Preprints.org

Article

Not peer-reviewed version

A Deep Learning Architecture for Detecting SQL Injection Attacks based on RNN Autoencoder Model

Maha Alghawazi *, Daniyal Alghazzawi, Suaad Alarifi

Posted Date: 11 July 2023

doi: 10.20944/preprints202307.0679.v1

Keywords: SQL injection attacks; Recurrent neural network (RNN) autoencoderANN; CNN; Decision Tree; Naïve Bayes; SVM; Random Forest; Logistic Regression

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

A Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model

Maha Alghawazi [1,†,‡], Daniyal Alghazzawi [2,‡] and Suaad Alarifi [2,*] on behalf of the Information Security Research Group, King Abdulaziz University

[1] Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 80200, Saudi Arabia; mmohammadalqhtani@stu.kau.edu.sa
[2] Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 80200, Saudi Arabia; dghazzawi@kau.edu.sa
* Correspondence: Salarifi@kau.edu.sa

Abstract: SQL injection attacks are one of the most common types of attacks on web applications. These attacks exploit vulnerabilities in the application's database access mechanisms, allowing attackers to execute unauthorized SQL queries. In this study, we propose an architecture for detecting SQL injection attacks using a recurrent neural network (RNN) autoencoder. The proposed architecture was trained on a publicly available dataset of SQL injection attacks. Then compared with several other machine learning models, including ANN, CNN, Decision Tree, Naïve Bayes, SVM, Random Forest, and Logistic Regression. The experimental result showed that the proposed approach achieved an accuracy of 94% and an F1 score of 92%, which demonstrate its effectiveness in detecting QL injection attacks with high accuracy in comparison with other models covered in the study.

Keywords: SQL injection attacks; Recurrent neural network (RNN); autoencoder ANN; CNN; decision tree; Naïve Bayes; SVM; random forest; logistic regression

1. Introduction

SQL injection attacks (SQLIAs) pose a severe security threat to web applications. These attacks involve the malicious execution of SQL queries on a server, enabling unauthorised access and retrieval of restricted data stored within databases [1]. Figure 1 illustrates the basic process of an SQLIA.

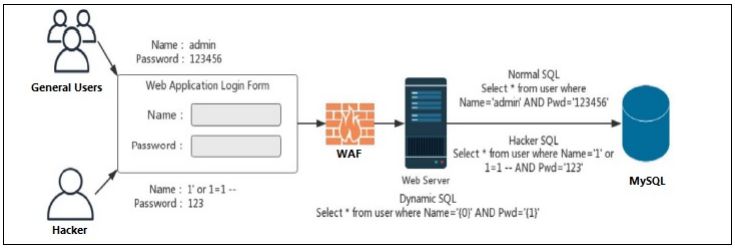

Figure 1. SQL injection attack process, adopted from [2].

Attackers exploit web applications by injecting SQL statements or using special symbols through user input to target the database tier and gain unauthorised access to valuable assets [2]. Due to the absence of proper validation in some web applications, which is usually the programmer fault, attackers can bypass authentication mechanisms and gain access to databases, enabling them to retrieve or manipulate data without appropriate authorisation [1].

Detection of such attacks is crucial to ensure the security and integrity of the web application and its associated data. To address this issue, a deep learning architecture based on the Recurrent Neural Network (RNN) Autoencoder model has been proposed for detecting SQL injection attacks.RNN autoencoder is the special case of the RNN based Encoder-Decoder (RNN-ED) model.The autoencoder consists of an encoder RNN that encodes the input sequence into a hidden state, and a decoder RNN

© 2023 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

that decodes the hidden state back into the original input sequence. The encoder and decoder RNNs are trained jointly using backpropagation to minimize the reconstruction error between the input and output sequences [3].In this study, we propose an architecture for an RNN autoencoder which is a combination of an autoencoder and a recurrent neural network (RNN) for SQL injection attacks detection. Figure 2 illustrates the architecture of the proposed model.

