Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

A Bio-Inspired Algorithm for Zero-Day SQL Injection Detection via Slime Mould Classifier

Ahmed Ibrahim Abdulameer *, Hasanen Alyasiri

Department of Computer Science, Faculty of Computer Science and Mathematics, University of Kufa, Najaf, Iraq
ahmedi.altufaily@student.uokufa.edu.iq

https://doi.org/10.46649/fjiece.v4.2.7a.23.9.2025

Abstract. *With the continuous development of the cybersecurity field, detecting unknown or zero-day attacks remains a significant challenge due to their unpredictable behavior. This paper proposes a novel biologically inspired approach to detecting unknown SQL Injection (SQLi) attacks using the Slime Mould Algorithm (SMA). This method leverages the adaptive and heuristic capabilities of the SMA to detect unknown attacks. A binary classification model was developed and trained on a benchmark dataset containing both natural queries and diverse SQLi attack vectors, including Out-of-Band, Boolean-based, Time-based, and Union-based injections. To ensure robustness and generalization, K-Fold cross-validation was employed. The SMA-based model demonstrated superior detection capabilities, particularly in identifying zero-day attacks that deviate from known patterns. The experimental results reveal promising detection rates across all attack types: 89.33% for Out-of-Band-based SQLi, 97.89% for Boolean-based SQLi, 90.27% for Time-based SQLi, and 96.69% for Union-based SQLi, and 91.51% for Error-based SQLi. These results underline the effectiveness of SMA in generalizing beyond seen data, a critical advantage in dynamic threat environments. Compared to traditional machine learning models, the SMA-based classifier achieved higher accuracy and F1 scores, confirming its potential as a powerful tool for web application security.*

Keywords: Slime Mould Algorithm, Zero-Day Attack, SQL Injection, Web Security

1. INTRODUCTION

Cybersecurity has become a critical concern with the rapid development of web-based technologies. Among the most dangerous threats are zero-day attacks, which exploit previously unknown vulnerabilities. Many web applications collect users' personal information and interact with users. So, they always connect to the database. Due to the large amount of valuable data stored in the database, it naturally becomes the target of attackers, so there are more and more SQLi attacks [1]. SQLi is a type of web attack that exploits a database query vulnerability to access and manipulate sensitive and important data [2]. A recent security report highlighted that 6.7% of all vulnerabilities discovered in open-source projects in 2024 are SQLi vulnerabilities, while 10% of vulnerabilities in closed-source projects were SQLi-related [3]. Despite improvements, the total number of SQLi vulnerabilities found in open-source projects is expected to increase from 2,264 in 2023 to over 2,400 by the end of 2024 [3]. According to the Web Attack Trend Report, SQLi remains one of the most common types of web attacks where it accounting for 37.36% of all detected web attacks during the monitored period. These figures underscore the persistent threat posed by SQLi and the urgent need for intrusion detection systems capable of accurately detecting and mitigating these attacks in real-time [4].

SQLi attack refers to the construction of special strings as parameters to be transmitted to web applications by submitting web forms or inputting query strings of domain names or page requests. These special strings often contain some executable statements in the SQL grammar, which make web applications mistake data as code to execute and ultimately deceive servers to execute malicious SQL commands. The main reason is that the web application does not filter the users' input data accurately, so the database is invaded. Existing methods for identifying SQLi attacks include regular matching [1]. Various detection and prevention techniques, such as stored procedures, input validation, whitelisting, parameterized queries, and Artificial Intelligence (AI), have been proposed to tackle SQLi threats [6][7]. AI techniques have been successfully applied to develop intelligent defense mechanisms across diverse domains, such as network security [8], mitigation of cross-site scripting attacks [9], and safeguarding mobile platforms [10]. Furthermore, machine learning (ML) methods have been tested and utilized to detect SQLi attacks, and the results are promising [6]. Bio-inspired algorithms, a subset of ML techniques, have been successfully employed to address optimization challenges across diverse domains, including cybersecurity [6].

In this paper, a bio-inspired algorithm named Slime Mould Algorithm (SMA) is used. This algorithm was introduced in 2020 [4]. SMA mimics the foraging behavior of Physarum polycephalum, a slime mould known for its efficient network formation. The algorithm mimics the organism's oscillation-based search mechanism, where virtual "slime agents" explore the solution space by dynamically adjusting their locations based on food quality (fitness). High-quality solutions draw more slime agents to them, leading to more focused searching in that area, while poorer solutions cause the agents to spread out and search more widely to keep moving forward. SMA balances exploration and exploitation through adaptive weight updates and stochastic components, mimicking the Mould's natural feedback system. SMA is known for optimization but has not been explored for classification. To our knowledge, this is the first time it has been used as a classifier. Our main contribution is to train SMA to detect zero-day attacks. However, most existing ML- and DL-based detection systems rely on static or signature-dependent models, which fail to identify novel or zero-day SQL injection patterns. Therefore, this study aims to develop a bio-inspired intrusion detection model based on the SMA that can dynamically adapt to unseen SQLi behaviors. The research addresses the gap in zero-day attack detection by leveraging the SMA’s adaptive exploration–exploitation mechanism to enhance detection accuracy and robustness.

This research details: Section 1 provides an introduction, while Section 2 describes related work. Section 3 defines SQLi and discusses its types. Section 4 explains the algorithm used in this research, detailing the dataset, feature extraction, and performance metrics. Section 5 presents the results for detecting unknown attacks and compares them with recent research on detecting unknown attacks. Section 6 presents the research conclusions and future directions.

2. RELATED WORK

Detecting zero-day attacks remains a significant challenge due to the evolving nature of cyber threats and the limitations of signature-based systems. Several researchers have explored intelligent and adaptive solutions

Cumi-Guzman et al. [11] developed a Random Forest-based classification model for detecting SQL Injection (SQLi) attacks, achieving a notable accuracy of 97.3%. The model was trained on a curated

dataset of 22,931 SQL statements using 82 syntactic and semantic features. While the classifier demonstrated strong generalization and interpretability, especially in identifying critical SQL elements linked to malicious activity, it showed vulnerability to overfitting due to the high dimensionality of the feature space.

Rosca et al. [12] proposed a ML architecture for SQLi attack detection that integrates syntactic normalization with semantic feature extraction. The integration allows for a more accurate attack detection. The study involved Azure ML Studio for 15 models and sampling combinations on a 90,000 SQL queries dataset, which consisted of normal and malicious queries. The Voting Ensemble, which is the most accurate model of their study, achieved excellent results, with 96.86% overall accuracy, a weighted F1-score of 96.77%, and a weighted AUC of 98.25%, demonstrating impressive classification proficiency. Ablation analysis showed that the feature named query_length was a necessary condition for the model to work, as system accuracy dropped to around 80.54% without it. Their results effectively argue the SQLi detection attack reliability of their lightweight interpretive feature.

