Enhancing SQL Injection Detection with Long Short-Term Memory Networks in Deep Learning

K. Takyi, R.M.O.M Gyening, M. Kobinnah, M. A. Boateng and S. Boadu-Acheampong

***Abstract*— The security risks posed by (Structured Queried Language) SQL injection attacks in web applications necessitate more advanced detection methods beyond conventional techniques. Deep learning methods such as Long Short-Term Memory (LSTM) networks have been employed to detect SQL injection because they can handle sequential data such as SQL queries. In SQL datasets, imbalances arise due to the infrequent presence of malicious SQL queries. In this study, we employ data augmentation techniques that mitigate this issue and enable robust model training. The augmentation involves substituting keywords with randomly selected synonyms exclusively within malicious SQL queries. This augmentation approach is implemented on a sizable dataset, resulting in 89,143 samples post-augmentation, distinguishing this research from the prevailing literature that predominantly employs smaller datasets. The outcomes underscore the model's robustness, yielding 99.4% accuracy, precision, and F1 score. Compared to LSTM-based methodologies for SQL injection (SQLi) detection, the proposed approach showcases superior accuracy and efficiency in identifying potential threats. This research significantly fortifies cybersecurity measures for online applications and databases.**

***Keywords*— Cybersecurity, Deep Learning, Long Short-Term Memory, Structured Query Language, SQL Injection Detection.**

Manuscript received October, 30, 2024.

K. Takyi is with the Computer Science Department, Kwame Nkrumah University of Science and Technology, PMB Kumasi, Ashanti Region, Ghana (corresponding author; phone number: +233245999949; e-mail: takyikate @knust.edu.gh).

R. M. O. M Gyening is with the Computer Science Department, Kwame Nkrumah University of Science and Technology, PMB Kumasi, Ashanti Region, Ghana (e-mail: rmo.mensah@knust.edu.gh).

M. Kobinnah is with the Computer Science Department, Kwame Nkrumah University of Science and Technology, PMB Kumasi, Ashanti Region, Ghana (e-mail: mkobbinah@gmail.com @gmail.com).

M. A. Boateng is with the Mathematics Department, Kwame Nkrumah University of Science and Technology, PMB Kumasi, Ashanti Region, Ghana (e-mail:boateng.ma@knust.edu.gh @knust.edu.gh).

S. Boadu-Acheampong is with the Computer Science Department, Kwame Nkrumah University of Science and Technology, PMB Kumasi, Ashanti Region, Ghana (e-mail: samuelsonacheampong@gmail.com @gmail.com).

I. INTRODUCTION

The Internet is rapidly changing, with 4.4 billion Internet users in 2019 and a growing number of online services [1]. This growth in access to confidential data, such as credit cards and social security numbers, has increased interest in hacking [2]. Cybercrime costs nearly $50 billion annually, with Structured Query Language (SQL) injections responsible for over a fifth of these attacks [3]. As new computer technology is developed, there will be no immediate end to SQL injection attacks, highlighting the need for continued security measures to protect users' personal information. SQL Injection (SQLi) is a prevalent cyberattack that can damage websites and visitors by putting incorrect SQL commands in web pages or domain names and tricking the server into running them [4]. The injection attack, as examined within this context, is well acknowledged for its significant severity owing to its capacity to undermine crucial security services, encompassing confidentiality, authentication, authorization, and integrity [5]. SQLi attacks may potentially result in many consequences, including but not limited to the unauthorized acquisition of data, infringements against privacy, financial ramifications, and detrimental impacts on the reputation of individuals or entities. Identifying and preventing SQLi attacks is crucial to minimize their possible implications. Traditional methods rely on rule-based or signature-based techniques, which have limitations in identifying new or unknown attack patterns.

