Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

AI-ENABLED SQL INJECTION ATTACK PREDICTION

G. Indira Priyadarshini, Associate Professor, Department of Information Technology, Vidya Jyothi Institute of Technology (Autonomous), Aziz Nagar, 500075, Hyderabad, Telangana, India, Email: priyadarshini@vjit.ac.in
M. Sharwani, Assistant Professor, Vidya Jyothi Institute of Technology (Autonomous), Aziz Nagar, 500075, Hyderabad, Telangana, India, Email: sharwaniit@vjit.ac.in
Dr. D. Marlene Grace Verghese, Associate Professor, Department of Information Technology, Vidya Jyothi Institute of Technology (Autonomous), Aziz Nagar, 500075, Hyderabad, Telangana, India, Email: degala.marlene@gmail.com

ABSTRACT
SQL injection attacks pose a serious threat to web applications, as they exploit vulnerabilities in the database layer by injecting malicious SQL code into user input fields. These attacks can have severe consequences, including unauthorized access, data breaches, and even the complete compromise of the application and underlying database. Although traditional methods like input validation and parameterized queries exist to counter SQL injection, they have their limitations. These methods often rely on manual coding practices and may not cover all possible attack vectors. As attackers continually evolve their techniques, there is a pressing need for advanced and automated solutions that can proactively identify and mitigate SQL injection attacks. This is where artificial intelligence (AI) proves its significance in predicting and combating SQL injection attacks. AI has the capacity to analyze vast amounts of data, detect patterns, and learn from previous attacks, making it an invaluable tool in this context. AI brings significant benefits to the prediction of SQL injection attacks. Its ability to detect anomalies, learn from new attack patterns, recognize complex patterns, reduce false positives, provide real-time protection, and scale to handle large applications makes it an indispensable tool. By leveraging AI, organizations can bolster their defenses against SQL injection attacks, mitigating risks and ensuring the security of their web applications and databases.

Keywords: SQL injection, Count vectorizer, Natural language processing, Artificial intelligence.

1. Introduction
SQL Injection is a type of cyber-attack that has been around for a long time. It involves injecting malicious SQL code into an application's input fields, which allows attackers to gain unauthorized access to the application's database. This can lead to severe consequences, such as data breaches and system compromises. In recent years, Artificial Intelligence (AI) and machine learning have become popular in various fields, including cybersecurity [1]. The idea of using AI to predict SQL Injection attacks emerged to bolster security measures and counter sophisticated attack techniques. By developing AI models that can analyze application input data, we can identify patterns that indicate the presence of an SQL Injection attack. The traditional methods used to prevent SQL Injection attacks rely on simple rule-based approaches or static pattern matching. However, these methods can sometimes be bypassed by well-crafted attacks. This is where AI-based prediction of SQL Injection attacks becomes essential. We need AI-based prediction because cyber attackers continuously evolve their methods, making it challenging to rely solely on traditional approaches [2]. AI-powered systems can process large amounts of data, discover hidden patterns, and adapt to new attack techniques, making them more effective in identifying SQL Injection attacks. The significance of AI-based prediction lies in its ability to enhance detection accuracy. AI models can learn from historical attack data and identify even subtle patterns that might go unnoticed by traditional methods. By doing so, they can reduce false positives, which helps minimize disruptions to legitimate user activities. Additionally, AI can serve as a proactive defense mechanism, continuously monitoring and protecting applications from potential threats, including novel and previously unseen SQL Injection attacks [3].

Artificial Intelligence, particularly machine learning, has shown promise in various cybersecurity applications due to its ability to analyze vast amounts of data, detect patterns, and make predictions. By harnessing the power of AI, security professionals can enhance their capabilities in detecting and mitigating SQL injection attacks [4].

Benefits of AI-based Prediction of SQL Injection Attacks:

- High Accuracy: AI models can achieve high accuracy in distinguishing between legitimate and malicious SQL queries, reducing false positives and false negatives.
- Real-time Detection: AI-based prediction can quickly assess incoming queries, providing a swift response to potential threats in real-time.
- Adaptability: The model can adapt to new attack patterns and variations, making it more resilient against emerging SQL injection techniques.
- Reduced Manual Effort: By automating the detection process, security teams can focus on other critical security tasks, allowing for a more efficient use of resources.
- Enhanced Security: Implementing AI-based prediction can significantly improve the security posture of web applications, safeguarding sensitive data and preventing unauthorized access.

