International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

A Discourse of AI-assisted Neural Networks Is Used to Anticipate and Distinguish Vindictive SQL Codes

Methuku Nagasindhuja[1], Venkataradhakrishnamurty[2], Nadendla Anil Kumar Chowdari[3]
[1]*Postgraduate in CSE department, Audisankara College of Engineering and Technology, Nellore.*
[2]*Associate Professor, Dept. Of CSE, Audisankara College of Engineering and Technology, Nellore.*
[3]*Assistant professor, Dept. Of CSE, Audisankara College of Engineering and Technology, Nellore.*

ABSTRACT: The rise of AI and neural networks has significantly influenced different fields, including online protection. This theory investigates the use of AI-Assisted to help Neural networks in distinguishing SQL infusion assaults, a predominant and harmful type of digital danger. SQL infusion includes embedding pernicious SQL explanations into a question, empowering aggressors to control data sets[1]. Artificial intelligence helped Neural networks offer a hearty arrangement by utilizing AI to perceive examples and oddities in SQL questions. This approach includes information assortment and preprocessing, model preparation with different Neural network structures, and ongoing examination for recognition and grouping. The upsides of this strategy incorporate high precision, versatility, and robotization, even though difficulties like information quality, computational asset necessities, and interpretability remain. By and large, AI-assisted intelligence helped Neural networks present a strong and versatile instrument for upgrading web application security and fighting SQL infusion assaults. It represents a danger to data sets as they exploit weaknesses in the data set layer by infusing SQL codes into client data sets [2]. The results of an effective assault make unapproved access data sets and assailants can get sufficiently close to delicate information. So, to avoid information breaks and unapproved access we need to distinguish whether the executed SQL code on the client side is malignant or not. Even though a few strategies like defined questions [3], getting away from characters and information approval are a few conventional procedures to distinguish SQL infusion, they have their constraints. These techniques frequently depend on manual coding rehearses and may not distinguish new assaults [3]. As aggressors consistently develop their methods to assault and get close enough to delicate information there is a requirement for cutting-edge arrangements that can proactively distinguish and relieve SQL infusion assaults. Simulated intelligence can break down tremendous measures of information, recognize designs, and gain from past assaults. Computer-based intelligence carries critical advantages to the forecast of SQL infusion assaults. Its capacity to distinguish abnormalities, gain from new assault designs, perceive complex examples, diminish misleading up-sides, give continuous assurance, and scale to deal with huge applications makes it a key device. Here we utilize a count vectorizer to make tokens and give these tokens to a Neural Network Algorithm i.e., Multi-Layer Perceptron to distinguish the vindictive SQL code [4]. By utilizing computerized reasoning, we can distinguish and relieve malignant SQL codes quickly and precisely to guarantee the security of data sets.
KEYWORDS: Artificial Intelligence, Natural Language Processing, Forecast, vindictive SQL codes, Machine Learning, Threat Detection, SQL Infusion, Neural Networks, count vectorizer, Multilayer Perceptron.

1. Introduction

The rise of Artificial Intelligence (AI) and neural networks has profoundly impacted numerous fields, including cybersecurity. One notable application of these technologies is in the detection and prevention of SQL injection attacks [5]. SQL injection is a method used by attackers to manipulate databases through malicious SQL statements inserted into entry fields. Traditional detection methods

DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

have often proven insufficient against these sophisticated threats [3]. AI-assisted neural networks, particularly those leveraging Natural Language Processing (NLP) and Recurrent Neural Networks (RNN) [2], offer a powerful solution for anticipating and distinguishing malicious SQL codes. SQL injection is a cyber-attack technique that exploits vulnerabilities in web applications by injecting malicious SQL queries [6]. These attacks can lead to unauthorized data access, data modification, or even the deletion of entire databases. Conventional detection methods, such as signature-based detection and manual code reviews, often fail to identify advanced SQL injection attempts, necessitating the adoption of more sophisticated approaches.

1.1 Role of AI and Neural Networks AI, especially through neural networks, provides an intelligent and adaptive approach to detecting malicious SQL codes [7]. Neural networks excel at recognizing complex patterns and anomalies within large datasets, making them particularly suitable for cybersecurity applications.