Alternatively, machine learning (ML) techniques can derive insights from previous examples of attacks and detect further attacks based on learned patterns. Deep learning algorithms have emerged as potential methods for detecting SQL injection threats in recent years due to their understanding of underlying behaviors and patterns. (Long Short-Term Memory) LSTM network models integrate feedback connections, allowing deep learning to handle sequential data and capture temporal relationships between them. The study aims to create a novel method for identifying SQLi attacks utilizing LSTM networks. The study contributes to the current literature on the deep learning approach to SQLi detection through;

1. Evaluating the effectiveness of LSTM networks in detecting SQLi attacks and contrasting their ability to perform with traditional and signature rule-based techniques.
2. Creating a deep learning model that uses LSTM networks to identify SQLi attacks in web-based applications.
3. Comprehensive assessment of the efficiency of the developed model on various datasets and scenarios and weighing its performance against that of an existing deep learning method.

A. Background and Related Works

This section investigates SQLi attacks, current conceptual literature, and the assessment of several efficient detection strategies and their performance. SQL Injection is a standard method hackers use to steal information from websites, targeting application layers [6]. This attack style exploits flaws in online applications, allowing hackers to insert malicious SQL queries into dynamic areas like login forms, reviews, search pages, and online ordering. These areas are vulnerable to SQL injection attacks, as they allow SQL instructions to query the database directly. The goal is to corrupt the execution of valid SQL commands by inserting malicious SQL instructions into the data plane input.

Some SQL injection threats include illegal or logically incorrect queries, where the attacker runs wrong queries to learn which servers contain vulnerabilities; buffer overflow, where malicious actors pollute the buffers to lead to floods and then exploit [7]. Another technique is when the attackers use blind SQLi to ask a database for true/false queries and then use the application's logic to guess the correct answer [8]. Lastly, hackers use the database server vulnerabilities to lead the server to generate incorrect SQL queries where they can perform SQL injection attacks on the server's failure message to the client Attacks [9]. It involves creating a database of known attack signatures and using pattern-matching algorithms to compare incoming SQL queries. The Snort Intrusion Detection System can identify SQL injection attacks. It uses a rule-based technique to detect pre-existing attack patterns, which can be customized to include user-specified rules [10].

Reference [11] developed a (Convolutional Neural Network) CNN model using (University of New South Wales - Network Benchmark) UNSW-NB15, (Knowledge Discovery and Data Mining) KDD99, and HTTP CSIC 2010 as training and validation datasets. Using word embedding, they obtained SQL injection payloads through data cleaning and transformed them into vectors. The model includes three padding convolution layers, three max-pooling layers, one complete connectivity layer, and one hidden layer. The convolution kernels are 3x3, 4x4, and 5x5, respectively. The entire connectivity and hidden layers are designed to avoid overfitting, and the hidden layer provides the final classification results. The experimental results show a 99.50% accuracy and 100% recall. Reference [12] also analyzed an intelligent transportation system's existing SQL injection detection algorithm. The paper proposed a long short-term memory (LSTM) based SQL injection attack detection method and a way to generate SQL injection samples to augment the dataset. This method can simulate SQL injection attacks and develop valid positive samples to solve the problem of overfitting. The experimental results showed that the proposed method's accuracy, precision, and F1 score were all above 92%.

An SQL injection detection method based on Artificial Neural Networks (ANN) has been presented by [13] a large amount of SQL injection data was analyzed to extract the relevant features; neural network models (Multilayer Perceptron) MLP and LSTM) were then trained. SQL injection attacks were successfully identified using (Adaptive Boosting) Adaboost. Reference [14] adopted the AdaBoost algorithm to detect SQL injection attacks but converted data into stumps. These stumps were classified as weak or strong based on their weight output.

Using SQL Injection Dataset, [15] proposed multiple ML methods. Logistic Regression, AdaBoost, Random Forest, Naive Bayes, and (Extreme Gradient Boosting Classifier) XGBoost in the detection of SQL Injection attacks. According to the paper, the best method for detecting SQL injection is Naïve Bayes. The limitation of the study is that the dataset may not fully represent the diversity and complexity of real-world SQL injection attacks. Additionally, the study did not consider the impact of different web application frameworks or technologies, which could affect the effectiveness of the machine learning methods in detecting SQL injection attacks.

II. MATERIALS AND METHODS

The proposed method for detecting SQLi attacks is shown in Fig. 1.

