Journal of Basrah Researches (Sciences) **50**(1), 340 (2024)
DOI: https://doi.org/10.56714/bjrs.50.1.27

A Survey on Bio-Inspired Algorithm for SQL Injection Attacks

Zainab H. Jody

Department of Computer Science, College of Computer Science and Mathematics, University of Kufa, Najaf, Iraq

ARTICLE INFO

Received 4 May 2024
Accepted 18 June 2024
Published 30 June 2024

Keywords:

1-Bio-inspired Algorithm, 2-an overview of the background of SQL injection, 3-SQL Injection Detection and prevention Techniques.

Citation: Zainab H. Jody, J. Basrah Res. (Sci.) 50(1), 340 (2024).
DOI:https://doi.org/10.56714/bjrs.50.1.27

ABSTRACT

SQL injection attacks cause significant threats to the security of online applications. It leverages vulnerabilities in database systems and can result in unauthorized access to and compromising sensitive data. This study investigates the use of bio-inspired algorithms to tackle such attacks, assessing their applications and potential for enhancing cybersecurity measures against SQL injection attacks. In this review, we describe the basic definition, causes, types, and prevention mechanisms of SQL injection attacks. In addition, we examine the use of various bio-inspired algorithms to solve the problem of SQL injection attacks. This study concludes the importance of continuously improving detection methods, particularly those adopting bio-inspired algorithms since they achieved promising results.

1. Introduction

The Internet is currently becoming a ubiquitous information infrastructure. The proliferation and advancement of Internet infrastructure have recently resulted in a surge in the data saved in databases [1]. Given the increasing user base and heavy reliance on digital data, it is crucial to implement spatial protection measures for many kinds of information, such as business data, financial transactions, health records, personal data, and online services [2]. Any web browser on any operating system can access all web applications on the Internet, a global non-profit initiative, focuses on enhancing software security. This community releases the "OWASP Top 10," which is a known publication focused on raising awareness between developers and enhancing web application protection. The statement signifies a widespread agreement over the most crucial security vulnerabilities of web applications [3]. In 2021, the OWASP Top 10 ranked injections as the third most critical online application security vulnerability, (see Table 1) [4], also released the CWE Top 25 Most Menacing Software Weaknesses This category ranks SQLIAs third

Web applications with poor construction are often vulnerable to malicious software attacks, particularly SQL injection attacks. Experts have recognized this vulnerability for over two decades and it remains a source of concern[5]. SQL has been the dominant accepted practice for many years

*Corresponding author email: zainabh.alfaham@student.uokufa.edu.iq

N: 1817-2695 (Print); 2411-524X (Online)
line at: https://jou.jobrs.edu.iq

as a standard for handling relational database management systems (DBMS). Since most cyber-physical systems contain safety-critical applications, any failure due to random mistakes or online attacks could significantly hinder their progress [6]. Hence, it is imperative to protect cyber-physical systems from experiencing such attacks.

SQL injection attacks against fact-driven web applications and systems have been a significant issue since the widespread integration of Internet web applications and SQL databases [7][8]. An SQLI attack occurs when an attacker injects malicious SQL code into a web application's database to exploit the vulnerability. An attacker exploits vulnerability in the web application's SQL implementation by injecting a malicious SQL statement into an input field. Put simply, the attacker will inject code into a specific area to extract or modify data or to obtain unauthorized access to the backend. According to sources [9]and [10], SQLI attack is a prevalent method that enables injurious SQL code to exploit database backends and gain access to concealed information. It is considered one of the most hazardous injection attacks as it poses a threat to essential security pillars such as confidentiality, authentication, authorization, and integrity [11].The material in question may encompass confidential corporate data, personal customer information, or lists of users. An experienced SQLI attacker can completely erase databases, exploit sensitive data without permission, and inadvertently gain administrative privileges in a DB of web attacks [12][13]. According to [14], the most common vulnerability in web applications is injection. Injection attacks exploit various vulnerabilities. The delivery of untrusted user input, which a web application then processes, is one of the issues [15]. SQLI attacks involve inserting malicious SQL commands into input forms or queries to gain unauthorized access to a database or change its data. This can include actions such as extracting the database contents, modifying or deleting the database content, and so on [16].Currently, most online applications rely on a back-end database to store user-gathered data and/or access user-chosen information [17].