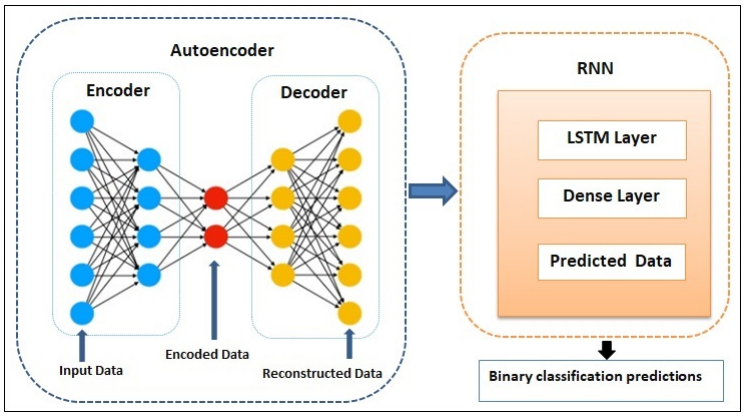

Figure 2. The RNN autoencoder architecture for SQL injection attacks detection.

In Figure 2, the proposed architecture consists of two main parts: the autoencoder and the RNN. The autoencoder contains an input layer, an encoder, and a decoder. The encoder takes the input data and compresses it into a lower-dimensional representation, which is then fed to the decoder. The decoder then reconstructs the input data from the encoded representation. The RNN is designed to take the compressed representation of the input data learned by the autoencoder and use it to make binary classification predictions [4]. RNN consists of an LSTM layer and a dense layer, which takes the encoded data from the Autoencoder as input and processes it through an LSTM layer, which is then fed to a dense layer to make a prediction on the output.

The aim of this study was to develop an architecture based on a recurrent neural network (RNN) autoencoder to detect SQL injection attacks. Moreover, the proposed approach that addresses this attack was discussed and compared. The research questions are:
Q1: Is the proposed RNN autoencoder based architecture effective for detecting SQL injection attacks?
Q2: How can the RNN autoencoder be optimized to improve its performance for detecting SQL injection attacks?
Q3: Can an RNN autoencoder outperform other SQL injection attacks machine learning detection models?

The main contributions of this paper are as follows:

- Proposing an SQLIAs detection architecture based on a recurrent neural network (RNN) autoencoder algorithm.
- A comparison between the proposed method and different machine learning techniques used for detecting and preventing of SQLIAs.

The paper is structured as follows: Section 2 reviews the related works about research in this area. The methodology is discussed in Section 3. Experiment results and discussion are shown in Section 4. The last section is the conclusion and future work.

2. Literature Review

This section explores a variety of ML and DL techniques found in the literature for the detection of SQL injection attacks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

Ketema [5] used a deep learning Convolutional Neural Network(CNN) to build a model to prevent an SQLI using a public benchmark dataset. The model was trained using deep learning, with different hyperparameters values and with five different scenarios. The model has achieved an accuracy of 97%. Roy et al. [6] present a method for detecting SQL injection attacks using machine learning classifiers. The authors use five ML classifiers: Logistic Regression, AdaBoost, Naive Bayes, XGBoost, and Random Forest to classify SQL queries as either legitimate or malicious. The proposed approach was trained and evaluated using a publicly available dataset of SQL injection attacks on Kaggle. The results of the study showed that the best performance was given by Naïve Bayes classifier with accuracy of 98.33%.Finally, authors performed a comparison with previous work. Overall, the study demonstrates the potential of machine learning classifiers in improving the accuracy and efficiency of SQL injection attack detection.

S.S. Anandha Krishnan et al. [7]proposes a machine learning-based approach for detecting SQL injection attacks. The authors argue that traditional signature-based approaches are ineffective against advanced attacks, and machine learning can help address this issue. The authors first describe the various types of SQL injection attacks and their impact on web applications. They then outline the proposed framework, which consists of preprocessing the data, feature extraction, model training, and evaluation. The results show that CNN classifier model performs better than the other classifiers in terms of accuracy, precision, recall, and F1-score. Rahul et al. [8] proposes a novel method of protecting against SQL injection and cross-site scripting (XSS) attacks by augmenting a web application firewall (WAF) with a honeypot. The WAF filters incoming traffic using established patterns, while the honeypot is designed to attract attackers and capture information about their attack methods, which is then used to improve the WAF's ability to detect and prevent future attacks. The proposed method is evaluated through experiments, and the results suggest that the combination of a honeypot and WAF can effectively protect web applications from these types of attacks.

