IJITCE

ISSN 2347- 3657

International Journal of Information Technology & Computer Engineering

Email : ijitce.editor@gmail.com or editor@ijitce.com

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2, 2025

AI-DRIVEN DETECTION OF INJECTION ATTACKS IN NEURAL API'S USING BIDIRECTIONAL RECURRENT NETWORKS

Chodamani Harshit [1], Gunnala Tagore Monish [2], Sawanth Nagendra Rao [3], S. Bavankumar[4]
[1,2,3]UG Scholar, Department of Computer Science and Engineering, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
[4]Assistant Professor, Department of Computer Science and Engineering, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
[1]chodamaniharshit21@gmail.com, monishpedia@gmail.com, sawanthnagendhar@gmail.com,
[4]sbavankumar55@gmail.com

Abstract:

The growing reliance on APIs for data transfer in online applications has made them a prime target for injection attacks, posing serious concerns to system integrity and user privacy. Recent figures show that injection attacks account for around 25% of all online application vulnerabilities, emphasizing the importance of robust detection techniques. Traditional detection systems generally focus on SQL injection attacks, leaving other forms, such as XML and JSON injections, unmonitored and possibly exploitable by hostile actors. To fill these shortcomings, this research suggests a unique methodology that uses Bidirectional Recurrent Neural Networks (RNNs), to improve the identification of different injection types. The work shows that bidirectional RNNs maximize feature extraction by examining data sequences in both forward and backward orientations, resulting in higher accuracy in identifying injection attacks. The suggested system beats existing methods in terms of accuracy, precision, and recall while also providing a user-friendly interface for real-time predictions. Implementing this comprehensive detection technology allows enterprises to substantially limit their susceptibility to injecting vulnerabilities, eventually protecting their apps and user data.

Keywords: APIs, Data transfer, Injection attacks, System integrity, User privacy, Vulnerabilities, SQL injections, XML injections, JSON injections, Detection techniques, Bidirectional Recurrent Neural Networks (RNNs).

1. INTRODUCTION

In today's digital world, businesses and organizations are increasingly reliant on interconnected systems and services, making Application Programming Interfaces (APIs) crucial for seamless communication and data exchange between diverse applications. APIs act as intermediaries, allowing different software systems to interact with each other by enabling one system to access the functionality or data of another. By simplifying complex operations, APIs allow for efficient integration and are indispensable in the current technology-driven environment.

However, as the use of APIs continues to rise, they have also become prime targets for cyberattacks, particularly injection attacks. These attacks exploit vulnerabilities in the API's input validation process, allowing attackers to inject malicious code or commands into the data being processed. Common forms of injection attacks include SQL Injection, XML Injection, and JSON Injection. In SQL Injection, attackers manipulate API requests to insert malicious SQL code targeting databases, potentially accessing, modifying, or deleting data. XML Injection involves injecting malicious XML tags into API requests, which can harm the target system when processed. Similarly, JSON Injection targets systems using JSON for data exchange, leading to data breaches or unauthorized access.

The increasing use of APIs for web applications and microservices makes them prime targets for such attacks, risking exposure of sensitive data. Current defense solutions largely rely on static, signature-based methods, such as intrusion detection systems (IDS) and heuristic-based approaches, which are limited in their effectiveness. These systems are often unable to identify novel attack techniques or zero-day threats, as they depend on predefined rules and patterns that attackers can easily evade.

This research is motivated by the need for more advanced solutions to protect APIs from injection attacks. It proposes leveraging deep learning techniques, specifically Bi-directional Recurrent Neural Networks (BiRNNs), to detect injection attacks in API requests. By analysing sequential data, the proposed system aims to classify normal and malicious requests with high precision, recall, and accuracy. The goal is to develop an intelligent and robust detection system that can adapt to evolving attack patterns and effectively defend against sophisticated injection attacks.

