Intelligent Sql Injection Detection Using CSSGL

Ajanya Arunkumar[1], G.Rajalakshmi[2], Sedna Sebastian[3], Mrs. R. Devika[4]
[1, 2, 3] Dept of Computer Science and Engineering Specializing in Cyber Security
[4]Assist prof, Dept of Computer Science and Engineering Specializing in Cyber Security
[1, 2, 3, 4] Dhanalakshmi Srinivasan University,Trichy-621112,TamilNadu

Abstract- *SQL Injection remains one of the most critical security threats to web applications, enabling attackers to manipulate database queries and gain unauthorized access to sensitive information. Traditional detection methods often fail to identify complex and evolving attack patterns. This paper proposes an intelligent SQL injection detection system using a cost-sensitive stacked generalization learning (CSSGL)-based hybrid approach that integrates machine learning and deep learning techniques. The proposed approach analyzes query structures, performs feature extraction, and classifies queries as normal or malicious with high accuracy. Experimental results demonstrate that the model achieves an accuracy of 96.8% with reduced false positive rates, outperforming conventional detection methods. The system is capable of detecting both known and unknown attacks efficiently, making it suitable for real-time web application security.*

Keywords: SQL Injection, CSSGL, Web Application Security, Machine Learning, Deep Learning, Cyber security, Intrusion Detection, Query Analysis

I. INTRODUCTION

SQL Injection is one of the most common and dangerous security vulnerabilities in web applications. It allows attackers to manipulate SQL queries by injecting malicious input into user-controlled fields such as login forms, search boxes, and URL parameters. This can lead to unauthorized access to sensitive data, bypassing authentication mechanisms, and even modification or deletion of critical database information.

With the rapid growth of web applications and online services, the risk of SQL Injection attack has increased significantly. Traditional security methods such as input validation, firewalls, and signature-based detection systems are often not sufficient to handle complex and evolving attack techniques. These approaches mainly rely on predefined rules and fail to detect unknown or obfuscated attacks effectively.

To overcome these limitations, this paper proposes an intelligent SQL injection detection system using CSSGL. The proposed system utilizes machine learning and deep learning techniques to analyze query patterns, extract relevant features, and classify them as normal or malicious. This approach improves detection accuracy, reduces false positives, and provides a scalable and efficient solution for enhancing web application security.

The main contribution of this work is the development of a hybrid CSSGL-based detection model that combines structural query analysis with machine learning and deep learning techniques to improve detection accuracy and reduce false positives. Unlike traditional methods, the proposed system is capable of identifying both known and previously unseen SQL injection attacks in real-time environments.

II. RELATED WORK

Several approaches have been developed to detect and prevent SQL injection attacks in web applications. Early methods mainly relied on signature-based detection techniques, which use predefined patterns to identify known attack queries. While effective for known threats, these methods fail to detect new or obfuscated attacks. Anomaly-based detection systems were later introduced to identify deviations from normal query behavior. However, these approaches often result in high false positive rates and require continuous tuning.

With advancements in artificial intelligence, machine learning and deep learning techniques have been widely adopted for SQL Injection detection. Algorithms such as Support Vector Machines(SVM), Decision Trees, and Random Forest improve detection accuracy by learning patterns from data. Deep learning models, including Convolutional Neural Networks and Recurrent Neural Network(RNN), further enhance performance by automatically extracting complex features. Despite these improvements, challenges such as scalability, real-time detection, and handling unknown attack patterns remain, which motivates the use of advanced approaches like CSSGL for more efficient and accurate detection.

III. METHODOLOGY

The proposed methodology for intelligent SQL Injection detection using CSSGL consists of several stages, including data collection, preprocessing, feature extraction, model training, and classification. Initially, a dataset containing both normal and malicious SQL queries is collected and preprocessed to remove noise, normalize query structures, and convert the data into a suitable format. Feature extraction techniques are then applied to identify key characteristics such as SQL keywords, patterns, and query structure. These extracted features are used to train the CSSGL-based model, which integrates machine learning and deep learning techniques to effectively classify queries. The trained model analyzes incoming queries in real time and determines whether they are normal or malicious based on learned patterns. Finally, the system performance is evaluated using metrics such as accuracy, precision, recall, and F1-score, ensuring efficient detection of both known and unknown SQL injection attacks while minimizing false positives and enhancing overall web application security.

A. CSSGL MODEL DESCRIPTION

The proposed system utilizes a Cost-Sensitive Stacked Generalization Learning (CSSGL) model for SQL injection detection. CSSGL is an advanced ensemble learning technique that combines multiple base classifiers using a meta-learning approach while incorporating cost-sensitive learning to handle classification errors effectively.