Zhang et al. [9] proposes a method for detecting SQL injection attacks using a deep neural network. The authors state that traditional methods of SQL injection attack detection have limitations, prompting the development of this new approach. The authors gather a dataset of clean queries and malicious queries and use it to train a deep neural network classifier with several layers. They then compared the result of the proposed method with the traditional machine learning algorithms include: KNN , DT, and LSTM algorithm.Liu et al. [10] proposes a new approach, called DeepSQLi, for the automated detection of SQL injection vulnerabilities in web applications using deep semantic learning techniques. DeepSQLi uses a deep neural network to learn the semantic meaning of SQL queries and identify potential injection vulnerabilities. The model is trained using a dataset of benign and malicious SQL queries and leverages multiple layers of convolutional and recurrent neural networks. The experimental results show that DeepSQLi outperforms SQLmap, such that more SQLi attacks can be identified faster with using a less number of test cases. Chen et al. [11] present a novel approach for detecting and preventing SQL injection attacks on web applications using deep learning algorithms. The authors train and evaluate the performance of a convolutional neural network (CNN) and a multilayer perceptron (MLP) and compare them in terms of accuracy, precision, recall, and F1-score metrics. The experimental results show that the CNN and MLP models both perform well for SQL injection attack detection.

In summary, deep learning-based approaches, have shown great promise in detecting SQL injection attacks. These approaches can learn the underlying patterns in the input data and detect any anomalies, making them more effective in detecting disguised attacks. In this research our goal is to explore the effectiveness of the proposed RNN Autoencoder in detecting SQL injection.

3. Materials and Methods

This study consists of three primary phases: data preparation, model training, and evaluation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

3.1. Data Preparation

3.1.1. Data Preprocessing

The Kaggle dataset [12] is utilized in this research to train, evaluate, and compare the performance of an RNN autoencoder with several classifiers. This dataset is specifically designed for SQLIAs and consists of both benign (normal) and SQL injection (malicious) traffic collected from multiple websites. The benign queries are labelled as 0s, while the malicious SQL injection queries are labelled as 1s. The dataset comprises 30919 records, with 19537 normal statements and 11382 malicious SQL injection statements.

In order to enhance the accuracy of our trained models, we performed data cleaning on the selected dataset. This involved removing any null values and eliminating duplicate records. The removal of missing or null values is crucial, as it prevents the model from learning incorrect relationships or making predictions based on incomplete data. After completing the cleaning process, the dataset consisted of a total of 30907 records, with 19529 normal statements and 11378 malicious statements as shown in Figure 3. Each record contains two main features: 'Query', which represents the statement itself, and 'Label', which indicates whether the statement is normal (0) or malicious (1).

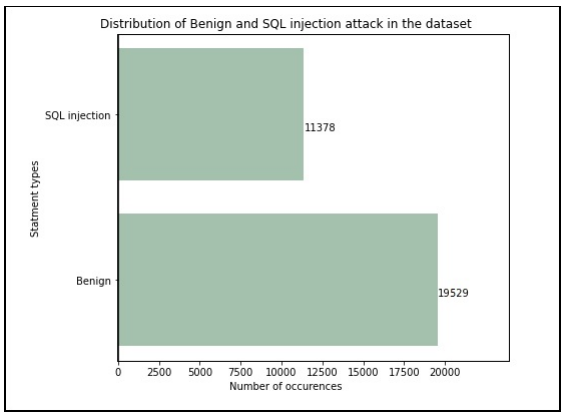

Figure 3. Distribution of benign and SQL injection attacks in the dataset.