To communicate with these users, we frequently use forms and data. Various hackers attempt to exploit this functionality by inserting malicious code into user inputs, which they then use to generate SQL queries. Insufficient validation of user inputs can lead to the successful execution of an SQL injection attack, which can have severe repercussions. As a result, it leads to the removal of the database or the extraction of sensitive and secret data from web application clients [18]. Multiple research studies have focused on the SQLI attack because of its significant impact on security. Some of these works solely focus on the post-detection of SQL injection (SQLI), while others aim to proactively avoid it. Traditional security methods have failed to detect various web attacks including SQLI adequately. Hence, there has been an increase in the adoption of new techniques to complement defence mechanisms such as Machine Learning (ML) algorithms. Bio-inspired algorithms, a subset of ML algorithms, have successfully evolved intelligent defence solutions for various domains, for instance, networks [19] cross-site scripting attacks [20] IoTs [21] and mobile [22].

This study presents a comprehensive survey of SQL injection attacks. We provide an overview of attacks' main attack sources, categories, and objectives. Furthermore, the primary suggested remedies that address this type of attack the topics were deliberated and contrasted.

The contributions of this study are:

1. Given the increasing incidence of SQLIA and its ability to undermine the security of web applications, it is necessary to review ways to improve detection techniques.
2. Machine learning techniques show potential to improve SQLI detection, with multiple studies reporting excellent accuracy, recall, and F1 scores in identifying malicious payloads.
3. Extensive discussion of bio-inspired Algorithms, such as Genetic Algorithm (GA), and Swarm Based Algorithms such as (ABC, ACO, CSA, GWO, and BT) new and effective ways to improve mechanisms aimed at preventing SQLI attacks.
4. A look at how combining bio-inspired approaches with machine learning have the potential to create advanced, adaptive defended systems against SQLI attacks.
5. Future research should prioritize the development of adaptive algorithms that can evolve in response to SQLI attack patterns. Furthermore, it is crucial to encourage collaboration between academics and industry to facilitate knowledge exchange and pooling of resources.

This survey paper will be organized as follows. Section 2: provides an overview of the

background of SQL injection. Section 3 provides SQL Injection Prevention Techniques. Section 4 explains Nature-Based Technology. Section 5. introduces a bio-inspired SQL injection application Section. 6. The Paper's Discussion Section 7. provides the paper's conclusion.

Table 1. The OWASP Top 10 Web Application Security Dangers table [4]

OWAPA TOP 10-2013	OWAPA TOP 10-2017	OWAPA TOP 10-2021
Injection	Injection	Broken access control
Broken Authentication and Session Management	Broken authentication	Cryptographic Failures
Cross-Site Scripting (XSS)	Sensitive data exposure	Injection
Insecure Direct Object References	XML external entities (XXE)(NEW)	Insecure Design (NEW)
Security Misconfiguration	Broken access control	Security Misconfiguration
Sensitive Data Exposure	Security misconfigurations	Vulnerable and Outdated Components
Missing Function Level Access Control	Cross-site scripting (XSS)	Identification and Authentication Failures
Cross-Site Request Forgery (CSRF)	Insecure deserialization (NEW)	Software and Data Integrity Failures (NEW)
Using Components with Known Vulnerabilities	Using components with known vulnerabilities	Security Logging and Monitoring Failures
Unvalidated Redirects and Forwards	Insufficient logging and monitoring	Server-Side Request Forgery (SSRF)(NEW)

2. SQLI Attack Overview

2.1 SQLIA Definition

Web-based applications often follow a three-tier design, consisting of a presentation tier for the user interface, a business tier for logical processes, and a data tier for handling data. It stores all the organized information. A SQLIA, or SQL Injection Attack, exploits vulnerabilities in all three levels of a system to carry out a successful attack[23]. Malevolent SQL commands, transmitted from the presentation layer to the business tier, modify the current SQL queries, thereby exploiting the database tier to gain access to assets. The absence of validation at both the display and business levels of online applications results in a successful SQL injection attack (SQLIA)[24].