1.2 Mechanism of AI-Assisted Neural Networks

A) Data Collection and Preprocessing:

Training Data: A diverse dataset comprising both legitimate and malicious SQL queries is essential for training the neural network [2].

Feature Extraction: Key features such as query length, structure, specific keywords, and the overall context of the query are extracted. These features help the neural network differentiate between benign and harmful SQL codes [5].

B) Natural Language Processing (NLP):

Text Analysis: NLP techniques analyze the textual content of SQL queries [1], identifying patterns and structures that indicate malicious intent.

Tokenization and Embedding: SQL queries are tokenized (i.e., broken down into individual words or symbols) and converted into numerical representations (embeddings) that the neural network can process [8].

C) Recurrent Neural Networks (RNN):

Sequential Data Handling: RNNs are designed to handle sequential data, making them ideal for analyzing SQL queries that have a sequential nature [6].

Memory and Context: RNNs, particularly Long Short-Term Memory (LSTM) networks, retain information from previous parts of the sequence [4], allowing them to understand the context of each token in the query.

Training Process: The RNN is trained using supervised learning, where it learns to classify queries based on labelled training data [2]. The network adjusts its parameters to minimize prediction errors through iterative training.

D) Detection and Classification:

Real-Time Analysis: The trained RNN analyzes incoming SQL queries in real time [7], detecting and flagging potential threats.

Anomaly Detection: The RNN identifies deviations from typical query patterns [6], which may indicate an SQL injection attempt.

Advantages of Using AI-Assisted Neural Networks

High Accuracy: NLP and RNN models achieve high accuracy in identifying malicious SQL codes due to their ability to understand complex patterns and contexts within the queries [8].

Adaptability: These models can be updated with new data to recognize emerging threats [4], ensuring they remain effective against evolving attack techniques [2].

Automation: AI automates the detection process [9], reducing the need for constant manual oversight and enabling continuous monitoring and protection.

Challenges and Considerations

Data Quality: The success of NLP and RNN models relies on the quality and comprehensiveness of the training data [3]. Inadequate or biased datasets can lead to inaccurate detection[1].

DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

Computational Resources: Training and running these models require substantial computational power, which may not be accessible to all organizations [4].
Interpretability: NLP and RNN models can be complex and challenging to interpret, which can be a drawback in understanding and verifying the model's decisions.

2. Overview of Literature

Relative Investigation of AI Algorithms for Expectation of SQL Infusions. Web applications are the most broadly used advanced stages due to their cross-platform similarity and the way that they needn't bother with clients to introduce anything to use them, making the utilization of online applications for an immense scope, and hence, the security chance of web applications is expanding [3], [5]. SQL infusion is one of the riskiest security attacks, harming an organization's standing, monetary misfortunes, and the protection of its clients [5]. Different order calculations can be utilized to decide if a specific code is noxious or plain. A portion of the neural organization and AI algorithms are Bayes classifier, LSTM, MLP, and SVM which can be utilized for the discovery of SQL Infusion assaults. They looked at different calculations on a typical dataset in this review to perceive how well they performed [2].

Trojan SQL: SQL infusion against regular language point of interaction to the information base.

The innovation of text-to-SQL has altogether upgraded the proficiency of getting to, what's more, controlling data sets. Nonetheless, restricted research has been led to concentrate on its weaknesses rising out of vindictive client collaboration [4]. By proposing Trojan SQL, an indirect access-based SQL infusion structure for text-to-SQL frameworks, they have shown how cutting-edge text-to-SQL parsers can be effectively deluded to deliver unsafe SQL proclamations that can discredit client inquiries or compromise delicate data about the information base. [3],[7]Trial results exhibited that both medium-sized models given tweaking and Enormous Language Models (LLM) based parsers utilizing provoking strategies are defenceless against this sort of assault, with assault achievement rates as high as almost 100% and 89%, individually. They trust that this study will raise more likely security.