2. Literature Survey

Alghawazi et al. [5] applied techniques from different areas to detect and deterrence of SQL injection attacks, for which to improve the detect ability of the attack, is not a new area of research but it is still relevant. Artificial intelligence and machine learning techniques have been tested and used to control SQL injection attacks, showing promising results. The main contribution of this paper is to cover relevant work related to different machine learning and deep learning models used to detect SQL injection attacks. With this systematic review, this work aimed to keep researchers up-to-date and contribute to the understanding of the intersection between SQL injection attacks and the artificial intelligence field. Zhang et al. [6] proposed a SQLNN deep neural network model. The core method is to convert the data into word vector form by word pause and then form a sparse matrix and pass it into the model for training to build a multi hidden layer deep neural network model containing ReLU function, which optimized the traditional loss function and introduces the Dropout method to improve the generalization ability of this model.

Uwagbole et al. [7] explored the generation of data set containing extraction from known attack patterns including SQL tokens and symbols present at injection points. Also, as a test case, this work build a web application that expects dictionary word list as vector variables to demonstrate massive quantities of learning data. The data set is pre-processed, labelled and feature hashing for supervised learning. This paper demonstrated a full proof of concept implementation of an ML predictive analytics and deployment of resultant web service that accurately predicts and prevents SQLIA with empirical evaluations presented in Confusion Matrix (CM) and Receiver Operating Curve (ROC). Gandhi et al. [8] proposed a hybrid CNN-BiLSTM based approach for SQLI attack detection. The proposed CNN-BiLSTM model had significant accuracy of 98% and superior performance compared to other machine learning algorithms. Also, paper presented a comparative study of different types of machine learning algorithms used for the purpose of SQLI attack detection. The study showed the performance of various algorithms based on accuracy, precision, recall, and F1 score with respect to proposed CNN-BiLSTM model in detection of SQL injection attacks.

Ali et al. [9] studied the top 10 security threats identified by the OWASP are injection attacks. The most common vulnerability is SQL injection and is the most dangerous security vulnerability due to the multiplicity of its types and the rapid changes that can be caused by SQL injection and may lead to financial loss, data leakage, and significant damage to the database, and this causes the site to be paralyzed. Machine learning is used to analysed and identified security vulnerabilities. It used classic machine learning algorithms and deep learning to evaluate the classified model using input validation features. Sharma et al. [10] used various classification algorithms to determine whether a particular code is malicious or plain. Some of the neural network and machine learning algorithms are Naive

Bayes classifier, LSTM, MLP, and SVM which can be used for the detection of SQL Injection attacks. This work compared various algorithms on a common dataset in this study.
Roy et al. [11] penetrated the logical section of the database. If the database has a logical flaw, the attackers send a new type of logical payload and get all of the user's credentials. Despite the fact that technology has advanced significantly in recent years, SQL injections can still be carried out by taking advantage of security flaws. Falor et al. [12] reviewed the different types of SQL Injection attacks and existing techniques for the detection of SQL injection attacks. We have compiled and prepared own dataset for the study including all major types of SQL attacks and have analysed the performance of Machine learning algorithms like Naïve Bayes, Decision trees, Support Vector Machine, and K-nearest neighbour. This work have also analysed the performance of Convolutional Neural Networks (CNN) on the dataset using performance measures like accuracy, precision, Recall, and area of the ROC curve. Tripathy et al. [13] investigated the potential of using machine learning techniques for SQL injection detection on the application level. The algorithms to be tested are classifiers trained on different malicious and benign payloads. They take a payload as input and decide whether the input contains a malicious code or not. The results showed that these algorithms can distinguish normal payloads from malicious payloads with a detection rate higher than 98%. The paper also compared the performance of different machine learning models in detecting SQL injection attacks.

