Forest Classifier	17
4.4	Result of KNN	17
4.5	Result of Naïve Bayes Algorithm	18

Chapter 1

Introduction

Our internet use in volume and also dealing with secretive data has grown significantly in the last few decades. We are spending time and resources to make this medium more and more secure. But every day there is an ever-growing threat to our privacy. We do not have the option to avoid the internet as it has become the backbone of modern infrastructure. So, the only option we have is to make this internet a safe place for the users to be in and share their valuable data on. If we do not ensure the safety of one's data the whole infrastructure will be at risk and the credibility of information will be questioned. In this era of technology, most of the institutions and organizations use a centralized back end database for storing and retrieving data. If the system is not secured enough then it becomes vulnerable to cyber-attacks, especially SQL Injection attack which retrieves and manipulates internal data through malicious SQL queries and statements. And we all know that "prevention is better than cure." So, it's better to be safe than to be sorry.

There are many sorts of common and critical attacks such as Trojan, SQL injection. SQL is a query language that was designed to access data, modify data, delete data from SQL databases like MySQL, Oracle, SQL Server. Many web applications and websites store all their data in the SQL database. We can also run OS commands through SQL commands. Successful injection can have very serious results like attackers can find the credentials of other users in the database as well as get the trade secrets through this. The attacker may alter the data of the existing database through this injection. This sort of attack may harm the intellectual property of many internet users. To prevent this, we must prevent attackers from this sort of attack to keep the users safe. This is the goal of our research.

In this paper, we will use several renowned machine learning algorithms to design several models which will be able to tell an SQL injection command apart from a regular harmless user input. We will then compare the results and identify the best suited algorithm for our job. Later on, we will be developing such a model that will detect, and take necessary steps to prevent SQL Injection attacks. For this, we will retrieve a dataset containing sufficient amounts of SQL injection attack samples along with regular user data and train our model with it. After that, we will feed test data sets for testing the accuracy of our model. We will use the algorithm which yielded the best accuracy on detecting SQL injection attacks for this task. Finally, we will incorporate our most successful model with a website we created. We will

connect our model to every input field so that it can provide complete security to SQL injection attempts throughout the entire website. This test will prove the effectiveness of our developed model in practical scenarios.

1.1 Motivation

Data is the heart and soul of any organization. To use this data to make decisions and predict any outcome would sometimes mean a life and death situation. In February of 2016 Bangladesh had a massive blow to its economy due to the fact that the Bangladesh Bank has been attacked by hackers and we lost more than 81 million USD [4]. According to Statistica, for mid-sized companies, the average cyber loss in the 2019 year amounted to 1.56 million U.S. dollars. In a survey of global companies carried out on May 2019, the average loss was calculated to be 4.7 million U.S. dollars over all sizes of companies [3]. Another important statistic regarding the SQL Injection attack shows that SQL injection attack is used by hackers in 51% of cases [9]. So, the importance of protecting a system from the SQL Injection attack is beyond description.

Therefore, we wanted to develop a model that will be able to detect SQL injection attempts and help us take necessary steps to prevent the attack.

1.2 Problem Statement

There are many ways to perform SQL injection attacks currently. There are tautology-based SQL injection attacks which bypass user authentication and extraction data by inserting a tautology in the WHERE clause of the SQL query. After that, there are Logically Incorrect queries where the attacker gathers some data and sends an incorrect query so that the server responds in form of error messages along with some important information regarding the database like table name, column name etc. Again, an attacker can add a wrong query with a correct one using the UNION keyword. Apart from these there are several other approaches such as the stored procedure method, piggy-backed queries, inference etc.

Certainly, the best way to prevent SQL injection attacks is coding in the most secure manner but in most cases that is not possible. Not every programmer has the best kind of coding experience and also it is not always possible to think about ways to prevent all kinds of vulnerabilities without getting distracted from the main purpose of a program. And even the best-case scenario new ways of getting past the defenses will always emerge as the attackers will not be idle either.

Since there are so many ways to get a harmful SQL query in, it has become extremely difficult to identify a potential attack and prevent it. There are a lot of models available which is supposed to prevent SQL injection attempts but these models either compare every input with a set amount of example data. The problem that arises with this approach is that in order to provide the most security, the flexibility and freedom of taking user inputs is affected. Again, if freedom is to be

maintained, the system becomes too vulnerable to SQL injection attempts. So, the limitations of the techniques to prevent SQL injection attacks is evident and as a result our precious systems are more vulnerable than ever.

1.3 Objective and contributions

Since there is no fixed format of SQL injection queries it is not possible to detect any kind of injection attempt by matching them to a static database of examples. So, we have decided to use machine learning algorithms in order to train our model so that it may detect any known kind of SQL injection attempts and even identify new approaches of attacks from its previous training. We are looking to train our model using a huge dataset containing all types of SQL injection queries as well as regular input. We will use the algorithm best suited for this task so that the model can identify SQL injection approaches with high accuracy and also reduce the rate of false positive alarms. Our model will serve as the barrier to any kind of harmful query from entering the database. It will block potential SQL injection attacks and notify the respective authority of the server.