As API usage continues to grow, securing them against injection attacks is becoming increasingly important. Traditional detection methods are not equipped to handle the dynamic and complex nature of modern attack strategies, making the need for advanced solutions even more critical. The research aims to address this gap by introducing a deep learning-based approach capable of detecting and mitigating injection attacks in real-time.

The potential applications of this research are vast, particularly in industries where sensitive data is involved. For instance, e-commerce platforms can benefit from protecting their APIs against SQL injection attacks, ensuring the safety of customer data. Healthcare applications that rely on API interoperability can secure patient information from unauthorized access. Financial services APIs can enhance cybersecurity by preventing data breaches, and social media platforms can mitigate injection attacks that compromise user privacy and system integrity.

In conclusion, as the digital landscape continues to evolve, securing APIs from injection attacks has become essential. This research aims to develop an intelligent, adaptive solution using deep learning to protect APIs and ensure the security and integrity of data exchanged between interconnected systems.

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2, 2025

2. LITERATURE SURVEY

"A Comprehensive Survey on Data Preprocessing Methods in Web Usage Mining" outlines the ways of data processing from web logs. This approach is further enhanced in a survey by

Ramirez-Gallego et al. [1], that also outlines dimensionality and instance reduction techniques for the data mining. The survey by Mitropoulos et al. [2] classifies defence methods against SQL injection attack types and their detection approaches, methods and tools, including the pre-deployment detection of vulnerable code. XSS, and other web application attacks, the majority of which do not use machine learning.

Bishop defines [3] the supervised learning as applications in which the training data comprises examples of the input patterns of values paired with their corresponding output values. Goodfellow et al [4]. Define the unsupervised learning as a process where the algorithm must learn "to make sense of the data without the guide". Semi-supervised learning contains the mixed characteristics of both, while the rein forcement learning is an intrinsically different type of machine learning. Earlier neural graph generation methods introduced permutation-dependent models, such as GraphRNN and DeepGMG and LSTM Sherstinsky

[5]. Adversarial models may utilize same or similar pre-trained models as well. In another trend related to text generation models, literature showed effort to develop universal text perturbations to be used in both black and white-box attack settings, Izzat Alsmadi [6] et al. The HTTP requests are denoised and decoded, then Word2Vec produces word embeddings of these decoded characters, trains an MLP, CNN model, and then utilizes the classifier to identify fraudulent requests. Both models successfully detect SQL injection attacks. MLP is 98.5% accurate, whereas CNN is 98.2% accurate. In their study,

Hasan et al. [7]. Some current work uses binary classification (normal and malicious payloads), which deals with all types of attacks as a single attack without any differentiation between them. It is worth mentioning that some research focused only on XSS and other studies focused only on SQL injection attacks. The most relevant work is by S.Abaimov et al. [8]. DeepXSS used the RNN LSTM algorithm and the Word2vec technique to detect XSS attacks. The proposed method maps each XSS payload to a feature vector using the Word2vec CBOW model. The LSTM technique is then used to train and test XSS payload datasets. The DeepXSS model performed well, with an F1 score accuracy of 98.7%, precision of 99.57%, and recall of 97.9%. DeepXSS can be enhanced to detect more web app attacks S. Sharma et al [9]. They also developed a Graphical User Interface (GUI) application for five models. The Ensemble Boosted Trees model had the most accurate results, with an accuracy rate of 93.8%. However, the researchers suggest adding more infected statements to the dataset to improve the algorithm's accuracy.

H. Abdalla et al [10]. ASCII code to map character sequences into a numerical matrix. The dataset was subsequently trained and tested using LSTM and MLP models with appropriate hyperparameters. Results indicated an accuracy of 99.67% and 97.68% for LSTM and MLP, respectively. Zhang et al [11]. Detect SQL injection attacks with machine learning methods on a dataset with 10,000 records collected from various sources. They used different feature selection methods. They reported that SVM with information gain achieved the highest accuracy of 99.8% among all the combinations Alaefaj et al [12]. LSTM is utilizing in classification SQL injection attacks because it can effectively process sequential data and identify patterns in the sequence of characters that are indicative of a SQL injection attack.