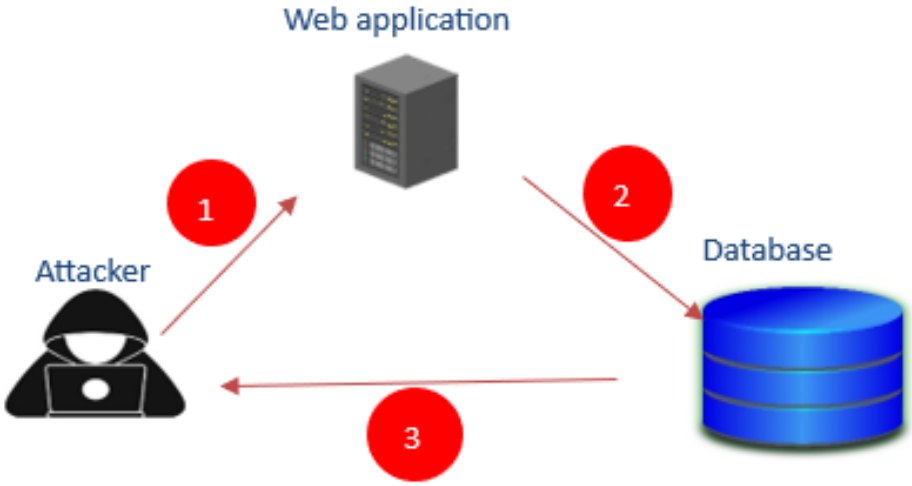

Fig.1. SQL injection attack work.

2.2 SQLIA Sources

Any application parameter that can be used in a database query could potentially include SQL injection vulnerabilities. The writers in [25][26], were provided Four sources which is how the SQL Injection Attack (SQLIA) gets started. These sources are user input, cookies, server variables, and stored injection.

2.2.1 Injection through user input

Web applications typically use forms to gather user data, such as during registration or login processes, or to allow users to define certain preferences or settings. Thus, information should to be retrieved, such as search results or an adapted view. These forms that have a "text field" might be vulnerable to exploitation by attackers who can inject malicious code. This can lead to unauthorized access to sensitive data (such as retrieving secret information) or performing unauthorized operations (such as manipulating a database). The typical fields include login name, password, address, phone number, credit card number, and search[27][25].

2.2.2 Injection through cookies

Modern web apps use cookies to store user preferences. The client computer stores cookies, which are files that include the state the online programs produce data. A web application that uses cookie contents to construct SQL queries may become vulnerable to an attacker's malicious code embedded in the cookies stored on his computer [24][25]

2.2.3 Injection through Server Variables

Server variables consist of a collection of parameters that store network headers, HTTP information, and environmental factors. In general, Web applications utilize server variables to conduct auditing, track usage data, and discover browsing trends. Malicious individuals can exploit this vulnerability by directly inserting an SQL injection attack into the server variables if they save these variables in a database without proper validation [25].

2.2.4 Stored injection

In this form of injection, sometimes referred to as second-order injection, malicious actors can insert detrimental inputs into a database, thereby initiating a response from the database. The appropriate command should follow each input to execute the SQL injection attack [28].

2.3 SQLI Attack Objectives

Hackers can utilize SQL injection (SQLI) attacks to achieve various goals. The primary objectives of a SQL injection (SQLI) attack are:

2.3.1 Identifying Injectable Parameters

Hackers start by identifying the vulnerabilities that they can manipulate to introduce malicious code. The sources listed in Subsection II-A may be Encompass these attributes. These parameters can refer to specific data, such as a "card number" stored in a cookie or a "username" entered in a form. By injecting SQL code, a malicious individual can alter the logic of the statement, causing it to run in a manner different from its intended functionality. One can exploit SQL injection vulnerability by injecting a single quote mark, commonly used in SQL to indicate the start or end of a string value. As a result, an application error would arise, exposing a potential vulnerability [29]

2.3.2. Performing database fingerprinting

The attacker must understand the database fingerprint to create a query syntax that the target database engine approves. We refer to the unique characteristics that distinguish a specific kind and version of a database system as the database fingerprint. Proprietary differences cause different database systems to use different SQL language syntax. Oracle SQL Server uses PL/SQL, while Microsoft SQL Server utilizes T-SQL. Consequently, the attacker needs to determine the specific kind and version of the web application's database to generate SQL input that can exploit its vulnerabilities. In addition, attackers can exploit the inherent vulnerability of the database's default settings [27]

2.3.3. Determining Database Schema

To proceed, the attacker must possess knowledge of the database schema, which includes information such as table names, column numbers and names, and column data types to effectively retrieve data from a database. Hackers construct a meticulous subsequent attack by exploiting the database structure to extract or manipulate data from the database [30].