Our model can be used to supplement the protective measures of any kind of website. It can be used to protect the database of various web applications. It is expected to serve as an added layer of protection to database servers of any kind of system. It should serve as a cheap but effective alternative to much expensive and complex protections against SQL injection attacks.

1.4 Thesis Structure

This report describes a model that has been designed to detect and prevent SQL injection attacks on websites.

Firstly, the Introduction Chapter (Chapter 1) describes the inspiration and motivation behind our work. The objective of our research and the description of our work is briefly discussed in this chapter.

After that, in the Background Chapter (Chapter 2), we have referred to some previous works which had a similar objective. The papers were analyzed in order to learn about multiple approaches towards the solution of the problem and their shortcomings. Furthermore, the algorithms that were considered and tested for the model are briefly discussed here.

In the Proposed Model Chapter (Chapter 3), we mentioned the source of our dataset and the procedure of data pre-processing. The workflow of our proposed model is briefly discussed in this section.

Then, the Experimentation and Result Analysis Chapter (Chapter 4) describes the procedure of our implementation. It shows our process of experimenting with different algorithms and finding out the most suitable algorithm for our model. Also,

the analysis of the results of our experimentation and the competency of our model is analyzed in this chapter.

Finally, in the conclusion section (Chapter 5), we have discussed about the significance that our model may hold in modern cyber-security. We have also mentioned the possible ways of improving our current model so that it can detect SQL injection attacks more accurately.

Chapter 2

Background

2.1 Literature Review

In paper [8], a method to detect complicated SQL injection attacks has been proposed which was based on adaptive deep forest algorithm. Here, an average of the earlier outputs was taken and used along with raw feature vector in order to concatenate input of each layer. This makes the deep forest structure more optimized in this paper. Later on, for utilizing the error rates in order to update the weights in every layer, they introduced an algorithm called AdaBoost based deep forest model. In this paper, it was found that, compared to the classical machine learning methods, the proposed model had a superior performance. The experimental results have showed this. There were two stages in the proposed model for detecting the SQL injection. These were off-line training phase and online testing phase. 10000 SQL injection samples were collected. Here, the features that were extracted from different datasets included UNION query, executer SQL commands, error-based injection and blind injection.

In paper [7], they proposed an algorithm which is based on a machine learning heuristic approach in order to prevent SQL injection attack. The advantages of dynamic and static analysis have been combinedly used in this paper through machine learning algorithm. A well-researched dataset was taken into account that covered all possible SQL statements. The paper used MATLAB in order to develop the model. They used 23 different machine learning classifiers in order to train and test the dataset. Then they choose the best 5 classifiers out of 25 on the basis of performance regarding true positive and true negative rates. After the end of training of the classification learners, they studied the accuracy of each classifier. They selected the most effective and accurate 5 classifiers in order to achieve an accuracy of 93.8%.

In another paper [12], ResNet has been used which acquires knowledge of SQLI attacks by itself. They proposed a ResNet model that was combined of 28 layers such as convolution layer, Dense layer, input layer, max pooling, and other such layers. To build this ResNet model they collected datasets from log documents of standard SQL Injection devices which consisted of all the different types of SQL Injection Attacks. Basically, the dataset was split into two sections. One section was training data and 85% of the information was regarded for training. Here the training data is again parted as validation data which is 20% and training data

which is 80%. Another one was testing data which consisted of 15% data. They employed Label Encoder to make numeric text from non-numeric text. They retained the lexicon of training and testing information and transformed these to matrix and also, they used tokenizer by providing expression number and vectorizing the lexicons by transforming each lexicon into a number which represented SQLi characteristics. After converting each token into a sequence of integers they executed RESNETSQL () with all layers and trained the design for some iterations so that the model can return accuracy, loss and also predict the output. Furthermore, they also implemented an LSTM method for comparing the outcomes of ResNet and an LSTM model with the dataset they collected. The outcome of Resnet model is 99% accuracy and that of the LSTM is 98% accuracy. Moreover, this ResNet model was implemented and programmed efficiently which can spontaneously detect all sorts of SQLi Attacks.