A. Data Acquisition and Data Aggregation

This study focuses on SQL injection attacks using a dataset obtained from Kaggle [16]. To gather the data, SQLi [17], SQLiV2 [17], and SQLiV3[18] was used. The obtained dataset thoroughly characterized SQL injection attacks, allowing the proposed method to be more generalizable to detect evolving attack queries. SQLi, SQLiV2, and SQLiV3 datasets were combined to create an aggregated dataset. The combined datasets provided advantages such as enhanced model generalization, improved feature coverage, data variance management, increased data volume, and diversity in training data. The increased

quantity of data lowered the chance of overfitting and provided a more accurate estimate of attack patterns, whereas the variety of the training data improved the model's resilience and adaptability. The datasets used are summarized in Table I.

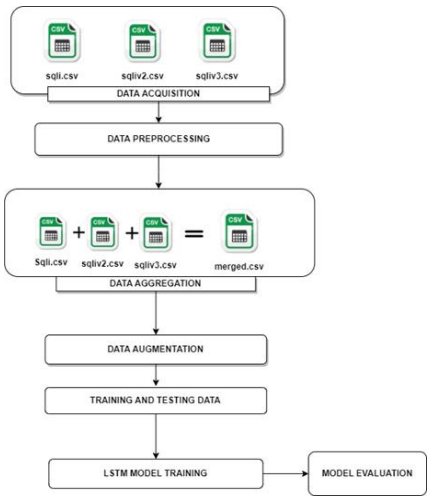

Figure 1. The Proposed Framework

B. Data Pre-processing and Tokenizer

The second stage, "data pre-processing," included essential tasks, such as cleaning the data, identifying anomalies, deleting duplicates, and rectifying errors. Data integrity was ensured through pre-processing procedures. To prepare raw SQL queries for use by deep learning models, tokenization is a vital step in the data pre-processing process. Using a tokenizer during data pre-processing enhances the accuracy and efficiency of detecting SQLi attacks since it helps find patterns and correlations in the data. Without regular expressions, queries would need to include commas and brackets.

C. Data Augmentation

To fix data imbalance problems, this study employs a random replacement data augmentation technique.

Algorithm; *augment_sql (sql_query, n)*.

***Input*:**

sql_query: A SQL query string.

n: The number of augmented queries to generate.

***Output*:**

A list of n augmented SQL query strings.

***Steps*:**

1. Initialize an empty list to store augmented queries.
2. Define a list of SQL keywords and their potential replacements.
3. Split the SQL query into words.
4. Repeat the augmentation process n times:
   a. Initialize an empty list to store the augmented words.
   b. Iterate through the words in the SQL query:
      i. If the word is an SQL keyword, replace it with a random synonym.
      ii. Otherwise, keep the word unchanged.
   c. Join the words back into a SQL query.
   d. Add the augmented query to the list.
5. Return the list of augmented queries.

Table I**.** Summary of datasets used

Datasets	Benign Samples	Malicious Samples	Total samples
SQLi	3,072	1,128	4,830
SQLiV2	22,305	11,456	33,761
SQLiV3	19,255	11,337	30,592
Total	44632	23921	68553

D. Training and Testing Data

The proposed LSTM model is trained on a dataset divided into training and testing sets, with 80% used for training, 10% for testing, and 10% for validation. The model learned to differentiate between legitimate SQL queries and those indicating SQL injection attacks. Backpropagation is applied to adjust LSTM cell weights and biases, reducing the prediction errors. The model's performance was evaluated during training using a segregated testing set containing unseen data. Dropout was employed to prevent overfitting.

E. LSTM Model Architecture

Fig. 2 displays an LSTM-base architecture which consists of three main layers: an embedding layer, an LSTM layer, and a dense layer. The embedding layer converts the input data into a 128-dimensional vector space. This layer is a systematic approach in natural language processing tasks, where words are typically represented as vectors. In this scenario, the input data is often textual, like SQL queries. The LSTM layer in a recurrent neural network uses 128 LSTM cells to store and update data from the embedding layer's output. Its goal is to capture the sequential character of input data, which is crucial for detecting SQL injection attempts. Dropout rates are 20% for LSTM and recurrent layers to prevent overfitting. This random ignore of specific neurons during training helps the model generalize better by preventing it from memorizing the training input. The LSTM-based model architecture handles textual inputs and sequential characters simultaneously, generating binary classification results to determine if the input text is a SQL injection attack. This design minimizes overfitting and increases generalization efficiency. The possibility that the input text was an SQL injection attack is represented by the returned value, which might be between 0 and 1.