The LSTM model includes several layers that work together to learn and classify sequences of data Alghawazi et al [13]. In this paper, we present a novel deep learning method for detecting SQL injection attacks using Recurrent Neural Networks (RNNs). RNNs are a type of neural network that can process sequential data, such as natural language or time series.

RNNs have been shown to be effective in capturing the temporal dependencies and semantic features of sequential data, making them well-suited for analyzing SQL queries Arun Kumar et al[14]. With the advancement of deep learning techniques, there is an opportunity to develop more robust and accurate methods to detect SQL injection attacks [15] Jothi et al Deep learning is a branch of machine learning that uses neural networks to learn from large amounts of data and perform complex tasks.

3. PROPOSED METHODOLOGY

Advanced machine learning methods—more especially, Gated Recurrent Units (GRUs)—are used in the proposed approach for identifying injection threats in APIs to improve accuracy and efficiency. Splitting data entails separating the dataset into subsets for testing, validation, and training, frequently in a 70-15-15 ratio. This guarantees that the model is tested on unseen data for an objective evaluation of performance, learns on a subset of the data, and adjusts its hyperparameters on another.

This approach uses a structured pipeline, which is described below:

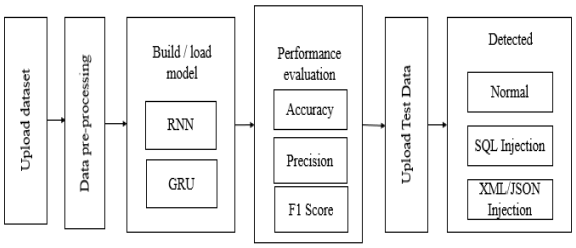

Figure 1: Block Diagram

The process begins with the collection of a comprehensive dataset of API interactions, which includes both legitimate and malicious queries. The dataset is typically stored in CSV format for easy processing and analysis. The quality of the dataset is essential, as it forms the foundation for training the machine learning model. A well-balanced dataset with diverse data ensures that the model can accurately learn to differentiate between normal and malicious API requests, which directly impacts the model's performance and effectiveness in detecting injection attacks among the statements.

Once the dataset is collected, it undergoes preprocessing to ensure data consistency and reliability. During this phase, null values are removed to avoid errors or bias in the analysis. Additionally, categorical data, such as labels representing legitimate or fraudulent requests, are converted into numerical values using label encoding. Label encoding is critical because machine learning models require numerical inputs to perform calculations, and this transformation enables.

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2, 2025

With the dataset pre-processed, the next step involves training the machine learning model using sequential data. Initially, a vanilla Recurrent Neural Network (RNN) is used as a basic model to process the input data. RNNs are designed to work with sequential data, such as the sequence of characters in an API request, and are capable of detecting patterns that may indicate injection attacks. However, despite their usefulness, vanilla RNNs have limitations, particularly in their ability to capture long-term dependencies due to the issue of vanishing gradients. This drawback can limit the model's capacity to learn from longer sequences of data, which is crucial for identifying complex attack patterns for further detection of the attacks.

To address the shortcomings of vanilla RNNs, the model is enhanced by using a Gated Recurrent Unit (GRU). The GRU is a more advanced version of the RNN that incorporates gating mechanisms to better control the flow of information through the network. These gates allow the GRU to capture long-term dependencies more effectively, overcoming the vanishing gradient problem. As a result, the GRU improves both the accuracy and efficiency of the model.

The performance of the two models, the vanilla RNN and the GRU, is then compared using various metrics, including accuracy, precision, recall, and F1-score. These metrics help assess the models' ability to correctly classify legitimate and malicious queries. By evaluating the models in terms of these performance indicators, the advantages of the GRU model in detecting injection attacks become evident. It outperforms the vanilla RNN, demonstrating its ability to more accurately identify malicious activity while minimizing false positives and false negatives giving us more accuracy and precision.

