mathematics

MDPI

Article

Effective SQL Injection Detection: A Fusion of Binary Olympiad Optimizer and Classification Algorithm

Bahman Arasteh [1,2,*], Asgarali Bouyer [1,3], Seyed Salar Sefati [1,4] and Razvan Craciunescu [4]

[1] Department of Software Engineering, Faculty of Engineering and Natural Science, Istinye University, Istanbul 34460, Turkey; sefati.seyedsalar@upb.ro (S.S.S.)
[2] Department of Computer Science, Khazar University, Baku AZ1096, Azerbaijan
[3] Faculty of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz 5375171379, Iran
[4] Faculty of Electronics, Telecommunications and Information Technology, National University for Science and Technology Politehnica Bucharest, 060042 Bucharest, Romania; razvan.craciunescu@upb.ro
* Correspondence: bahman.arasteh@istinye.edu.tr

Abstract: Since SQL injection allows attackers to interact with the database of applications, it is regarded as a significant security problem. By applying machine learning algorithms, SQL injection attacks can be identified. **Problem**: In the training stage of machine learning methods, effective features are used to develop an optimal classifier that is highly accurate. The specification of the features with the highest efficacy is considered to be an NP-complete combinatorial optimization challenge. Selecting the most effective features refers to the procedure of identifying the smallest and most effective features in the dataset. The rationale behind this paper is to optimize the accuracy, precision, and sensitivity parameters of the SQL injection attack detection method. **Method**: In this paper, a method for identifying SQL injection attacks was suggested. In the first step, a particular training dataset that included 13 features was developed. In the second step, to specify the best features of the dataset, a specific binary variety of the Olympiad optimization algorithm was developed. Various machine learning algorithms were used to create the optimal attack detector. **Results**: Based on the experiments carried out, the suggested SQL injection detector using an artificial neural network and the feature selector can achieve 99.35% accuracy, 100% precision, and 100% sensitivity. Owing to selecting about 30% of the effective features, the proposed method enhanced the efficacy of SQL injection detectors.

Keywords: security; SQL injection attacks; binary Olympiad optimization algorithm; feature selection; machine learning algorithms; accuracy

MSC: 68W01; 68T20

Citation: Arasteh, B.; Bouyer, A.; Sefati, S.S.; Craciunescu, R. Effective SQL Injection Detection: A Fusion of Binary Olympiad Optimizer and Classification Algorithm. *Mathematics* **2024**, *12*, 2917. https://doi.org/10.3390/math12182917

Academic Editor: Ioannis Tsoulos

Received: 15 August 2024
Revised: 11 September 2024
Accepted: 14 September 2024
Published: 19 September 2024

1. Introduction

The security of software is regarded as a significant quality parameter for a software product [1]. SQLi (SQL injection) is a drawback of web security that provides an opportunity for attackers to interact with a database of software applications. SQL queries may be directly exploited by attackers to obtain data from databases; in this way, they can obtain unlimited data and achieve illegal admission and entrance to the database. OWASP (Open Worldwide Application Security Project) contends that SQL injection attacks are highly vulnerable and are regarded as one of 10 web program security threats [1–3]. The majority of traditional security techniques are not strong enough to protect databases of web applications against SQLi attacks. Machine learning (ML) algorithms have been applied to create SQLi detectors, and the created classifiers categorize attacks and normal SQL queries.

The dataset features that are employed in the training step of ML algorithms have a considerable impact on the efficiency and performance of the created classifier. Spotting

the most efficient dataset features is regarded as a combinatorial NP-complete optimization problem. Selecting the most effective features (optimal) enhances the accuracy, precision, and performance of the created SQLi detector. Numerous metaheuristic search algorithms have been applied to determine the best dataset features. High error rates, low accuracy in the developed SQLi detectors, inadequate precision of the developed SQLi detectors, plurality of features used by learning algorithms to develop SQLi detectors, and poor convergence rate are regarded as the chief shortcomings of earlier SQLi techniques. Having mentioned the shortcomings of the earlier SQLi techniques and methods, the purposes of this study are to optimize the SQLi detectors' performance (accuracy, precision, sensitivity, and F1), identify the minimal number of effective features, and enhance the stability and reliability of SQLi detection techniques.

In this study, an efficient method for identifying SQLi attacks is suggested by applying minimal effective features of the training datasets. A particular training dataset that included 13 features, was first developed. Next, owing to the specific merits of the Olympiad optimization algorithm (OOA), an improved binary version of OOA (referred to as BOOA) was suggested as a feature selector; it was aimed at specifying the best features for the dataset. The ideal dataset selected via BOOA is applied by various ML algorithms to create efficient SQLi detectors. Eventually, the operation of the proposed feature selection method is investigated and tested using the test data. Major achievements of this paper are the development of a binary variety of Olympiad optimization algorithms for determining dataset features, identification of the most effective SQL-querys' attributes, and development of an effective and efficient classifier for identifying SQLi attacks.

The rest of the paper is organized as follows: Section 2 reports and reviews related SQLi techniques in the related literature. Section 3 introduces and discusses the proposed SQLi method. Section 4 reports the experimental procedure and study results. Section 5 provides the conclusion and directions for further research.

2. Related Works

Numerous methods and techniques have been proposed to identify SQLi attacks. Here, some of the related works under certain classifications are reported. Conventional SQLi techniques apply blacklists to prevent illegal and unauthorized entities. Related studies in this domain have proposed several methods. For SQLi attack prevention, input validation methods have been classified into whitelist and blacklist methods [4]. Penetration testing was proposed in [5] to make up for the drawbacks of the blacklist technique. Similar to the blacklisting technique, the Boyer-Moore string-matching algorithm was proposed in [6] to determine the degree to which a string corresponds to the associated URL using an injected SQL string to identify an SQL attack. An automated method was proposed in [7] to avoid SQLi attacks during implementation. The merits of Blackbox methods versus XSS and SQL attacks were examined in [8]. Hsiu-Chuan Huang proposed a novel vulnerability scanner for web applications. It applies penetration testing to avoid and identify SQLi attacks.

WAVES was proposed in [9] to examine SQLi vulnerabilities in web applications. The application of a web crawler is aimed at detecting spots in a web application that is utilized for SQLi attacks. Owing to the exploitation of machine learning approaches for guiding testing, this procedure optimizes penetration testing methods. Nevertheless, similar to black box testing and fuzz testing, this procedure does not ensure completeness. The work in [10] developed a detection technique for SQLi attacks that eliminates values from the SQL query features of webpages. In this method, dynamic and static analyses are combined. The performed tests show that the proposed technique is effective.

Analyzing SQL queries in web applications is a notable technique for preventing SQLi. Instead of detecting SQLi, this method aims to verify the probability of user input SQLi [9]. By capitalizing on the Java String Analysis (JSA) library, the study reported in [11] validated user inputs and avoided SQLi attacks. Nonetheless, if the malicious (attack) input data have the right syntax, it fails to avoid SQLi attacks. Moreover, such a library cannot support languages other than Java language. The work reported in [12] combines static

analysis with automatic reasoning. Hence, it can be argued that it is a successful technique for identifying and avoiding SQLi attacks; however, tautology is regarded as an exception, that is, it cannot detect other types of SQLi attacks. A novel SQLi detector was proposed by Stephen Thomas et al. [13] by gathering simple SQL statements and implementing them to authenticate user input. It cannot identify SQLi attacks on its own; rather, it is aimed at avoiding attacks by eliminating vulnerabilities regarding how SQL queries are composed. This method may also be used for web applications composed in Java.

This type of analysis investigates the responses of web applications after scanning them. In contrast to static analysis, this approach can detect the vulnerability of SQLi attacks regardless of any modifications to web applications; hence, this is regarded as a merit lacking in the static approach. On the other hand, researchers should address the shortcomings related to the pages of web applications. A method called Sania was developed and proposed by Yuji Kasuga et al. [14]. It is intended to defend web applications against SQLi attacks. This technique gathers natural queries not only between the client and web application, but also between the web application and database. A web scan technique was developed in [15]. It has advanced features for investigating and identifying interruptions in PHP-based web applications. In this way, this technique is aimed at detecting vulnerable spots in web applications related to SQLi attacks. However, a shortcoming of this method is that it functions merely on PHP-based applications; hence, it fails to identify SQLi attacks in other types of web applications (like ASP).

It refers to an amalgamation of static and dynamic methods. It enjoys the advantages of both static and dynamic analyses. Moreover, this method can examine webpages and generate SQL queries to assess the outcomes. An example of such a hybrid analysis is the AMNESIA method, which was developed and proposed by William et al. [16]. It is a blend of static and dynamic analysis methods. Indeed, it has a static step and a dynamic step; in the static analysis step, AMNESIA is aimed at constructing patterns for various kinds of queries. Applications may produce different queries when a database is available. In dynamic analysis, this method first deals with all queries. Then, it transmits them to the database and statistically analyzes each query. Another instance of such a hybrid method is the SQL Guard [17], which was developed and proposed by Buehrer et al. This method is intended for static and dynamic SQL queries created by users to identify SQLi using a survey tree. SQLCheck is regarded as an additional instance within this category. It is based on attack injection. Both the SQLCheck and SQLGuard methods apply a concealed key to specify the limit of user inputs while they are examined by the runtime controller.

The work reported in [18] proposed the SQLrand technique, which is a randomization-based method. It is designed to set instructions that allow developers to apply instructions rather than SQL keywords. Hence, a proxy filter injects queries into the database and eliminates keywords from users. Since the SQL code of attackers does not consist of random instructions, the commands that can be injected result in syntactically wrong queries. The query-profile-based method was developed and proposed in [18]. It can detect SQLi attacks regardless of the modification of web applications. Nevertheless, the web application should be re-profiled when it is modified. By capitalizing on ML algorithms, an IDS method (intrusion detection) was developed by Valeur et al. [19] for detecting SQLi attacks. It exploits several anomaly identification patterns. As a module, this technique establishes a connection between the applications and the database; the queries completed by the application are stopped, and then, they are transmitted to the attack detection system. It compares and contrasts them with the created model throughout the implementation of a web program to examine alterations.