In this paper [15], they have done research on some ML strategies which uses a payload as input and decides if the input contains a malicious code or not to detect SQLi attacks. Firstly, they collected datasets from different public repositories and then did cleaning and preprocessing by using feature engineering. After that, they shuffled the dataset that contained 7576 harmful SQL statements and 100496 correct inputs so that biases can be removed. Then to implement the classifier at first, they created a set of decision trees from a randomly selected subset of the training set by using Random Forest classifier. They also used AdaBoost Classifier that combines multiple classifiers by setting weights and trains the data in each loop for increasing accuracy of unusual predictions. All features were merged for providing the inputs to the classifiers and chi-squared scoring method was employed for the custom feature space. Additionally, they implemented functions of python for feature extraction. To determine features of SQL they calculated the amount of keywords in SQL by finding the strings inputted by users with the probable list of keywords. After creating the space of features, to systematize harmful SQL statements from normal statements they used ML models. Finally, they fitted training data into the model and evaluated the best model and also tested unknown data in the models. When they implemented and evaluated the classifier, the feature vectors were generated from the collected input data where with five features and achieved high accuracy. On the other hand, the final optimized models had ten features that gave better accuracy and can detect SQLi attacks and also can successfully predict and classify the legal and illegal queries by ensuring higher than 97% accuracy.

The paper [14] proposes a model that will be able to identify SQL injection vulnerabilities. The authors of this paper have also focused more on deep learning-based ML algorithms. The authors trained the proposed classifier using several machine learning algorithms in order to determine the best suited algorithm for their work. Random Forest, Logistic Regression and SVM were used as generic algorithms and LSTM, RNN, CNN and MLP were the deep learning algorithms that were used. Ten folds cross validation was used for training and testing the data. 18 different IVS attributes were taken from the PHP code files for training and testing the classifier models. The authors used precision, recall, accuracy and F-measure to compare the performances of the chosen algorithms. They also used BOW to exclude comments, variable names, keywords. Thirdly, Word2vec was used to know the condition of the

traits. It was seen that the proposed classifier was performing better while using the deep learning-based algorithms. Finally, the authors found that implementing CNN algorithm with the help of BOW technique yielded the highest precision which was 95.4% but a multilayer perceptron that was trained using BOW had a recall that was highest, which was 63.7% as well as the highest f-measure which was 0.746. The authors also mentioned that they would improve their research further by expanding their dataset.

The paper [6] has represented different forms and classification of SQL injection attack. They also proposed a model which is like a pattern lock and it works in three parts where the first one checks the ASCII characters, then the tokens are being checked and finally the threshold values are examined. In this way, the model ensures that only the correct queries are handed to the database server. This paper has mentioned some types of attacks under different categories for example client-side attacks, attacks based on disclosure of information, attacks made with the help of command execution and authentication-based attacks. According to the paper, SQL injection attacks are grouped into six kinds i.e., tautologies, union queries, stored procedure, logically incorrect queries, inference, piggy-backed queries, etc. In the proposed model, the user interface would take input from the user. After that it would compare query length and pattern values of all the inputs. If both of the values are same then calculation of the anomaly value will be done. The given query will be rejected by the model if the score for anomaly exceeds the threshold set beforehand. If the threshold was not exceeded then the query will not be rejected. Also, some ways of preventing SQLI are studied like SQL Cheat Sheet which is available in OWASP, avoiding detailed error message, using queries that are parameterized, outlining of automatic and dynamic access control list, usage of function that would block quote, giving only the necessary permissions to users etc. In the end the authors acknowledged that the security of their proposed system has some loop holes and their application was not running fast enough. They mentioned that the model could be improved by adding cross site script prevention mechanism to make the model faster and more secure.

In paper [10], the authors have proposed a short-term memory-based detection system for SQL injection attack. The proposed method in the paper is capable of learning the effective way of representing the data automatically. It is also claimed to hold a significant advantage while facing massive and complex high dimensional data. Furthermore, the authors have proposed an injection sample generation model which is capable of generating valid SQL injection attack samples. The authors mentioned that recent proposed methods such as static analysis, dynamic analysis, instruction randomization etc. can detect only a small set of SQL injection attacks. The system proposed by the authors contains two main phases, i.e. the off-line training phase and on-line testing phase. In the proposed system LSTM is used to control three gates which in turn control the contents in the unit state. After that, in order to determine the percentage of the state of the cell in the earlier moment is maintained in the current state. The authors collected 6052 SQL injection samples for training and testing. Finally, the authors have shared the results of using various feature vector transformation methods. Word2vec method yielded an accuracy of 93.47% while BoW method yielded an accuracy of 91.93%.

In paper [11], the authors have claimed that with the inception of B/S mode application development, the developers are nowadays more inclined towards using it but due to lack of proper experience the authenticity of user input is not being verified properly, leading to SQL injection attacks. The authors have pointed out the basis of SQL injection, types of SQL injection attacks and discussed about ways to prevent SQL injection attacks in this paper. It has been mentioned in the paper that SQL injection lets the attacker evade the authentication mechanism entirely and manipulate the database located on the remote server. As database commands are allowed to be mixed with user data in the SQL syntax, the system may interpret some user data as database commands. According to the authors there are two major processes of SQL injection attacks i.e. inserting code straight into the user input variables which is concatenated with SQL command, or the second method being to store harmful code in tables in the form of strings and later connecting it to dynamic SQL commands. The paper also mentions several types of SQL Injection attacks such as not filtering the escape character, incorrect type handling, blind SQL injection, conditional response, time delay etc. Finally, the authors have shared some ways to prevent SQL injection attacks, e.g. filtering out the keywords in SQL statements, naming database tables and fields in a way that is hard to guess, setting register globals to off in PHP configuration file, not showing errors in browser, not omitting small quotation marks and single quotation marks, encapsulating common methods in order to prevent direct leaking of SQL statements etc.