2.3.4. Extracting data

These attacks employ techniques to extract data from the database. This vulnerability poses a significant risk to a web application as it allows unauthorized access to sensitive information. It may include sensitive and highly classified information. The most common type of SQLIA comprises attacks aimed at achieving this objective [28]

2.3.5. Database alteration

These attacks have the objective of altering or updating data that is stored in a database. For instance, a hacker can greatly decrease the cost of an online transaction by modifying the pricing information that is often stored in a database. An internet chat database may potentially be targeted by adding a malicious link as a potential attack strategy [31]

2.3.6. Performing denial of service

This attack can manifest in various forms, including disabling a web application's database, encrypting or deleting database tables, and so on, to deny access to other users [29]

2.3.7. Bypassing authentication

This exploit aims to bypass the online application's authentication protocols. The exploit exploits the privileges and rights of another user, typically one with a superior status. If the intruding party is successful in carrying out such an attack, it may revoke one's rights and privileges [29]

2.3.8. Executing remote commands

Executable code known as remote commands is present on the compromised database server. These instructions could either be functions or stored routines. This information can be easily accessed by database users. In this type of attack, the attackers attempt to execute arbitrary instructions on the database. This can lead to the execution of shutdown commands or cause disruption and denial of service to the database [32].

2.3.9. Performing privilege escalation

These exploits leverage implementation flaws or logical weaknesses in the database to try to elevate the attacker's privileges. These attacks focus on exploiting the privileges of the database user instead of trying to bypass authentication. Once the attacker gains root privilege, this technique can have severe consequences [30].

2.4. SQLI Attack Categories

There are various types and styles of SQLI attacks [5][33]We go over and categorize the primary SQLI attack types in this section, which are:

2.4.1 Tautologies

An attacker makes use of the SQL statement's condition format following (WHERE) clustering. Verifying the username and password is the prerequisite for any login inquiry. The attacker meanwhile introduces an input for the always-true condition. Such an attack resulted in access to a web application. [31].

2.4.2. Union query

In SQL (Structured Query Language), the Union statement is frequently used to combine two or more SQL queries. Structured Query Language (SQL) A UNION-based injection attack is a specific form of attack that exploits the capabilities of UNION to append an additional query to extract data from a system. For example, during the login process, an attacker may execute a data extraction

statement (SELECT * FROM Customers) to retrieve all customer details. The list contains the numbers [34][34]. An example of a Union query is SELECT * FROM users WHERE username = 'admin' and password= 'moh@20' UNION ALL SELECT * FROM users;[36].

2.4.3. Error-Based

The web page will display the database error message. The SQL query statement is experiencing an issue, and the "union query" is unable to function in this vulnerability. In contrast to basic SQL injection, the back-end code is currently only retrieving the initial result and cannot extract all of it. As a result, the "union query" is not functioning effectively. The error-based exploitation technique involves the theft of data by exploiting error messages displayed on a webpage [36].

2.4.4. Blind SQLi

Blind SQL injection is a technique that involves utilizing SQL statements to ask a series of questions that get either true or false responses. This method can be employed to gain unauthorized access to sensitive information on a website[37].

2.4.4.1. Boolean-Based

This type of SQL injection uses Boolean values (true or false). The malicious SQL query causes the web application to generate an alternative result. The result of the query determines whether it is true or false. To change the syntax of the original query's WHERE clause, for example, "aaa O R 2 = 2" has been entered as the password in the SQL query "SELECT ∗ FROM people WHERE password = aaa OR 2 = 2." The resulting SQL query uses a logical operator OR to separate two distinct conditions. The second condition, "2 = 2," has to be true even when the first, "password = aaa," might not. Thus, if at least one of the operands is true, the logical operator OR returns true, compelling the web application to produce a different result [38].

2.4.4.2. Time-Based

A time-based SQL injection attack is a sophisticated method of SQL injection that falls under inferential or blind attacks. Unlike other attack types, this technique does not involve data transfer. Instead, attackers manipulate complex SQL statements to alter the execution time of queries on the target computer. If the attack is successful, the response will be delayed, giving the attacker ample time to methodically identify and analyze the database and devise a plan of attack [37].

2.4.5. Piggy-backed query

A piggy-backed query attack involves an attacker intentionally injecting additional queries to retrieve, modify, or add data. Adversaries introduce supplementary elements. Additional queries to the original query cause the DBMS to receive multiple SQL requests. The provided query is a prime example of a piggy-backed query SQL injection attack. The following is an example of a drop table (SELECT * FROM books WHERE id=5; drop table users;) [17].

2.4.6. Stored procedures

The database engine may support a stored developer-implemented procedure, as well as default-coded procedures and triggers. The attacker injects a remote execution call for such procedures, as an example where the attacker tries to shut down the database [39]. (SELECT * FROM users WHERE id=5; exec (SHUTDOWN);) .

2.4.7. Alternate encoding

The attacker encoded the injection to trick the web server. Moreover, to fraud any filtering method within the web server, an example is shown as SELECT * FROM users WHERE id=5; exec (char(0x736875746 46f776e)); [31].