Maha et al. [13] the author proposes an architecture for detecting SQLi attacks using a recurrent neural network (RNN) auto encoder model. The experimental results show that the proposed approach achieved an accuracy of 94% and an F1-score of 92%, outperforming various other machine learning models like ANN, CNN, DT, NB, SVM, RF, and LR

M. Shahbaz et al. [14] used a CNN-based model achieving 98.16% accuracy, but their research lacks comparisons with other deep learning models, and deep learning models require large computational resources, making them unsuitable for limited computational resources.

Jarudat et al. [15], four distinct machine learning techniques, namely gradient boosting (GB), MLP, LR, and k-nearest neighbor (KNN), were used to improve model performance and identify the most effective configuration. The tree-based pipeline optimizer (TPOT) and genetic algorithm (GA) were used. The dataset provided by the Canadian Institute 2023, which includes various types of attacks, served as the basis for testing the model. Notably, the accuracy values achieved by GB for precision, recall, and F1 score were 95%, 94%, and 95%, respectively.

Lu and Traore [16] were among the early pioneers to apply Genetic Programming (GP) for intrusion detection, successfully evolving rules to identify previously unseen variants of DoS attacks in the DARPA 1999 dataset.

Blasco et al. [17] guided the GP evolution process using advanced IDS evaluation metrics, enhancing its effectiveness in zero-day detection.

Alyasiri et al. [8] evaluated three evolutionary computing techniques, namely GP, GE, and CGP to detect known and unknown cyberattacks for web and network attacks. By removing specific attack types from the training phase, their system simulated zero-day scenarios and demonstrated strong generalization, with CGP achieving the highest detection rate for unknown attacks. These results confirm the effectiveness of evolutionary models in identifying previously unseen threats across multiple datasets..

Waheed and Alyasiri [18] proposed an evolutionary learning-based method for Android malware detection using the evtree algorithm. Their model was trained and tested on the CICMalDroid2020 dataset, achieving 99.11% detection accuracy, 96.14% precision, and an F1-score of 97.60%. The results showed strong performance in identifying unknown malware excluded from training, validating its suitability for detecting zero-day threats.

Kamarudin et al. [19] proposed a hybrid intrusion detection system using LogitBoost combined with Random Forest to identify both known and unknown web attacks. The system was evaluated using the NSL-KDD and UNSW-NB15 datasets, achieving an accuracy in detecting new attack patterns. The model's ability to identify anomalies and zero-day attacks was validated by high precision and recall rates on the test datasets.

In contrast to previously mentioned works, this study introduces SMA as a nature-inspired optimization technique for detecting unknown SQL injection attacks. Using the SMA, the proposed system improves the decision boundary and enhances the detection accuracy of previously undetected threats. Experimental results confirm the model's ability to generalize effectively, outperforming many existing approaches.

3. BACKGROUND ON SQL INJECTION ATTACKS AND TYPES

Web-based applications typically adhere to a three-tier architecture: a presentation tier for the user interface, a business tier for logical operations, and a data tier for data management. It retains all the structured data. A SQLi attack capitalizes on weaknesses across all three tiers of a system to execute a successful breach [20]. Malicious SQL instructions, delivered from the presentation layer to the business tier, alter existing SQL queries, leveraging the database tier to access resources. The lack of validation at both the presentation and operational tiers of web applications facilitates a successful SQLi attack [21].

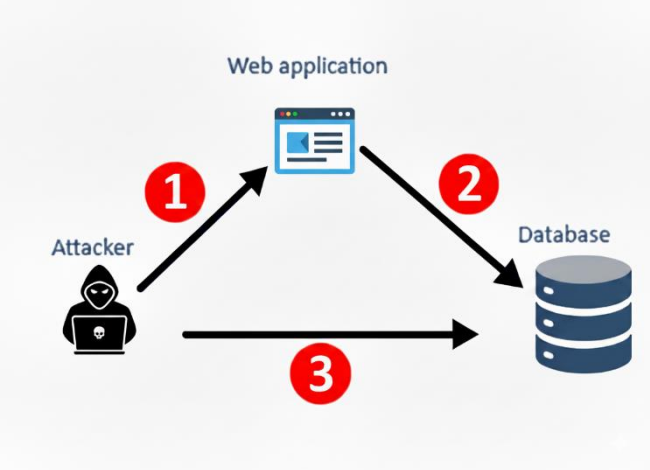

Figure (1) SQLi Attack Workflow

3.1. Types of SQL Injection

SQLi can be employed in several ways to induce significant issues. Through the utilization of SQLi, a malicious individual could circumvent the authentication process, get unauthorized entry, and manipulate, and eradicate data stored within a database. SQLi can, in certain instances, facilitate the execution of commands on the operating system, hence enabling an attacker to possibly escalate to more destructive operations within a system. The backend network is protected by a firewall. SQL Injection can be categorized into three primary classifications -In-band SQLi, Inferential SQLi Additionally, there is a type of SQL injection called Out-of-band SQLi [22].

Union-based SQLi is a method of SQLi that exploits the UNION SQL operator to merge the outcomes of many SELECT statements into one result, which is subsequently included in the HTTP response [**23**].

The term blind means that the SQLi is performed when the programmer has set a generic custom error message in case web application encounters an error [**24**]. Without displaying error messages, database vulnerabilities can be protected. The hacker has to deal with a database system that does not display error messages and as an alternative, hackers submit a series of "TRUE" and "FALSE" queries via SQL queries [25]. Information about the database will be revealed through the results of these queries.

Time-based SQLi means hackers obtain information on database based on response times. SQL command is sent to database with code to force the database to wait for a specified amount of time during the execution of the queries. The response time indicates whether the result of the query is true or false. While waiting for the query to be processed, the attacker able to execute another query and might inject malicious code in the query [26].

Blind SQLi refers to a type of SQLi when the attacker does not receive a response from the targeted program through the same communication channel. Instead, they are able to manipulate the application to send data to a remote endpoint that they have access over.

Out-of-band SQL injection can occur if the server being used has instructions that can initiate DNS or HTTP requests. Nevertheless, this applies to all widely used SQL servers [27].

Error-based SQLi is an in-band SQLi technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site or logged to a file with restricted access instead [36].