Artificial Intelligence for Prediction of SQL Infusion Assault

SQL infusion assaults represent a danger to web applications, as they exploit weaknesses in the data set layer by infusing noxious SQL code into client input fields. These assaults can have serious outcomes, including unapproved access, information breaks, and, surprisingly, the total split of the difference between the application and hidden data set. Even though customary techniques like information approval and defined questions exist to counter SQL infusion [1], they have their constraints. These strategies frequently depend on manual coding rehearses and, what's more, may not cover all conceivable assault vectors. As aggressors constantly develop their procedures, there is a squeezing need for cutting-edge and mechanized arrangements that can proactively distinguish and relieve SQL infusion assaults. This is where Artificial consciousness (artificial intelligence) demonstrates its importance in anticipating and battling SQL infusion assaults [11]. Artificial intelligence can examine tremendous measures of information, identify designs, and gain from past assaults, making it an important apparatus in this unique circumstance [3]. Artificial intelligence carries huge advantages to the expectation of SQL infusion assaults. Its capacity to distinguish inconsistencies, gain from new assault designs, perceive complex examples, lessen misleading up-sides, give ongoing insurance, and scale to deal with enormous applications makes it an essential device. By utilizing Artificial intelligence, Associations can reinforce their guards against the SQL Infusion assaults and moderate the dangers [11] [12].

SQL Infusion Discovery Utilizing RNN

SQL infusion assaults are a typical kind of digital assault that exploits weaknesses in web applications to get to data sets through vindictive SQL questions. These assaults present a danger to the security and trustworthiness of web applications and their information [12]. The current strategies for

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

distinguishing SQL infusion assaults depend on predefined decisions that can be without any problem evaded by refined aggressors [6]. Hence, there is a requirement for a more compelling strategy for recognizing SQL infusion assaults. In this exploration, they proposed a book strategy for identifying SQL infusion assaults utilizing intermittent neural organizations (RNN) [3], which are a sort of profound learning model that can catch the sentence structure and semantic elements of SQL questions. They have prepared an RNN model on a dataset of harmless and noxious SQL inquiries and involved it to characterize inquiries as either harmless or pernicious. They assessed their strategy on a benchmark dataset and contrasted it and the current rule-based strategies [19]. Test results show that their technique accomplished high precision and beat the standard-based techniques for distinguishing SQL infusion assaults [15]. This examination adds to the field of web application security by giving a new and powerful answer for safeguarding web applications from SQL infusion assaults utilizing profound learning. This technique has both functional and hypothetical ramifications, as it very well may be handily incorporated into existing web application security systems to give an extra layer of assurance against SQL infusion assaults [3], and it can likewise propel the comprehension of how profound learning models can be applied to normal language handling undertakings

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

TRADITIONAL METHODOLOGIES

Traditional methodologies for a project focused on vindictive SQL code utilizing AI-enabled regular language handling would normally include [3]:

Input Approval:

Input Approval, otherwise called information approval, is the trying of any info (or information) given by a client or application against anticipated standards [8]. Input approval forestalls noxious or inadequately qualified information from entering a data framework. Applications ought to check and approve all info that went into a framework to forestall assaults and mix-ups. Input approval is likewise significant while getting information from outside gatherings or sources [15]. Mistaken input approval can permit infusion assaults [18], memory spillage, and at last a split-the-difference framework. Input approval can involve two particular arrangements of measures for approval. These models for examination can be a permit list or a deny list. Permit list information approval is ideal over deny-posting information approval [11]. Deny-posting depends on IT staff knowing all go after that exists (what to explicitly deny) and might be utilized against your application or framework [12]. Permit records depend on IT staff understanding what orders and information types are allowed or OK, and allowing them to go through their frameworks (having frameworks dismissing Every pernicious datum checked by the channels).

Parameterized Queries:

Signature-Based Discovery:

Signature-based Discovery discovery techniques depend on predefined examples or marks of known SQL infusion assault vectors [13]. These marks are normally founded on catchphrases, punctuation,

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

or noxious payloads generally connected with SQL infusion assaults. While powerful in recognizing realized assault designs, signature-based approaches are restricted by their powerlessness to distinguish novel or jumbled assaults that don't match existing marks [10].

Syntax-Based Analysis:

Language structure-based investigation includes parsing and breaking down the grammar of SQL inquiries to distinguish expected weaknesses or dubious examples characteristic of SQL infusion [4]. This approach frequently includes the utilization of static code examination strategies to review SQL inquiries for dangerous info dealing with [2], [3], the connection of client input into SQL explanations, or other normal coding blunders that might prompt SQL infusion weaknesses.