2.4.8. Illegal/logically incorrect queries

This type of attacker exploits an incorrectly executed database query [40]. Database error messages often contain vital information that allows an attacker to gain knowledge about the exact details of the application's database. The attack's objective encompasses finding injectable parameters, performing database fingerprinting, and extracting data are the tasks involved. This attack enables an assailant to get vital information about the structure and operation of the backend database of a web service [41, 42]. The attack is believed to be a rehearsal for future attacks to gain intelligence. This attack exploits the vulnerability that arises from the detailed nature of default error pages on application servers [43]. For instance: SELECT * FROM users WHERE username = 'test" and password test'; [32].

Table 2: SQLIA Exporter, Objectives, and Category Categorization [40][31][32]

Category Parameter	Categorization
Attacking Sources	User input
	Cookies
	Server variables
	Second order injection
Attacking Objectives	Database fingerprinting
	Analysing schema
	Extracting data
	Amending data
	Executing dos
	Equivocating detection
	Bypassing authentication
	Remote control
	Privilege intensification
Attacking Categories	Tautology
	Illegal/logically incorrect queries
	Union query
	Piggyback query
	Stored procedure
	Blind query
	Alternate encoding

3. SQL Injection Prevention Techniques

3.1 Stored Procedures

Stored Procedures provide an extra level of protection for the database, in addition to using Prepared Statements. It enables the program to accept input data for processing instead of SQL code for execution, avoiding its need [41]. The critical difference between a prepared statement and a stored procedure is that the SQL code for a stored procedure is created and saved within the database server and subsequently invoked via the web application [42].

Suppose a user's access to the database is only authorized through stored procedure techniques. In that case, there is no need to explicitly grant the user access to any database table to get data properly. In this manner, the database remains secure [38].

3.2 Input Validation

Validating the input data as the primary line of defense is crucial to mitigate SQL injection attacks. The user's input must undergo validation against predetermined criteria to ensure its compatibility with the software's processing capabilities [43]. The primary objective of input validation is to exclude malicious code from being transmitted to the database as a component of an SQL query. Limiting data types, the range of data, and checking the character set validation will again

be in the diverse category. Data type validation is all about the verification of the data given by the user inclusive of amount, characters, and symbols to number and string type [44].The range will be checked to make sure the input to the user is within the set range. Character set verification because of the requirement that only an acceptable character set was used under the supervision of the software by having a user type nothing but such a character set. Validation of the input can reveal the presence of certain specified characters or symbols that may be pointed out as SQL injection attacks. Detecting an attempt to introduce malicious code into a SQL query can be accomplished by examining user-provided data for the existence of a single quote character ('). Despite its significance, input validation is frequently disregarded during web application development, thus rendering them susceptible to SQL injection attacks [42]. Client-side validation can be achieved using JavaScript or another client-side scripting language. In contrast, server-side validation can be performed. One can use server-side scripting languages like PHP or ASP.NET to create dynamic web pages.

3.3 Whitelisting

Whitelisting is the system administrators use to prevent SQL injection attacks by way of nullifying or eliminating possibly harmful keywords or transformations in SQL queries. Another tool that differs but on the contrary is block listing. This mechanism aims at tracking all the keywords that are disallowed to be inserted in a SQL query containing UNION or Boolean-like statements. Being equipped with these keywords allows the system either to remove them automatically or to raise an error right away. This move is done to forestall possible attacks [46].

3.4 Parameterized Query

A parameterized query is a commonly used approach to mitigate SQL injection. It involves placing an additional layer between the user and the database. This layer constructs the query in a predetermined structure, with only variables serving as placeholders that will be replaced with user input. The intermediary layer verifies the query's validity before transmitting it to the database. If any issues arise with the query, they will be promptly identified. The question mark symbol ("?") in the query below is utilized as a placeholder in a parameterized query [31].
Retrieve all records from the Customers table. Specify the condition where the username and password match the given values.

3.5 Escaping all input supplied by the user

This approach specifically enables DBMS to distinguish between user input and SQL commands. To do so, the developers must be able to configure In the PHP environment, the configuration files provide settings for magic_ quotes_ gpc and magic_ quotes_ runtime [47].

3.6 Use the principle of least privilege

In this approach, users should only have access to the specific table they need, not full database privileges [45].

3.7 Web Application Firewall (WAF) Instant Protection

The system possesses the capacity to detect and block harmful network traffic before it reaches the web application. Customizing the system allows for the identification and prevention of known SQL injection attacks, thereby enhancing security measures in opposition to such attacks [45].