4. METHODOLOGYS

4.1. Dataset Preparation

In this study, open-source data was collected from the Kaggle repository. Several scattered data sets were collected, cleaned, and remove the duplicate. The data for this study came from two main sources:

1. The SQLi-XSS dataset [28], which contains various data from web attacks, particularly SQLi attacks.
2. The SQL injection dataset [29], which includes both malicious and natural data, making it suitable for an intrusion detection system.

The data was processed and filtered to ensure accuracy, and the most important features were extracted, enhancing the system's ability to accurately distinguish between malicious and natural data. A total of 64,172 queries were collected, classified into two categories: approximately 25,865 normal and 38,307 malicious. A multi-class dataset was created that included four attacks, in addition to normal queries based on their pattern and methodology of execution. This ensures the system's ability to effectively predict different types of attacks. The collected dataset included five types of SQLi namely Boolean-based, Out-of-band, Union-based, Time-based, and Error-based. Each attack type was categorized accordingly, allowing the dataset to be structured for multi-class classification. Classification facilitates a more detailed detection approach, enabling the model to distinguish between normal and malicious queries and classify the specific type of SQLi attack.

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

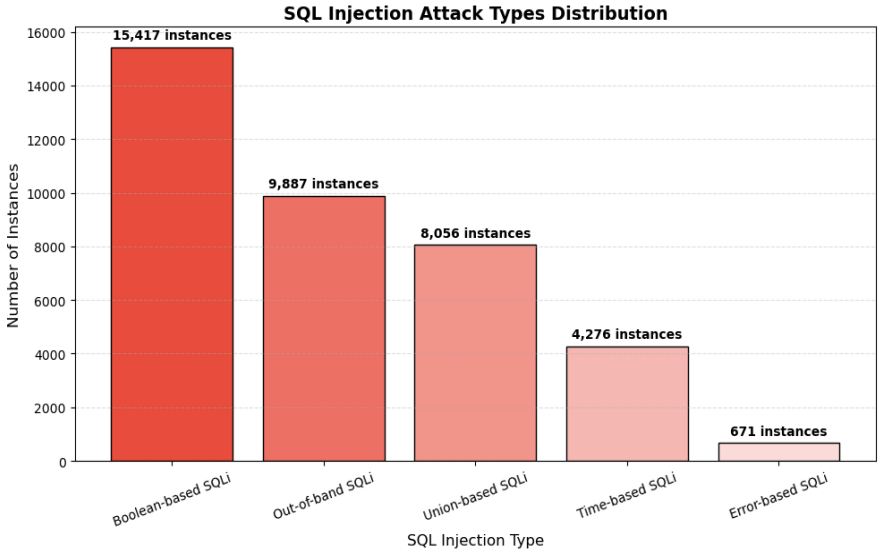

Figure (2) Dataset Category Distribution

In this study, 26 features were extracted from the dataset to improve the performance of the classification system. The feature selection process was guided by domain expertise and statistical analysis to ensure their suitability for the classification task. The selected features covered various aspects of SQL queries, such as their structure, logical conditions, and the use of special symbols. This helps the model distinguish between normal and malicious queries. The feature extraction process involved analysing text, metadata, and linguistic patterns to detect any indication of an SQLi attack. Preprocessing techniques, such as normalization and feature segmentation, were also used to ensure data consistency and smooth operation on different datasets. The selected features serve as the primary input for the machine learning model and help improve classification because they focus on several important aspects of queries. The selection of these features is consistent with previous studies in cybersecurity and machine learning. For example, Recio-Garcia et al. [30] emphasized the importance of analyzing keywords and grammatical patterns, considering features such as “contains contains union”, and “contains location”. Comi-Guzman et al. [31] showed that even minor changes in query structure affect classification. For this purpose, they used features such as query length, number of numbers, and ratio of keywords. Tang et al. [32] focused on the role of special characters in SQL injection detection using neural networks, justifying the use of features such as “contains a single quote”, “contains a semicolon”, and the proportion of special characters. Gao et al. [33] analyzed query behaviour, demonstrating the importance of evaluating logical operators such as “contains AND”, “contains OR”, and “contains Equals” to determine malicious intent. Overall, the features they chose are based on extensive research and have proven effective in accurately classifying SQL injection attacks.

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

4.2. Feature Correlation Matrix

Feature correlation is one of the basic statistical methods to understand the relationship and interdependence between features in a dataset. It gives an understanding of the way changes in one feature may impact or be linked up with changes in another. Correlation analysis is especially important in the context of security, as it helps identify the most relevant and informative features of the data that separate normal behaviour from malicious entities. Analysts can identify redundancy in the dataset by assessing how much features are correlated with each other. Feature correlation analysis was conducted to identify redundant or highly correlated features within the dataset. As shown in Figure (3), features such as query_length and word_count showed strong positive correlation, indicating potential redundancy. On the other hand, features like Query_Entropy and uppercase_count demonstrated low correlation with others, making them valuable candidates for classification retaining all of them might lead to overfitting and increased computational complexity, since highly correlated features often convey similar information. Thus, correlation analysis can help reduce dimensionality by finding and removing unnecessary features, which in turn enhances the efficacy of machine learning algorithms by streamlining the features and improving generalization. The correlation matrix is commonly prepared in this analysis, which shows pairwise correlations for all the features. It aids with selecting features that are highly correlated to target class and have little correlation among themselves, which is good as input for predictive models training.

