ISSN 2278-3091
Volume 11, No.5, September - October 2022
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse051152022.pdf
https://doi.org/10.30534/ijatcse/2022/051152022

A Deep Learning Approach for The Detection of Structured Query Language Injection Vulnerability

Ogini, P.B[1], Dr. E.O Taylor[2], Dr. N.D Nwiabu[3]
[1]Rivers State University, Nigeria, Princewill.ogini1@ust.edu.ng
[21]Rivers State University, Nigeria, Taylor.onate@ust.edu.ng
[31]Rivers State University, Nigeria, Nwiabu.nuka@ust.edu.ng

Received Date: August 17, 2022 Accepted Date: September 23, 2022 Published Date: October 06, 2022

ABSTRACT

With the rapid development of Web 2.0 technology, network applications have gradually become an indispensable part of our lives. At the same time, Web applications are confronted with more challenges. As announced by the OWASP (open web application security project) organization, injection attack has been the first of the top 10 security vulnerabilities in 2013 and 2017, and SQL injection attack is one of the most important types among the injection attacks. Due to the rapid growth of SQL injection attacks on web application, this research developed a deep learning model in detecting SQL injection attack. The model was trained on a dataset that contains about 30,635 queries, which includes both injected and non-injected queries. The dataset was gotten from Kaggle database. The dataset was then processed by removing null and duplicate values. Further pre-processing was carried out in terms of tokenization and conversion of text to arrays. CountVectorizer () function was used for data normalization in converting the dataset to arrays in form of 0s and 1s. After the pre-processing stage, Feature selection was done on the dataset using the tfidvectoriser. The selected features were passed to the deep feed forward neural network for training. The model was trained on a step of 20 epochs, the model achieved an accuracy of 97.65%. Confusion matrix depicts the total number of correct prediction and the total number of false classifications. The confusion matrix shows that out of 590 classifications on attacks that are of normal, the model predicted correctly for 572 and predicted falsely for 16 times. Then for attacks that are of SQL injection, the model predicted correctly 251 times and predicted falsely for just 1. This shows the performance of the model is in good shape. The model was saved and deployed to web using python flask for easy testing and usage. The model was compared with other existing models and it outperformed the existing model in terms of accuracy. This research can further be extended by using combinations of deep learning algorithms. It can further be extended by deploying the model to android applications.

Key words: Deep Learning, Feed Forward Neural Network, SQL Injection Attack, Web Applications.

1. INTRODUCTION

The fast development of Internet technology has resulted in an explosion of network information. Web applications provide us with convenience," but they also pose a significant network security risk. Qihoo 360 conducted security tests on 1.979 million websites in China at the end of 2016 and found that 46.3 percent of web applications had security vulnerabilities, with SQL injection attack (SQLIA) and cross-site scripting attack (XSS) vulnerabilities accounting for the most [1]. SQL injection attacks are one of the most common network security flaws that must not be overlooked. SQL injection was used to target Sony's Play Station Network in April 2011. There were about 77 million accounts impacted, with 12 million credit cards taken. User accounts, passwords, addresses, and credit card spending details were all hacked, causing Sony to lose up to $170 million (about twice the cost of a high-end private jet) in the process. In February 2017, the Russian hacker "Rasputin successfully infiltrated a database server by exploiting SQL injection vulnerabilities. More than 20 colleges and government agencies in the United Kingdom and the United States had a huge amount of sensitive material stolen [2].

SQL injection attacks might theoretically affect any database-driven Web application system. Because the SQL injection attack is like a user's normal access to the system, it can be carried out by submitting Web forms, query strings, or page requests, and it is more covert, whereas the current Web application firewall (WAF) based on feature matching algorithms (rule base) struggles to cover all SQL injection attack variants [3]. Users' given data is frequently related with database application security risks. The Structured Query Language (SQL) is a querying, operating and administration language for database applications.

A robust SQL injection detection architecture that can detect all SQL injection attack types and is flexible enough to update when a new type of attack arises is critical. All detection methods were serialized, wasting a lot of time and computational resources. Due to the complexity of attack representations, creating a broad model to detect all attacks is similarly difficult. On the other hand,

many characteristics have varying influence on determining the types of attacks.