On paper [2] Professor Sonewar and professor Thosar, aims to detect the SQL attacks on three-tier Web Applications. Two web applications were implemented by them, SWA with the option of uploading and downloading files and DWA in which a web application for blogs is created where users can upload an article, comment on an article, create an account, and browse articles. The authors stated that the request rate and reply rate satisfied their expectations. They further mentioned that this approach causes a problem of overhead in the webserver. But unlike the system that uses lightweight virtualization to contain each user, their system decreases the overhead. The authors with the help of httperf were able to check their system for up to 250 requests per second to the webserver. They mentioned that Vanilla had better performance than their system as container overhead is absent.

On paper [1], Kamtuo, Soomlek from the department of computer science at Khon Kaen University is working on preventing SQL injection server-side using machine learning. The authors stated that the most commonly found web application susceptibility in the National Security Agency from the USA is SQL injection. The framework is designed for SQL commands datasets extraction to mark as input attributes. The input attribute will be sent to the machine learning models and SQL injection will be predicted. The framework is created to complement the ability of validating SQL syntax, detection and prediction of SQL injection in development of web application. The methods used for detection and prediction of SQL injection, e.g., machine learning models, research techniques, and plans are mentioned in the paper. The authors stated that, a variable that receives value from user input is a dependent variable and so there is a vulnerability to SQL injection when someone inserts hostile SQL commands mixing with the basic tasks as user input. The au-

thors were successful at preventing such vulnerabilities by training the model with 1100 examples and finally figured out which commands to avoid.

2.2 Algorithms

The Algorithms that were used to design some models to test their performances are shortly described below.

2.2.1 Random Forest

Random forest algorithm uses supervised learning which can be applied in both classification and regression-based models. It is one of the more popular algorithms due to its flexibility. The algorithm basically creates a number of decision-trees and combines them to predict more accurately. The more trees there are the higher will be the accuracy and the probability of overfitting will also go down.

The algorithm first selects random data from a given dataset. After that, for every sample of data it will create one decision tree. The algorithm gets a result from each of the decision trees. Then, every result is subjected to voting and the result that wins is selected as the final result.

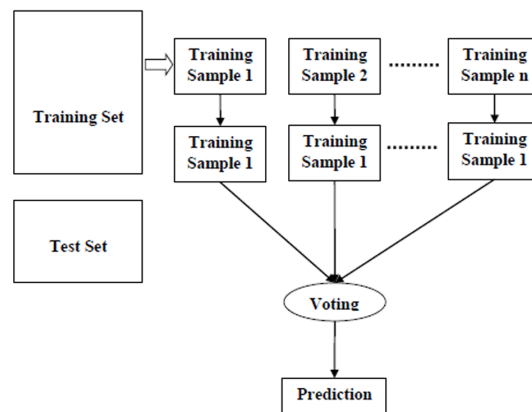


Figure 2.1: Random Forest Algorithm

2.2.2 KNN

KNN or K-nearest neighbors algorithm is also a kind of supervised Machine Learning algorithm and is used for both classification and regression problems. KNN algorithm does not have a separate training phase. The algorithm works in a kind

of a straight forward assumption that the similar samples exist close to each other.

Firstly, the training and testing data is loaded. ‘K’ is initialized as the chosen number of neighbors. Then, for each point of training data the following steps are followed. The distance between the K neighbors are calculated. The K nearest neighbors are taken based on the calculated distance. After that, the number of K neighbors are counted from every category. Finally, new data points are assigned to the category with the maximum number of neighbors. [5]

Although the algorithm is easy to implement, the algorithm gets progressively slower as the amount of data increases.