Peculiarity Detection:

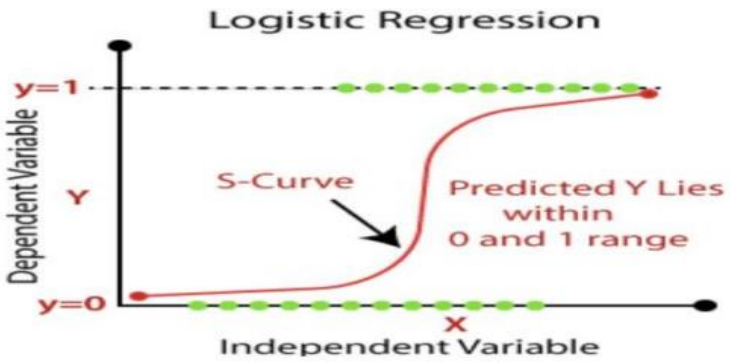
Anomaly detection in the context of malicious SQL codes involves leveraging statistical and machine-learning techniques to identify abnormal patterns in SQL queries that may indicate potential SQL injection attacks [6]. By establishing a baseline of normal query behaviour through historical data, anomaly detection algorithms can detect deviations such as unusual syntax, unexpected query structures, or atypical database operations [9],[14]. These anomalies are scored based on the deviation from the established norms, enabling security systems to prioritize and flag suspicious queries for further investigation. This proactive approach helps in mitigating the risks posed by SQL injection attacks [11], offering a dynamic defence mechanism that can adapt to evolving attack strategies and safeguard databases from unauthorized access and manipulation.

3.2 EXISTING ALGORITHM

For a venture zeroed in on foreseeing malignant SQL code utilizing Artificial intelligence-empowered normal language handling, a few existing calculations and procedures can be used. Here are some of the usually utilized algorithms:

LOGISTIC REGRESSION:

Logistic regression is a commonly used statistical method for binary classification tasks, including text classification [13]. To use logistic regression for text classification, we first need to address the message as mathematical highlights that can be utilized as Input to the model. One famous methodology for this is to utilize the pack of words portrayal [12], where we address each report as a vector of word frequencies. calculated relapse is a straightforward however compelling technique for text grouping errands and can be utilized as a gauge model or joined with additional complicated models in troupe draws near [11]. Nonetheless, it might require assistance with additional complicated connections among highlights and marks and may not catch the full scope of examples in regular language information.



Given a series of input/output pairs: (x, y)
For each observation x

- We represent x(i) by a feature vector [x1, x2 ..., xn]
- We compute an output: a predicted classy` belongs to {0,1}

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

For feature xi, weight wi tells is how important is xi Weights: one per feature W = [w1 , w2 , ..., wn] (Sometimes represented as θ)

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

If this sum is high, we say y=1; if low, then y=0

$$z = w \cdot x + b$$

SIGMOID FUNCTION

Using a sigmoid function algorithm to detect malicious SQL attacks involves leveraging the function's ability to transform input values into a probability score between 0 and 1. Here's how such an approach could be conceptualized:

Input Representation: Encode SQL queries or query components (such as keywords, operators, and structural patterns) into numerical features that can be fed into the sigmoid function [10]. This encoding captures the essence of each query in a format suitable for mathematical processing.

Sigmoid Function Application: Apply the sigmoid function to the encoded features of each SQL query. The function will transform these features into a probability score [14], where values closer to 1 indicate a higher likelihood that the query is malicious, and values closer to 0 indicate a lower likelihood[3].

Threshold Setting: Establish a threshold probability score (e.g., 0.5) above which a query is classified as potentially malicious [9]. Queries scoring above this threshold are flagged for further investigation or mitigation measures [5].

Training and Adaptation: Train the sigmoid function model using a labelled dataset of SQL queries, where each query is categorized as either benign or malicious [13]. This training process allows the function to learn the distinguishing characteristics of malicious queries and adjust its parameters accordingly.