2. RELATED WORKS

Peng et al. [4] used the deep neural network Long Short Term Memory (LSTM) and the Multilayer Perceptron (MLP) in training data sets to extract the features of HTTP traffic in the training sets, and the final predictive capacity of the testing sets was over 99 percent. The deep neural network employs ReLU as the hidden layer's activation function, updates the weight parameters continuously using the gradient descent algorithm, and completes the training in 50 epoch rounds. The proposed strategy yielded a 95.5 percent accuracy result.

Zainab et al. [5] published a survey report on traditional and current varieties of Structured Query Language Injection Attack (SQLIA), their functioning mechanisms, and detection and mitigation approaches for traditional and modern types of SQLIA. They assess the detection and prevention strategies in terms of their capacity to detect, prevent, or partially stop the attack for evaluation. In terms of the findings, the efficacy of some of their strategies has to be improved in order to defeat the SQLIA.

Deep learning is proposed by Huafeng et al. [6] to detect SQL injection attacks in network traffic. They chose the target features based on the SQL injection attack's characteristics and used requests from urls or post packets as training data. They also used a deep belief network (DBN) model to train the selected features and sample data, resulting in a model that could be used to identify an SQL Injection attack.

Kelvin [7] demonstrated the use of input validation and sanitization features collected from source code files to train and evaluate classifier models using classical and deep learning-based machine learning methods. The algorithms they used are Convolutional Neural Network, Multi-Layer Perception and Random Forest Classifier. Their experimental result shows that Convolutional Neural Network has the highest accuracy result of about 95.3%, followed by Random Forest Classifier 93.3% and Multi-Layer Perception of about 92.3%.

Musaab [8] offers a machine learning-based heuristic approach. They trained and tested 23 different machine learning classifiers using a dataset of 616 SQL statements. They chose the best five classifiers based on detection accuracy and constructed a Graphical User Interface (GUI) program around them. The proposed algorithm (Ensemble Based Tree) was put to the test, and the results revealed that it could detect SQL injection attacks with a high degree of 93.8 percent accuracy.

To detect SQL injection attacks, Kranthikumar and Leela [9] presented a pattern-based classification named REGEX. They compared the pattern-based classifier's results to those of SVM (Support Vector Machine), Gradient Boosting Algorithm, and Naive Bayes classifiers, which are all machine learning classifiers. The comparison was done on a synthesized dataset of 20474 SQL queries, and it indicated that the REGEX classifier had 97 percent accuracy and took 3.98 seconds to compute, which was faster than the other machine learning algorithms.

For identifying SQLi in online applications, Maruf et al. [10] presents a deep learning-based approach. To rank the characteristics from the dataset, the solution uses both correlation and chi-squared approaches. Not only in feature selection, but also in the detection process, they used a feed-forward network strategy. Above 1850 recorded datasets, their proposed method had an accuracy of 98.04 percent, demonstrating its higher efficiency over other existing machine learning solutions.

Ding et al. [11] proposed employing a natural language processing model and a deep learning framework to identify SQL injection. The strategy can enhance accuracy and reduce false alarms by allowing the machine to learn the language model features of SQL injection attacks automatically, eliminating human interaction and offering some protection against zero-day attacks that never happen. Their proposed strategy yielded a precision of around 98 percent.

Kevin [12] proposed a Deep Learning algorithm in detecting SQL injection attack. The Deep learning algorithm he used here was that of Convolutional Neural Network (CNN). They used a training dataset that consists of vulnerable PHP files gathered from two dissimilar sources. They selected files written in PHP, one of the most popular language to develop web applications. For feature extraction, they made use of word2vec in selecting features. The word2vec was used in creating word embedding's, which finds the relationship between words. They made use of word2vec based models because word2vec have shown much success in text classification in previous works, and it may be effective in analyzing which functions or commands in codes most contributed to SQL injection vulnerabilities. Finally, the extracted text was used in training the model using Convolutional Neural Network (CNN) model. The CNN model was trained to detect SQL injection attack. The CNN achieved a training accuracy of about 95.5%.