3. Proposed System

3.1 Overview

Collect a diverse dataset that includes both legitimate user input and various types of SQL Injection attacks. This dataset should be representative of the application's user base and encompass different attack scenarios. Next, we preprocess the collected data, removing any noise, irrelevant information, and personally identifiable data that may violate privacy regulations. After preprocessing, we perform feature engineering, extracting relevant features from the data that effectively represent the characteristics of input patterns and potential attacks. The next step is to select appropriate AI algorithm for the task. Depending on the available data and prediction requirements, we can choose from supervised learning (ANN) or unsupervised learning (such as anomaly detection). With the algorithms selected, we proceed to train the AI model using the pre-processed dataset. During this phase, we adjust the model's parameters to optimize its performance in predicting SQL Injection attacks. To ensure the model's effectiveness and reliability, we evaluate its performance on a separate validation dataset and fine-tune it further if needed. We conduct extensive testing to validate its capabilities in real-world scenarios. Once the model has been thoroughly tested and proven effective, we integrate it into the application's security infrastructure. This enables the AI model to predict and prevent SQL Injection attacks in real-time. As the threat landscape constantly evolves, continuous monitoring and updating of the AI model are crucial. Regularly updating the model with new data allows it to adapt to emerging attack patterns and maintain its effectiveness over time.
Figure 1 shows the proposed system design. The detailed description of proposed system described as follows:
Step 1: Data Preparation Start by obtaining a SQL injection dataset, which contains examples of both SQL injection attacks and non-attacks (legitimate SQL queries).
Step 2: Feature Extraction with Count Vectorizer: Use the Count Vectorizer to convert the tokenized text data into numerical features. The result will be a matrix where rows represent data points (documents) and columns represent the counts of words (features) in each document.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

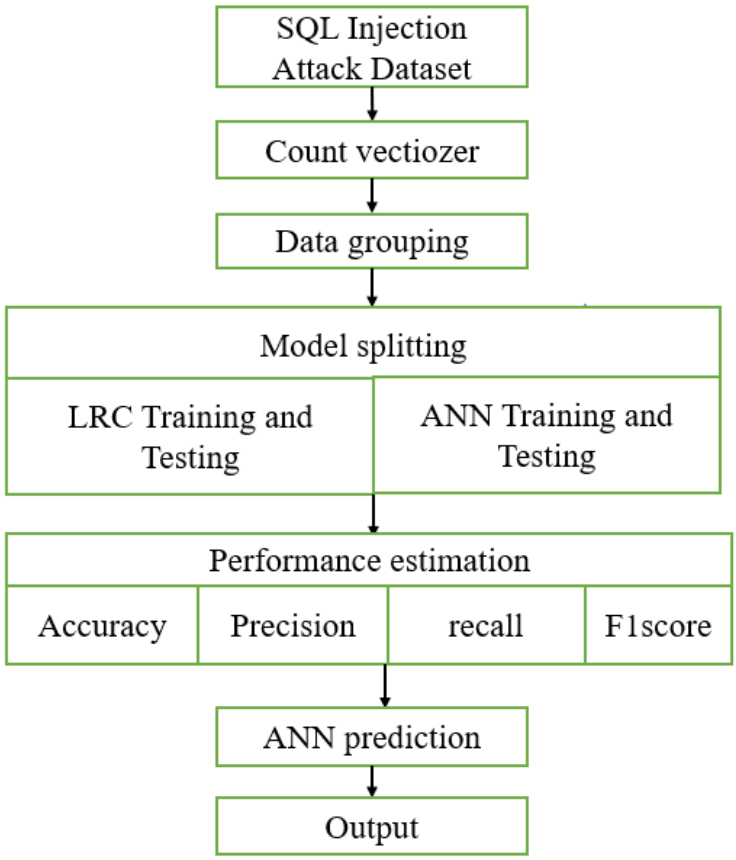


Fig. 1: Block diagram of proposed system.

Step 3: Model Training: Train both the Logistic Regression and ANN models on the training data.
Step 4: Model Evaluation: Evaluate the models' performance using appropriate evaluation metrics. Common metrics for binary classification problems like SQL injection detection include accuracy, precision, recall, F1-score, and ROC AUC.

3.2 Count Vectorizer

Machines cannot understand characters and words. So, when dealing with text data we need to represent it in numbers to be understood by the machine. Count vectorizer is a method to convert text to numerical data. Count Vectorizer converts a collection of text documents to a matrix of token counts: the occurrences of tokens in each document. This implementation produces a sparse representation of the counts. It creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that text sample.