A significant restriction of learning-based techniques is that they do not ensure 100% detection. Indeed, this is attributed to the quality and training technique exploited in the respective model. Hence, these techniques may result in false positives and negative consequences. The Naive Bayes algorithm was proposed by Joshi et al. [20]; it was developed as a Bayes' theorem-based classification pattern in machine learning. The technique assumes that feature availability in the data pattern is independent of the availability of other fea-

tures. The Naive Bayes algorithm is easy to run; however, it may not detect different SQL injection attacks, particularly when a specific SQL injection is utilized.

3. Proposed Method

In this paper, an efficient method for identifying SQLi attacks has been developed. At the outset, a specific training dataset including 13 features was generated. Then, in the second step, given the distinctive aspects of OOA, a binary type of OOA was produced to specify the best dataset features. Several ML techniques exploit the optimal datasets that BOOA filters. The techniques of ANN (neural networks), DT (decision tree), SVM (Support Vector Machine), and KNN (K-Nearest Neighbors) were applied to develop the correct classification pattern for identifying SQLi. In the final testing step, the test dataset was used to investigate and evaluate the operation of the proposed feature selection method. The workflow of the proposed technique is depicted in Figure 1.

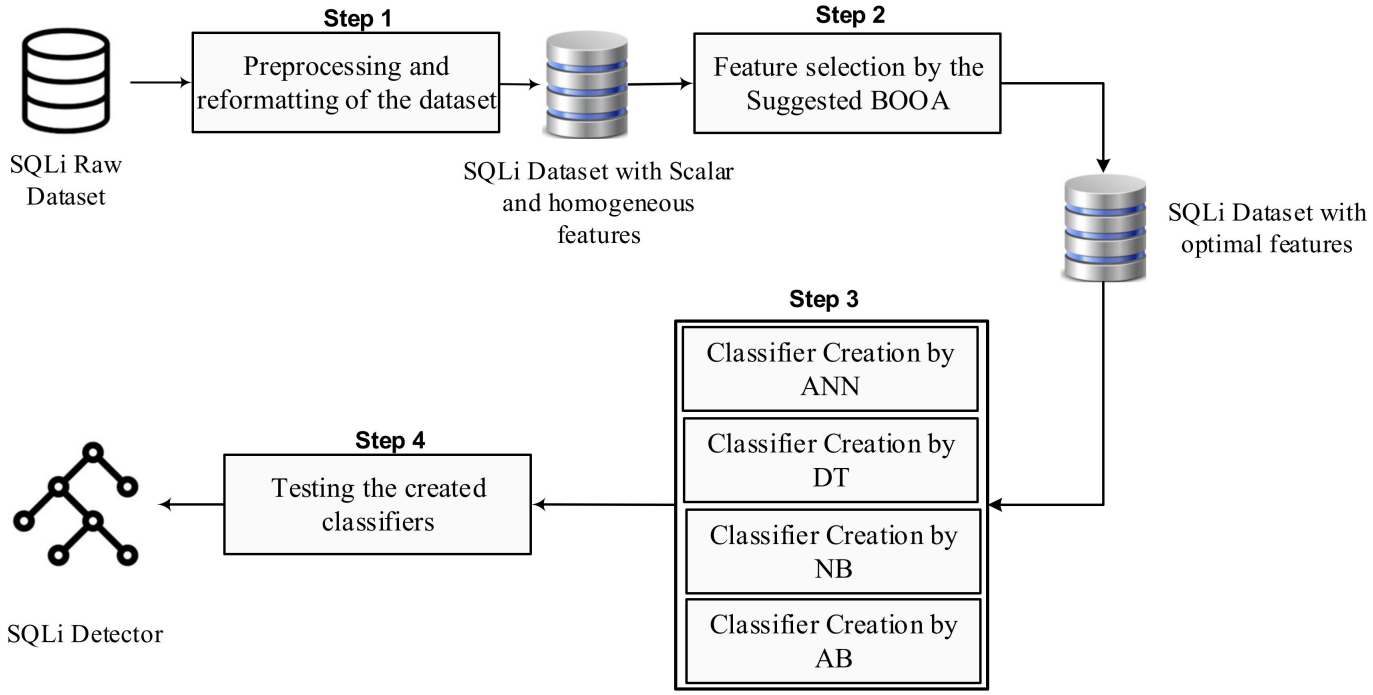


Figure 1. Workflow of the proposed technique.

3.1. Dataset Generation

This subsection elaborates on the needed dataset for training the supervised machine learning algorithm. SQL query profiling process was applied to generate the SQLi dataset. The experimental system includes web programs with a MySQL server back-end, HTTP APIs, and front-end. SQL query generation application was used to generate SQLi traffic, including both normal and SQLi queries. The intended MySQL traffic between the user interface code, server-side codes, and database was seized. Then, the gathered raw datasets were processed, and the correlation between them was computed to generate a distinct dataset [21]. The records of the collected dataset are provided in the SQL query generation application. Attacks and normal queries are included within the respective dataset for a given web program. The queries can be executed using various SQL formats and complexities. Complex and Simple queries are available within the related dataset. The dataset row (record) shows a normal SQL query or attack (SQL injection). A single record within the dataset includes SQL query features, such as single quotes, intermediate data, comments, and semicolons. Table 1 indicates SQLi attacks that were recorded in the dataset.

Table 1. SQL injection records in the SQLi dataset.

Query
SELECT * FROM employee UNION SELECT * clear FROM result ORDER BY diploma
SELECT * FROM items WHERE item_id=1 OR 'a'='a'SELECT * FROM items WHERE item_id=1 OR 'a'='a'
SELECT name, family, tell, sum1 FROM customer INNER JOIN (SELECT cust_id, SUM(amount) AS sum1 FROM payment GROUP BY cust_id) AS payment1
ON customer.cu_id=payment1.cust_id WHERE cu_id=@cu_id;
SELECT customer1.cu_id, customer1.name, customer1.family, factor.fact_id, factor.fdate, factor.amount
FROM (SELECT * FROM customer WHERE cu_id=@cust_id) AS customer1 INNER JOIN factor ON customer1.cu_id=fa tor.cust_id;

A given raw dataset should be cleaned, filtered, and changed to a dataset with homogenous numeric features. The dataset used in this paper consists of 1027 retrieved SQL queries, namely both normal and SQL injection queries derived from a text dataset. It includes 1027 records, each of which denotes an SQL query. In the dataset, 554 records were regarded as malicious (attack) queries (SQL injection queries) and 473 as normal queries. Dataset records are categorized and labeled as 0 (normal query) and 1 (SQLi query).

This paper utilized an efficient numeric training dataset, which includes 13 numeric features [22]. It should be noted that the similarity of features within the given dataset leads to a more effective operation of machine learning algorithms concerning accuracy and precision criteria. According to the first step of the method, SQLi datasets with thousands of SQLi queries were investigated; then, 13 effective traits (features) were identified. After that, the chosen original dataset was examined, and 13 features were removed from the query codes. It should be noted that the features are numeric; 12 of them are independent, and one is dependent, which denotes the query label. Each derived feature has a numeric label. The following are regarded as derived features: query length, constants, number of nested queries, logical operators, and punctuation. In the end, the generated dataset will consist of 1027 records; each record includes 13 features. Table 2 specifies the values of the features for the 19 randomly selected records from the dataset. Also, the features of the produced training dataset are shown in Table 3.

Table 2. The arrangement of the datasets includes 13 numeric features [22].

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	10	1	1	2	0	3	2	1	0	1	0	0	1
2	17	1	0	1	0	10	1	0	1	1	0	2	1
3	18	1	0	2	0	10	1	0	0	0	0	0	0
4	17	1	0	2	0	10	1	0	0	1	0	2	1
5	31	1	0	5	0	0	1	0	0	2	0	4	1
6	28	2	1	7	0	13	2	0	0	2	0	0	1
7	11	2	1	1	0	4	2	0	0	0	0	0	1
8	18	1	0	3	0	8	1	1	0	4	0	0	0
9	18	1	0	3	0	8	1	1	0	4	0	0	0
10	47	2	0	2	0	22	1	2	0	4	0	5	1
11	10	1	0	1	0	3	1	0	0	0	0	0	0
12	47	2	0	2	0	23	1	2	0	4	0	5	1
13	14	1	0	3	0	4	1	1	0	0	0	0	0
14	35	1	0	1	0	18	1	2	0	4	0	4	1
15	36	1	0	1	0	18	1	2	0	4	0	4	1
16	41	1	0	1	0	19	1	2	0	4	0	4	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0
19	49	2	0	2	0	22	1	1	0	4	0	5	1

Table 3. The specification of the features in the dataset [22].

#Numbers	Specification
1	"Length of SQL Query"
2	"Nesting Level Query"
3	"Num. of UNION Operator in the SQL Query"
4	"Num. of Constant Value in the SQL Query"
5	"Num. of OR Operator in the SQL Query"
6	"Num. of Specific Character in the SQL Query"
7	"Type of SQL Query in the SQL Query"
8	"Num. of AND Operator in the SQL Query"
9	"Num. of double quotation Character (") in the SQL Query"
10	"Num. of single quotation Character (') in the SQL Query"
11	"Num. of Null Value in the SQL Query"
12	"Num. of Parenthesis in the SQL Query"
13	"Class (Attack or Not Attack)"

3.2. Selecting Best Features via Olympiad Optimization Algorithm

After the dataset is generated, the next step is related to developing a binary alternative to the Olympiad optimization algorithm, namely, BOOA. This algorithm is aimed at selecting features, particularly detecting optimal features from the training dataset. This procedure is carried out before the related classification model is generated using ML algorithms. OOA, which is considered a swarm-based imitation algorithm, applies both global and local search procedures. Using a divide-and-conquer methodology and performing as a population and group-oriented heuristic method, this novel alternative to the Olympiad optimization algorithm sorts out optimization problems. Indeed, every single member of the OOA population emulates a student's learning behavior while preparing for an Olympiad exam [21]. In this situation, each 'student' is denoted by a vector with a length of 13, which reveals the number of features. The structure of a student in the proposed BOOA is depicted in Figure 2. Firstly, the value of each cell is set to 1, which indicates that all features are included in the training. Student vector changes in a population, including n rows and 13 columns. The population varies via iterative teaching and learning interactions among population members (students). Algorithm1 indicates the pseudo-code of the suggested OOA for identifying the best features of the SQLi dataset.