3. DESIGN METHODOLOGY

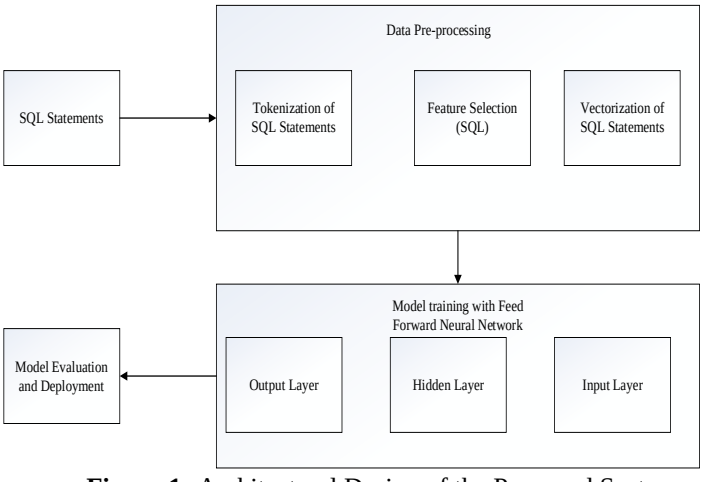

Figure 1: Architectural Design of the Proposed System

Dataset: The dataset comprises of about 30650 structured queries. The queries comprise of both safe and unsafe (SQL injection attack) queries. The dataset is made of four columns. These columns comprise of the following

1. Query: The query comprises of various structured query statements for both normal and anomalous statements.
2. Length: The length comprises of the total length of each structured query statements.
3. Attack: The attack column comprises of the kind of attack that is being carried out, if it is sqli or just a none sqli
4. Label: The label column describes the structured query statement to be anomalous or normal statement.

	Query	Length	Attack	Label
0	1' where 6406=6406;select count(*) from rdb$fi...	115	sqli	anom
1	1) and 8514=(select count(*) from domain.domai...	111	sqli	anom
2	-3136%') or 3400=6002	21	sqli	anom
3	1) where 7956=7956 or sleep(5)#	31	sqli	anom
4	-7387'))) order by 1--	22	sqli	anom
5	1')) as gfzb where 7904=7904;begin dbms_lock.s...	60	sqli	anom
6	1))) union all select null,null,null#	37	sqli	anom
7	-5622" where 7970=7970 union all select 7970,7...	66	sqli	anom
8	1")) and 4386=utl_inaddr.get_host_address(chr(...	227	sqli	anom
9	-9860%" union all select 6996,6996,6996,6996,6...	55	sqli	anom
10	1));select dbms_pipe.receive_message(chr(66)||...	101	sqli	anom
11	-2604)) as sekb where 6897=6897 or 1000=7683	44	sqli	anom
12	1%"))) waitfor delay '0:0:5'--	30	sqli	anom
13	1") as nraw where 7984=7984 and elt(3114=3114,...	56	sqli	anom
14	1');begin dbms_lock.sleep(5); end and ('jzlr'=...	51	sqli	anom

Figure 2: Dataset sample

Pre-Processing: The data pre-processing has to do with transforming the dataset into a well suitable standard that is appropriate for training the Feed Forward Classifier model. The processing consists of the cleaning of the data and tokenization. The cleaning of the data has to do with the removal of noise. By noise we mean removing parenthesis, capital letters, non-alpha numeric characters. The data cleaning also comprises of the removal or filling of Nan values. The pre-processing also has to do with tokenization, breaking the words into tokens. The processing also has to do with the conversion of the dataset to arrays in form of zeros and ones (0s and 1s).

Feature Selection: Feature selection is an important process in training a model. In other to get the best out of the model, certain features need to be selected or extracted from the dataset. Tfidvectorizer was used in selecting or extracting the most notable features that was used in training the Feed Forward Classifier model.

Model Training: "The model was trained using Feed Forward Classifier in a robust model that was used in detecting SQL Injection attacks on web applications. The Feed Forward Classifier model was trained on the SQL injection dataset. The Feed Forward Classifier is a Recurrent Neural Network algorithm. The Feed Forward Classifier model was built using Tensorflow Framework with Keras application. Keras Sequential API which means we build the network up one layer at a time. The layers are as follows:

1. An Embedding which maps each input word to a 100-dimensional vector. The embedding can use pre-trained weights (more in a second) which we supply in the weight's parameter. Trainable can be set False if we do not want to update the embeddings.
2. A Masking layer to mask any words that do not have a pre-trained embedding which is represented as all zeros. This layer should not be used when training the embeddings.
3. The heart of the network: a layer of Feed Forward Classifier cells with dropout to prevent overfitting. Since we are only using one Feed Forward Classifier, it does not return the sequences, for using two or more layers, make sure to return sequences.
4. A fully-connected Dense layer with relu activation. This adds additional representational capacity to the network.
5. A Dropout layer to prevent overfitting to the training data.
6. A Dense fully-connected output layer. This produces a probability for every word in the vocab using softmax activation.