Various conventional methods or instruments are available for detecting or minimizing SQL injection attacks. Automated scanners can be used to detect SQL injection vulnerabilities. Automated scanners utilize several techniques, such as static analysis, dynamic analysis, Boolean-based testing, and error-based testing, to identify SQL injection vulnerabilities. However, those scanners are not capable of detecting all sorts of SQL injection vulnerabilities. Some organizations fail to detect these attacks due to various challenges they face, including outdated code, limited resources for testing and implementing changes, a lack of awareness about application security, and infrequent updates to their web applications. These factors contribute to vulnerabilities. To address these issues, we must employ statistical and artificial intelligence techniques, such as machine learning and deep learning. This study examines different machine learning and deep learning approaches and their applications in detecting and mitigating SQL injection threats. The next part presents the key findings of the literature

review, focusing on current and significant effect mechanisms. Next, we will discuss the performance of the top three approaches, their outcomes, and the debates surrounding them.

3.8 Machine Learning Algorithms

Using machine learning to find SQL injection attacks is a useful and effective way to deal with this growing security risk. These programs can get around the problems with traditional rule-based methods. The algorithms work by learning to recognize the unique traits and patterns of legal requests. They then use this knowledge to find any oddities or deviations that could be signs of an attempt at SQL injection [46]. Instead of sticking to set rules, these systems use machine learning to adapt to new attack tactics. Anomaly detection models, for example, can learn the patterns of common questions and then sound an alarm when they come across questions that don't follow the learned patterns. Similarly, we can train classification models on label datasets of attacks and benign requests to identify the features of SQL injection efforts. Models like recurrent neural networks in natural language processing can detect semantic and syntactic anomalies in SQL queries, treating them as structured language. Furthermore, we use ensemble techniques that mix several machine learning models to enhance general detection accuracy and robustness against evasion efforts. These methods, driven by machine learning, have the main benefit of being able to adapt to the strategies used by SQL injection attackers. Traditional static, rule-based defences are less flexible and effective than the models that can learn to identify new attack patterns through constant training on fresh data. But putting into practice efficient ML-based SQL injection detection also has its share of difficulties, such as getting representative training data, reducing model bias, and guaranteeing real-time performance. These issues are the main focus of current research to fully utilize machine learning for robust, flexible SQL injection defences [9].

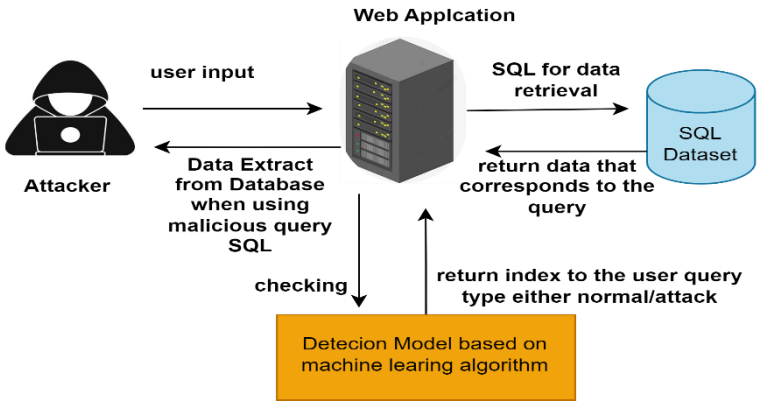

Fig.2. The fundamental method for employing ML to identify SQL injection attacks.

4. Nature -Based Technology

Nature offers a vast and rich variety of complex and challenging events, which serve as a valuable source of innovation for addressing complications in computer science [48]. Nature-inspired algorithms are metaheuristics designed to efficiently solve optimization problems that mimic natural processes, leading to a new era in computation [49]. Bio-inspired algorithms will initiate a novel epoch in the field of computer science. Academic collaboration across several disciplines, including computer science, ecology, AI, environmental science, and biology, is necessary to gain a broader understanding and analysis of each step or interaction at a micro-level. This will lead to significantly more impactful and remarkable results [50]. Environmental-based technologies are highly efficient optimization algorithms that can create a substantial impact [51]. Conventional problem-solving procedures encompass precise techniques such as logical reasoning, mathematical programming, and