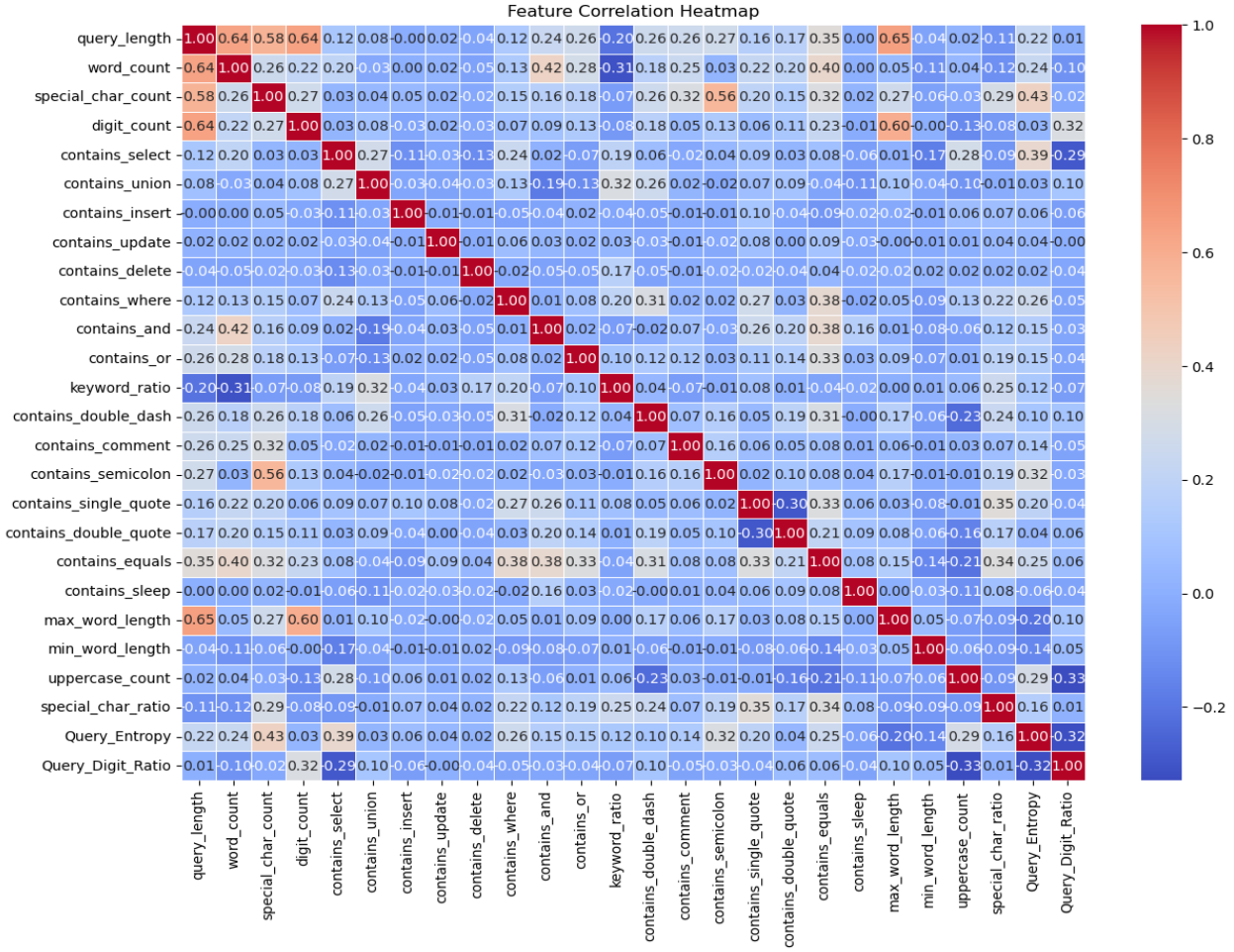

Figure (3) Feature Correlation Matrix

4.3 Proposed System

The proposed system introduces an innovative approach to attack detection. It uses SMA algorithm to accurately detect SQLi attacks. This algorithm is inspired by the behaviour of slime Mould, which exhibits remarkable problem-solving capabilities when searching for food. In the context of attack detection, it is used to achieve the following:

1. Search space exploration: The algorithm intelligently searches for optimal solutions between exploration (Global search) and exploitation (local search).
2. Adaptation to data pattern: The slime Mould algorithm adapts by dynamically adjusting search paths based on the quality of the solution, making it highly adaptable to different attack patterns
3. Classification boundary optimization: By adjusting its parameters, it can effectively separate different data classes, such as SQL attacks.

The performance of SMA depends largely on its parameters, which are carefully tuned to achieve optimal results. These parameters include:

1. Pop_size: This represents the number of solutions in the set. As the object size increases, the diversity of solutions increases, but it also increases computational complexity. In this system, we used 150 to balance finding the optimal solution with computational efficiency.

2- epoch: This parameter specifies the maximum number of iterations the algorithm can run. Increasing the number allows for a more thorough exploration of the search space but increases training time. In this system, the algorithm was set to run for 1,000 iterations to ensure convergence to the optimal solution.

3- P-t: The exploration-exploitation balance parameter controls the behaviour of the slim Mould oscillations, balancing exploration and exploitation. A smaller value encourages exploitation, while a larger value encourages exploration (0.1) was chosen to balance exploration and exploitation [34]

4- Search Space Boundaries (`lb` and ub):

-These parameters define the lower and upper bounds of the search space.

-The search space is constrained to ensure that the solutions remain within feasible limits.

-In this system, the search space is defined as **[-1, 1]** for each dimension, ensuring that the weights remain within a reasonable range.

5- Objective Function:

-The objective function evaluates the fitness of each solution based on classification performance (e.g., F1-score).

- The F1-score retrieval was chosen because the data was unbalanced to find a higher accuracy for the system

The Slime Mould Algorithm (SMA) works through the stages of food search, as it simulates the behaviour of slime Mould in nature when exploring food resources and exploiting them in an intelligent way, which helps it find optimal solutions. The figure 4 illustrates the basic steps of the SMA algorithm.