A Count Vectorizer, also known as a CountVectorizer, is a text preprocessing technique widely used in natural language processing (NLP) and machine learning. It is part of the process of converting a collection of text documents into a numerical format that machine learning algorithms can work with. Below, I'll provide a detailed analysis of Count Vectorizer, including what it is, how it works, and its applications. Count Vectorizer is a technique for converting a text corpus (a collection of documents) into a matrix of token counts. In simpler terms, it transforms text data into numerical data that machine learning models can understand. It's a fundamental step in various NLP tasks such as text classification, sentiment analysis, topic modeling, and more.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

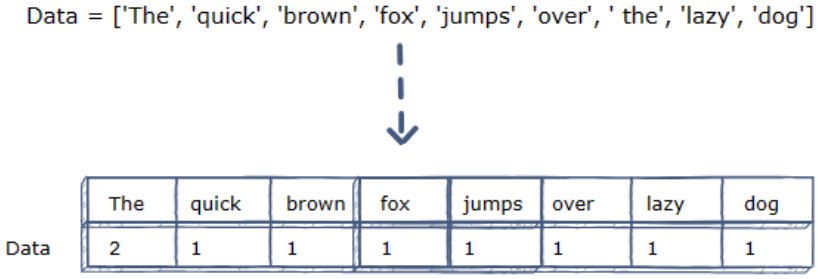


Fig. 2: Example of count vectorizer.

Here's how Count Vectorizer works:

Step 1: Tokenization: The first step is to tokenize the text, which means breaking it into individual words or tokens. Tokenization typically involves removing punctuation, splitting text on spaces, and handling special cases like contractions.

Step 2: Vocabulary Creation: Count Vectorizer builds a vocabulary from all the unique tokens (words) in the corpus. Each word becomes a feature in the vocabulary, and the position of the word in the vocabulary is recorded.

Step 3: Counting Tokens: For each document in the corpus, Count Vectorizer counts how many times each word from the vocabulary appears in that document. These counts are stored in a matrix where each row corresponds to a document, and each column corresponds to a word in the vocabulary.

Step 4: Sparse Matrix: The result is often a sparse matrix, as most documents only contain a subset of the vocabulary's words. Sparse matrices are efficient for storage and computation because they store only non-zero values.

3.3 Dataset Splitting

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So, we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

3.4 ANN Classifier

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images. Interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the “phenomenal world” with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt’s perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt’s model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

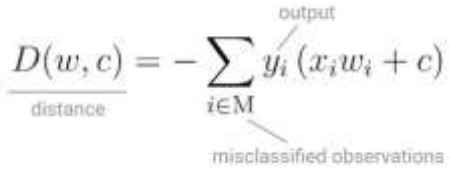
Fig. 3: Perceptron neuron model (left) and threshold logic (right).

Threshold *T* represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary.

It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:



To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

But, if you look at Deep Learning papers and algorithms from the last decade, you'll see the most of them use the Rectified Linear Unit (ReLU) as the neuron's activation function. The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input. The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

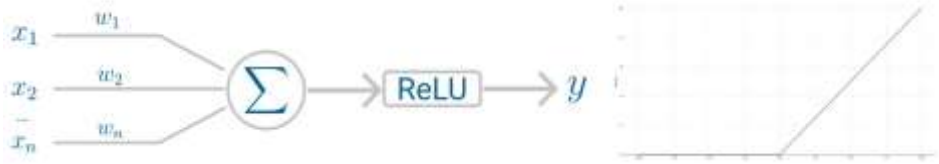


Fig. 4: Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

3.3.2 ANN

The ANN was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear. A ANN has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a ANN can use any arbitrary activation function. ANN falls under the category of feedforward algorithms because inputs are

combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer. If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning. This is where Backpropagation comes into play.