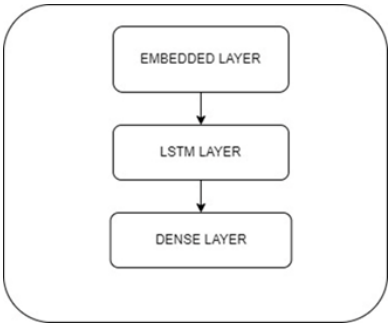

Figure 2. LSTM Architecture

F. Proposed LSTM Model Training

The LSTM model is employed in the research to detect SQL injection attacks utilizing 89,143 samples. An embedding layer, an LSTM layer with dropout and recurrent dropout, and a dense layer with sigmoidal activation made up the model's three layers. To avoid overfitting, the LSTM layer contained 128 units, a 20% dropout rate, and a 20% recurrent dropout rate. The embedding layer supplied a 128-dimensional vector representation of the input data. For SQL injection attacks, the dense layer produced binary classification output. 32 batches and 5 epochs of a binary cross-entropy loss function are used to train the LSTM network model. The Adam optimizer adjusted model weights based on backpropagation gradients. A 10% validation split is used to monitor and Evaluate the model's performance on new data. The model processed the entire training dataset across epochs and accurately detected SQL injection attacks.

G. Model Evaluation

During the evaluation phase, various metrics are employed to assess the performance of a model. These metrics include accuracy, recall, F1 score, precision, ROC curve, confusion matrix, and false positive rate. The evaluation of a neural network model's performance on test data sets is based on four distinct combinations, represented by the symbols TP (True Positive), FP (False Positive), FN (False Negative), and TN (True Negative) in the model's outputs.

The accuracy of the model may be determined using the formula [19]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (1)$$

The term recall is used to depict the sensitivity or true positive rate, which denotes the proportion of positive instances that are accurately identified as positive. Recall value is computed as follows [20]:

$$\mathrm{Re}\,call = \frac{TP}{FN + TP} \qquad (2)$$

Precision is a metric that quantifies the likelihood of an instance being correctly classified. The precision score is computed as follows [13]:

$$\mathrm{Pr}\,ecision = \frac{TP}{FP + TP} \qquad (3)$$

The F1 score is a metric utilized to evaluate the accuracy of a model, which considers both its precision and recall [21]:

$$F1Score = \frac{2 * recall * precision}{recall * precision} \qquad (4)$$

The False Positive Rate (FPR) is a significant metric in various fields of study, as it denotes genuine users who are identified as generating malevolent requests. The calculation of the FPR can be performed using the following formula. In scenarios involving high throughput applications, many authorized users may be unable to gain entry to the system [12].

$$FPR = \frac{FP + TN}{FP} \qquad (5)$$

Receiver Operating Characteristics (ROC) analysis examines an algorithm's performance under various operating conditions. The ROC graph will demonstrate the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at different categorization levels. The model was assessed at various threshold settings, and TPR versus FPR was displayed to derive ROC curve points.

The confusion matrix is a specific table layout that allows visualization of the performance of an algorithm [22], as shown in Figure 3.

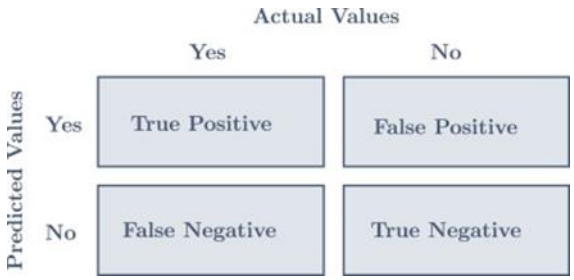

Figure 3. Confusion Matrix

III. Results and Discussion

This section presents the study's results and analysis and showcases the research outcomes on the effectiveness of LSTM networks in identifying SQL injection attacks. The experiments' results described in this section are compared to similar studies.