heuristics. When confronted with challenging and intricate optimization issues, the heuristic strategy proves superior, especially when conventional methods are unsuccessful. Several biological processes can be conceptualized as constrained optimization strategies. BIAs is heuristic algorithms that emulate nature's methodology. Due to their tendency to make numerous arbitrary choices, they can be classified as a subset of randomized algorithms [49]. The process design is contingent upon the nature of the problem, the evaluation of the fitness function, and the generation of solutions. Recently, there has been a surge in articles discussing the effectiveness of bio-inspired tactics in many computer science domains. The literature also shows a growing interest in using bio-inspired approaches to solve diverse challenges. The two predominant and prosperous BIA disciplines are EA and SBA, both shaped by the principles of natural evolution and the collective behavior of animals. This will be further improved to enable the categorization of algorithms based on their natural inspiration, offering a more comprehensive understanding of the subject [52]. The objective is to evaluate Business Impact Analyses (BIAs), along with taxonomy information and the many application domains. The bio-inspired algorithms can be categorized into three primary groups [53], (1) Evolutionary algorithms, (2) Swarm Based, and (3) Human Based.

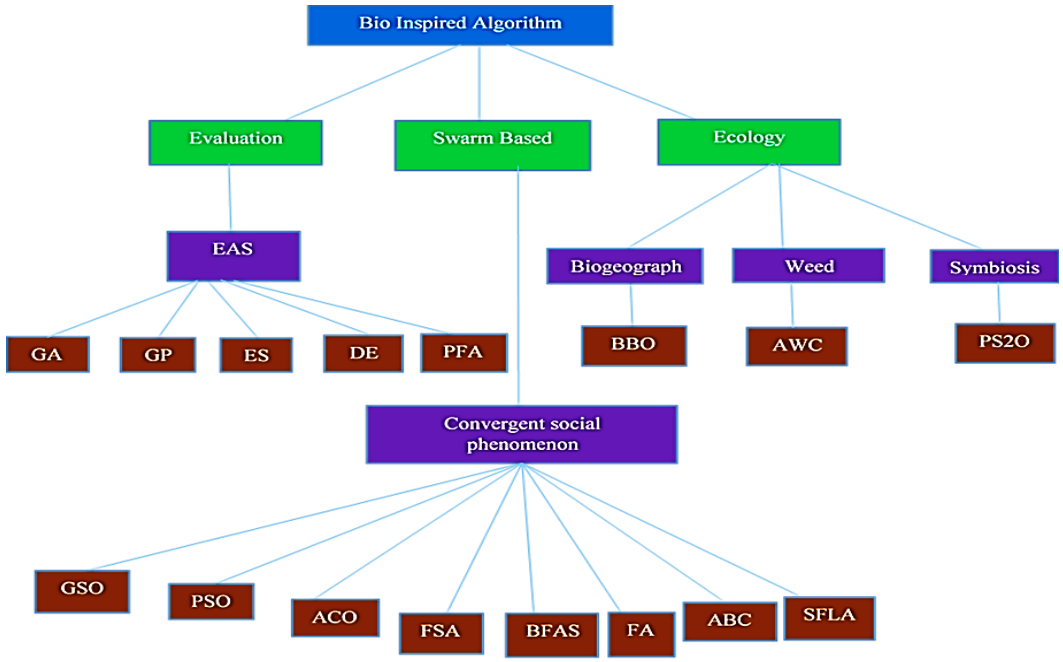

Fig.3. Bio-Inspired Optimization Algorithms[54].

5. A SQL injection application with a bio-inspired approach

5.1 Genetic Algorithm (GA)

The method of optimization is rooted in genetics and natural selection. This technique was initially introduced by John Holland in the 1975s and is currently recognized as a biologically inspired method for search and optimization in a recent study[55].

The author [56], discussed the development and utilization of a genetic algorithm-driven system model aimed at optimizing SQL injection test data. The primary objective is to enhance the precision and effectiveness of identifying and curtailing SQL injection attacks. In particular, we create an elaborate system workflow through various integral steps such as testing & assessment, genetic algorithm optimization, filtering, and enriching the test data set— allowing us to fine-tune the test data on an automated basis to boost its resistance towards SQL injection attacks. The findings indicate that a system model developed using evolutionary algorithms (based on the testing) can significantly

augment both the accuracy as well as speed in detecting SQL injections— making it more effective in stopping these attacks with optimal resource utilization.

The utilization of four distinct machine learning techniques, namely gradient boost (GB), multilayer perceptron (MLP), logistic regression (LR), and K nearest neighbour (KNN), was implemented by the author in [57], To enhance the model's performance and identify the most effective configuration, a tree-based pipeline optimization tool (TPOT) and the genetic algorithm (GA) were employed. The dataset provided by the Canadian Institute 2023, encompassing various attack types, served as the basis for testing the model. Notably, the accuracy values achieved by GB for accuracy, recall, and F1-score were 95%, 94%, and 95% respectively.