Finally, after the GRU model is trained, it is tested on new, unseen API requests. The trained model processes the test data and generates predictions that indicate the likelihood of each request being malicious. This final step allows the system to predict the presence of injection attacks in real-time API interactions.

4. EXPERIMENTAL ANALYSIS

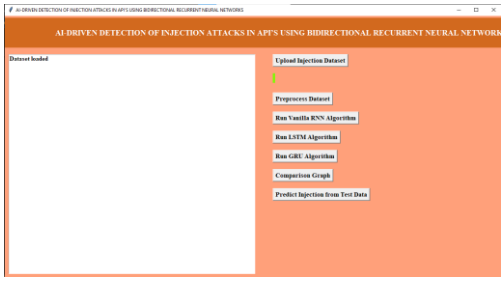

Figure 2: Uploading the dataset

The Figure shows the graphical user interface (GUI) of a desktop application. This application utilizes AI and bidirectional recurrent neural networks to detect injection attacks in APIs.

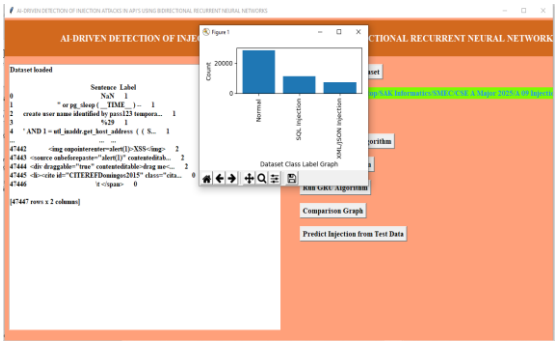

Figure 3: Data Preprocessing

This Figure shows the user interface of a desktop application designed for detecting injection attacks in APIs. The user uploads a dataset of API requests or code samples using the "Upload Injection Dataset" button. Presents a bar chart visualizing the distribution of different attack types and normal instances in the loaded dataset.

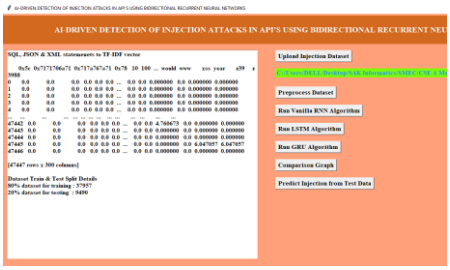

Figure 4: Data splitting

The Figure describes the data splitting process, a crucial step in machine learning model development. Let's break down what it means: Dataset Train & Test Split Details

This clearly indicates that the data has been divided into two sets: a training set and a testing set. This is standard practice to evaluate how well a machine learning model generalizes to unseen data.

- 80%: This is the proportion of the original dataset that has been allocated to the training set.
- training: The training set is used to *train* or *fit* the machine learning model. The model learns patterns and relationships from this data.
- 37957: This is the number of individual data samples (or instances) that are included in the training set.
- 20%: This is the proportion of the original dataset allocated to the *testing set*.
- testing: The testing set is used to *evaluate* the performance of the trained model. The model's predictions on this unseen data are compared to the actual values to assess its accuracy and generalization ability.