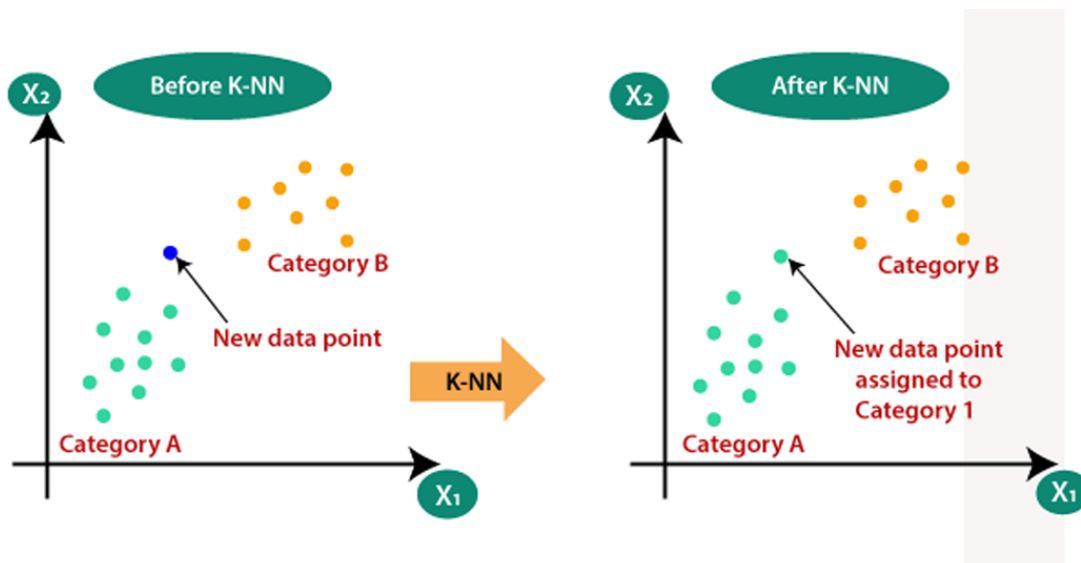


Figure 2.2: KNN Algorithm

2.2.3 Naïve Bayes Algorithm

Naïve Bayes algorithm is based on Bayes’ Theorem. Its unique approach is to assume that something in a class has no relation to other features in the class. Basically, every feature that is to be classified is independent of each other. After that, every feature is considered to have the same importance.

Bayes’ Theorem finds out the probability of an event A occurring given that event B has already occurred.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

2.2.4 Logistic Regression

Logistic regression is a kind of classification algorithm that is used to classify a discrete set of classes. A Logistic function is used in this case to model a binary

dependent variable. There are several types of logistic regression. They are, Binary Logistic Regression, Multinomial Logistic Regression and Ordinal Logistic Regression. We have used the Binary Logistic Regression for our experiments [4].

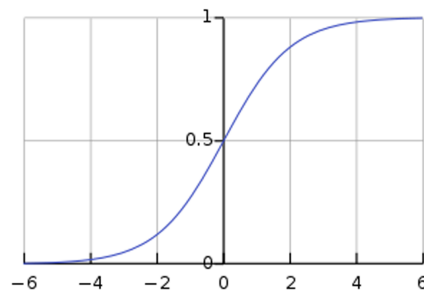


Figure 2.3: A logistic function

2.2.5 Neural Network Classifier

Neural Networks is an artificial structure of electronic network of neurons crudely mimicking the neural structure of animal brain. The records are processed one at a time by the neurons. The neurons in a neural network are arranged into three layers, input layer, hidden layer and output layer. The network works in a feed-forward process i.e. every neuron receives an input, applies a function to the input and then feeds the modified input forward to the neurons in the next layer. So, the output of a neuron works as the input of a neuron from next layer. This process continues until the final output is generated. Each function consists of a weight which is multiplied with the input to produce an output. The difficulty of constructing a proper neural network lies in finding the perfect weight and an additional bias value

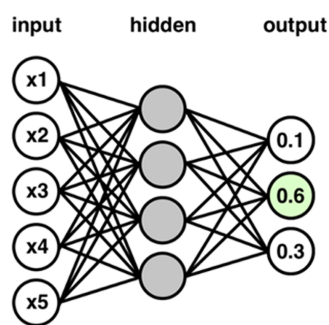


Figure 2.4: A simple Neural Network

Chapter 3

Proposed model

3.1 Dataset description

The sample dataset that we have used regarding SQL Injection Attack was taken from Kaggle’s website. Our sample dataset contains 34,048 unique values. The sample dataset contains two Columns of data. The first column represents a Sentence which needs to be detected as normal or SQL Injection Attack and second Column represents a numeric value to determine whether it is a normal statement or SQL Injection query. Here, the value 1 has been used to represent the sentence as a SQL Injection query and 0 has been used for representing the sentence as a normal statement. There were 22,305 positive samples and 11,781 negative samples in the sample dataset.

3.1.1 Data preprocessing

In our sample dataset, we found that, there were some null values. So, at first, we identified the null values in our dataset and removed them before training phase. Since our dataset contains sentence which is actually a text value, so it is needed to be converted to int value before starting the training. So, in order to do this, we used count Vectorizer class to convert the text value into its corresponding numeric value. It should be noted that, we have chosen words which have been present in at least two text documents and up to 70% of documents and converted them to tokens for training our dataset. Here, we have selected 1500 most occurring words as features for training our dataset.