3.1.2. Balancing and Sampling

Stratified sampling was applied, which ensures that the training and testing sets have a similar proportion of each class. This is important for imbalanced datasets like the SQL injection dataset, where the number of malicious queries is much smaller than the number of benign queries [13].

3.2. Model Training

In this experiment, we divided the dataset into two parts: 80% for training and 20% for testing. This division allows us to train the proposed approach on a majority of the data and assess their performance on unseen samples.

3.3. Model evaluation

After training the RNN autoencoder model on the training set, we applied them to the testing set and calculated various performance metrics such as (ROC) curve, accuracy, precision, recall, and F1-score to measure the effectiveness of the RNN autoencoder in detecting SQLIAs.The mathematical representation of these metrics are calculated as follows:

The accuracy metric measures the percentage of correctly classified samples [14], and it is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

Precision, another important metric, represents the probability that a sample will be correctly classified [14]. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity or true positive rate, indicates the proportion of positive samples that are correctly classified [14]. The recall score is calculated as:

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

The F1 score is a combined metric that considers both precision and recall, providing a balanced measure of model performance [15]. It is calculated as:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

TN is the true negative rate. It indicates the number of correctly predicted normal requests.
TP is the true positive rate. It indicates the number of correctly predicted malicious requests.
FN is the false negative rate. It indicates the number of incorrectly predicted normal requests.
FP is the false positive rate. It indicates the number of incorrectly predicted malicious requests.

4. Results and Discussion

This section provides a description of the experimental results.The Python environment was used to implement the system.Table 1 summarizes the performance of RNN autoencoder in terms of evaluation metrics.

Table 1. Performance metrics for the proposed model.

Performance metrics	Result
Accuracy	94%
Precision	95%
Recall	90%
F1-Score1	92%

The results from Table 1, show that the RNN autoencoder approach does perform better in terms of prediction accuracy.RNN autoencoder achieved an accuracy of 94% and an F1 score of 92%. Further, we use the receiver operating characteristic (ROC) curve for checking the performance of the proposed approach. It is a graph that shows the relationship between the true positive rate (TPR) and false positive rate (FPR) for different classification thresholds [16].

The AUC value of the RNN autoencoder model is shown in Figure 4. We obtained the value of 0.94, which indicates that our model can successfully separate 94% of positive and negative rates.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

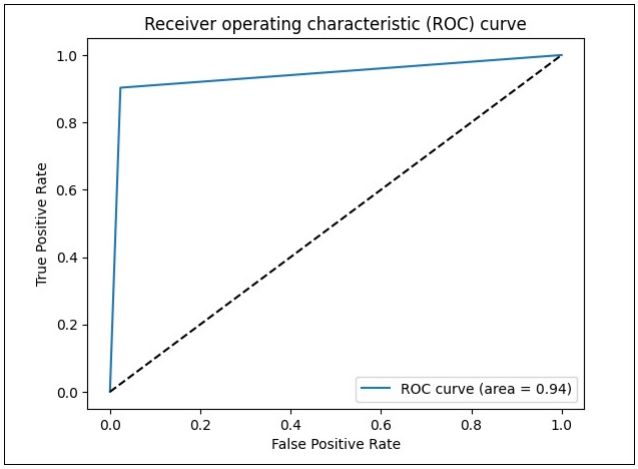

Figure 4. Receiver Operating Curve (ROC) of our proposed approach.

Regarding (RQ1) based on the results provided, it appears that the proposed RNN autoencoder model is performing well in correctly identifying instances of SQL injection attacks in the dataset and can be effective for detecting SQL injection attacks.

Regarding (RQ2), one of the most used methods to optimize the RNN autoencoder to improve its performance for detecting SQL injection attacks is to adjust the hyper-parameters of the model such as epochs [16].To find the optimal number of epochs to train a model. We will be experimenting with various numbers of epochs and checking how it will effects the accuracy . In the first iteration we initialize epochs to 10.

At epochs = 10 , we get an accuracy of 88%. From Figure 5, we can infer that the validation error decreases. Next, we set the epochs to 50.