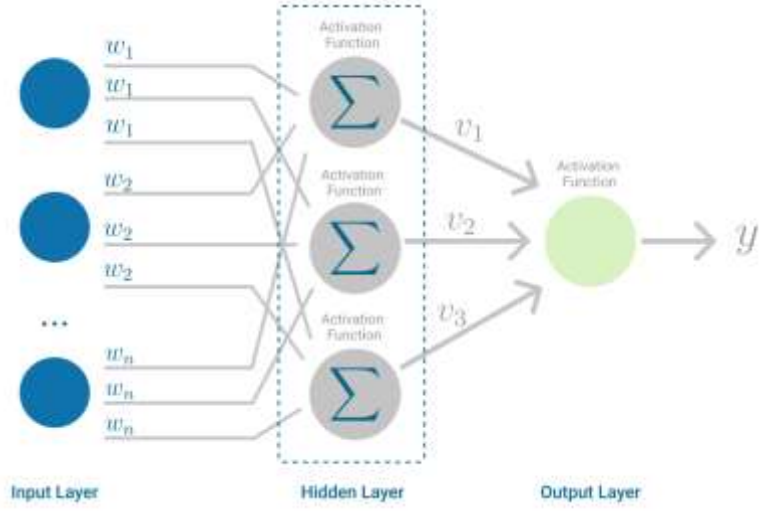


Fig. 5: Architecture of ANN.

Backpropagation: Backpropagation is the learning mechanism that allows the ANN to iteratively adjust the weights in the network, with the goal of minimizing the cost function. There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative because Gradient Descent is typically the optimization function used in ANN. In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network. One iteration of Gradient Descent is defined as follows:

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

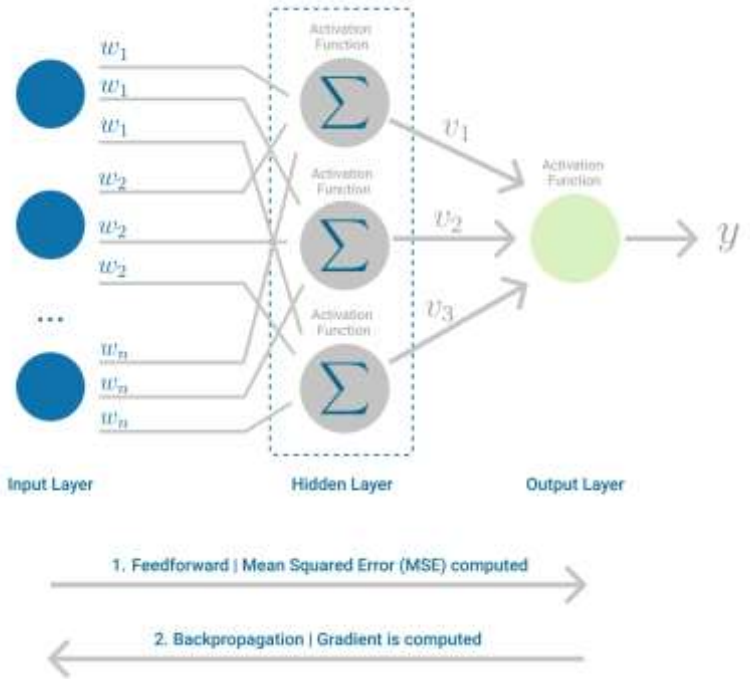


Fig. 6: ANN, highlighting the Feedforward and Backpropagation steps.

4. RESULTS AND DISCUSSION

Figure 7 represents a sample or a portion of the dataset that is being used for the task of detecting SQL injection attacks. It displays a few rows of data, showing both the text sentences (features) and their corresponding labels (whether they are SQL injection attacks or not). Figure 8 shows a portion of the dataset in tabular form (a DataFrame) after various preprocessing steps have been applied to the text features. Preprocessing could include tokenization, removal of stopwords, and vectorization, as mentioned in the code you provided. Each row likely represents a sample, and each column represents a feature (e.g., word or token).

	Sentence	Label
0	a	1
1	a'	1
2	a' --	1
3	a' or 1 = 1; --	1
4	@	1
...	...	...
4195	org/?option = com_k2 <a href = "http://corfopym	0
4196	com/?option = com_k2 <act> <![CDATA[procMemb...	0
4197	picsearch	0
4198	com/is?-WZx-uhyLezKNiYLvAbKL3W4oh5F749nr2KUmFF...	0
4199	de]]> </email_address> <find_account_answer...	0

4200 rows × 2 columns

Figure 7: Sample dataset used for detection of SQL injection attack.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4195	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4196	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4197	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4198	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4199	0	0	1	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0

4200 rows × 4717 columns

Figure 8: Data frame of features after preprocessing.