Real-Time Detection: Implement the trained sigmoid function model in a real-time detection system [9]. As new SQL queries are processed [15], apply the function to determine the probability of each query being malicious. Automatically flag queries that exceed the predefined threshold for human review or automated response [11].

Feedback and Iteration: Continuously update and refine the sigmoid function model based on feedback from detected incidents and new data. This iterative process improves the accuracy and effectiveness of the detection algorithm over time.

```python
import math

# Example sigmoid function
def sigmoid(x):
    return 1 / (1 + math.exp(-x))

# Example SQL query feature extraction (dummy example)
def extract_features(sql_query):
    # Dummy feature extraction; replace with actual feature extraction logic
    if "DROP TABLE" in sql_query:
        return [1]
    else:
        return [0]

# Example malicious SQL detection using sigmoid function
def detect_malicious_sql(sql_query):
    features = extract_features(sql_query)
    # Compute score using some logic based on features (dummy example)
    score = sum(features)  # Simplified score calculation; replace with actual logic
    probability = sigmoid(score)
    return probability

# Example usage
sql_query = "SELECT * FROM users WHERE username='admin' OR 1=1;"
probability = detect_malicious_sql(sql_query)
print(f"Probability of SQL query being malicious: {probability}")
```

In this example:

DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

sigmoid(x) is a simplified implementation of the sigmoid function.
extract features (SQL query) is a placeholder for actual feature extraction logic that would convert a SQL query into numerical features.
Detect malicious SQL (SQL query): computes a score based on extracted features (dummy logic here) and applies the sigmoid function to determine the probability of the query being malicious.
This approach demonstrates the basic concept of using a sigmoid function for malicious SQL attack detection [11], highlighting its potential application in cybersecurity and anomaly detection systems.

4. PROPOSED SYSTEM OVERVIEW:

Gather a different dataset that incorporates both genuine client input and different sorts of SQL Infusion assaults. This dataset ought to be illustrative of the application's client base and include different assault situations. Then, we preprocess the gathered information, eliminating any commotion, superfluous data, and recognizable information that might abuse protection guidelines. After preprocessing, we perform this by designing and extricating important elements from the information that address the attributes of info examples and potential assaults [10]. The following stage is to choose a proper artificial intelligence calculation for the errand. Contingent upon the accessible information and forecast necessities, we can browse directed learning (ANN) or solo learning (like irregularity identification) [3]. With the calculations chosen, we continue to prepare the simulated intelligence model utilizing the pre-handled dataset. During this stage, we change the model's boundaries to enhance its presentation in foreseeing SQL Infusion assaults. To guarantee the model's viability and dependability [16], we assess its exhibition on a different approval dataset and tweak it further if necessary.

TEXT PREPROCESSING: Text preprocessing is a technique to clean text information and prepare it to take care of information in the model [17]. Text information contains clamour in different structures like feelings, accentuation, and text in an alternate case. At the point when we discuss Human Language then, there are various approaches to saying the same thing, and this is just the principal issue we need to manage because machines won't comprehend words [3] [15], they need numbers so we want to switch text over completely to numbers in a productive way. Strategies for text preprocessing are:

1. Grasping issue explanation: The initial step before executing any AI project is understanding the issue [9].

2. Load Information: Select the necessary Dataset for the task we are executing and afterwards transfer the Dataset [13].

3. Text lowercase: We lowercase the text to diminish the size of the jargon of our text information [17].

4. Eliminate numbers: We can either eliminate numbers or convert the numbers into their text-based portrayals. We can utilize normal articulations to eliminate the numbers [18].

5. Remove punctuation marks

We remove punctuations so that we don't have different forms of the same word. If we don't remove the punctuation, then be. been, been! will be treated separately [15].

6. Remove White space:

We can utilize the join and split capability to eliminate every one of the blank areas in a string [16].

7. Eliminate default stop words:

Stop words will be words that don't add to the significance of a sentence. Thus, they can securely be eliminated without bringing about any adjustment of the importance of the sentence [17]. The NLTK library has a bunch of stop words and we can utilize these to eliminate and prevent words from our text, and what's more, return a rundown of word tokens [11].