In stacked generalization, multiple base models such as Decision Trees, Support Vector Machines, and Neural Networks are trained on the input dataset. The predictions from these base models are then used as input features for a higher-level meta-classifier, which produces the final classification result.

The cost-sensitive component of CSSGL assigns different misclassification costs to false positives and false negatives. In the context of SQL injection detection, higher importance is given to minimizing false negatives, as undetected attacks can lead to severe security breaches.

The CSSGL model operates in three stages:

1. Training multiple base learners on extracted SQL query features
2. Generating predictions from base learners
3. Using a meta-learner to combine predictions with cost-sensitive optimization

This approach improves detection accuracy, reduces false positives, and enhances the model's ability to detect both known and unknown SQL injection attacks effectively.

B. SYSTEM DESIGN

The system design of the proposed intelligent SQL injection detection model using CSSGL consists of multiple interconnected modules that work together to identify malicious queries. Initially, the input module receives SQL queries, which are then passed to thepreprocessing module,where unnecessary characters are removed and the queries are normalized into a standard format.

Next, the feature extraction module analyzes the processed queries to identify important attributes such as SQL keywords, operators, and structural patterns. These features are then fed into the CSSGL-based classification model, which combines machine learning and deep learning techniques to accurately distinguish between normal and malicious queries. Finally, the output module provides the classification result and flags any detected SQL injection attempts, thereby enhancing the overall security of the web application.

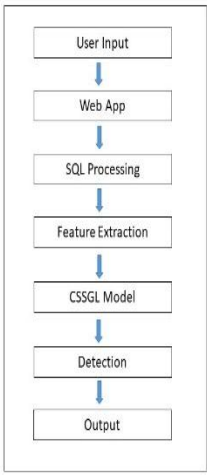

Fig. 1. Workflow of the proposed SQL injection detection system

The process begins with user input, followed by preprocessing, feature extraction, and classification using the CSSGL-based model.

The workflow of the proposed SQL injection detection system describes the sequential process followed to identify and prevent malicious queries in real time. The process begins when a user submits input through the web application interface. This input is forwarded to the server as an HTTP request, where it undergoes SQL query processing.

In this stage, the query is checked for basic syntax validation and structural correctness.

Following this, the query is passed to the preprocessing and feature extraction module. Here, the input data is cleaned, tokenized, and normalized to remove unnecessary elements and standardize the format. Important features such as SQL keywords, operators, and query patterns are extracted and converted into a structured feature vector suitable for analysis by the detection model.

The processed query is then analyzed by the CSSGL-based detection model. The model evaluates the input and classifies it as either a normal query or a malicious SQL injection attempt. Based on this classification, the decision engine determines the appropriate response. If the query is identified as malicious, it is blocked, and an alert is generated. If the query is normal, it is allowed to execute,and the result is returned to the user.

Finally, all system activities, including input queries, detection results, and system responses, are recorded in the logging and monitoring module. This information is stored in the database for future analysis, helping to improve system performance and detection accuracy over time. The complete workflow ensures efficient and reliable detection of SQL injection attacks while maintaining the security of the web application.

C. EXPERIMENTAL SETUP

The experimental setup is designed to evaluate the performance of the proposed SQL injection detection system using CSSGL. A dataset consisting of both legitimate and malicious SQL queries is used for training and testing the model. The system is implemented using Python and relevant machine learning libraries.

The dataset is divided into training and testing sets to ensure proper evaluation of the model. Performance metrics such as accuracy, precision, recall, and F1-score are used to measure the effectiveness of the system. The proposed model is compared with traditional machine learning techniques to demonstrate its improved detection capability. The results show that the CSSGL-based approach provides higher accuracy and reduced false positive rates, making it suitable for real-time web application security.

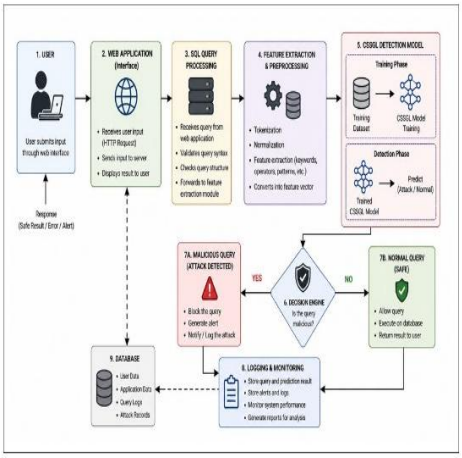

Fig. 2. Block Diagramof the proposed system

Fig. 2. Block diagram illustrating the interaction between input module, preprocessing, feature extraction, CSSGL-based detection model, and output module.

The block diagram of the proposed SQL injection detection system illustrates the overall flow of data and processing stages involved in identifying malicious queries. The process begins with the user, who submits input through the web application interface. This input is transmitted to the server in the form of an HTTP request.