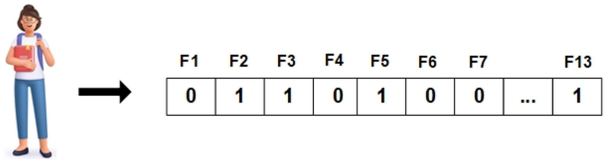


Figure 2. The agent's structure in the BOOA as a binary array for specifying features.

Functioning as a divide-and-conquer algorithm, the OOA integrates local and global search techniques. As depicted in Algorithm 1, individuals, known as students within the whole population, go through sequential division, which leads to equal subgroups. Algorithm 2 sketches the learning operator as the primary segment of the OOA, which is implemented in MATLAB. The respective subgroups apply unique imitation methods to examine different regions in a broad solution environment. There is a competitive learning context among these individuals. All individuals are supplied with a memory to save their learning development. Students within the OOA are structured and run in the form of a numerical array, collectively creating a student population that denotes solutions. The

categorization of the population into n teams is regarded as the first OOA phase, which follows sorting. Each team consists of m students, and the initial team is considered the global best, and the last team is regarded as the global worst. Each team examines its local solution environment; the first student in each team is selected as its respective best student.

Students within the population are arranged into teams. They attempt to learn from the best students in the adjacent teams (local students) or the global best team. The recommended learning operator includes both global and local search devices within the OOA. It operates as the main operator for finding the optimal solution. The above-mentioned operator is designed to improve population knowledge (fitness) and sort individuals according to their knowledge (fitness function). By applying a learning operator, which includes four key steps, the OOA is intended to optimize population knowledge. In the first step, students within each team integrate and comprehend the knowledge of related students in the neighboring team. This stage requires knowledge transmission from one team to another team in a serial manner. If weaker students within the team are unable to enhance their knowledge, the second optimization stage will be run to mutate the best student in the team to create diversity. The mutation operation requires a local search for the best student in the weaker team. If progress is not made in the second stage, the third stage begins. That is, all students within the weak learner team received lessons from the global best team (the first team).

Algorithm 1. Pseudo code of the BOOA [21].

```
BCOOA Algorithm {
1.   Initialize the first population;
2.       itr=0;
3.       for i=1 to maxitr do
4.       {
5.            Partition population into n team
6.            i=2;
7.            while (i<=n)
8.            {
9.                 Imitation of team(i) from team (i-1);
10.                  i = i + 1;
11.           }
12.           for i=1 to n do in parallel
13.           {
14.           imitation of team(i) members from the local best member;
15.           }
16.           for i=1 to n do in parallel
17.           {
18.                mutation of team(i)'s best student;
19.           }
20.           mutation of global best student;
21.           merging teams into a single total population;
22.           Evaluating the fitness of students by Equation (4);
23.      }
24.      return the global best as the best selected features;
25.  }
```

If students from the global team fail to transfer knowledge to the worst team, the mutation operator is initiated for the globally best student. Owing to presenting minor diversity, the mutation of the globally best student is intended to prevent local optima. Finally, individuals (search agents) from diverse teams collaborate to generate a new population. Iterative application of the learning operator occurs in the student population. Algorithm 2 shows the pseudo-code for the learning operator run in each OOA iteration. In the proposed approach, the learn operator (also referred to as the imitation operator) is

actualized through a crossover operation, which involves replacing certain bits (cells) in the weakest individual with the corresponding elements from the best individual (student). The LearnCount parameter denotes the number of bits from the weaker element to be substituted by equivalent bits from the stronger element. The optimal value for LearnCount is determined empirically.

Algorithm 2. Learning operator's algorithm in the in the OOA.

```
Function studentType Olampiyad_Learn(BestStd, WorstStd)
{
    nVar = length (Student_array);
    LearnCount = ceil ((30 * nVar) / 100); % imitation count
    count = 0;
    i = 0;
    NewStd = WorstStd;
    while (count < Learn Count and i < nVar)
    {
        aL = randi (nVar);
        if (WorstStd(aL) != BestStd (aL))
        {
            NewStd (aL) = BestStd (aL);
            count = count + 1;
        }
        i = i + 1;
    }
    Return (NewStd);
}
```

3.3. Feature Selection via Olympiad Optimization Algorithm

A novel binary technique according to the Sigmoid function was developed and put forth for OOA. It should be noted that a serial number is included within each solution (student) in the population of this technique. Since the problem involves choosing a feature, the binary reply must be either 0 or 1. If the feature value is 1, it will be designated for the new dataset; otherwise, it will not be chosen if its value is 0. The purpose of using the Sigmoid relation [23] was to modify the values of the individual array into binary values. The BOOA aims to select useful features for ML algorithms. Regarding certain issues, such as feature selection, the solutions are limited to binary digits and generate binary alternatives. Learn the algorithm (Algorithm 2) in the BOOA and revise and refresh the student array. Every student is intrigued by the algorithm to be the best student. The location of the ith student regarding the d dimension and t iteration is denoted by $\vec{X}_{\mathrm{i}}^{d}(t)$ vector. Hence, as depicted in Equation (1) and according to the developed model, the Sigmoid function is used as a binary pattern for modifying the location of the solution in the OOA. The Sigmoid transfer function produces an output of 0 or 1. Indeed, a threshold value is required to convert the output into a binary (discrete) value. A random threshold value depicted in Equation (2) is included within the Sigmoid function for selecting the feature by changing it into a binary solution. Owing to Equations (1) and (2), the existing solutions for the BOOA population are obliged to change within a binary (discrete) search space. Next, the equations are meticulously fed into the BOOA.

$$SG\left(X_i^d(t)\right) = \frac{1}{1+\mathrm{e}^{-X_i^d(t)}} \tag{1}$$

$$X_i^d(t+1) = \begin{cases} 0 & if\ rand < SG\left(X_i^d(t)\right) \\ 1 & if\ rand \geq SG\left(X_i^d(t)\right) \end{cases} \tag{2}$$

Then, regarding classification problems, the BOOA is adjusted to choose features. Based on the principle of permutation and the feature vector with n binary value, there exist 2^n varying feature alternatives which generate a huge and banal space; all 2^n options should be thoroughly investigated. In this manner, the BOOA is adjusted to examine the matching feature space to discover the ideal feature mixture. It should be noted that the ideal feature combination is the one with the highest classification efficacy and the lowest number of feature selections. Equation (3) is used to measure the fitness function; this function is applied for appraising individual positions in the BOOA. Equation (3) can be expressed as follows:

$$Fitness = \alpha\gamma_R(D) + \beta\frac{|C-R|}{|C|} \tag{3}$$

In Equation (3), $\gamma_R(D)$ refers to the categorization quality of features, which is specified with decision D. Furthermore, C denotes the general number of features. The variable R indicates the number of selected features. $\frac{|C-R|}{C}$ refers to the ratio of features that were not selected. Moreover, α and β are regarded as coefficients and denoted as $\alpha \in [0,1], \beta = 1-\alpha$. The fitness function is aimed at enhancing the classification quality $\gamma_R(D)$. The value of $\frac{|C-R|}{C}$ shows the proportion of the unselected features to the total features. The classification error rate is substituted with a fitness function to create a minimization function. In addition, rather than using the number of unselected features (Equation (4)), the fitness function uses the number of selected features. In Equation (4), $E_R(D)$ denotes the error rate; $|R|$ indicates the number of selected features; also, $|C|$ refers to total features. Furthermore, α and β indicate constant values for checking classification accuracy; they are aimed at decreasing the quantity of the selected features in this way: $\alpha \in [0,1], \beta = 1-\alpha$.

$$Fitness = \alpha E_R(D) + \beta\frac{|R|}{|C|} \tag{4}$$

4. Experiment Results and Discussion

At the outset, the required scalar (numeric) training dataset was developed to investigate and examine the proposed method. After that, the proposed feature selection (BOOA algorithm) was executed in MATLAB. ANN, DT, SVM, and KNN algorithms were run in two ways. In the first style, the ML algorithms were run independently of the selective features. However, in the second style, the ML algorithms were run by applying feature selection algorithms. Two various classification models were developed in the experiments. Firstly, ML algorithms train the classification model by capitalizing on the dataset, including all traits (features). In the 2nd run, machine learning (classification) algorithms activate and operate the BOOA to choose optimal features. Next, by capitalizing on the dataset with optimal features, machine learning algorithms are applied to train the classification model. The experiments have been conducted on a computer with Intel (Core i7, 3.4 GHz), 16 GB, and Windows 11.

4.1. SQLi Dataset

In this paper, the SQLi dataset was aimed at training and testing the ANN, DT, SVM, and KNN algorithms. Consequently, 70% of the dataset was devoted to training the model, and only 30% was employed to test the developed classification model. Furthermore, 30% of the training dataset was applied throughout the testing step. The chosen training and test datasets are characterized by an unvarying distribution; they include normal and malicious queries.

These features are specified and determined according to the significant features in the SQL injection and the allocated weight to each feature concerning their significance. In this study, nominal features, namely, query length, number of parentheses, number of AND operators, number of special characters, number of UNION commands, and other features, were taken into account. Applying numerical features for training machine learning algorithms is preferably recommended to enhance the efficiency of the classification

models. Thus, the nominal features extracted from the initial dataset were transformed into numerical features. The format and datatype of the dataset features are shown in Tables 2 and 3. Features were chosen for each query, and the row of the dataset refers to the features of the query (see Table 2). Moreover, the last column in the table shows that each row (query) may be either malicious or normal. In this paper, the converted dataset was applied for training and testing machine learning algorithms. The specifications of the numeric training dataset are listed in Table 4.