8. Stemming

Stemming is the method involved with getting the root type of a word. Stem or root is the part to which inflectional appends (- ed, - ize, - de, - s, and so forth) are added. The stem of a word is made

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

by eliminating the prefix or postfix of a word. [18] Thus, stemming a word may not outcome in real words. On the off chance that the text isn't in tokens, then we want to change it into tokens. After we have changed over strings of text into tokens, we can change the word tokens into their root structure. There are three calculations for stemming [19]. These are the Watchman Stemmer, the Snowball Stemmer and the Lancaster Stemmer. Watchman Stemmer is the most widely recognized among them.

5. PROPOSED ALGORITHM

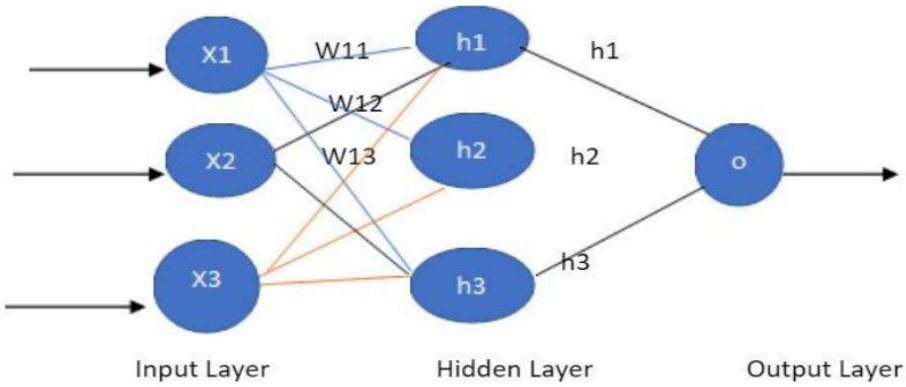
5.1 INTRODUCTION TO MLP

A multi-layer perceptron (MLP) Neural network has a place with the feedforward neural organization. Straight to the point Rosenblatt first characterized the word Perceptron in quite a while perceptron program [9]. Perceptron is a fundamental unit of a fake neural network that characterizes the counterfeit neuron in the neural network [11]. It is a regulated learning calculation containing hubs' qualities, initiation capabilities, sources of info, and loads to ascertain the result.

The Multi-Layer Perceptron (MLP) Neural Organization works just in the forward heading. All hubs are completely associated with the organization [13]. Every hub passes its worth to the approaching hub as it were in the forward course. The MLP neural network utilizes a Backpropagation calculation to increment the exactness of the preparation model.

5.2 Construction OF MLP

This organization has three fundamental layers that consolidate to shape a total Counterfeit Neural Network. These layers are as per the following:



Input Layer

It is the underlying or beginning layer of the multi-layer perceptron. It takes input from the preparation informational index and advances it to the secret layer [18], [12]. There are n input hubs in the information layer. The number of info hubs relies upon the quantity of dataset highlights. Each info vector variable is appropriated to every one of the hubs of the secret layer.

Hidden Layer

It is the core of all Counterfeit neural networks. This layer contains all calculations of the neural networks. The edges of the secret layer have loads duplicated by the hub values. This layer utilizes the enactment capability [12]. There can be a couple of stowed-away layers in the model [19]. A few secret layer hubs ought to be precise as a couple of hubs in the secret layer make the model incapable of working proficiently with complex information. More hubs will bring about an overfitting issue.

Output Layer

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

This layer gives the assessed result of the Neural networks, [11] [20]. The quantity of hubs in the Output layer relies upon the kind of issue. For a solitary designated variable, utilize one hub. N order issue, ANN involves N hubs in the result layer.

WORKING OF MULTILAYER PERCEPTRON NEURAL NETWORK •□The information hub addresses the element of the dataset.

•Each information hub passes the vector input worth to the secret layer.
•In the secret layer, each edge has some weight duplicated by the info variable. All the creation values from the secret hubs are added together. To create the result
•The enactment capability is utilized in the secret layer to distinguish the dynamic hubs.
•The result is passed to the result layer.
•Compute the distinction between anticipated and real results at the result layer.
•The model proposes backpropagation after ascertaining the anticipated result

Benefits:

High Exactness: simulated intelligence models can accomplish high precision in recognizing genuine furthermore, malevolent SQL questions [13], and diminish misleading up-sides and bogus negatives.