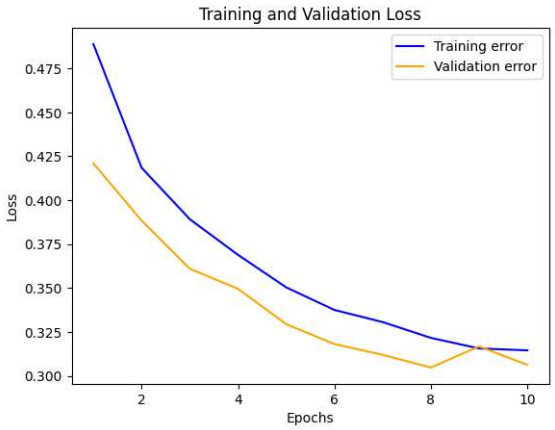

Figure 5. Loss in SQL injection dataset using 10 epochs.

From Figure 6, the accuracy of the model increased to 94% at epochs 50.Next we try to increase the number of epochs to 100.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

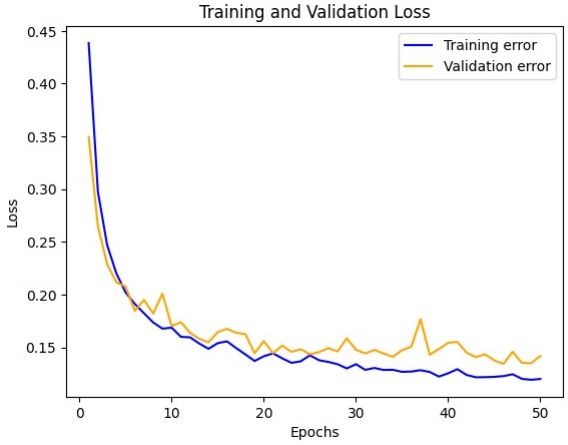

Figure 6. Loss in SQL injection dataset using 50 epochs.

From Figure 7, at epoch = 100 the accuracy increased to 95% but the validation error increases this may case an overfitting. Using a small number of epochs the model cannot capture the underlying patterns in the data and may cause underfitting. And training the model using many epochs it may lead to overfitting where we the model is learning even noise or unwanted parts[17]. So from the this experiment we can stop the training process early at around 50 epoch to get better permormance from the model without underfitting or overfitting the model.In summary, Table 2 summarize the choice of the different hyper-parameters.

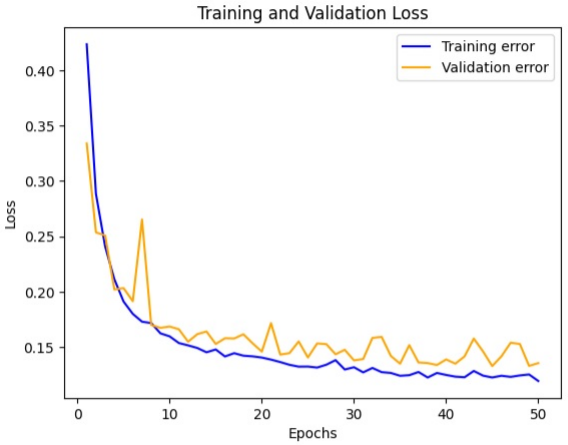

Figure 7. Loss in SQL injection dataset using 100 epochs.

Table 2. Values of the several hyper-parameters.

Hyperparameters	Value
Number of Hidden layers	3
Hidden layer size (neurons)	64 units
Optimizer	Adam
Loss function	binary cross-entropy
Activation function	ReLU and sigmoid
Number of epochs	50
Batch size	128

The proposed model achieves the best performance when trained for 50 epochs using Adam optimizer, and batch size of 128, ReLU activation function for encoder layer, and sigmoid activation function for decoder layer in the autoencoder and output layer in the RNN.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

We compared the performance of the proposed approach with the performance of several classifiers including ANN, CNN, Decision Tree, Naïve Bayes, SVM, Random Forest, and Logistic Regression. The results are presented in Figure 8.