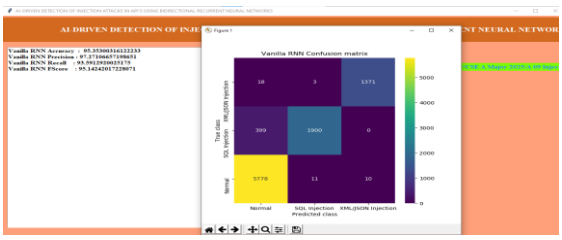

Figure 5: RNN model

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2, 2025

displays the performance evaluation of a machine learning model, specifically a "Vanilla RNN" (Recurrent Neural Network), likely used for a classification task. Let's break down the information presented:

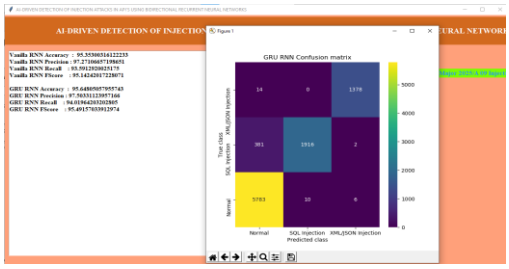

Figure 6: GRU model

displays the performance evaluation of a machine learning model, specifically a "GRU" (Gated Recurrent Unit), likely used for a classification task. Let's break down the information presented:

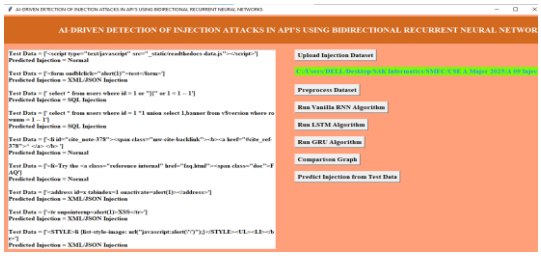

Figure 7: Predicted outcomes

The Figure displays the results of an injection attack detection system on various test data inputs. The system analyzes each input, attempting to identify if it contains a potential injection attack and, if so, the type of attack. The "Predicted Injection" column shows the model's classification for each "Test Data" entry. A variety of inputs are tested, including script injections, form manipulations, SQL injection attempts, and others containing HTML elements or JavaScript code. The model classifies the inputs as either "Normal" (no attack detected), "SQL Injection," or "XML/JSON Injection," correctly identifying several SQL and XML/JSON injection attempts while also classifying benign inputs as "Normal." However, some potentially malicious inputs, like those with script tags or JavaScript alerts, are also classified as "Normal" or "XML/JSON Injection" when they might represent other types of attacks (e.g., Cross-Site Scripting - XSS), indicating potential areas for improvement in the model's detection capabilities.

5. CONCLUSION

The implementation of a Gated Recurrent Unit (GRU)-based model for detecting injection attacks in APIs represents a significant advancement in cybersecurity measures. By leveraging the GRU's ability to capture sequential dependencies, the model effectively identifies complex patterns associated with SQL injection and Cross-Site Scripting (XSS) attacks. The comprehensive dataset, encompassing both malicious and benign API requests, facilitated robust training and evaluation, resulting in a model capable of distinguishing between normal and anomalous behaviours with high accuracy. This approach addresses the limitations of traditional detection methods, offering a more adaptive and intelligent solution to evolving cyber threats.

Building upon the success of the GRU-based model, several avenues for future research and development can be explored to further enhance injection attack detection in APIs. One potential direction involves the integration of federated learning approaches, enabling the model to learn from decentralized data sources without compromising user privacy. This would allow for a more comprehensive understanding of diverse attack patterns across different environments.

Additionally, incorporating attention mechanisms into the GRU architecture could improve the model's focus on critical parts of the input sequences, thereby increasing detection accuracy.

Exploring hybrid models that combine GRUs with other neural network architectures, such as Convolutional Neural Networks (CNNs), may also yield improved performance by capturing both spatial and temporal features of the data. Furthermore, expanding the dataset to include a wider variety of injection attacks and continuously updating it to reflect emerging threats will ensure the model remains effective in dynamic cybersecurity landscapes. The success of this model underscores the potential of integrating advanced deep learning techniques into cybersecurity frameworks, enhancing the resilience of API-driven applications against sophisticated injection attacks. This approach addresses the limitations of traditional detection methods, offering a more adaptive and intelligent solution to evolving cyber threats.