Constant Recognition: simulated intelligence-based forecasts can rapidly evaluate approaching inquiries, giving a quick reaction to expected dangers continuously [12].

Versatility: The model can adjust to new go-after examples and varieties, making it more strong against rising SQL infusion procedures [16].

Diminished Manual Exertion: Via robotizing the recognition cycle, security groups can zero in on other basic security undertakings, considering a more proficient utilization of assets [17].

Upgraded Security: Carrying out artificial intelligence-based expectations can altogether work on the security stance of web applications, defending delicate information and forestalling unapproved access [21].

CONCLUSION:

All in all, SQL infusion assaults represent a huge danger to web applications, possibly prompting unapproved access, information breaks, and a complete split of the difference between the application and hidden data set [19]. While conventional strategies like info approval and defined inquiries offer some degree of security, they have constraints and may not cover all assault vectors [13]. Subsequently, this work carried out ANN to proactively recognize and relieve SQL infusion assaults. It can distinguish peculiarities and gain from new assault designs [17], which empowers it to perceive complex assault vectors that conventional techniques could miss. Moreover, it can lessen misleading up-sides, give continuous security, and scale to effectively deal with huge applications. These capacities make artificial intelligence a basic device in guarding against SQL infusion assaults [21]

References:

[1] Sharma, Vishal, and Sachin Kumar. "Comparative Study of Machine Learning Algorithms for Prediction of SQL Injections." In Computer Vision and Robotics: Proceedings of CVR 2022, pp. 455-466. Singapore: Springer Nature Singapore, 2023.
[2] Zhang, Jinchuan, Yan Zhou, Binyuan Hui, Yaxin Liu, Ziming Li, and Songlin Hu. "Trojansql: Sql injection against natural language interface to database." In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 4344- 4359. 2023.
[3] Poornima, A., Kunusoth Sai Pavan, Shiva Venkata Sai Kumar, Suthari Rithika, and Gadila Vishal. "Artificial Intelligence for Prediction of SQL Injection Attack." [4] Thalji, Nisrean, Ali Raza, Mohammad Shariful Islam, Nagwan Abdel Samee, and Mona
M. Jamjoom. "AE-Net: Novel Autoencoder-Based Deep Features for SQL Injection Attack Detection." IEEE Access 11 (2023): 135507-135516.

DOI:10.46647/ijetms.2024.v08i03.070 ISSN: 2581-4621

[5] Saha, Barun Kumar, Paul Gordon, and Tore Gillbrand. "NLINQ: A natural language interface for querying network performance." Applied Intelligence 53, no. 23 (2023): 28848-28864.
[6] Zhang, Shengyuan, Yuhong Li, and Quan Jiang. "Feature Ratio Method: A Payload Feature Extraction and Detection Approach for SQL Injection Attacks." In 2023 3rd AsiaPacific Conference on Communications Technology and Computer Science (ACCTCS), pp. 172-175. IEEE, 2023.
[7] ALAzzawi, Abdulbasit. "SQL Injection Detection Using RNN Deep Learning Model." Journal of Applied Engineering and Technological Science (JAETS) 5, no. 1 (2023): 531-541.
[8] Pahuja, Vanshika, Rajat Dubey, Cybersecurity Expert, and Ishu Sharma. "Machine Learning-Based Detection and Mitigation of XML SQL Injection Attacks."
[9] Crespo-Martínez, Ignacio Samuel, Adrián Campazas-Vega, Ángel Manuel GuerreroHigueras, Virginia Riego-DelCastillo, Claudia Álvarez-Aparicio, and Camino FernándezLlamas. "SQL injection attack detection in network flow data." Computers & Security 127 (2023): 103093
[10] SIAHAAN, ELFRIDA BA, and ABBA SUGANDA GIRSANG. "IMPROVING DETECTION SQL INJECTION ATTACKS USING REFERRER HEADER AND URI WEBLOG." Journal of Theoretical and Applied Information Technology 101, no. 18 (2023)
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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 (Bigdata Security), 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.
[21] Hubskyi, O., Babenko, T., Myrutenko, L., Oksiiuk, O. (2021). Detection of SQL Injection attacks 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