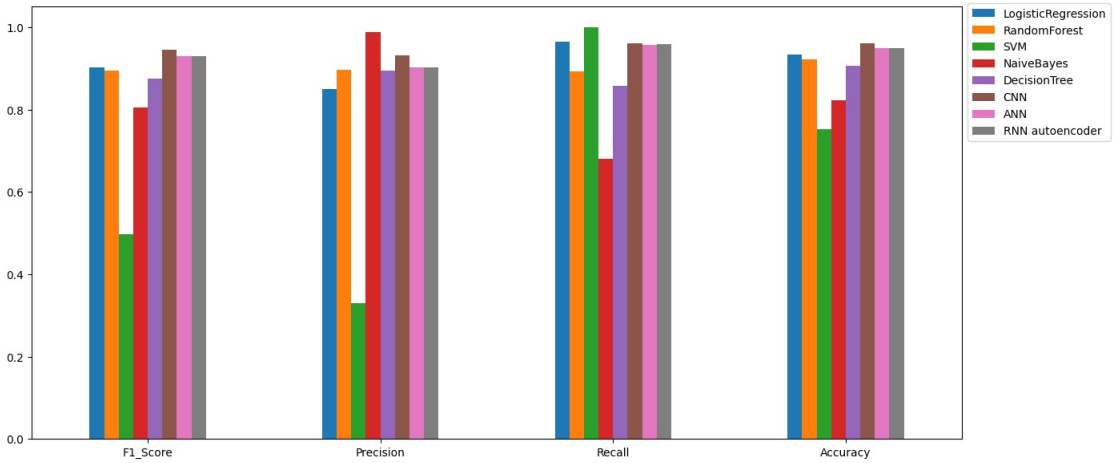

Figure 8. The Evaluation Metrics Comparison for different ML algorithms.

The results in Figure 8 show that the RNN autoencoder and ANN are effective in detecting SQL injection attacks, achieving high accuracy of 94% and F1 scores of 92%. The RF , LR, and DT models also performed well, achieving accuracy scores of 92% , 93%, and 90% respectively, and F1 scores of 89%,90%, and 87%. The CNN model had the highest accuracy of 96% and F1 score of 49%, indicating its potential for detecting SQL injection attacks. However, the Naive Bayes and SVM models had the lower accuracy and F1 score, they achieving accuracy scores of 82% and 75% respectively, and F1 scores of 80% and 49%.

Regarding (RQ3), the results indicate that the RNN autoencoder approach outperforms some of the other algorithms, including logistic regression, decision trees, random forest, SVM, and Naive Bayes, in terms of accuracy, precision, recall, and F1-score. The RNN autoencoder approach also performs comparably to some of the other algorithms, including CNN, and ANN, in terms of these metrics.

This suggests that the RNN autoencoder approach is a promising method for detecting SQL injection attacks and may be more effective than some traditional machine learning algorithms. The key advantage of the RNN autoencoder approach is that it can learn a compressed representation of the input data, which can capture the underlying patterns and relationships in the data more effectively than traditional methods.

5. Conclusions

A deep learning architecture model based on an RNN autoencoder has been proposed for detecting SQL injection attacks. The autoencoder is trained to learn a compressed representation of the input data, while the RNN uses this compressed representation to make binary classification predictions. In this study, the RNN autoencoder was trained with different optimization techniques on a public SQL injection dataset. The performance of the model is evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1-score. Additionally, a ROC curve was calculated to evaluate the model's performance. The experimental result showed that the proposed approach achieved an accuracy of 94% and an F1 score of 92%, which indicates that RNN autoencoder is a promising method for detecting SQL injection attacks. As part of future research, we plan to explore the use of a more complex architecture of RNN autoencoder to detect SQL injection attacks. Additionally, we acknowledge that the dataset used in this study was relatively small, and we recommend expanding the dataset and implementing the models in real-world scenarios for future investigations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