Table 4. Characteristics of the dataset.

Dataset	Total Samples	Normal Samples	Attack Samples
# Normal and attack queries	1027	473	554

To implement and execute the proposed method, MATLAB version 2020b was utilized. Some parameters such as *accuracy*, *precision,* and *sensitivity* were taken into consideration as the classification criteria that were applied to investigate the proposed method. These metrics were applied as the fitness function in the BOOA. The parameters regarding the developed binary OOA were experimentally calibrated throughout the experiments. The highest values for the BOOA parameters are given in Table 5.

Table 5. Values for the BOOA parameters.

Calibration Parameter	Optimal Value
Population Size(#students)	40
# teams	10
Rate of Learning	Random values [0.2–0.8]
Size of teams	4
Max iterations	100
Imitation count	1

4.2. BOOA-Based Feature Selection for SQLi Detection

The algorithm proposed in this paper was run and executed in a scenario referred to as BOOA; it was executed on a specific dataset to specify the effectiveness of the decision tree, neural network, SVM, and K-Nearest Neighbor algorithms. The following standards and parameters were taken into consideration for evaluating the proposed method:

- Accuracy
- Sensitivity
- Precision
- F1
- Number of selected features.

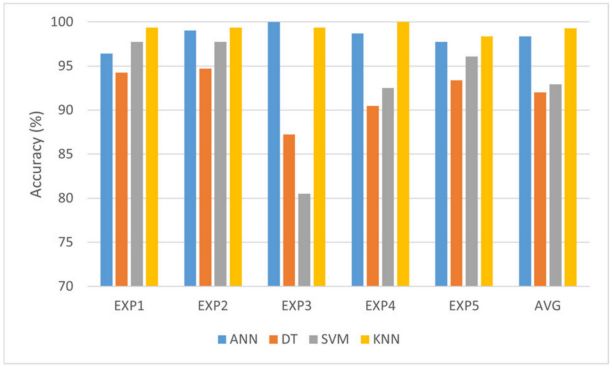
The above-mentioned parameters were taken into consideration as the BOOA fitness function. Numerous investigations and experiments were carried out on the above-mentioned data to answer the following research questions:

- RQ1: Is the developed feature selector (BOOA) significantly effective in terms of accuracy, sensitivity, precision, and F1 parameters of SQLi detectors?
- RQ2: Is the developed feature selector significantly effective in reducing the number of required features in the training dataset?
- RQ3: Is the developed feature selector significantly stable as a heuristic-based method?
- RQ4: Is the success rate and convergence speed of the proposed heuristic-based feature selector reliable?

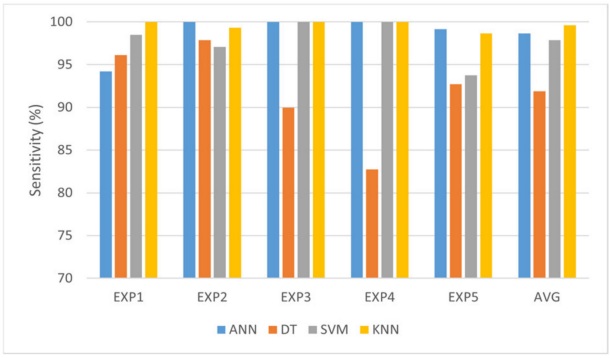
In this paper, two sets of ML experiments were carried out; firstly, ANN, DT, SVM, and KNN were operationalized on the entire dataset, which included 13 features. However, in the 2nd experiment, The ML algorithms were implemented on the optimal dataset, which was filtered with the selective features by the BOOA. The obtained results were analyzed and investigated to determine the efficacy of the suggested feature selector in the SQL injection dataset.

4.3. The Performance of the SQLi Detectors without the Feature Selector

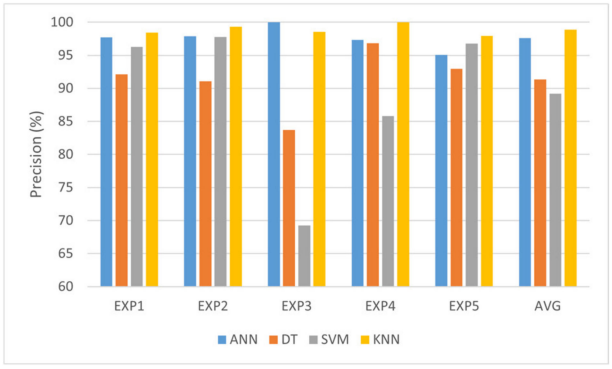
In the first set of experiments, only the classification algorithms (ANN, DT, SVM, and KNN) were used to generate an SQLi classifier using the generated training dataset. In the experiments, all features were applied in the training phase. Figure 3 shows the results of five sets of experiments investigating the efficacy of the SQLi detectors created by different ML algorithms, regardless of the proposed feature selector. It was observed that the SQLi classifier generated by KNN and ANN operated more efficiently than the classifier developed by the SVM and DT algorithms in terms of accuracy, sensitivity, precision, and F1. Indeed, the SQLi detectors created by KNN and ANN perform better than the other SQLi detectors. On the other hand, the SQLi detector generated by DT has a lower efficacy.



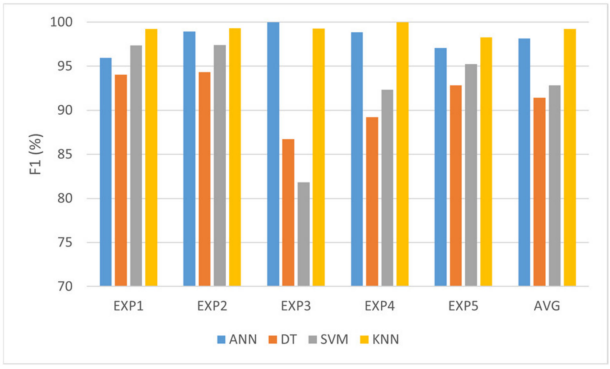
(**a**) The accuracy value of the created SQLi detectors by the ML algorithms



(**b**) The sensitivity of the created SQLi detectors by the ML algorithms



(**c**) The Precision value of the created SQLi detectors by the ML algorithms



(**d**) The F1 value of the created SQLi detectors by the ML algorithms

Figure 3. The average performance of the SQLi detectors developed by different ML algorithms, regardless of feature selection, during five runs.

Figure 4 shows the time required for classifier creation (training and testing) using different ML algorithms. Regarding the results, the time required by the ANN is higher than that of other algorithms. The average execution time of the KNN in the dataset is lower than that of the ANN and DT. At the same time, the efficacy of the created SQL detector by the KNN is superior to that of other detectors in terms of accuracy, sensitivity,

precision, and F1. Overall, the effectiveness and efficiency of the KNN without a feature selector are higher than those of the other ML algorithms in the SQLi classification problem.

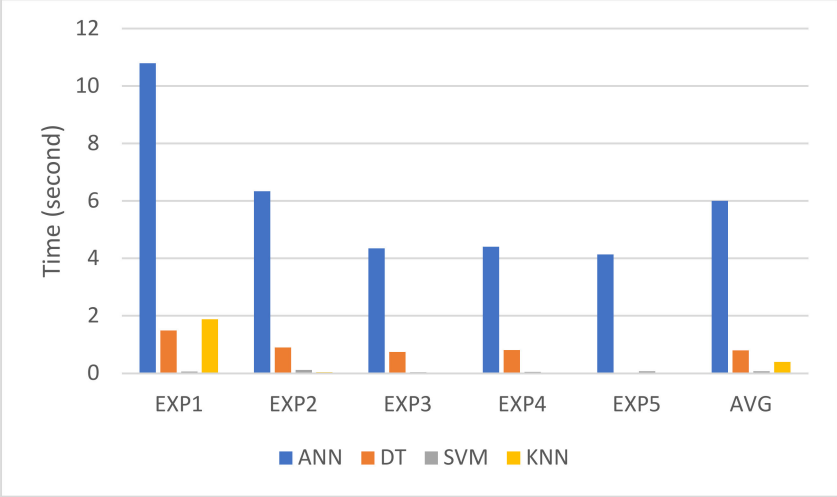


Figure 4. The time required for SQLi classifier creation by different ML algorithms, regardless of feature selection, during five runs.