The architecture of the Feed Forward algorithm can be seen in figure 3.

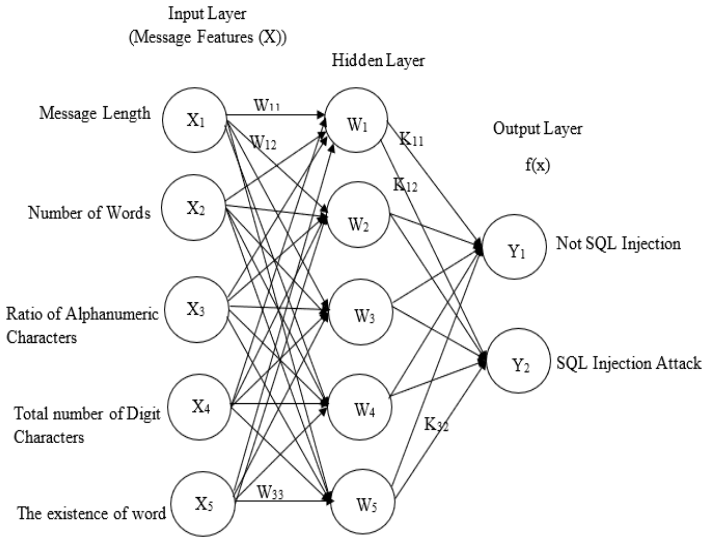

Figure 3: Architecture of Feed Forward Neural Algorithm

(X_1, X_2 and X_3, X_4, X_5) denotes the input features. The hidden layer neurons transform their input from the first layer with a linear function $W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5$ to the next layer.

Calculation of output for the first (hidden layer) layer consists of multiplying the input vector by the first weight matrix,wij. Output of hidden layer is given by:

$$y_i = \sum_{i,j=1}^{n} x_i * w_{i,j} \quad (3.1)$$

The output of output layer is given by:

$$z_i = \sum_{i,j=1}^{n} y_i * k_{i,j} \quad (3.2)$$

Where,

$x_i = (x_1, x_2, x_3)$

$y_i = (y_1, y_2, y_3)$

$z_i = (z_1, z_2, z_3)$

w and *k* are synaptic weight matrices of hidden layer and output layer.

Activation function F is applied to it there by producing the final output signal 'OUT;'

$$OUT = f(x) = \frac{1}{1 + e^{-x}} \quad (3.3)$$

Where,

F is sigmoid function".

4. EVALUATION AND TESTING

Plotting a classification report on the trained Feed Forward Classifier model was used to assess its performance. The Classification report is used to assess the quality of Feed Forward neural model predictions by determining how many are True and how many are False. True Positives, False Positives, True Negatives, and False Negatives are all terms used to describe the results of a forecast. The Feed Forward neural model SQLI detection and classification report is shown in the classification.

Table 1 Evaluation Matrix

Definition	Formula
Accuracy	$\frac{TP+TN}{TP+FN+TN+FP}$
Error	1-accuracy
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
F-measure	$\frac{2\ Precison \times Recall}{Precision+Recall}$