A. *Epoch-Varying Test Results*

The LSTM model's performance in detecting SQLi attacks was evaluated over five epochs. In the initial epoch, the model achieved a training accuracy of 98.58% with a training loss of 0.0469. The validation accuracy was 99.36%, and the validation loss was 0.0224, as displayed in Fig. 4.

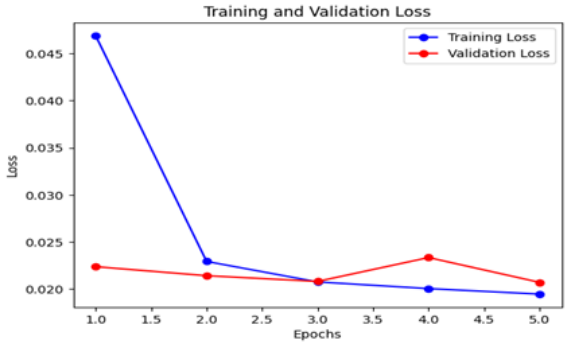

Figure 4. Training and Validation loss of the epoch

The model's performance increased during training, peaking in the fifth epoch with a training accuracy of 99.55% and a training loss of 0.0195. Validation accuracy was 99.44% then, with a validation loss of 0.0207, as shown in Fig. 5.

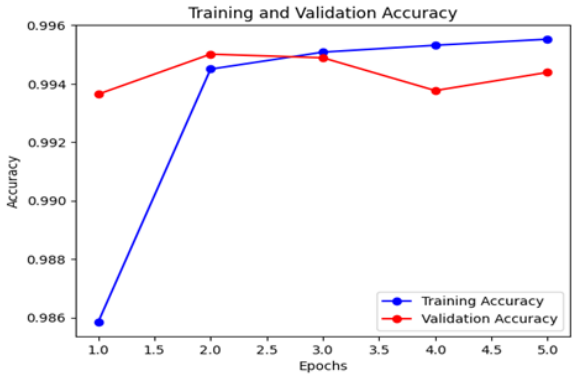

Figure 5. Training and Validation Accuracy

These results show that the LSTM model can successfully train to learn and detect SQL injection attacks

B. *Results of the model*

The LSTM model performed exceptionally well in detecting SQLi attacks, obtaining a 99.4% detection rate. It effectively reduced false positives while maintaining a 99.4% precision rating. In addition, the model correctly classified 99.4% of the data, resulting in a remarkable overall accuracy of 99.4%. This performance is comparable to the study [23], which obtained 91% for the same metrics. The f1 score of 99.4%, which incorporates precision and recall, indicates a balanced performance. High true negatives (4412), high true positives (4446), and low false positives (28 and 29) were revealed by the confusion matrix. These findings support the LSTM model's outstanding performance in identifying SQLi attacks, demonstrating its capacity to identify assaults while minimizing false positives reliably. Fig. 6 shows the confusion matrix of the model, and Fig. 7 unveils the false positive rate, true positive rate, and thresholds.

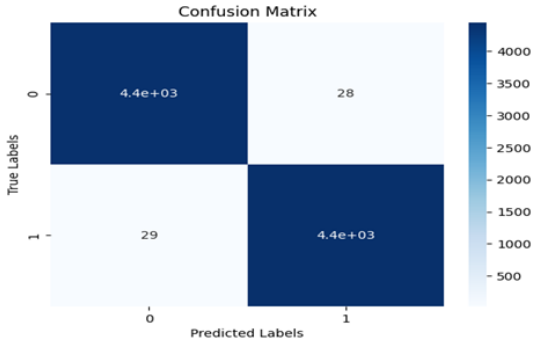

Figure 6. Confusion matrix

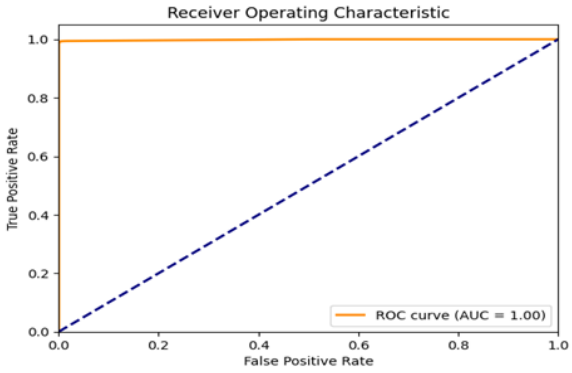

Figure 7. Thresholds, FP and TP

Table 1. Results of model performance

Accuracy	Recall	Precision	F1	TN	FP	FN	TP
99.4%	99.4%	99.4%	99.4%	4,412	28	29	4,446