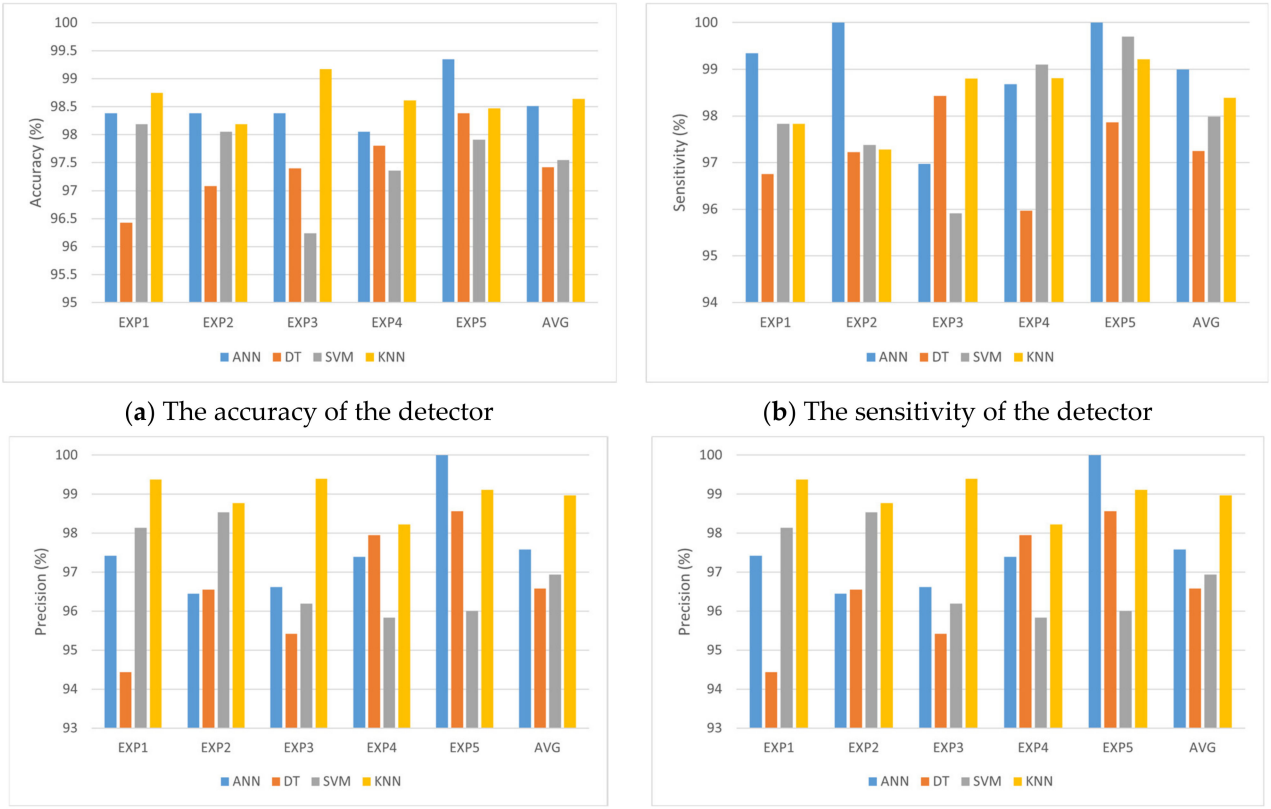
4.4. Standards for Evaluating the Proposed Method

Numerous experiments were carried out to evaluate the performance of the suggested SQLi detector using a feature selector (RQ1). The proposed BOOA algorithm was implemented to select the optimal features in the SQLi dataset. The dataset filtered by the BOOA was applied to train different ML algorithms. After that, the test data, which were selected from the SQLi dataset, was utilized to test the generated SQLi detectors. The effectiveness of the BOOA algorithm regarding the performance of the SQLi detectors created by the ANN, SVM, DT, and KNN is depicted in Figure 5. The results revealed that the accuracy of the generated SQLi detector regarding BOOA + ANN was approximately 98.50%, whereas this figure is about 98.37%. Moreover, the sensitivity of the generated SQLi detectors by ANN and KNN with BOOA were 98.99% and 98.39%, respectively. However, the sensitivity of the BOOA + ANN was notably better than that of the other algorithms.

Regarding the results shown in Figure 5.c, the precision of the SQL detectors has increased significantly. The precision of the generated SQLi detectors by BOOA + ANN is 97.57%. Meanwhile, the precision of the generated SQLi detector by the BOOA + KNN is about 98.97%. The precision of the SQLI detectors created by ANN and KNN is higher than that of the other detectors. The F-measure indicates the harmonic mean of the recall and precision of the generated SQLi classifiers. Figure 5.d illustrates the F1 value for the generated SQLi detectors by the BOOA + Ml algorithms. The ML algorithms perform better for the selected features. In the five experiments that were conducted, the SQLi created by BOOA + ANN and BOOA + KNN performed better in terms of F1. The F1 of the generated SQLi detector by the BOOA + ANN and BOOA + KNN is 98.42% and 98.52%, respectively.

The average accuracy, sensitivity, precision, and F1 of the created SQLi classifiers are depicted in Figure 6. The average value of accuracy of the created SQLi detectors without the feature selector is about 95.64%, while this figure for the SQLi detector created by BOOA + ML is about 98.02%. Indeed, the suggested feature selector has a considerable impact on the accuracy metric. The suggested feature selector identifies the most effective traits (features) in the training dataset and eliminates the effectless features. The effectless features of the training dataset may have negative effects on the performance of the SQLi classifier. The average sensitivity of the SQLi detectors without a feature selector on the raw dataset is about 96.99%, whereas this figure for

the suggested SQLi detector that utilizes the feature selector, is about 98.15%. Similarly, the average precision of the suggested SQLi detector was enhanced from 94.24% to 97.51%. The average value of F1 for the suggested SQLi detector is enhanced to 97.81% as a result of using the feature selector. Overall, the average performance of the feature-selection-based SQLi detector is considerably higher than that of the detectors that were created on the raw dataset (without feature selection.



(**a**) The accuracy of the detector

(**b**) The sensitivity of the detector

(**c**) The precision of the detector

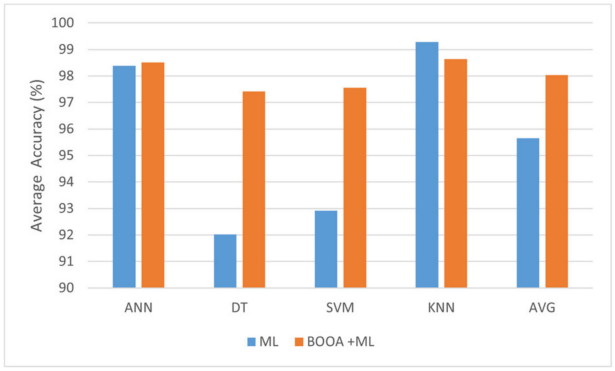
(**d**) The F1 of the detector

Figure 5. The performance of the generated SQLi detectors produced by BOOA + ML throughout five runs.

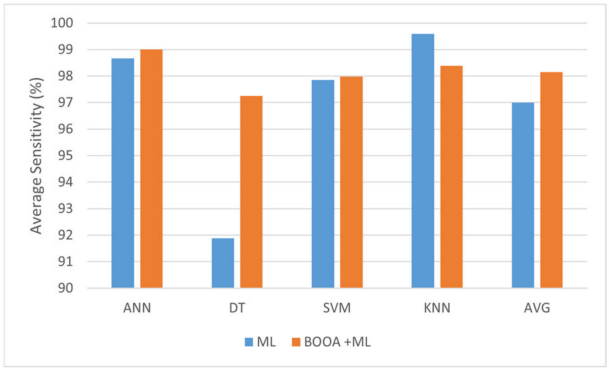
The average accuracy obtained for the generated models by the BOOA + ANN (filtered ANN) was better than that of the other algorithms. The proposed feature selection algorithm (BOOA) had a more significant impact on the accuracy of the classifiers generated by the ANN. In a similar vein, the average precision of the SQLi detector generated by BOOA + ANN is better than that of the models generated by other algorithms. The average precision values of the developed SQLi detection by various training algorithms are depicted in Figure 6. The obtained fitness values of the suggested feature selector, along with those of the ANN, DT, SVM, and KNN (ML classification algorithms), are shown in Figure 7.

The fitness value includes the number of selected features and the error rate of the generated SQLi detector by the selected features. The lower the fitness value, the lower the error rate of the SQLi detector. The generated SQLi detectors by the ANN and DT (classification algorithms) have lower fitness values. A lower fitness value indicates a lower quantity of selected features and a higher accuracy of the generated SQLi detector. Eliminating the effectless features and enhancing the overall accuracy of the generated SQLi detector are the main objectives of the suggested BOOA-based feature selectors. Figure 8 shows the required feature selection and training time by the suggested BOOA for different classification algorithms. The average feature selection

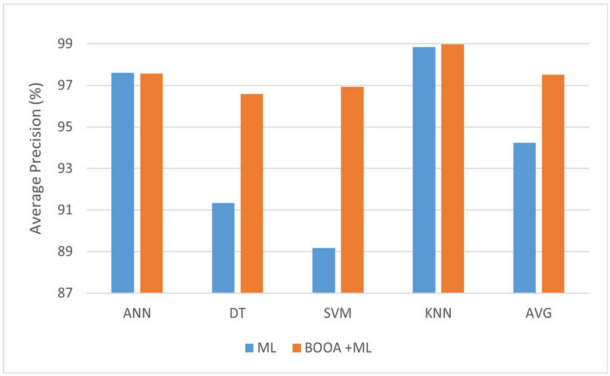
time in the BOOA + ANN is about 365.48 milliseconds. The feature selection time of the BOOA in DT, SVM, and KNN is considerably lower than that of AAN.



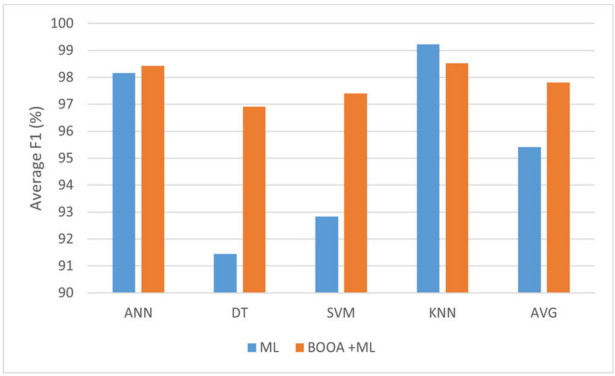
(**a**) The AVG. value of accuracy in the SQLi detectors



(**b**) The AVG. value of sensitivity in the SQLi detectors



(**c**) The AVG. value of the precision in the SQLi detectors



(**d**) The AVG. value of the F1 in the SQLi detectors

Figure 6. Comparison of the average performance of the generated SQLi detectors produced by ML and the BOOA + ML.