5. RESULTS AND DISCUSSIONS

From the experiment conducted, figure 2 shows the first fifteen rows of the structured query language (SQL) dataset. From figure 2, the dataset is made up of four columns namely, Query, Length, Attack, Label. The Query column contains various queries written in structured query language (SQL) format; the total number of queries written is 30635. The length column contains the length of the SQL statements. By length, we mean how long or how many characters is contained in the query, including space. The Attack column signifies what type of attack is being carried out (being SQL injection attack or a normal query). The label column contains anomalous and normal. The anomalous signifies that it is destructive while the normal describes that it is normal SQL query. Figure 4 is a heat map function in python which is being used to check for missing values. The white lines in figure 4 shows that some rows in the label column are missing. In other to have a well trainable model, the data needs to be cleaned. That is to say that null or missing values, needs to be removed. Figure 5 shows that the missing values in the dataset has been removed completely. After this process, feature extraction was applied on the dataset to select the most important feature. Figure 4 shows that after feature extraction, the most notable features that are suitable for training the deep learning model are the query column and the label column. Before passing the data to the deep learning model, the query column needs to pass through tokenization process. This is to say that the query column needs to be tokenized and converted to arrays. Figure 7 shows the tokenized and converted data. Figure 9 shows the accuracy obtained for both training and validation test. The training and validation accuracy are used in testing the performance of the model during training and on a test dataset. The model achieved a training result of about 98% and a test result of about 98%. Figure 10 shows the losses of the model for both training and testing data. The model had a loss value below o.1 for both training and testing. Figure 10 shows the classification report of the model. The classification report is a summation of accuracy, precision, recall and f- measure. Precision has to do with the correct classification of the model in terms of false positive, false negative, true positive and true negative. The precision score of the model is about 100% correct classification for queries that are normal and 94% correct classification for queries that are of SQL injection attack. The support shows the total number of classifications that was carried out by the model. Figure 11 shows the confusion matrix of the proposed system. Confusion matrix depicts the total number of correct prediction and the total number of false classifications. The confusion matrix shows that out of 590 classifications on attacks that are of normal, the model predicted correctly for 572 and predicted falsely for 16 times. Then for attacks that are of SQL injection, the model correctly predicted 251 times and predicted falsely for just 1. This shows the performance of the model is in decent shape.

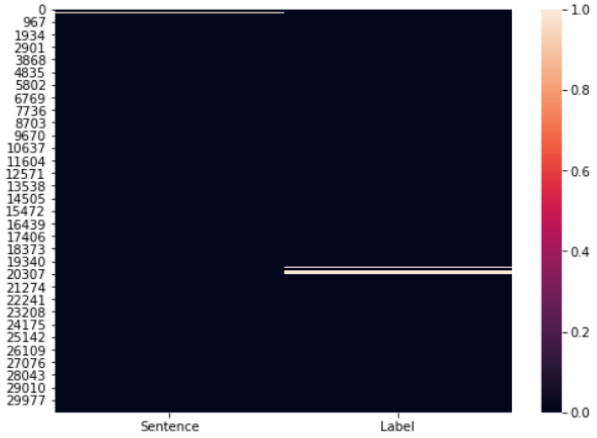

Figure 4: Result of dataset Heat map

The thick white line shows that there are some missing values in row 19307 and 20307.

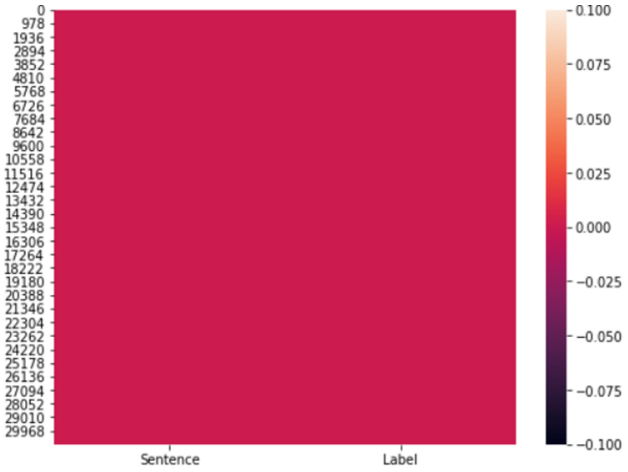

Figure 5: Shows that the missing values has been removed from the dataset

	Sentence	Label
0	" or pg_sleep (__TIME__) --	1
2	AND 1 = utl_inaddr.get_host_address (...	1
3	select * from users where id = '1' or @ @1 ...	1
4	select * from users where id = 1 or 1#" (...	1
5	select name from syscolumns where id = ...	1
6	select * from users where id = 1 +$+ or 1 =...	1
7	1; (load_file (char (47,101,116,99,47...	1
8	select * from users where id = '1' or ||/1 ...	1
9	select * from users where id = '1' or \.<\ ...	1
10	? or 1 = 1 --	1
11	) or ('a' = 'a	1
12	admin' or 1 = 1#	1
13	select * from users where id = 1 or " ( ]...	1
14	or 1 = 1 --	1
15	AND 1 = utl_inaddr.get_host_address (...	1