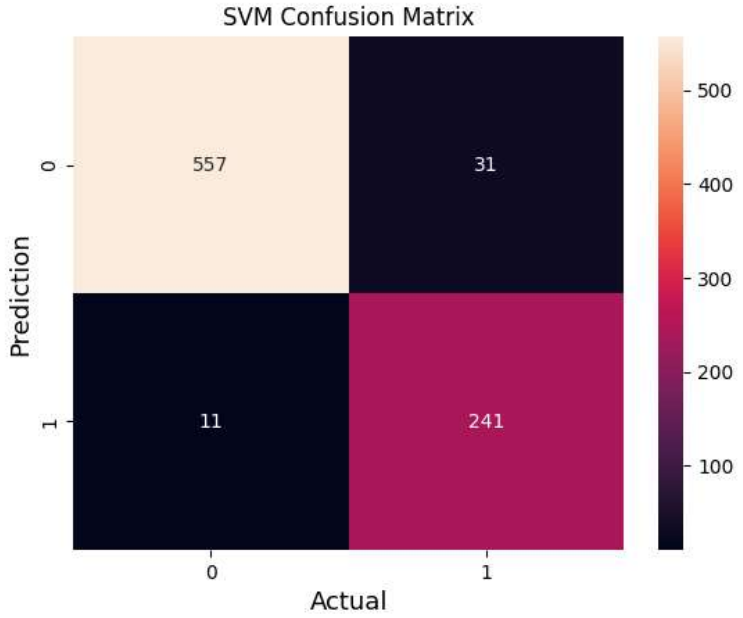


Figure 9: Confusion matrix heatmap of SVM classifier.

SVM classification_report

	precision	recall	f1-score	support
0	0.98	0.95	0.96	588
1	0.89	0.96	0.92	252
accuracy			0.95	840
macro avg	0.93	0.95	0.94	840
weighted avg	0.95	0.95	0.95	840

Figure 10: Classification report of SVM classifier.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 3, No. 3, March : 2024

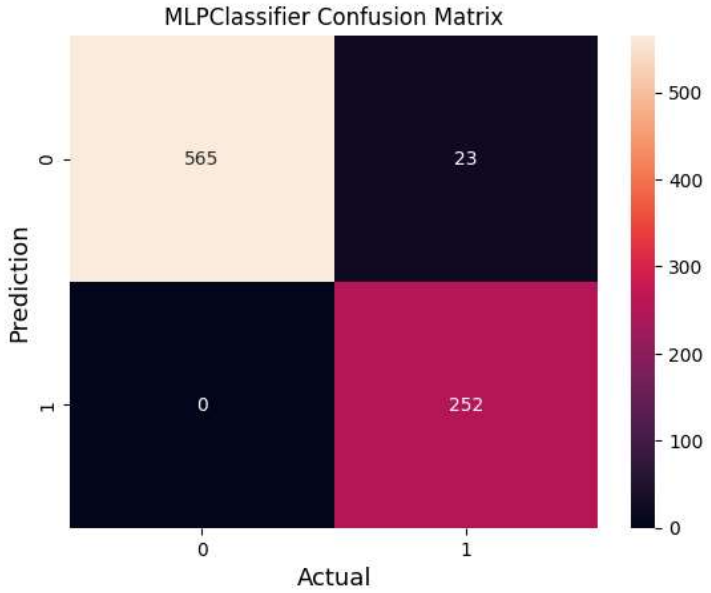


Figure 11: Confusion matrix heatmap for MLP Classifier.

MLPClassifier classification_report

	precision	recall	f1-score	support
0	1.00	0.96	0.98	588
1	0.92	1.00	0.96	252
accuracy			0.97	840
macro avg	0.96	0.98	0.97	840
weighted avg	0.97	0.97	0.97	840

Figure 12: Classification report of MLP Classifier.