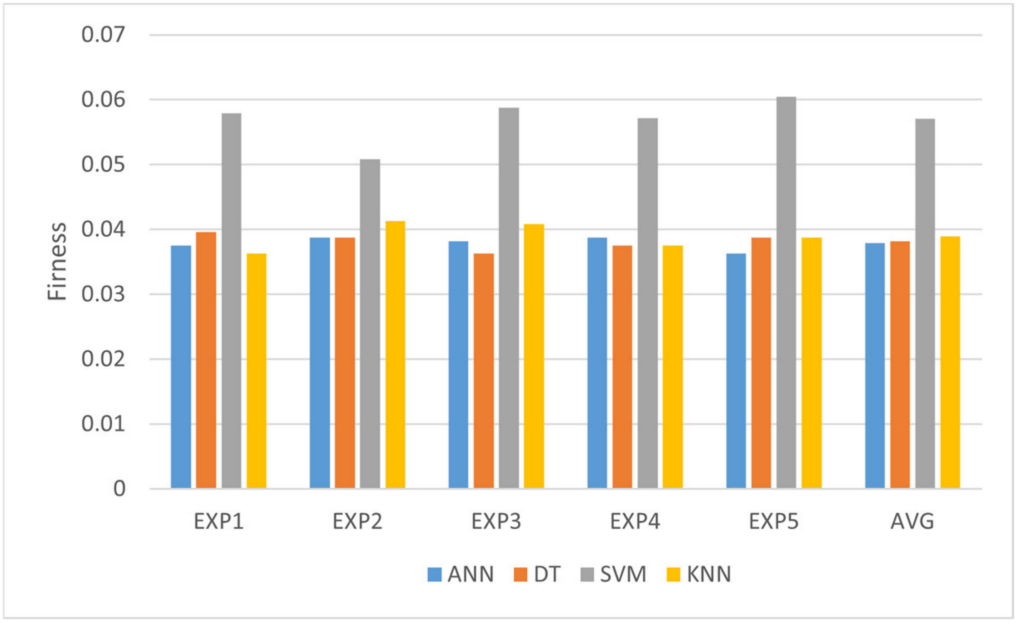


Figure 7. The fitness of the suggested BOOA as a feature selector in different ML algorithms.

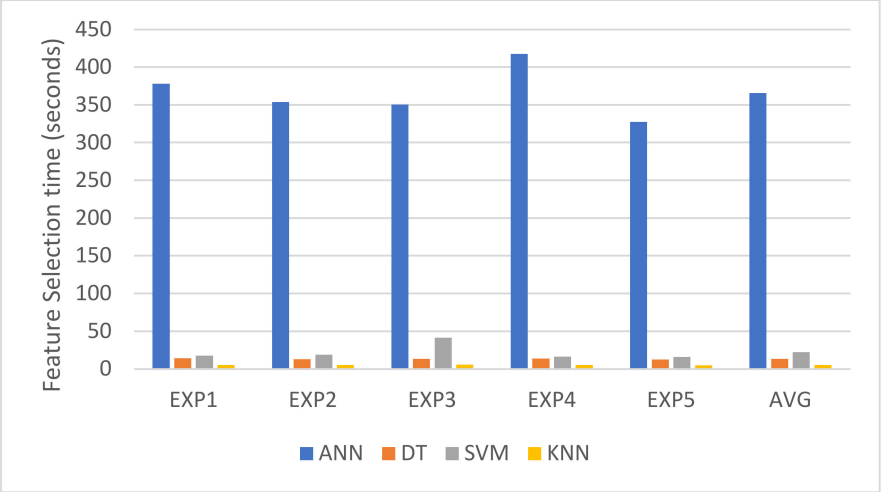


Figure 8. The feature selection and training time by the suggested BOOA in different ML algorithms.

4.5. The Impact of BOOA in Reducing the Quantity of Features (RQ2)

The features designated by the BOOA for the best implementation (the optimal fitness) during the five runs are given in Table 6. It was found that the average number of features chosen by the BOOA is fewer than those selected by the BOOA. The number of the selected features by the BOOA during several implementations is depicted in Table 6. The findings revealed that BOOA outperformed the pure ML models in terms of the number of selected features. Table 6 shows the selected features and calculated fitness using the suggested BOOA in the different classification algorithms. The average quantity of the selected features and the average fitness value are indicated in Table 6. On average, the number of features selected by the BOOA was about 3.6 when it was used along with the ANN. The number of the selected features by the BOOA in the DT, SVM, and KNN are 3.2, 3.8, and 3.2. The average number of features selected by the BOOA + ANN is higher than that of the other algorithms. The average fitness of the SQLi detectors created by the BOOA + ANN is 0.0378, which is lower (better) than that of the other algorithms. The generated SQLi detector by the BOOA + ANN performs better than the other detectors. The BOOA + ANN selected four features (1, 2, 4, and 12) as the best features of the raw dataset. The best fitness obtained by the BOOA + ANN in the conducted experiments is 0.03627. The accuracy of the best SQLi detector is 99.35%. The selected features by the BOOA + ANN in the best execution (lowest fitness) are as follows:

- Nesting Level in the SQL query
- Length of SQL query
- Number of constant values in the SQL query
- Number of parentheses in the SQL query.

Table 6. The selected features selected by the BOOA and the obtained fitness in different ML algorithms during five executions.

	BOOA + ANN		**BOOA + DT**		**BOOA + SVM**		**BOOA + KNN**	
	Selected Features	**Fitness**	**Selected Features**	**Fitness**	**Selected Features**	**Fitness**	**Selected Features**	**Fitness**
EXP1	2, 4, 6	0.0375	1, 4, 5, 12	0.0395	1, 3, 4, 10	0.0579	2, 4, 6	0.0362
EXP2	3, 4, 6	0.0387	1, 4, 6	0.0387	2, 4, 9, 12	0.0508	1, 4, 12	0.0412
EXP3	3, 4, 6, 12	0.0381	2, 4, 6	0.0362	1, 2, 4	0.0587	2, 4, 6, 12	0.0408
EXP4	3, 4, 6, 12	0.0387	1, 4, 12	0.0377	2, 4, 6, 10	0.0571	1, 4, 6	0.0375
EXP5	1, 2, 4, 12	0.0362	1, 4, 12	0.0387	2, 4, 6, 9	0.0604	1, 4, 12	0.0387

In the worst-case execution of the SQLi detector created by BOOA + ANN, only four features (3, 4, 6. 12) have been selected. The worst fitness value is 0.03877. Indeed, the suggested SQLi detector using BOOA produces similar results in the best and worst-case executions.

Given the results depicted in Tables 7 and 8, it should be highlighted that the developed SQLi detection by the BOOA + ANN notably outperformed the developed SQLi detector created using the raw dataset. Ineffective and overlapping features were selected by BOOA. Table 7 shows the average number of the selected features by different SQLi detectors during five times executions. Furthermore, the average fitness values calculated by different methods during five executions are shown in Table 7. In the testing phase, the fitness of the selected features by BOOA was investigated. The accuracy, sensitivity, precision, and F1 of the trained model were taken into consideration in investigating the fitness of the selected feature. The raw training dataset consists of 12 features. Even though the number of chosen features by BOOA + ANN is not minimal, it manages to provide the highest performance in terms of accuracy, sensitivity, and precision. In general, owing to the selection of 30% of the best features, the proposed technique was able to enhance the effectiveness of the detection system. On the whole, merging ANN and BOOA led to the development of the SQLi detection model, which outperformed the other algorithms. In line with the findings of the study, the BOOA outperformed the other techniques in terms of sensitivity, accuracy, and precision. Moreover, it takes less time than the other techniques to select features. Thus, the proposed method significantly reduces the time required to detect the attacks injected by SQL.

Table 7. The average number of the features selected by the BOOA and the average value of fitness obtained by the BOOA + ML algorithms.

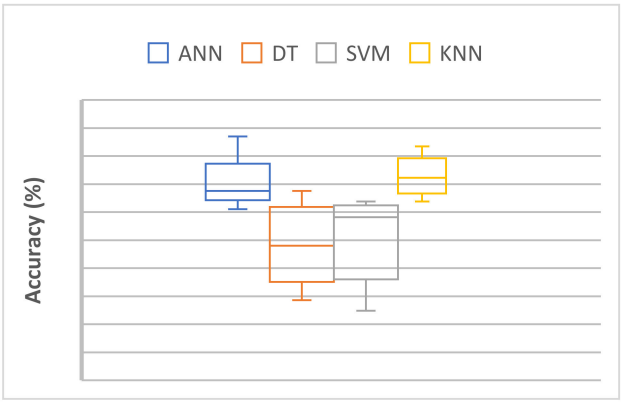
	BOOA + ANN	**BOOA + DT**	**BOOA + SVM**	**BOOA + KNN**
AVG Num of Selected Features	3.6	3.2	3.8	3.2
AVG Fitness	0.0378	0.0381	0.0570	0.0389

Table 8. Comparison of the performance of different SQLi detection methods on the same dataset.