Figure 6: Training data

	0	1	2	3	4	5	6	7	8	9	...	4707	4708	4709	4710	4711	4712	4713	4714	4715	4716
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Figure 7: Tokenized and converted data

In other to have a well trainable data, the dataset in figure 4.5 need to be tokenized and converted to array. This was achieved using Count-Vectorizer (), stop-words and tokenizer ()

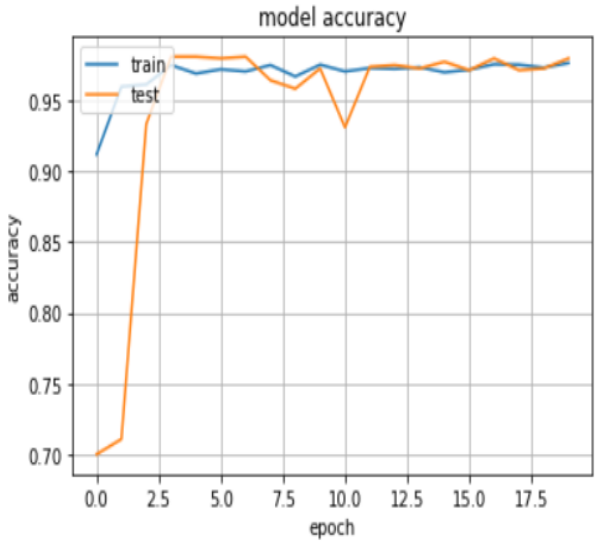

Figure 8: A graphical representation of Training Accuracy Vs Training Epochs.