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

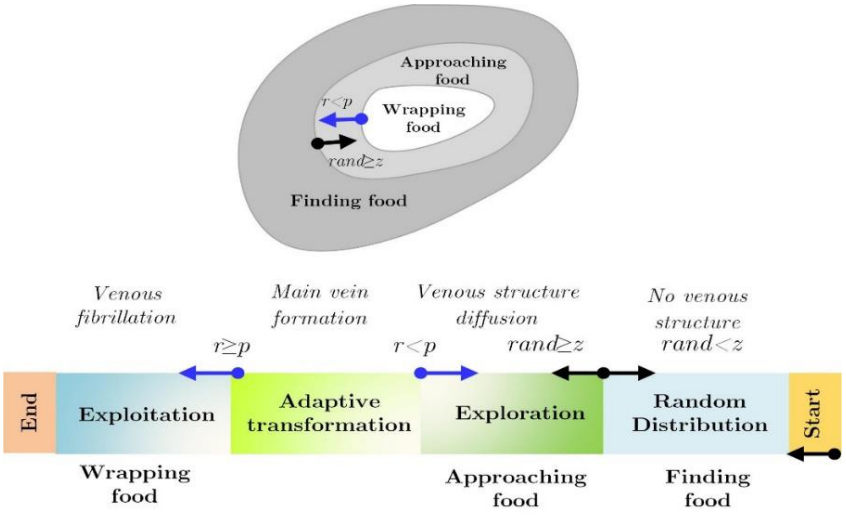

Figure (4) Illustrates SMA algorithm steps

In the proposed method, the SMA is optimized to act as an independent classifier, enabling it to make decisions by analyzing various patterns and solve complex problems arising from the behavior of SMA in their search for food. This algorithm is based on simulating slime mould movement in the environment. In the proposed SMA-based classification framework, each candidate solution encodes a vector of raw scores corresponding to the possible class labels. These scores are passed through the softmax function to transform them into normalized probability distributions, ensuring comparability across classes. The predicted class is then determined by selecting the class with the maximum probability using the argmax operation. A (softmax) activation function was used to convert the results into easily interpreted probabilities, which in turn convert the outputs into probabilities that reflect the degree of affiliation enabling the system to perform with high confidence [35].Comparative experiments with traditional algorithms show that the proposed approach has a better classification performance in intrusion detection system, and can effectively distinguish between both malicious data and normal data, achieve real-time detection of two types of data while ensuring scalability and separability in practical applications. This is a methodology that prevent SQLi attacks. The system architecture we propose includes data gathering, loading into the model, cleaning and processing, feature preprocessing and extraction, standardization and binary classification. During the training phase K-stratified cross-validation is used to maintain robustness and generalization capability of the model. This procedure splits the data into a few folds and trains using some but not all of them, averages their results while validating on one or more of the other folds, resulting in less overfitting and greater stability of results.

To simulate a real-world zero-day attack scenario, where the system must detect a new attack variant it has never encountered before. The instances of a variant of SQLi attacks in the dataset were removed from the training set and added to the testing set. Then, the best evolved SMA classifier was tested against the removed SQLi variant. The complete workflow of the proposed SMA classifier framework is outlined in Algorithm 1 and described in Figure 5.

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

Algorithm1 : The Proposed System
Input: Dataset with Normal and SQLi Query **Output**: Best SMA model
1. Start **2. Preprocess the data** **a. Features Extraction** **b. Label Encoder** **c. Standardization** **3. Split the data into training and testing sets.** **4. Initialize the SMA population (solutions).** **5. Score =np.dot (X_Train, SMA Weight.T)** **6. Repeat until the maximum number of iterations is reached:** **a. Apply the Softmax function outputs.** **b. Evaluate each solution using the F1-score metric.** **c. Update weights and positions using SMA update equations.** **7. Select the best SMA solution.** **8. Evaluate the final model on the testing set.** **9. Report the performance metrics: Accuracy, Recall, Precision, F1-score** **10. End**

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

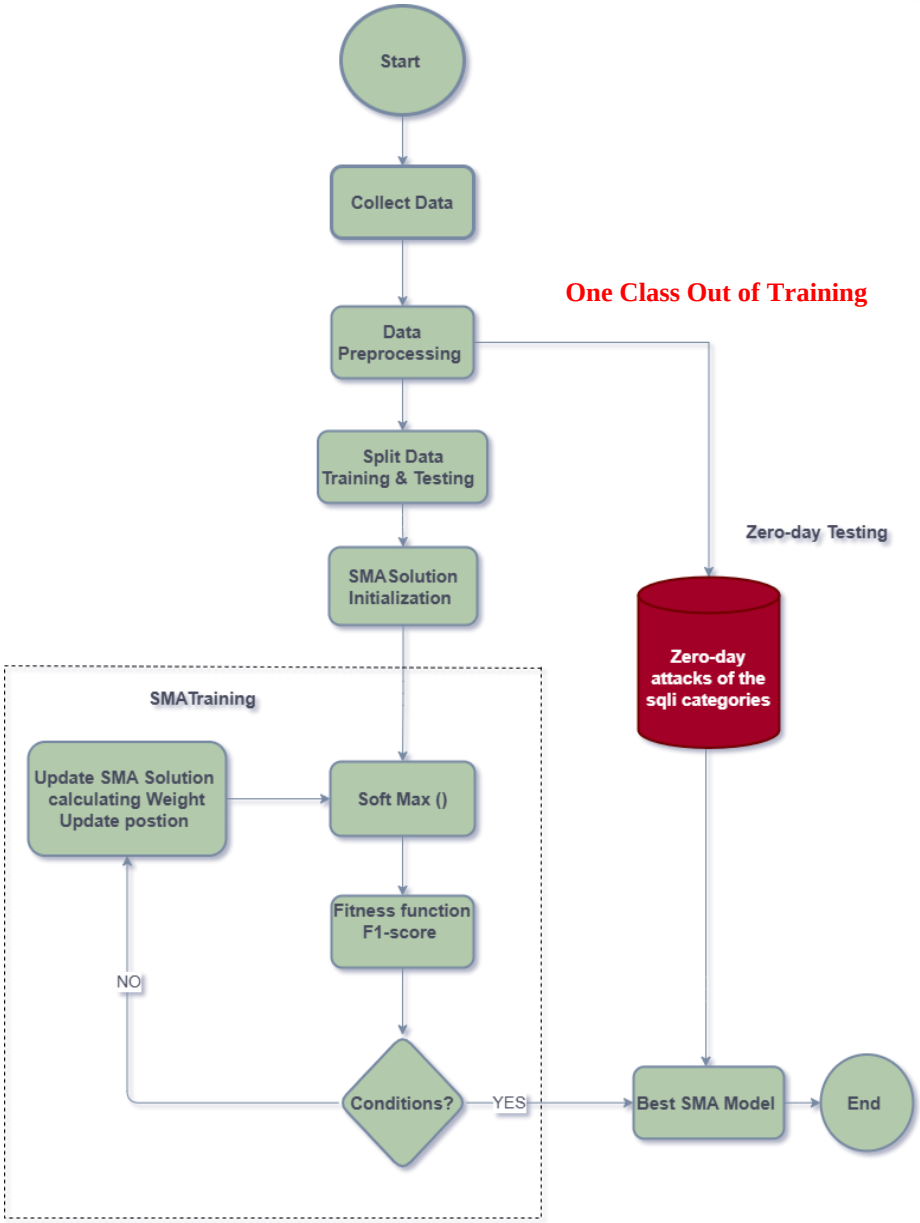

Figure (5) Architecture of the Proposed SMA-Based method for Detecting SQL Injection and Zero-Day Attack.

4.3. Training and Evaluation

The model was trained and evaluated using an open-source SQLi attack classification dataset. Preprocessing and feature extraction, including encoding and standardization, were supervised. SMA was used as a classifier, and a Soft Max activation function transformed the raw system outputs into probability distributions to ensure the reliability of the results. We evaluated the system using accuracy metrics, F1 score, and classification report analysis. These metrics provided a comprehensive assessment of the system's ability to distinguish between malicious and normal queries and detect zero-day attacks. All experiments were conducted on a Lenovo PC with an Intel Core™ i3-3110M processor. A 2.40 GHz processor, 12 GB of RAM, and Windows 10 Pro (64-bit, x64) are implemented using Python and leverage key machine learning and nature-inspired algorithmic libraries, including NumPy, Matplotlib, Pandas, Scikit-learn, Time, Seaborn, and Mealpy.