However, the dataset we selected contained SQL queries which were marked as 1 or true and regular texts which were marked as 0 or false. Initially the datasets did not contain any regular texts that contained keywords from SQL queries. For example, “INSERT SELECT OR” is not an SQL query as the syntax does not follow the rules of SQL although multiple common keywords that are used in SQL queries are present in the string. So, it is just a regular text. But, in our dataset such kind of regular texts was absent. As a result, if our model was trained using this unaltered dataset, it would identify regular texts that contained SQL query keywords as a potential SQL injection attack.

Our model was meant to be incorporated to the input fields of a website, where inputs like “SELECT1234INSERT” may be valid in input fields like a password field. Even if it was not valid for the field alarming the website owner over such a harmless input would be nonsensical. Harmless inputs like these which contained common keywords from SQL would be blocked as a potential threat and the owner of the website would be notified with a false positive alarm.

Therefore, we added many samples of such kind of strings in the dataset that might look harmful for containing SQL keywords but is actually harmless as syntactically they do not form any SQL query. Strings like “for select into”, “hello insert ok”, “or and into” etc. were entered into the dataset as a ‘0’ or false which meant they were regular strings.

3.2 Model description (Workflow)

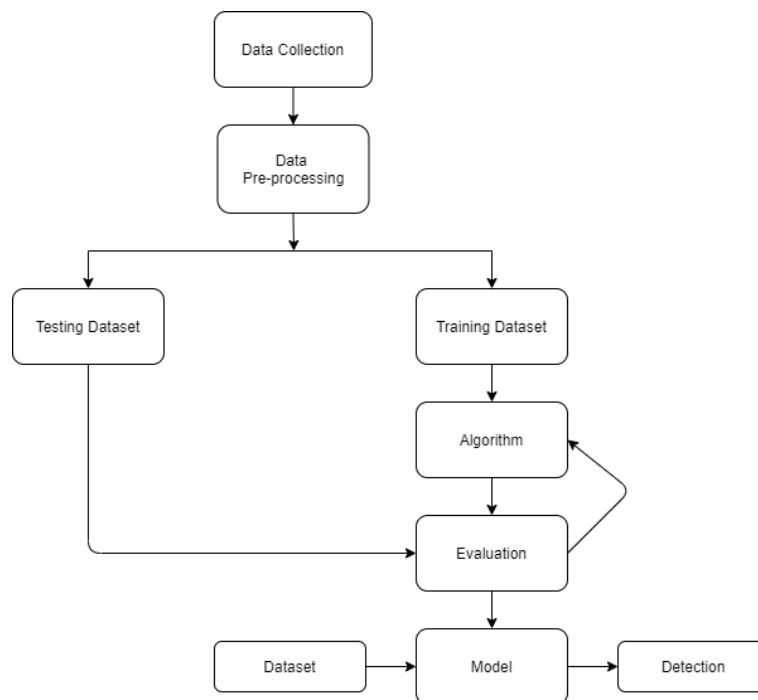


Figure 3.1: Workflow of our proposed model.

At first, after researching about SQL injection attacks and going through many relevant papers and articles we gathered sufficient knowledge regarding SQL injection. A dataset was chosen containing many SQL queries as well as some harmless strings. They were classified into two groups, 1 and 0. 1 standing for SQL injection attack string and 0 standing for regular strings. After the dataset has been collected, the dataset was processed.

After that, we implemented several machine learning algorithms and created several models. Our goal was to find out which algorithm yielded the best accuracy while identifying the SQLi queries. Then, we compared the results and identified the algorithm that had the highest accuracy. Finally, we used this algorithm to design our final model which is able to detect SQLi attacks and provide relevant warning. The workflow of our proposed model is as follows:

The dataset was taken from [13]. As the gathered data contained few irrelevant data, data pre-processing was required. Following the 80/20 rule for machine learning we spent most of the time of our study in this step. Data pre-processing means cleaning raw data into clean data, i.e. keeping the data that is most relevant for training. Basically, we removed noisy data and removed data duplication. We removed the null values found in the dataset. After that, we trained and tested our model using the dataset simultaneously. After that, we evaluated the results again and again and continued to improve our model. We finally developed a model that performed properly with our chosen dataset.

Our dataset was categorized into three categories: training data for training our algorithm, testing data for checking if the algorithm is providing the expected results and validation set was used to predict the ability of the model to function on unseen data.

A website was developed in order to test the competency of the model in practical situations. Finally, the model was connected to the website and tested.

Chapter 4

Experimentation and Result Analysis

4.1 Experimentation

For this paper, we have implemented 5 different algorithms to train our sample dataset to observe their accuracy. The algorithms that we have used are Logistic Regression, Neural Network Classifier, Random Forest, KNN and Naïve Bayes. Here we have used 80% of sample dataset for training and the rest 20% of data for testing.