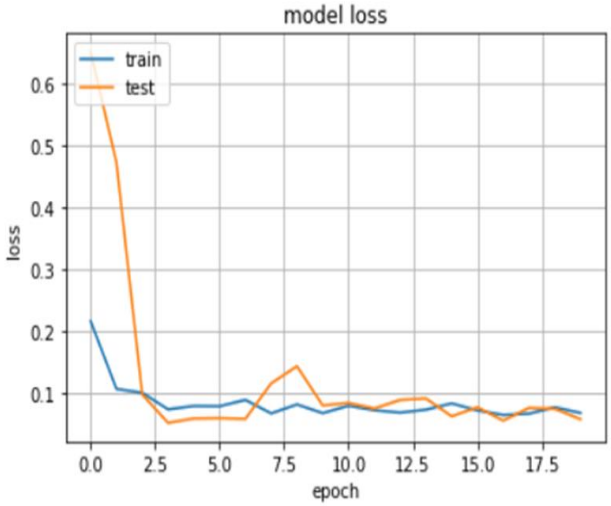

Figure 9: A graphical representation of Training Loss Values Vs Training Epochs.

```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	0.97	0.99	588
1	0.94	1.00	0.97	252
accuracy			0.98	840
macro avg	0.97	0.98	0.98	840
weighted avg	0.98	0.98	0.98	840

Figure 10: Classification report of Deep Learning

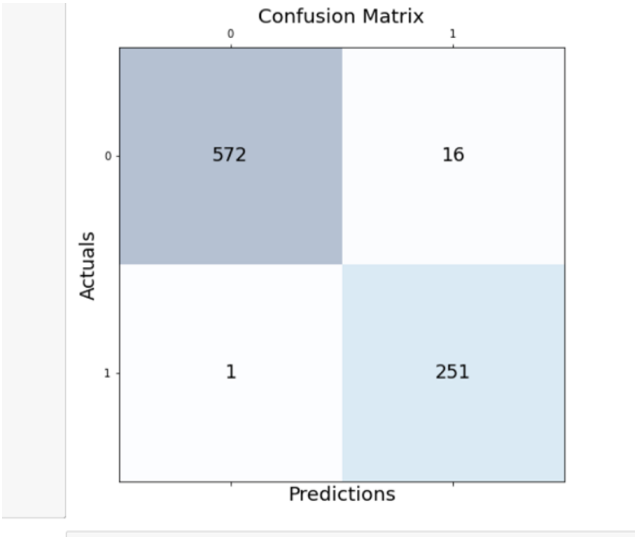

Figure 11: Confusion Matrix of the proposed Feed Forward Neural Network

The confusion matrix shows the predicted result vs the actual prediction.

Table 2: Proposed System versus Existing System

System	Model	Training Data	Accuracy
A Machine Learning based Approach to Identify SQL Injection Vulnerabilities	Convolutional Neural Network	9750	95.3%
A Model to detect SQL Injection attacks using Deep Forward Neural Network	Feed Forward Neural Network	30,635	97.65%

Table 2 shows a comparative analysis between our system and the existing system proposed by Kevin [12]. The table shows that our system outperformed the existing system in terms of accuracy. The existing system had an accuracy result of about 95.3% while our system had an accuracy result of 97.65%

6. CONCLUSION AND FUTURE WORK

Due to the rapid growth of SQL injection attacks on web application, this research developed a deep learning model in detecting SQL injection attack. This paper presents a deep learning algorithm in detecting SQL Injection Attacks on web applications with high accuracy detection rate. The system detects advanced SQL injection (Second Order Attack, and Hybrid Attack). The implementation of this system was carried out beyond analysis and testing of model's performance using test data, but a real time implementation of SQL injection attacks was carried out by creating a web application using Python flask. The system achieved an accuracy rate of 97.65%. To enhance the efficiency of the system, more SQL statements (both injected and non-injected statements) need to be considered for training and testing our model. This research can further be extended by using combinations of deep learning algorithms. It can further be extended by deploying the model to android applications. Our system is also scalable in the sense that any enhancement can be easily implemented with minor modification.

REFERENCES

[1] Uwagbole, S., Buchanan, J. and Lu, F. (2017). Applied machine learning predictive analytics to SQL injection attack detection and prevention. *Proceeding of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbob, Portugal,* 8-12, 1087-1090

[2] Ding, C., Qiseng, Y., Chunwang, W. and Jun, Z. (2021). SQL Injection Attack Detection and Prevention Techniques Using Deep Learning. *Journal of Physics: Conference Series*, 1757

[3] Subasi, A., Alzahrani, S., Aljuhani, A. and Aljedani, M. (2018). Comparison of Decision Tree Algorithms for Spam E-mail Filtering," *In Proceedings of the International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia*, 1-5.

[4] Peng, T., Weidong, Q., Zheng, H and Huijuan, L. (2018). SQL Injection Behavior Mining Based Deep Learning. *International Conference on Advanced Data Mining and Application*, 445 – 454.

[5] Zainab, S. A. and Manal, F. Y. (2017). Detection and Prevention of SQL Injection Attack: A Survey. *International Journal of Computer Science and Mobile Computing*, 6(8), 5-17.

[6] Huafeng, Z., Bo, Z. and Hui, Y. (2019). SQL Injection Detection Based on Deep Belief Network. *SQL Injection Detection Based on Deep Belief Network. Issue,* 20, 1-6.

[7] Kevin, Z. (2019). A Machine Learning based Approach to Identify SQL Injection Vulnerabilities. *934th IEEE/ACM International Conference on Automated Software Engineering (ASE),*
1286-1288.

[8] Musaab, F. (2020). Intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Secure Application*, (50), 102419,

[9] Kranthikumar, B. and Leela, V. (2020). SQL injection detection using REGEX classifier. *Journal of Xi'an University of Architecture & Technology*, 12(6), 800-809.

[10] Maruf, H., Badlishah, A. and Tonmoy, G. (2021). SQL Injection Vulnerability Detection Using Deep Learning: A Feature-based Approach. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI*) 9(3)702~718.

[11] Ding, C., Qiseng, Y., Chunwang, W. and Jun, Z. (2021). SQL Injection Attack Detection and Prevention Techniques Using Deep Learning. *Journal of Physics: Conference Series,* 1757

[12] Kevin, Z. (2019). A Machine Learning based Approach to Identify SQL Injection Vulnerabilities. *934th IEEE/ACM International Conference on Automated Software Engineering (ASE),* 1286-1288.