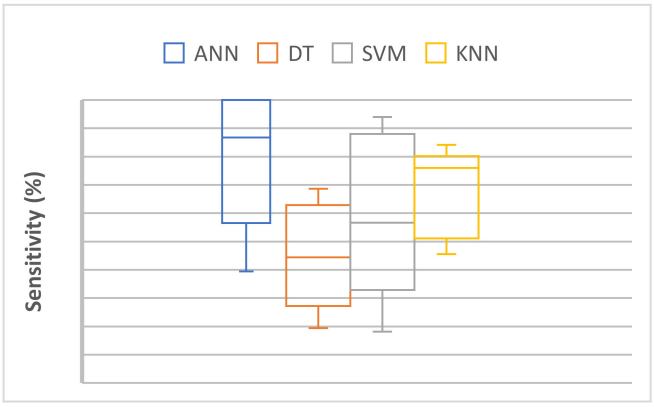
	Random Forest	**Two-Class SVM**	**Naïve Bayes**	**Logistic Regression**	**AVG. BOOA + ANN**
Accuracy	93.6%	98.60%	93.30%	95.10%	98.50%
Sensitivity	57.7%	99.80%	89.00%	56.00%	98.99%
Precision	77.4%	97.40%	100.00%	98.50%	97.57%
F1	66.0%	98.60%	87.20%	71.30%	98.42%

4.6. Stability of the Suggested Method (RQ3)

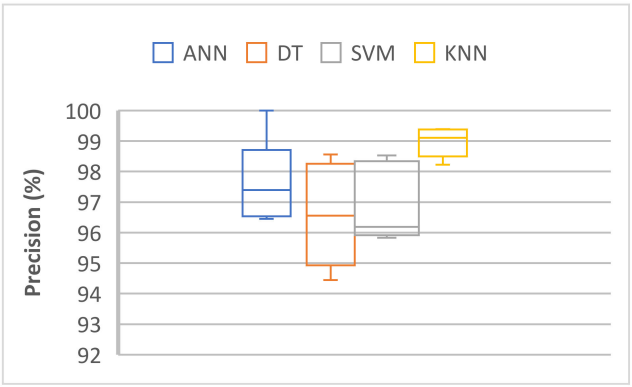
The stability of the suggested feature-selecting algorithm was taken into consideration in the third research question of the paper. The stability of metaheuristic algorithms should be investigated under various circumstances (according to their best, average, and worst outputs during various executions). The developed BOOA was executed at different times under identical conditions to investigate and examine the best, average, and worst outputs. Figure 9 shows the variation and distribution in the values of the obtained performance criteria (accuracy, sensitivity, precision, F1, and fitness) during five times executions.



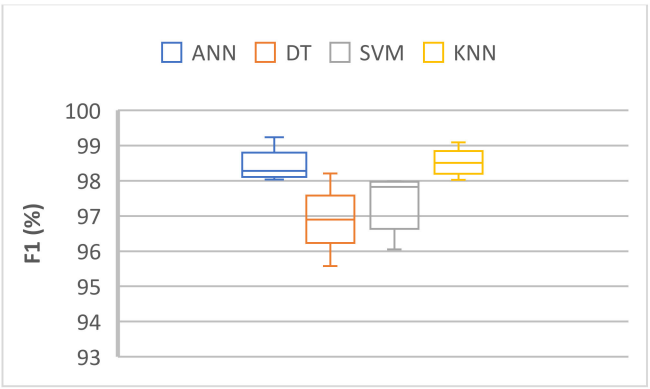
(**a**) The distribution and variation of the accuracy values obtained from five times executions of the SQLi detectors



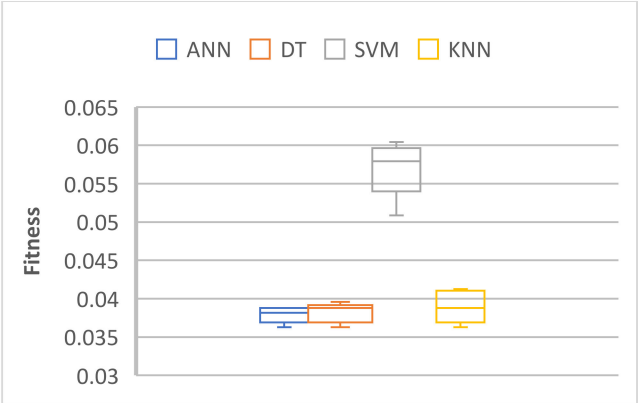
(**b**) The distribution and variation of the sensitivity values obtained from five times executions of the SQLi detectors



(**c**) The distribution and variation of the precision values obtained from five times executions of the SQLi detectors



(**d**) The distribution and variation of the F1 values obtained from five times executions of the SQLi detectors



(**e**) The distribution and variation of the fitness values obtained from five times executions of the SQLi detectors

Figure 9. The variation in the values of the evaluation criteria obtained from five times executions of the generated SQLi detectors by the BOOA + ANN, BOOA + DT, BOOA + SVM, and BOOA + KNN.

The five values of the accuracy parameters obtained by the SQLi detectors generated via various machine learning algorithms are depicted in Figure 9.a. As shown in this figure, BOOA + ANN and BOOA + KNN were able to achieve higher accuracy values with lower distributions than the other methods. Furthermore, the sensitivity parameter for the generated SQLi detector obtained via various algorithms in five times executions is depicted in Figure 9.b. In line with these findings, it can be maintained that ANN produced better sensitivity than the other methods once it is applied together with BOOA.

The obtained values for the precision parameter of the generated SQLi detectors by several algorithms at five executions are shown in Figure 9.c. It was observed that the ANN and KNN achieved better precision values as soon as the training dataset was filtered using BOOA. The standard deviation among the precision values of the SQLi detectors created by the BOOA + ANN and BOOA + KNN is lower than that of the other methods. The SQLi created by DT had lower precision and higher standard deviation (distribution). Regarding the results shown in Figure 9.d, the generated SQLi by the BOOA + AAN has a higher F1 value and lower standard deviation. In contrast, the SQLi detector created by DT has a lower F1 value and a higher value variation in different executions.

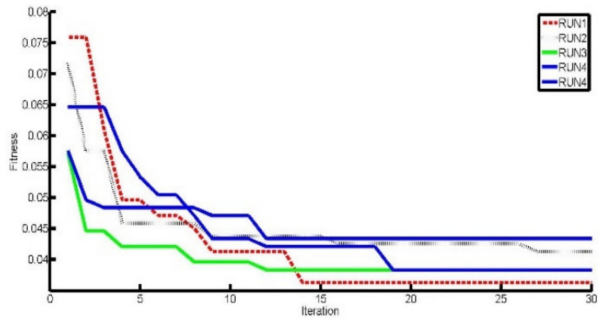
All in all, the closer values were generated for accuracy, sensitivity, precision, and F1 by the suggested SQLi detector using BOOA + ANN. The lower standard deviation among the results obtained by BOOA + ANN in different executions indicates a higher stability of the suggested feature selector (BOOA). In the conducted experiments, the best accuracy (99.35%) was provided by the BOOA + ANN. Furthermore, the best values of sensitivity, precision, and F1 were 100%, 100%, and 99.23%, respectively, which were obtained by the BOOA + ANN. The findings demonstrate the notable effectiveness of the developed BOOA as a feature selection method for enhancing the accuracy, sensitivity, precision, F1, and stability of the generated SQLi detector. The dataset filtered by BOOA contained four features (1, 2, 4, and 12) in the best-case execution and three features (2, 4, and 6) in the worst-case execution. Moreover, it was found that applying the BOOA algorithm decreases the discrepancy between the accuracy of the generated detectors in the best and worst cases. In general, the findings indicate that in the SQLi classification problem, combining BOOA and ANN produced better results than those of other algorithms.

4.7. Stability of the Suggested Method (RQ3)

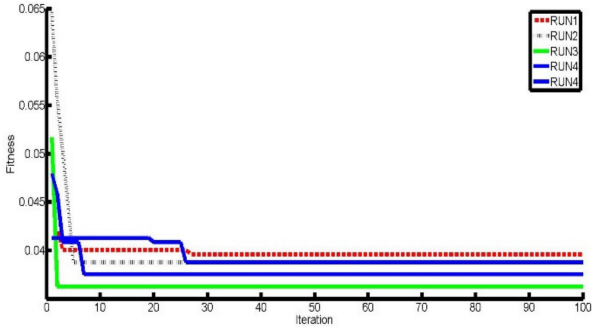
A set of experiments for investigating the success rate and convergence of the proposed method have been performed. Figure 10 indicates the convergence of the suggested BOOA in different ML algorithms during five times executions. The results show the convergence of the BOOA in finding the best features in the SQLi dataset. The results shown in Figure 10 are based on the fitness function. The fitness function was calculated using Equation (4). A lower value of the fitness shows the minimal quantity of the selected features and the lower error rate of the classifying SQL queries in the test set.

Table 8 compares the performance of different SQLi detection methods on the same dataset. The suggested BOOA + ANN has a higher performance and lower detection time. In the suggested method, the ML algorithms were conducted on the effective dataset, which includes 30% of the best features of the main dataset. The effective features were selected by the developed BOOA. The suggested method has a lower detection time than the other methods.

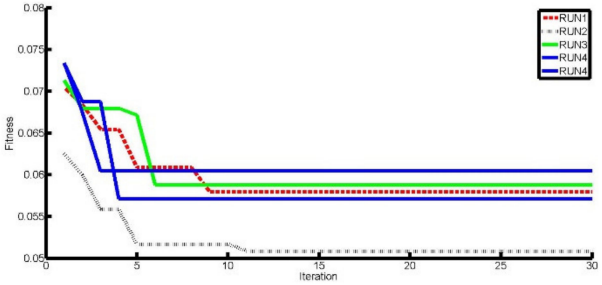
Deep learning (DL) is a subset of machine learning (ML) that uses neural networks with multiple layers to model complicated patterns in data. DL models are especially useful for jobs requiring large datasets and high-level abstraction, such as image and speech recognition. Deep learning algorithms frequently use numerous hidden layers in neural networks to automatically learn hierarchical data representations. ML models are frequently effective for tasks with small data. They are easier to fine-tune and optimize for certain activities. Regarding the size of the utilized dataset and the complexity of the SQLi detection problem, the ML algorithms were selected in this study. Table 9 shows the adjusted values for the parameters of the utilized ML algorithms.



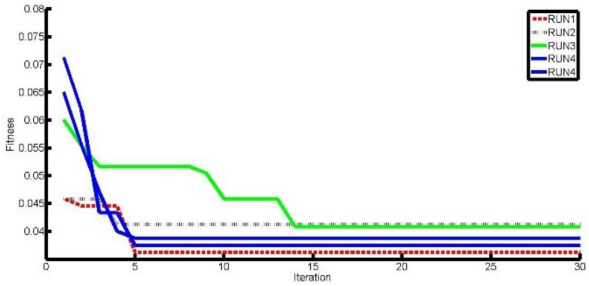
(**a**) The convergence of the BOOA along with the ANN



(**b**) The convergence of the BOOA along with the DT



(**c**) The convergence of the BOOA along with the SVM



(**d**) The convergence of the BOOA along with the KNN

Figure 10. Comparison of the convergence speed of the BOOA-based feature selector along with the ML algorithms.