4.4. Performance Metrics

The metrics used to evaluate the performance of the proposed system are accuracy, recall, precision, and F1-score. These classification measures are based on the confusion matrix, serve to evaluate performance. Accuracy measures the percentage of total queries in a dataset that are correctly classified by the SMA classifier. It is calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

In this content, True Positive (TP) shows the count accurately detected as SQLi. When valid queries are accurately classified as normal, it is called a True Negative (TN). Queries that are not SQLi but were incorrectly classified as SQLi are known as False Positives (FP). A False Negative (FN) occurs when unsolicited queries are mistakenly identified as normal SQL queries. Recall, also known as sensitivity, determines how well the model captures actual SQLi queries. It is defined as the percentage of true positives identified out of all actual SQLi queries:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Precision reflects the proportion of positive identifications that were actually correct. In other words, it quantifies how many of the queries flagged as SQLi truly are SQLi:

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

The F1-Score reflects the harmonic mean of precision and recall, offering a balanced metric. It is measures as:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

5. RESULTS AND DISCUSSION

5.1. K-Fold Cross-Validation Results

Table 1 shows the performance of SMA-based classifier when evaluated using 5-fold cross-validation, measuring accuracy, precision, recall, and F1-score for both training and testing phases. The results reveal consistently high performance across all folds, with minimal variation, indicating strong model generalization and stability. The average testing accuracy of 98.82% and F1-score of 98.55% demonstrate the model's effectiveness in distinguishing between legitimate and malicious SQL queries. The high recall value of 99.65% highlights the classifier's ability to detect nearly all attack instances, thereby minimizing false negatives. Similarly, a precision of 97.49% confirms that the majority of detected attacks were correctly identified, reducing false alarms and ensuring reliability in detection outcomes. Furthermore, the very low Standard Deviations (SD) across all metrics ($\leq \pm 0.15$) indicate consistent performance across the folds, signifying that the model is not overfitting and can maintain strong predictive capability on unseen data. The close alignment between training and testing results further supports the model's excellent generalization ability and robustness. Overall, these results confirm the effectiveness of the SMA-based classifier as a reliable and adaptive bio-inspired approach for detecting SQLi attacks in dynamic web environments.

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

Table 1. K-Fold Cross-Validation Results (%)

Folds	Training				Testing			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
1	98.91	97.68	99.66	98.66	98.93	97.75	99.67	98.70
2	98.88	97.73	99.54	98.63	98.76	97.35	99.59	98.46
3	98.79	97.52	99.55	98.50	98.75	97.52	99.49	98.49
4	98.73	97.18	99.75	98.45	98.85	97.48	99.73	98.59
5	98.86	97.51	99.73	98.61	98.82	97.34	99.74	98.53
AVG	98.83	97.51	99.65	98.57	98.82	97.49	99.65	98.55
SD	± 0.07	± 0.19	± 0.09	± 0.08	± 0.07	± 0.15	± 0.10	± 0.09

As the one of executing time (test time) is an important metric for our approach, this timely executed will influence on effectively and efficiency in testing process. Figure (6) shows the SMA model has a test time of 0.0086 ± 0.0057 seconds This test time efficiency is important for facilitating faster model iterations, identifying performance issues more quickly.

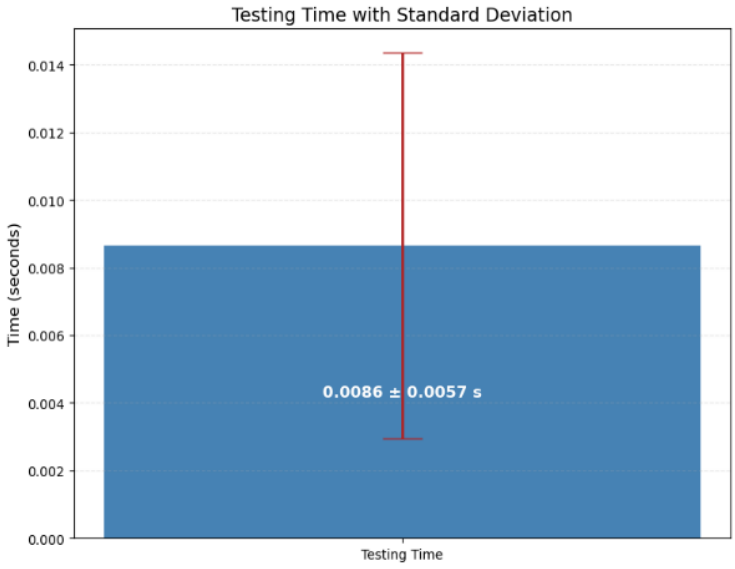

Figure (6) Testing Time for the Proposed Approach.

5.2. Performance on Detecting Zero-Day Attacks

To evaluate the model's ability to detect previously unseen or zero-day attacks, we tested it against a set of samples excluded from the training data. The SMA classifier was able to successfully generalize and detect patterns indicative of zero-day behaviours. The SMA-based System demonstrated superior detection capabilities, particularly in identifying zero-day attacks that deviate from known patterns. The experimental results reveal promising detection rates across all attack types: 89.33% for Out-of-Band-based SQLi, 97.89% for Boolean-based SQLi, 90.27% for Time-based SQLi, and 96.69% for Union-based SQLi and 91.51%for Error-based SQLi. These findings underline the effectiveness of SMA in generalizing beyond seen data, a critical advantage in dynamic environments as shown figure (7).

Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

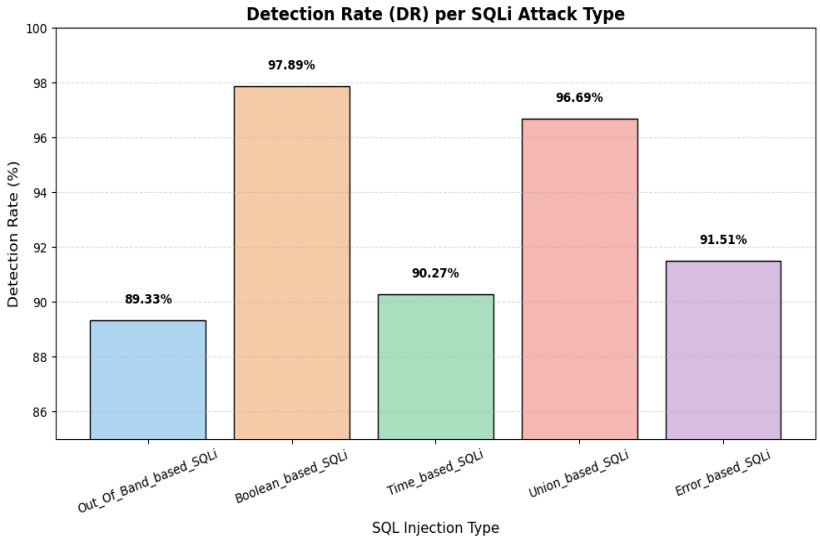

Figure (7) Performance on Detecting Zero-Day Attacks

5.3. Feature Importance

The proposed approach deployed SMA algorithm to effectively choose and compute the precise predictors for detecting SQLi attacks via extracted features. However, SMA model not only utilised the best feature but also supplied importance scores that demonstrate the influence of every individual feature on the classification performance. To evaluate the stabilities of the used features, their importance score, and SD are calculated over each fold of cross-validation. These results showed that a certain number of features were highly important with low SD, indicating their high and consistent impact in the classification process. Figure 8 demonstrates the feature importance scores with their corresponding SD extracted from SMA model. It shows that uppercase_count, query_length, and contains_delete are the most influential features, contributing significantly more to the model's performance than the remaining attributes.

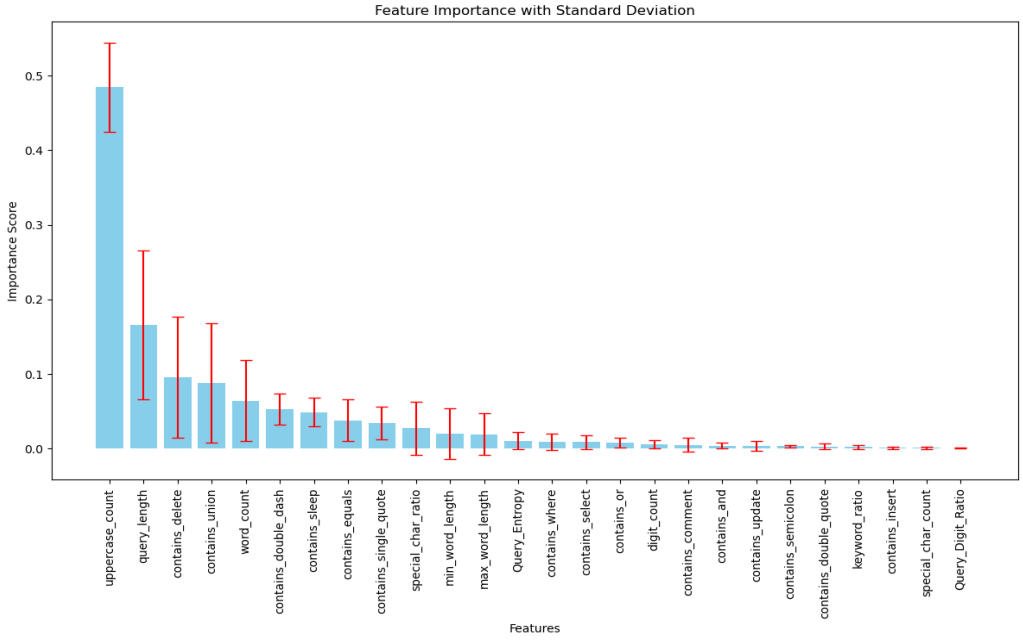

Figure (8) Feature Importance with Standard Deviation calculated using SMA and 5 -Fold Cross-Validation

5.5. Comparing the Results with the Most Advanced Studies

The efficacy of the suggested method was evaluated against contemporary research on SQLi detection, using current works for this assessment. Table (2) shows that the suggested method performed better compared to state-of-the-art techniques, with an accuracy of 98.83%.

Table (2) Achieved Results compared with recent studies on detecting SQLi Attacks.

Ref no.	Algorithm	Year	Dataset size	Accuracy	F1-score
Maha et al. [13]	RNN	2023	30,907	94	92
Jarudat et al. [15]	GA+GB	2023	2,300	95	94
M. Shahbaz et al. [14]	CNN	2024	109,520	98.16	98.06
Cumi-Guzman et al. [11]	RF	2024	22,931	97.3	97.2
Rosca et al. [12]	Ensemble	2025	90,000	96.86	96.77
Proposed System	SMA	2025	64,172	98.82	98.55

6. CONCLUSION AND FUTURE WORKS

In this research, we show the implementation of the Slime Mould algorithm to ensure a safe web environment. we introduce a unique solution for identifying zero-day attacks using the Slime Mould Algorithm as the classifier instead of its traditional usage as an optimization technique. SMA is quickly adaptable to complex and dynamic cybersecurity data making it agile to identify new patterns associated with zero-day.

The future work plans to enhance the dimension of the research to involve more SQLi attacks including a second order injection, cookie injection, etc., as well as others web attacks especially Cross-Site Scripting, Cross-Site Request Forgery and command injection - also malicious attacks in mobile devices and IOT contexts. Next to apply the algorithm to other branch of cybersecurity

REFERENCES

[1] M. Qbea'h, M. Alshraideh, and K. E. Sabri, ''Detecting and preventing SQL injection attacks: A formal approach,'' in Proc. Cybersecur. Cyber- forensics Conf. (CCC), Aug. 2016, pp. 123–129.
[2] OWASP, "2024 SQL Injection," [Online]. Available: https://owasp.org/wwwcommunity/attacks/SQL_Injection. [Accessed: Jan. 01, 2025].
[3] Aikido Security, "SQL Injection Vulnerability Report," 2024. [Online]. Available: https://www.aikidosec.com/blog/sql-injection-vulnerabilities-2024. [Accessed: Jan. 01, 2025].
[4] AIONCLOUD, "2025 Web Attack Trend Report," Feb. 2025. [Online]. Available: https://www.aioncloud.com/2025-02-web-attack-trend-report/
[5] Y. Wei, Z. Othman, K. M. Daud, Q. Luo, and Y. Zhou, "Advances in Slime Mould Algorithm: A Comprehensive Survey," Biomimetics, vol. 9, no. 1, p. 31, Jan. 2024.
[6] Z. H. Al-Araji, "A Survey on Bio-Inspired Algorithm for SQL Injection Attacks: Survey on Bio-Inspired Algorithm for SQL Injection Attacks", J. Basrah Res. (Sci.), vol. 50, no. 1, p. 340, Jun. 2024
[7] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review," Journal of Cybersecurity and Privacy, vol. 2, no. 4, pp. 764–777, 2022. doi: 10.3390/jcp2040039.
[8] H. Alyasiri, J. A. Clark, and D. Kudenko, "Evolutionary computation algorithms for detecting known and unknown attacks," in Innovative Security Solutions for Information Technology and Communications: 11th International Conference, SecITC 2018, Bucharest, Romania, November 8–9, 2018, Revised Selected Papers 11, Springer, 2019, pp. 170–184