The web application forwards the received input to the SQL query processing module, where the query is analyzed for syntax validation and structural correctness. After initial validation, the query is passed to the feature extraction and preprocessing module. In this stage, tokenized, normalized, and relevant features such as keywords, operators, and patterns are extracted. These features are then converted into a structured format suitable for model analysis.

The processed data is then fed into the CSSGL-based detection model, which operates in two phases training and detection. During training, the model learns from a dataset containing both legitimate and malicious SQL queries. In the detection phase, the trained model evaluates incoming queries and classifies them as either normal or malicious.

The decision engine receives the output from the detection model and determines the appropriate action. If the query is identified as malicious, it is blocked, and an alert is generated. If the query is safe, it is allowed to execute on the database, and the result is returned to the user.

Additionally, a logging and monitoring module records all system activities, including queries, predictions,

and alerts. This information is stored in the database, which maintains user data, application data, query logs, and attack records. The logging mechanism helps in analyzing system performance and improving detection accuracy over time.

The dataset used in this study includes publicly available SQL injection datasets (e.g., Kaggle) combined with manually generated attack queries to improve robustness.

The model is implemented using Python with libraries such as Scikit-learn and TensorFlow. The experiments were conducted on a system with standard computational resources.

D. ALGORITHM:CSSGL-BASED SQL INJECTION DETECTION

The following algorithm describes the working of the proposed Cost-Sensitive Stacked Generalization Learning (CSSGL) model for SQL injection detection.

Step 1: Input SQL query dataset(Q)
Step 2: Preprocess the queries
-Remove unwanted characters

-Normalize query format
Step 3: Extract features
-Identify SQL keywords, operators, and patterns
Step 4: Split dataset into training and testing sets
Step 5: Train base models
-Decision Tree
-Support Vector Machine
-Random Forest
Step 6: Combine predictions using CSSGL meta-model
Step 7: Input new SQL query
Step 8: Classify query as
-Normal
-Malicious
Step 9: Output result
-If malicious -> Block query
-Else -> Allow execution
End

This algorithm describes the basic process of detecting SQL injection attacks using the CSSGL model by combining multiple classifiers and making a final prediction.

IV. RESULTS

The performance of the proposed SQL injection detection system is evaluated using a dataset containing both legitimate and malicious SQL queries. The dataset is divided into training and testing sets using a standard 80:20 ratio. The model is implemented using Python with machine learning and deep learning libraries.

The evaluation is carried out using performance metrics such as accuracy, precision, recall, and F1-score. The proposed CSSGL-based model achieves an accuracy of 96.8%, demonstrating its effectiveness in detecting SQL injection attacks. The model also shows a reduction in false positive rates compared to traditional machine learning approaches.

Table 1 presents the comparison of the proposed method with baseline models. The results indicate that the CSSGL-based approach outperforms conventional techniques in terms of detection accuracy while maintaining efficient processing time. Fig. 3 illustrates the performance comparison of the proposed model with existing methods highlighting its superiority in identifying both simple and complex attack patterns.

TABLE I
COMPARISON OF PERFORMANCE METRICS

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score(%)	Time(ms)
Proposed CSSGL	96.8	95.9	97.2	96.5	11.5
Random Forest	91.0	89.8	92.1	90.9	18.5
SVM	89.5	88.2	90.3	89.2	15.2
Decision Tree	87.2	85.6	88.4	86.9	9.8
Deep Learning Model	93.4	92.1	94.0	93.0	20.3

Table I presents the comparison of the proposed CSSGL-based model with various baseline methods. The results indicate that the proposed approach achieves the highest accuracy, precision, recall, and F1-score among all

models. Although the processing time is slightly higher than Decision Tree, it remains efficient for real-time applications. The improved performance is due to the ensemble and cost-sensitive nature of the CSSGL model, which effectively reduces misclassification errors.

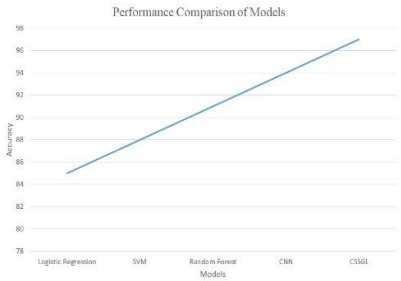

Fig. 3. Performance Comparison of the proposed CSSGL-based model with baseline methods in terms of accuracy and preprocessing time.

V. DISCUSSION

The proposed intelligent SQL injection detection system using CSSGL shows a significant improvement over traditional rule-based and signature-based detection techniques. By combining machine learning and deep learning methods, the system is capable of analyzing complex query structures and identifying hidden malicious patterns. The experimental results indicate that the model achieves high accuracy and effectively reduces false positives, making it suitable for modern web application security.