Table 9. The calibrated values of the parameters of the ML algorithms in the experiments.

ANN Parameter Setting		
Ns	Num. of Records	Num of records in Training Dataset
Ni	Num. of Features	Num of features in Training Dataset
No	Num. of output Neurons	1
alpha	scaling factor to calculate the number of hidden neurons	2
Nh	Num. Neurons in the hidden layer	round(Ns/(alpha * (Ni + No)));
SVM Parameter Setting		
C	Regularization Parameter	1
MaxIter	Maximum iteration	100,000
Cache Size	Kernel Cache Limit	50 MB
DT Parameter Setting		
D	Maximum Depth	Unlimited
Sp	Minimum Samples for Splitting	10
Ms	Minimum Samples per Leaf	1
Mf	Maximum Features	12 (all features)
KNN Parameter Setting		
K	Number of Neighbors	[3–5]
W	Weighting of Neighbors	“Equal”
DM	Distance Metric	“Euclidean”

5. Conclusions

In this paper, a feature selection-based technique for identifying SQL injection attacks was proposed. A binary variant of the Olympiad optimization algorithm was developed to select the most effective features of the SQLi dataset. The findings of this study revealed that the proposed BOOA + ANN can produce 99.34% accuracy, 100% precision, and 100% sensitivity. By selecting 30% of the best features, the proposed method managed to enhance

the effectiveness of the attack detection systems. It should be highlighted that selecting a limited number of effective features is regarded as one of the main advantages of the proposed method. In addition, better accuracy, precision, sensitivity, and F1 are considered as the other merits of the method.

Studying generated feature selection algorithms based on deep learning methods is recommended as another direction for further research. Utilization of chaos theory [24] to enhance the feature selectors' performance in the SQLi detection problem. Taking the API-level features into account may result in performance enhancement. Another direction for further research, machine learning and heuristic algorithms proposed in [25–27], can potentially be applied to develop efficient SQLi detection models. Future studies could also explore how cloud computing can leverage various learning methods, deep learning, and boosting algorithms [28,29] to effectively detect SQL injection attacks. Finally, generating a binary variety of other effective heuristic algorithms and studying their performance in the SQLi detection area are regarded as other potential directions for further research.

Author Contributions: Conceptualization, B.A. and A.B.; methodology, B.A.; software, S.S.S.; validation, B.A., A.B. and R.C.; formal analysis, B.A.; investigation, B.A.; resources, B.A.; data curation, S.S.S.; writing—original draft preparation, S.S.S.; writing—review and editing, B.A.; visualization, R.C.; supervision, R.C; project administration, R.C.; funding acquisition, R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is (also) supported by NetZeRoCities competence center, funded by European Union—NextGenerationEU and Romanian Government, under National Recovery and Resilience Plan for Romania, contract no. 760007/30.12.2022 with Romanian Ministry of Research, Innovation and Digitalisation, through specific research project P5-Smart City and Digital Twins. This work was supported by the Project "Mobility and Training foR beyond 5G ecosystems (MOTOR5G)" funded by the European Union's Horizon 2020 Program under the Marie Skłodowska Curie Actions (MSCA) Innovative Training Network (ITN) under Grant 861219.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Marashdeh, Z.; Suwais, K.; Alia, M. A Survey on SQL Injection Attacks: Detection and Challenges. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 957–962.
2. Huang, H.-C.; Zhang, Z.-K.; Cheng, H.-W.; Shieh, S.W. Web Application Security: Threats, Countermeasures, and Pitfalls. *Computer* **2017**, *50*, 81–85. [CrossRef]
3. Sefati, S.S.; Fartu, O.; Nor, A.M.; Halunga, S. Enhancing Internet of Things Security and Efficiency: Anomaly Detection via Proof of Stake Blockchain Techniques. In Proceedings of the 2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Osaka, Japan, 19–22 February 2024; pp. 591–595.
4. Hu, H. Research on the technology of detecting the SQL injection attack and non-intrusive prevention in WEB system. *AIP Conf. Proc.* **2017**, *1839*, 20205. [CrossRef]
5. Mouelhi, T.; El Kateb, D.; Le Traon, Y. Inroads in Testing Access Control. In *Advances in Computers*; Academic Press Inc.: Cambridge, MA, USA, 2015; Volume 99, pp. 195–222.
6. Ramachandran, M. Software security requirements management as an emerging cloud computing service. *Int. J. Inf. Manag.* **2016**, *36*, 580–590. [CrossRef]
7. Ibarra-Fiallos, S.; Higuera, J.B.; Intriago-Pazmino, M.; Higuera, J.R.B.; Montalvo, J.A.S.; Cubo, J. Effective Filter for Common Injection Attacks in Online Web Applications. *IEEE Access* **2021**, *9*, 10378–10391. [CrossRef]
8. Clarke, J. Platform level defenses. In *SQL Injection Attacks and Defense*, 2nd ed.; Elsevier: Amsterdam, The Netherlands, 2012; pp. 409–442.
9. Tian, W.; Yang, J.-F.; Xu, J.; Si, G.-N. Attack Model Based Penetration Test for SQL Injection Vulnerability. In Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, Izmir, Turkey, 16–20 July 2012; pp. 589–594. [CrossRef]
10. Buja, G.; Jalil, K.B.A.; Ali, F.B.H.M.; Rahman, T.F.A. Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack. In Proceedings of the ISCAIE 2014—2014 IEEE Symposium on Computer Applications and Industrial Electronics, Penang, Malaysia, 7–8 April 2014; pp. 60–64. [CrossRef]
11. Masri, W.; Sleiman, S. SQLPIL: SQL injection prevention by input labeling. *Secur. Commun. Netw.* **2015**, *8*, 2545–2560. [CrossRef]

12. Parvez, M.; Zavarsky, P.; Khoury, N. Analysis of the effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015; pp. 186–191. [CrossRef]
13. Huang, Y.-W.; Huang, S.-K.; Lin, T.-P.; Tsai, C.-H. Web application security assessment by fault injection and behavior monitoring. In Proceedings of the twelfth international conference on World Wide Web—WWW '03, Budapest, Hungary, 20–24 May 2003; p. 148. [CrossRef]
14. Lee, I.; Jeong, S.; Yeo, S.; Moon, J. A novel method for SQL injection attack detection based on removing SQL query attribute values. *Math. Comput. Model.* **2012**, *55*, 58–68. [CrossRef]
15. Gould, C.; Su, Z.; Devanbu, P. JDBC checker: A static analysis tool for SQL/JDBC applications. In Proceedings of the 26th International Conference on Software Engineering, Edinburgh, UK, 28 May 2004; Volume 26, pp. 697–698. [CrossRef]
16. Wassermann, G.; Su, Z. An analysis framework for security in Web applications. In Proceedings of the FSE Workshop on Specification and Verification of component-Based Systems (SAVCBS 2004), Newport Beach, CA, USA, 31 October–1 November 2004; p. 70.
17. Thomas, S.; Williams, L. Using Automated Fix Generation to Secure SQL Statements. In Proceedings of the Third International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007), Minneapolis, MN, USA, 20–26 May 2007; p. 9.
18. Kosuga, Y.; Kono, K.; Hanaoka, M.; Hishiyama, M.; Takahama, Y. Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In Proceedings of the Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007; pp. 107–117. [CrossRef]
19. Ali, A.B.M.; Shakhatreh, A.Y.I.; Abdullah, M.S.; Alostad, J. SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. *Procedia Comput. Sci.* **2011**, *3*, 453–458. [CrossRef]
20. William, W.G.; Orso, A. AMNESIA: Analysis and monitoring for NEutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, Long Beach, CA, USA, 7–11 November 2005.
21. Buehrer, G.T.; Weide, B.W.; Sivilotti, P.A.G. Using parse tree validation to prevent SQL injection attacks. In Proceedings of the 5th international workshop on Software engineering and middleware—SEM '05, Lisbon, Portugal, 5–6 September 2005; p. 106. [CrossRef]
22. Park, J.C.; Noh, B.N. SQL injection attack detection: Profiling of web application parameter using the sequence pairwise alignment. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4298, pp. 74–82. [CrossRef]
23. Valeur, F.; Mutz, D.; Vigna, G. A learning-based approach to the detection of SQL attacks. In Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment: Second International Conference, DIMVA 2005, Vienna, Austria, 7–8 July 2005; pp. 123–140. [CrossRef]
24. Joshi, A.; Geetha, V. SQL Injection detection using machine learning. In Proceedings of the 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, Kanyakumari, India, 10–11 July 2014; pp. 1111–1115. [CrossRef]
25. Arasteh, B.; Sadegi, R.; Arasteh, K.; Gunes, P.; Kiani, F.; Torkamanian-Afshar, M. A bioinspired discrete heuristic algorithm to generate the effective structural model of a program source code. *J. King Saud Univ. Comput. Inf. Sci.* **2023**, *35*, 101655. [CrossRef]
26. Arasteh, B.; Aghaei, B.; Farzad, B.; Arasteh, K.; Kiani, F.; Torkamanian-Afshar, M. Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms. *Neural Comput. Appl.* **2024**, *36*, 6771–6792. [CrossRef]
27. Arasteh, B. Clustered design-model generation from a program source code using chaos-based metaheuristic algorithms. *Neural Comput. Appl.* **2023**, *35*, 3283–3305. [CrossRef]
28. Arasteh, B.; Razieh, S.; Keyvan, A. ARAZ: A software modules clustering method using the combination of particle swarm optimization and genetic algorithms. *Intell. Decis. Technol.* **2020**, *14*, 449–462. [CrossRef]
29. Sefati, S.; Mousavinasab, M.; Zareh Farkhady, R. Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: Performance evaluation. *J. Supercomput.* **2022**, *78*, 18–42. [CrossRef]