In Logistic regression, we first imported necessary libraries and stored all the tokens generated in preprocessing into an array. The 'sentence' column was used as the independent variable and the 'label' column was used as the dependent variable. The data was divided into training set and testing set. After that, we trained our model. Here the value of random state was put as 0. After training, we put our model to test by feeding the testing dataset. The accuracy of this model was found to be approximately 95.7%. The classification report is given below.

	precision	recall	f1-score	support
0.0	0.94	1.00	0.97	4520
1.0	0.99	0.88	0.93	2288
accuracy			0.96	6808
macro avg	0.97	0.94	0.95	6808
weighted avg	0.96	0.96	0.96	6808

Figure 4.1: Result of Logistic Regression

In Neural Network classifier, we used sigmoid function in order to evaluate the model. We added around 1024 dense layers. The value for the dropout layer was put as 0.5. It was done in order to prevent over-fitting of our model. Here, the output of the sigmoid function if anything was above 0.5 it was considered as a SQL Injection Query, and anything below 0.5 was considered as a normal statement. Batch size for training phase was 15. In addition to that, we used 10 epochs for training. After testing phase, the accuracy score was found to be approximately 95.8%. Below is the classification report for Neural Network classifier.

	precision	recall	f1-score	support
0.0	0.94	1.00	0.97	4520
1.0	0.99	0.88	0.93	2288
accuracy			0.96	6808
macro avg	0.97	0.94	0.95	6808
weighted avg	0.96	0.96	0.96	6808

Figure 4.2: Result of Neural Network Classifier

In Random Forest classification, we used the value for random state as 0. In addition to that, we used number of estimator's value to be 1000. It is the number which determines the number of trees to be built before taking maximum voting or averages of prediction. We have used a higher value here, because a higher value will make our model stable and more accurate. The accuracy of this model was found to be 95.7%. Below we have given the complete classification report for this model.

	precision	recall	f1-score	support
0.0	0.94	0.99	0.97	4520
1.0	0.99	0.88	0.93	2288
accuracy			0.96	6808
macro avg	0.97	0.94	0.95	6808
weighted avg	0.96	0.96	0.96	6808

Figure 4.3: Result of Random Forest Classifier

In KNN, the value for nearest neighbor used for training our dataset was 1, since we have only two values for the 'label' column (1 or 0). After that, we trained our model using the training set. We then tested our model by feeding the test dataset. Following this, the accuracy of the model was found to be 95.5%. We have also generated the classification report for this algorithm.

	precision	recall	f1-score	support
0.0	0.94	0.99	0.97	4520
1.0	0.98	0.88	0.93	2288
accuracy			0.96	6808
macro avg	0.96	0.94	0.95	6808
weighted avg	0.96	0.96	0.96	6808

Figure 4.4: Result of KNN

In Naïve Bayes algorithm, we first imported Multinomial NB, Pipeline and Count Vectorizer. Here, we put the value of random state as 42. We generated our model using Pipeline and trained our training set data with the model. After that, we have fed test sample dataset in order to evaluate the algorithm. It was found that, this algorithm provided with 97.8% accuracy. The classification report for this algorithm has been given below.

	precision	recall	f1-score	support
0.0	0.99	0.98	0.98	6750
1.0	0.96	0.98	0.97	3462
accuracy			0.98	10212
macro avg	0.97	0.98	0.98	10212
weighted avg	0.98	0.98	0.98	10212

Figure 4.5: Result of Naïve Bayes Algorithm

4.2 Result Analysis

Here we have compared the accuracy of all the algorithms implemented

Algorithm	Logistic Regression	Neural Network Classifier	Random Forest Classification	KNN	Naïve Bayes Algorithm
Accuracy	0.957	0.958	0.957	0.955	0.978
Percentage	95.7%	95.8%	95.7%	95.5%	97.8%

Table 4.1: Comparison between different implemented algorithms.

In this table, we have compared accuracies of all the algorithms which we have implemented in this paper. This table shows that, of all the algorithms, Naïve Bayes algorithm has an astounding accuracy of 97.8% whereas Logistic Regression, Neural Network Classifier and Random Forest have an accuracy of almost 96%. Finally, KNN has the lowest accuracy among all algorithms, which is 95.5%.

We also developed a website in order to use an algorithm for giving protection and safety against SQL Injection Attack. The name of our website is BharaChai Ltd. It is a website where people can see different places which are available for rent. People can create account, login and search for places to rent. The website also allows for search by specific locations to see various places up for renting. People can also put up their own places for renting. We did not use any kind of framework for building this website. We used general CSS, PHP and HTML for building the website. As a result, the website had no additional security to

SQL Injection Attacks. For this, we tried to use SQL statements to get information from the database of our website. Here, we found a flaw in the search option of our website. It was found that whenever we wrote “name%’) Union select 1,”asdm@gmail.com”,email,pass,”dn”,3,2,”nd.com” from user where (’a’ like’%a” in the search option and hit enter, all the databases login id and password were revealed to us. So the website was vulnerable to SQL injection attacks.