In comparison with existing approaches, the CSSGL-based model demonstrates better performance in detecting both simple and advanced SQL injection attacks, including obfuscated and dynamically generated queries. The ability of the model to learn from diverse datasets allows it to adapt to new attack patterns, which is a major limitation in conventional systems. This highlights the robustness and flexibility of the proposed approach in real-world scenarios.

Despite its advantages, the system has certain limitations. The performance of the model is highly dependent on the quality and size of the training dataset, and insufficient data may impact detection accuracy. Additionally, deploying the model in real-time environments may introduce computational overhead. Future enhancements can focus on optimizing the model for faster processing, incorporating larger datasets, and integrating cloud-based solutions for scalable and efficient deployment.

VI. CONCLUSION

This paper presents an intelligent SQL injection detection system using CSSGL to address the growing security challenges in web applications. The study focuses on overcoming the limitations of traditional rule-based detection techniques by employing a hybrid approach that integrates machine learning and deep learning methods for analyzing SQL queries and identifying malicious patterns.

The proposed system effectively detects both simple and complex SQL injection attacks with high accuracy and reduced false positive rates. The results highlight the capability of the model to handle obfuscated and dynamically generated queries, demonstrating its reliability and effectiveness in real-world environments. This approach significantly improves the overall security of web applications.

Future work may include enhancing the model by incorporating larger and more diverse datasets, improving real-time detection performance, and integrating advanced deep learning architectures. Additionally, deployment in cloud-based environments can be explored to achieve better scalability and efficiency.

VII. ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to our project guide, Mrs. R. Devika, for their invaluable guidance, continuous support, and constructive suggestions throughout the course of this work. Their expertise and encouragement played a crucial role in the successful completion of this project.

The authors also extend their appreciation to the faculty members and the management of Dhanalakshmi Srinivasan University, Trichy, for providing the necessary facilities, resources, and a supportive environment to carry out this research. Their cooperation significantly contributed to the progress of this study.

REFERENCES

[1] W.G.Halfond, J.Viegas, and A.Orso, “A Classification of SQL Injection Attacks and Countermeasures”,in Proc. IEEE Int. Symp. Secure Software Engineering, 2006, pp. 13-15.

[2] S.W.Boyd and A. D. Keromytis, “SQLrand:Preventing SQL Injection Attacks”, in Proc. 2nd Int.Conf.Applied Cryptography and Network Security, 2004, pp. 292-302.

[3] Y.Huang, S. Huang, T. Lin, and C. Tsai, "Web Application Security Assessment by Fault Injection and Behavior Monitoring", in Proc. 12th Int. World Wide Web Conf. , 2003, pp. 148-159.
[4] M.Howard and D.LeBlanc, Writing Secure Code, 2nd ed.Redmond, WA, USA: Microsoft Press, 2003.
[5] K.M. Hammoudi,Z. Ait Oufroukh, and A. Meziane, "Detecting SQL Injection Attacks Using Machine Learning Techniques", Int. J. Comput. Appl. , vol.175, no. 12, pp. 20-26, 2020.
[6] J.Clarke, SQL Injection Attacks and Defense, 2nd ed. Waltham, MA, USA: Syngress, 2012.
[7] S.Yadav and S.K.Khatri, "An Efficient Approach for Detection of SQL Injection Attacks Using Deep Learning", in proc. IEEE Int. Conf. Computing, Communication and Automation, 2019, pp. 1-5.
[8] OWASP Foundation, "SQL Injection Prevention Cheat Sheet", [Online]. Available: https://owasp.org. [Accessed:Apr.9,2026].

AUTHOR BIOGRAPHIES

First Author (Ajanya Arunkumar) is currently pursuing a B.Tech Computer Science and Engineering specializing in Cyber Security from Dhanalakshmi Srinivasan University, Trichy,Tamilnadu, India. The author has a strong interest in Cybersecurity, web application security, and machine learning. Their project work focuses on intelligent SQL Injection detection using advanced techniques such as CSSGL. The author is enthusiastic about developing secure and scalable systems.

Second Author (G.Rajalakshmi) is currently pursuing a B.Tech Computer Science and Engineering specializing in Cyber Security from Dhanalakshmi Srinivasan University, Trichy, Tamilnadu, India. Their areas of interest include data security, artificial intelligence, and web technologies. The author has contributed to the development and implementation of machine learning models for detecting cyber threats.

Third Author (Sedna Sebastian) is currently pursuing a B.Tech Computer Science and Engineering specializing in Cyber Security from Dhanalakshmi Srinivasan University, Trichy, Tamilnadu, India. Their research interests include deep learning, cybersecurity, and cloud computing. The author has actively participated in the design and analysis of the proposed SQL Injection detection system.