Table 3. Metric performance of existing methods

Authors	Recall	Precision	Accuracy	F1 Score
[24]	97%	98%	98%	-
[25]	74%	90%	96%	81%
[26]	96%	85%	94%	-
[23]	91%	91%	91%	91%
Proposed model	99%	99%	99%	99%

C. Model comparison

Various models have been explored for their effectiveness in research related to detecting SQL injection threats. Reference [23] employed an MLP model, whereas [25] and [23] utilized LSTM to detect SQL injection attacks. [26] employed the CNN model for the same task. Table 4 illustrates the superior performance of the suggested LSTM-based model in detecting SQL injection threats compared to competing models. The LSTM model achieved outstanding scores of 99% for Recall, Precision, Accuracy, and F1, surpassing the models presented by [23]–[26] and which obtained lower ratings across assessment measures. The accuracy of the LSTM-based model in identifying SQL injection attacks suggests its potential to enhance security measures against these types of vulnerabilities.

IV. CONCLUSION AND FUTURE WORKS

Detecting SQL injection attacks using neural networks and LSTM is the focus of our study. The accuracy score was computed to assess how well the proposed model performed. According to the study's findings, the LSTM-based model achieved an excellent accuracy score of 99.4%. This means that our proposed model correctly identified positive and negative situations 99.4% of the time. The study's high accuracy score for LSTM highlights its reliability and resilience in identifying SQL injection attacks and its potential to improve web application security against SQL injection attacks.

Deep learning approaches such as attention mechanisms and CNNs should be investigated to increase detection accuracy and capture spatial relationships when detecting SQL injection attempts using LSTM networks. Improving the interpretability of LSTM-based systems is also essential since it provides insights into decision-making processes.

V. ACKNOWLEDGMENT

We would like to acknowledge the Computer Science Department of Kwame Nkrumah University of Science and Technology for their support and encouragement.

REFERENCES

[1] M. M. Bujnowska-Fedak, J. Waligóra, and A. Mastalerz-Migas, "The Internet as a source of health information and services," *Advancements and Innovations in Health Sciences*, pp. 1–16, 2019. https://doi.org/10.1007/5584_2019_396

[2] M. Y. Bae, H. K. Lim, and D. J. Cho, "A study on security diagnosis using automated Google hacking tools-focusing on the US government website," *Journal of Advances in Information* Technology, vol. 7, no. 2, pp. 93–97, 2016.

[3] P. Sadotra and C. Sharma, "SQL Injection Impact on Web Server and Their Risk Mitigation Policy Implementation Techniques: An Ultimate solution to Prevent Computer Network from Illegal Intrusion.," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, 2017.

[4] D. A. Kindy and A.-S. K. Pathan, "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques," in 2011 *IEEE 15th international symposium on consumer electronics (ISCE)*, IEEE, 2011, pp. 468–471. https://doi.org/10.1109/ISCE.2011.5973873

[5] G. Deepa, P. S. Thilagam, F. A. Khan, A. Praseed, A. R. Pais, and N. Palsetia, "Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications," *Int J Inf Secur*, vol. 17, pp. 105–120, 2018. https://doi.org/10.1007/s10207-016-0359-4

[6] W. B. Demilie and F. G. Deriba, "Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques," *J Big Data*, vol. 9, no. 1, p. 124, 2022. https://doi.org/10.1186/s40537-022-00678-0

[7] G. R. Chowdary, S. Neeraj, and S. Sparsha, "Machine Learning Approaches for Different Cyberthreats," *BMS Institute of technology and Management*, 2021. https://doi.org/10.13140/RG.2.2.23437.67040

[8] M. H. U. Sharif, "Web Attacks Analysis and Mitigation Techniques," *International Journal of Engineering Research & Technology (IJERT)*, pp. 10–12, 2022.