5. Conclusion

In conclusion, SQL injection attacks pose a significant threat to web applications, potentially leading to unauthorized access, data breaches, and complete compromise of the application and underlying database. While traditional methods such as input validation and parameterized queries offer some level of protection, they have limitations and may not cover all attack vectors. Therefore, this work implemented ANN to proactively identify and mitigate SQL injection attacks. It can detect anomalies and learn from new attack patterns, which enables it to recognize complex attack vectors that traditional methods might miss. Furthermore, it can reduce false positives, provide real-time protection, and scale to handle large applications efficiently. These capabilities make AI an indispensable tool in defending against SQL injection attacks.

References

[1] Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: ASystematic Review. IEEE Access 2020,8, 35403–35419.
[2] Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of using Machine Learning Techniquesfor Intrusion Detection. IEEE Commun. Surv. Tutor. 2018,21, 686–728.
[3] Yan, R.; Xiao, X.; Hu, G.; Peng, S.; Jiang, Y. New deep learning method to detect code injection attacks on hybrid applications. J.Syst. Softw. 2018,137, 67–77.
[4] Aliero, M.S.; Qureshi, K.N.; Pasha, M.F.; Ghani, I.; Yauri, R.A. Systematic Review Analysis with SQLIA Detection and PreventionApproaches. Wirel. Pers. Commun. 2020,112, 2297–2333.
[5] Alghawazi, Maha & Alghazzawi, Daniyal & Alarifi, Suaad. (2022). Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. Journal of Cybersecurity and Privacy. 2. 764-777. 10.3390/jcp2040039.

[6] W. Zhang, Y. Li, X. Li, M. Shao, Y. Mi, H. Zhang, G. Zhi, "Deep Neural Network-Based SQL Injection Detection Method", Security and Communication Networks, vol. 2022, Article ID 4836289, 9 pages, 2022. https://doi.org/10.1155/2022/4836289

[7] S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 2017, pp. 1087-1090, doi: 10.23919/INM.2017.7987433.

[8] N. Gandhi, J. Patel, R. Sisodiya, N. Doshi and S. Mishra, "A CNN-BiLSTM based Approach for Detection of SQL Injection Attacks," 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2021, pp. 378-383, doi: 10.1109/ICCIKE51210.2021.9410675.

[9] M. H. Ali AL-Maliki; Mahdi Nsaif Jasim. "Review of SQL injection attacks: Detection, to enhance the security of the website from client-side attacks". International Journal of Nonlinear Analysis and Applications, 13, 1, 2022, 3773-3782. doi: 10.22075/ijnaa.2022.6152

[10] Sharma, V., Kumar, S. (2023). Comparative Study of Machine Learning Algorithms for Prediction of SQL Injections. In: Shukla, P.K., Singh, K.P., Tripathi, A.K., Engelbrecht, A. (eds) Computer Vision and Robotics. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-19-7892-0_36

[11] P. Roy, R. Kumar and P. Rani, "SQL Injection Attack Detection by Machine Learning Classifier," 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 2022, pp. 394-400, doi: 10.1109/ICAAIC53929.2022.9792964.

[12] Falor, A., Hirani, M., Vedant, H., Mehta, P., Krishnan, D. (2022). A Deep Learning Approach for Detection of SQL Injection Attacks Using Convolutional Neural Networks. In: Gupta, D., Polkowski, Z., Khanna, A., Bhattacharyya, S., Castillo, O. (eds) Proceedings of Data Analytics and Management . Lecture Notes on Data Engineering and Communications Technologies, vol 91. Springer, Singapore. https://doi.org/10.1007/978-981-16-6285-0_24

[13] D. Tripathy, R. Gohil and T. Halabi, "Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning," 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), Baltimore, MD, USA, 2020, pp. 145-150, doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00035.

[14] Hubskyi, O., Babenko, T., Myrutenko, L., Oksiiuk, O. (2021). Detection of SQL Injection Attack Using Neural Networks. In: Shkarlet, S., Morozov, A., Palagin, A. (eds) Mathematical Modeling and Simulation of Systems (MODS'2020). MODS 2020. Advances in Intelligent Systems and Computing, vol 1265. Springer, Cham. https://doi.org/10.1007/978-3-030-58124-4_27