Author Contributions: Conceptualization, M.A. and D.A.; methodology, M.A.; software, M.A.; vali- dation, M.A., D.A. and S.A.; formal analysis, O.R.; investigation, M.A.; resources, M.A.; data curation, M.A.; writing—original draft preparation, M.A.; writing—review and editing, S.A.; visualization, M.A.; supervision, D.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant No. IFPDP-284-22. The authors, therefore, acknowledge with thanks to DSR technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SQLIA	SQL injection attacks
RNN-ED	RNN-based Encoder-Decoder
IDSs	Intrusion detection systems
ML	Machine learning
DL	Deep learning
NB	Naive Bayes classifier
DT	Decision Tree
LR	Logistic Regression
RF	Random Forests
SVM	Support Vector Machines
CNN	Convolutional Neural Network
ANN	Artificial Neural Networks
MLP	Multi-layer Perceptron
RNN	Recurrent Neural Networks
LSTM	Long short-term memory

References

1. H.R., Y.W.; Kottegoda, H.; Andaraweera, D.; Palihena, P. A comprehensive review of methods for SQL injection attack detection and prevention **2022**.
2. Chen, D.; Yan, Q.; Wu, C.; Zhao, J. SQL Injection Attack Detection and Prevention Techniques Using Deep Learning. *Journal of Physics: Conference Series* **2021**, *1757*.
3. Yu, W.; Kim, I.Y.; Mechefske, C. Analysis of different RNN autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing* **2021**, *149*, 107322.
4. Do, J.S.; Kareem, A.B.; Hur, J.W. LSTM-Autoencoder for Vibration Anomaly Detection in Vertical Carousel Storage and Retrieval System (VCSRS). *Sensors* **2023**, *23*, 1009.
5. Ketema, A. DEVELOPING SQL INJECTION PREVENTION MODEL USING DEEP LEARNING TECHNIQUE. PhD thesis, St. Mary's University, 2022.
6. Roy, P.; Kumar, R.; Rani, P. SQL Injection Attack Detection by Machine Learning Classifier. 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC). IEEE, 2022, pp. 394–400.
7. Krishnan, S.A.; Sabu, A.N.; Sajan, P.P.; Sreedeep, A. SQL Injection Detection Using Machine Learning. *REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS* **2021**, *11*, 300–310.
8. Rahul, S.; Vajrala, C.; Thangaraju, B. A Novel Method of Honeypot Inclusive WAF to Protect from SQL Injection and XSS. 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON). IEEE, 2021, Vol. 1, pp. 135–140.
9. Zhang, W.; Li, Y.; Li, X.; Shao, M.; Mi, Y.; Zhang, H.; Zhi, G. Deep Neural Network-Based SQL Injection Detection Method. *Security and Communication Networks* **2022**, *2022*.
10. Liu, M.; Li, K.; Chen, T. DeepSQLi: Deep semantic learning for testing SQL injection. Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2020, pp. 286–297.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2023 doi:10.20944/preprints202307.0679.v1

11. Chen, D.; Yan, Q.; Wu, C.; Zhao, J. Sql injection attack detection and prevention techniques using deep learning. Journal of Physics: Conference Series. IOP Publishing, 2021, Vol. 1757, p. 012055.
12. Shah, S.S.H. SQL Injection Dataset, 2021.
13. Chindove, H.; Brown, D. Adaptive Machine Learning Based Network Intrusion Detection. Proceedings of the International Conference on Artificial Intelligence and its Applications, 2021, pp. 1–6.
14. Mwaruwa, M.C. Long Short Term Memory Based Detection Of Web Based Sql Injection Attacks. PhD thesis, UoN, 2019.
15. Ahmad, M.S.; Shah, S.M. Supervised machine learning approaches for attack detection in the IoT network. In *Internet of Things and Its Applications*; Springer, 2022; pp. 247–260.
16. Said Elsayed, M.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Network anomaly detection using LSTM based autoencoder. Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, 2020, pp. 37–45.
17. Afaq, S.; Rao, S. Significance of epochs on training a neural network. *Int. J. Sci. Technol. Res* **2020**, *9*, 485–488.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.