Since Naïve Bayes Algorithm had the most accuracy, we used the model that used Naïve Bayes and wrote a python script to add it to our website. We designed it in such a way that, whenever anyone tries to put SQL query in the input fields within the website, it immediately detects it and notifies the admin through email. We used a python module called pickle in order to integrate the model to our website. Afterwards, when testing the system after the incorporation it was found that the model was able to identify every SQL query that was inserted into the input fields as a threat i.e., false negatives were mostly absent. However, the accuracy of the model dropped drastically due to false positive alarms. The model kept identifying non-SQL strings as a threat when one or more SQL keywords were present in the string. We were able to reduce the rate of false positive alarms by adding regular strings that contained SQL keywords to the training set of the model. The model was then able to identify harmless SQL keyword containing strings as regular input much more accurately. But, the rate of false positive alarms was still much more than that of false negatives.

Chapter 5

Conclusion

At present we have become so much dependent on the internet that it will not be a mistake to call it an essential element of our lives just as food, clothing and shelter. In fact, we are also dependent on internet for acquiring and managing all the other essentials in our life. We rely on websites and web applications for banking, shopping, businesses, jobs and so on. The smooth operation of these media is absolutely necessary for the smooth operation of not just our personal lives but at organization level, state level, national level and in some cases even globally.

Furthermore, all the websites and web applications that we are affiliated with both directly and indirectly, contains so many sensitive information about us that it may almost seem impossible. Should this information fall in the hands of the wrong individual or group, our organizations or even our personal lives could be destroyed. But it is an alarming truth that we are getting more and more vulnerable as technology develops. We are getting exposed to increasing number of kinds of attacks. SQL injection attack is a very special kind, since it may serve as the medium for several other kind of severe attacks.

SQL injection attack is currently one of the most popular method of attacking the database of a system. Many kinds of attacks come and go with time, but SQL injection is a technique that lingers. It continues to haunt the securities of our websites from multiple fronts. It has so many faces that it is hard to keep up with it using traditional defenses. It can only be dealt with consistently using an evolving defense system. Our defenses need to evolve along with the evolution of the threats. The model we proposed in this paper uses such a machine learning algorithm so that it can keep learning about new kinds of SQLi attack attempts and identify them in the future. It will be able to protect any website from SQL injection attacks as demonstrated in the website we used to test it. We have plans to improve our current model by adding more datasets which contains SQL statements that will not compromise the security of the website. Adding these types of dataset will help us to train our model more effectively and decrease the false positive rate of our model. This will help to improve the efficiency of our model. We hope that the model we developed will work as an added layer of defense against such attempts and keep our databases more protected.

Bibliography

- [1] K. Kamtuo and C. Soomlek, “Machine learning for sql injection prevention on server-side scripting,” in *2016 International Computer Science and Engineering Conference (ICSEC)*, IEEE, 2016, pp. 1–6.
- [2] P. A. Sonewar and S. D. Thosar, “Detection of sql injection and xss attacks in three tier web applications,” in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, IEEE, 2016, pp. 1–4.
- [3] Stasista, “Average cyber losses to global companies in the last fiscal year as of may 2019, by company size.” [online],” *Wired*, 2016.
- [4] K. Zetter, “That insane, \$81m bangladesh bank heist? here’s what we know,” *Wired*, 2016.
- [5] S. Jaiswal, “K-nearest neighbor(knn) algorithm for machine learning,” in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, java point, 2018, javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.
- [6] P. N. Joshi, N. Ravishankar, M. Raju, and N. C. Ravi, “Encountering sql injection in web applications,” in *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, 2018, pp. 257–261.
- [7] M. Hasan, Z. Balbahaith, and M. Tarique, “Detection of sql injection attacks: A machine learning approach,” in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, IEEE, 2019, pp. 1–6.
- [8] Q. Li, W. Li, J. Wang, and M. Cheng, “A sql injection detection method based on adaptive deep forest,” *IEEE Access*, vol. 7, pp. 145 385–145 394, 2019.
- [9] —, “Sql injections: Used in 51% of cases by hackers,” *IEEE Access*, vol. 7, pp. 145 385–145 394, 2019.
- [10] Q. Li, F. Wang, J. Wang, and W. Li, “Lstm-based sql injection detection method for intelligent transportation system,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4182–4191, 2019.
- [11] L. Ma, D. Zhao, Y. Gao, and C. Zhao, “Research on sql injection attack and prevention technology based on web,” in *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, IEEE, 2019, pp. 176–179.
- [12] S. Nagasundari, P. B. Honnavali, *et al.*, “Sql injection attack detection using resnet,” in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2019, pp. 1–7.

- [13] S. S. H. Shah, "Sql injection dataset.," Kaggle.com, 2019, kaggle.com/syedsaqlainhussain/sql-injection-dataset?select=sqli2.csv.
- [14] K. Zhang, "A machine learning based approach to identify sql injection vulnerabilities," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2019, pp. 1286–1288.
- [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 and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, IEEE, 2020, pp. 145–150.