[9] R. Dorai and V. Kannan, "SQL injection-database attack revolution and prevention," *J. Int'l Com. L. & Tech.*, vol. 6, p. 224, 2011.

[10] H. M. Elshafie, T. M. Mahmoud, and A. A. Ali, "Improving the performance of the snort intrusion detection using clonal selection," in *2019 International Conference on Innovative Trends in Computer Engineering (ITCE), IEEE*, 2019, pp. 104–110. https://doi.org/10.1109/ITCE.2019.8646601

[11] A. Luo, W. Huang, and W. Fan, "A CNN-based Approach to the Detection of SQL Injection Attacks," in *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), IEEE*, 2019, pp. 320–324. https://doi.org/10.1109/ITCE.2019.8646601

[12] Q. Li, F. Wang, J. Wang, and W. Li, "LSTM-based SQL injection detection method for intelligent transportation system," *IEEE Trans Veh Technol*, vol. 68, no. 5, pp. 4182–4191, 2019. https://doi.org/10.1109/TVT.2019.2893675

[13] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Detection of SQL injection based on artificial neural network," *Knowl Based Syst*, vol. 190, p. 105528, 2020. https://doi.org/10.1016/j.knosys.2020.105528.

[14] A. Sivasangari, J. Jyotsna, and K. Pravalika, "SQL Injection Attack Detection using Machine Learning Algorithm," in *Proceedings of the 5th International Conference on Trends in Electronics and Informatics, ICOEI 2021, Institute of Electrical and Electronics Engineers Inc.*, Jun. 2021, pp. 1166–1169. https://doi.org/10.1109/ICOEI51242.2021.9452914

[15] 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), IEEE*, 2022, pp. 394–400. https://doi.org/10.1109/ICAAIC53929.2022.9792964

[16] Hussain, S. S. (2019). SQL injection dataset [Data set]. *Kaggle*. https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

[17] Hussain, S. S. (2020). SQL injection dataset [Data set]. *Kaggle*. https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

[18] Hussain, S. S. (2021). SQL injection dataset [Data set]. *Kaggle*. https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

[19] D. Wichers and J. Williams, "Owasp top-10 2017," *OWASP Foundation*, vol. 3, p. 4, 2017.

[20] Y. Fang, Y. Li, L. Liu, and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning," in *Proceedings of the 2018 international conference on computing and artificial intelligence*, 2018, pp. 47–51. https://doi.org/10.1145/3194452.3194469

[21] *M. T. Muslihi and D. Alghazzawi*, "Detecting SQL Injection On Web Application Using Deep Learning Techniques: A Systematic Literature Review," in *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 2020, pp. 1–6. https://doi.org/10.1109/ICVEE50212.2020.9243198

[22] S. Fraihat, S. Makhadmeh, M. Awad, M. A. Al-Betar, and A. Al-Redhaei, "Intrusion detection system for large-scale IoT NetFlow networks using machine learning with modified Arithmetic Optimization Algorithm," *Internet of Things*, p. 100819, 2023.

[23] N. Joshi Padma, N. Ravishankar, M. B. Raju, and N. C. Ravi, "Surgical Striking Sql Injection Attacks Using Lstm," *Indian Journal of Computer Science and Engineering*, vol. 13, no. 1, pp. 208–220, Jan. 2022

[24] K. R. Jothi, N. Pandey, P. Beriwal, and A. Amarajan, "An efficient SQL injection detection system using deep learning," in *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), IEEE*, 2021, pp. 442–445. https://doi.org/10.1109/ICCIKE51210.2021.9410674

[25] I. C. Potinteu and R. Varga, "Detecting Injection Attacks using Long Short-Term Memory," in *Proceedings - 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing, ICCP 2020, Institute of Electrical and Electronics Engineers Inc.*, Sep. 2020, pp. 163–169. https://doi.org/10.1109/ICCP51029.2020.9266177

[26] 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 *Proceedings of Data Analytics and Management: ICDAM 2021*, Volume 2, Springer, 2022, pp. 293–304. https://doi.org/10.1007/978-981-16-6285-0_24