[9] H. Alyasiri, Evolving Rules for Detecting Cross-Site Scripting Attacks Using Genetic Programming, vol. 1347. Springer Singapore, 2021. doi: 10.1007/978-981-33-6835-4_42.
[10] Z. Z. Jundi and H. Alyasiri, "Android Malware Detection Based on Grammatical Evaluation Algorithm and XGBoost," in 2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT), IEEE, 2023, pp. 70–75
[11] F. K. Alarfaj and N. A. Khan, "Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks," Appl. Sci., vol. 13, no. 7, 2023, doi: 10.3390/app13074365.
[12] Rosca, C.-M.; Stancu, A.; Popescu, C. Machine Learning Models for SQL Injection Detection. Electronics 2025, 14, 3420. https://doi.org/10.3390/electronics14173420.
[13] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Deep learning architecture for detecting SQLinjection attacks based on RNN autoencoder model," Mathematics, vol. 11, no. 15, p. 3286, 2023.
[14] M. Shahbaz, G. Mumtaz, S. Zubair, and M. Rehman, "Evaluating CNN Effectiveness in SQL Injection Attack Detection," Journal of Computing & Biomedical Informatics, vol. 7, no. 2, 2024. DOI: 10.56979/702/2024.
[15] Jarudat, et al. (2023). Genetic Algorithm-Based Model for SQL Injection Detection Using TPOT and GB Optimization. Journal of Applied Security Research.
[16] W. Lu and I. Traore, "Detecting new forms of network intrusion using genetic programming," *Computational Intelligence*, vol. 20, no. 3, pp. 475–494, 2004. doi: 10.1111/j.1467-8640.2004.00241.x.
[17] J. Blasco, A. Orfila, and A. Ribagorda, "Improving network intrusion detection by means of domain-aware genetic programming," in *Proc. Int. Conf. on Availability, Reliability and Security (ARES)*, IEEE, pp. 327–332, 2010.
[18] W. F. Waheed and H. Alyasiri, "Evolving trees for detecting android malware using evolutionary learning," *Int. J. Nonlinear Anal. Appl.*, vol. 14, no. 1, pp. 753–761, 2023. doi: 10.22075/ijnaa.2022.6874.
[19] M. H. Kamarudin, A. A. Aziz, and A. Selamat, "A LogitBoost-based algorithm for detecting known and unknown web attacks," *IEEE Access*, vol. 5, pp. 26190–26203, 2017.
[20] Y. Wimukthi, H. R. Sri, H. Kottegoda, D. Andaraweera, and P. Palihena, "A comprehensive review of methods for SQL injection attack detection and prevention SEE PROFILE A comprehensive review of methods for SQL injection attack detection and prevention," no. October, pp. 1–10,2022,[Online]. Available: https://www.researchgate.net/publication/364935556
[21] M. Nasereddin, A. Alkhamaiseh, M. Qasaimeh, and R. Al-, "A systematic review of detection and prevention techniques of SQL injection A systematic review of detection and prevention techniques of SQL injection attacks," Inf. Secur. J. A Glob. Perspect., vol. 00, no. 00, pp. 1–14, 2021, doi: 10.1080/19393555.2021.1995537.
[22] T. Pattewar, H. Patil, H. Patil, N. Patil, M. Taneja, and T. Wadile, "Detection of SQL injection using machine learning: a survey," Int. Res. J. Eng. Technol.(IRJET), vol. 6, no. 11, pp. 239–246, 2019.
[23] P. Suri, "DATA PROTECTION : SQL INJECTION PREVENTION," no. 01, pp. 2716–2732, 2024.
[24] M.Amirulluqman Azman, Mohd Fadzli Marhusin and Rossilawati Sulaiman, ―Machine Learning-Based Technique to Detect SQL Injection Attack‖, Journal of Computer Science, 17 (3), 2021, pp.296-303
[25] J.Minhas and Kumar Raman, ―Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries‖, International Journal of Computer Network and Information Security; 5(2), Feb 2013, pp.1-9.
[26] M.Amin Mohd Yunus, Muhammad Zainulariff Brohan and Nazri Mohd Nawi. ―Review of SQL Injection: Problems and Prevention‖. International Journal On Informatics Visualization, vol 2, 2018, No 3 –2

[27] I. Muscat, “SQLi part 6: Out-of-band SQLi,” November 16, 2015. Accessed: Mar. 01, 2025. [Online]. Available: https://www.acunetix.com/blog/articles/sqli-part-6-out-of-band-sqli/

[28] A.Trinity,“SQLIandXSSDataset,”Kaggle,[Online]. Available: https://www.kaggle.com/datasets/alextrinity/sqli-xss-dataset [Accessed: Jan. 01, 2025].

[29]S.S.Hussain,“SQLInjectionDataset,”Kaggle,[Online]. Available: https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset [Accessed: Jan. 01, 2025].

[30] Recio-García, J. A., Orozco-del-Castillo, M. G., & Soladrero, J. A. (2023). Case-based explanation of classification models for the detection of SQL injection attacks.

[31] Cumi-Guzman, B. A., Espinosa-Chim, A. D., Orozco-del-Castillo, M. G., & Recio-García, J. A. (2024). Counterfactual Explanation of a Classification Model for Detecting SQL Injection Attacks.

[32] Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural networks.

[33] Gao, H., Zhu, J., Liu, L., Xu, J., Wu, Y., & Liu, A. (2019). Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining.

[34] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Slime mould algorithm: A new method for stochastic optimization," Future Generation Computer Systems, vol. 111, pp. 300-323, Oct. 2020. [Online]. Available: https://doi.org/10.1016/j.future.2020.03.055

[35] M. Abadi et al., "Deep learning with differential privacy," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16), Vienna, Austria, Oct. 2016, pp. 308-318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[36] P. Kumar and R. K. Pateriya, “A Survey on SQL Injection Attacks, Detection and Prevention Techniques,” no. July, 2012.