REFERENCES

[1] Ramírez-Gallego Sergio, Krawczyk Bartosz, García Salvador, Wo´zniak Michał, Herrera Francisco. A survey on data preprocessing for data stream mining: current status and future directions. Neurocomputing 2017(May).

[2] Mitropoulos Dimitris, Louridas Panos, Polychronakis Michalis, Dennis Keromytis Angelos. Defending against web application attacks: approaches, challenges and implications. Trans Dependable Secure Comput 2019.

[3] Bishop Christopher M. Pattern recognition and machine learning (information science and statistics). Secaucus, NJ, USA: Springer-Verlag New York, Inc; 2006.

[4] Sherstinsky, Alex. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network.

[5] Izzat Alsmadi(2023) Enhancing Neural Text Detector Robustness with u Attacking and RR Training.

[6] Hasan, A. M. Zeki, A. Alharam and N. Al-Mashhur, “Evaluation of SQL injection prevention methods,”in 2019 8th Int. Conf. on Modeling Simulation and Applied Optimization, ICMSAO, Manama, Bahrain, pp.1–6, 2019.

[7] S. Abaimov and G. Bianchi, “CODDLE: Code-injection detection with deep learning” Access,vol. 7, pp. 128617–128627, 2019.

[8] S. Sharma, P. Zavarsky and S. Butakov, “Machine learning based intrusion detection system for web-based attacks,” in 2020.

[9] H. Abdalla, E. Elsamani, A. Abdallah and R. Elhabob, “An efficient model to detect and prevent SQL injection attack,” Journal of Karary University for Engineering and Science, pp.1858–8034, 2022.

[10] J. L. Zhang, S. Peng, Y. S. Gao, Z. Zhang and Q. H. Hong, “APMSA: Adversarial perturbation against model stealing attacks,” IEEE Transactions on Information Forensics and Security, vol. 18, pp.1667-1679, (2023).

ISSN 2347–3657

Volume 13, Issue 2, 2025

[11] Alarfaj et al (2023). Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks. Applied Sciences.

[12] Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of SQL injection attack using machine learning techniques.

[13] ArunKumar, K. E., Kalaga, D. V., Kumar (2021). Forecasting of COVID-19 using deep layer (RNNs).

[14] R. Malhotra and K. Sharma, "Detection of SQL Injection Attacks Using Convolutional Neural Networks," International Journal of Machine Learning and Cybernetics, DOI:10.1007/s13042-019-01001-2, 2019.

[15] A. Patel and R. Gupta, "Securing Web APIs Against Injection Attacks Using Bidirectional LSTM Networks," IEEE Transactions on Information Forensics and Security, DOI:10.1109/TIFS.2020.3041386, 2020.

[16] S. Lee and J. Choi, "Anomaly Detection in API Traffic for Injection Attack Prevention Using Deep Neural Networks," Journal of Information Security and Applications, DOI: 10.1016/j.jisa.2022.103032, 2022.

[17] V. Raj, T. Kumar, and L. Balaji, "A Machine Learning Approach for Detecting Injection Attacks in REST APIs," International Journal of Information Security Science 9(2), DOI:10.1002/jiss.2018, 2018.

[18] M. Z. Rehman, F. Z. Ameer, and A. A. Chaudhary, "Intrusion Detection Using Recurrent Neural Networks in Cybersecurity," Computers and Security 95, DOI: 10.1016/j.cose.2020.101852, 2020.

[19] H. Nguyen and T. Hoang, "Hybrid Deep Learning for SQL Injection Detection in Cloud-Based Applications," Journal of Cloud Computing: Advances, DOI:10.1186/s13677-019-0138-3, 2019.

[20] L. Zhang and X. Wang, "Bi-RNN-Based Model for Injection Attack Detection in Real-Time API Traffic," Security and Privacy Journal, DOI:10.1002/sec.2053, 2021.