5.2 Artificial Bee Colony Algorithm (ABCA)

In 2005, “Karaboga” introduced the (ABC) Technique for the first time[58]. It is designed to optimize numerical problems by using a population-based approach. The algorithm is a variant of the behavior of honey bee swarms in the foraging process. This simple algorithm has been praised for its robustness and effectiveness in solving various optimization problems.

The author [59], followed the complete ABC method to do a thorough code assessment of the possible vulnerabilities. The lead dataset for the ABC algorithm contributes to the better performance and quality of the result. To enhance the results, the researcher highlighted the benefits of adding more data. They also suggested looking into dynamic genome-specific testing that might yield captivating alternatives. In general, the researcher proposed an approach to determine SQL injection danger in web applications with the aid of the ABC method. They spotted the necessity for multi-languages. Such a piece of the system which is based on PHP analysis was developed by them. The researcher highlighted that by widening the database and assessing the dynamism of the testing, the answer may be advanced.

5.3 Cuckoo Search Algorithm (CSA)

The (CSA) is derived from the cuckoo's breeding behaviour, which involves laying eggs in the nests of other birds. If the eggs are not found and removed, they will hatch, and the cuckoos will grow into adulthood. The algorithm, which Renew Yang and Deb first developed in 2009[60], is a reflection of this strategy. From an initial population of cuckoos, it continuously adapts and "lays eggs" in many possible "habitats" in an attempt to find the optimum "nest," which is comparable to the best solution to an optimization issue.

The author [61], employed the Cuckoo Search Algorithm (CSA) to tackle input-type validation flaws in HTTP requests. This approach protects online application systems from security breaches such as SQL injections and cross-site scripting (XSS) attacks. The researcher obtained the following outcomes:

- An accuracy of 0.937
- Precision of 0.883
- A misclassification rate of 0.063
- A detection omission rate of 0.034

The study proposed a novel approach, the Cuckoo Search-Based Exploratory Scale (CSES), for identifying and mitigating HTTP requests susceptible to attacks. An evaluation was conducted on the CSES model, which demonstrated high accuracy in identifying sensitive requests while minimizing false alarms.

5.4 Bat Algorithm (BT)

The method, presented by Yang in 2010[62], is a modern swarm intelligence algorithm that draws inspiration from the echolocation system of microbats. The Bat Algorithm (BT) algorithm has been successfully implemented and fine-tuned for eight widely recognized optimization problems. A comparative analysis has been performed between the suggested algorithm and other pre-existing algorithms [63].

The author[64]. It has been shown in this study that great accuracy was attained through optimization using the BAT method. The identification of SQL injection attacks, a serious security risk to numerous data-intensive industries, is the main objective of this study. It assesses how well a probabilistic neural network (PNN) that has been BAT algorithm-optimized performs in detecting these kinds of attacks. The optimized PNN obtained 99.19% accuracy, 0.995% precision, 0.981% recall, and an F-measure of 0.928%, according to the results. This performance is faster than previous research and has a lower false-positive rate. The main issues that have been found are the algorithm's intricacy and its susceptibility to irrelevant or noisy data elements.

5.5 Ant Colony Optimization Algorithm (ACO)

Marco Dorigo developed[65], the ant Colony Optimization (ACO) metaheuristic in 1992 was based on biological and behavioural. Pheromone laying and using pheromone inspiration is to choose the quickest path creation of the initial ACO algorithm. Numerous academics have worked on this topic and published their findings since the introduction of the first algorithm of this kind. Even though the early results were not very encouraging, more recent advancements have elevated this metaheuristic
to a prominent position in swarm intelligence.

In [66], the study presents a brand-new, two-phase technique that uses Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), and Genetic Algorithm (GA) to identify SQL injection vulnerabilities. The method Optimizer makes the vulnerability search process more efficient by using the previously described methods after examining the code's SQL queries. The framework as it exists now is designed for PHP applications, primarily targeting SQL injection; however, it may be expanded to cover other programming languages and security flaws in the future.

5.6 Grey Wolf Optimization Algorithm (GWO)

2014[67], work by "Seyedali Mirjalili" et al. was motivated by the predatory performance of grey wolf packs. One of the more modern metaheuristic swarm intelligence algorithms is the Grey Wolf Optimizer (GWO). Due to its exceptional advantages over other swarm intelligence techniques—namely, the fact that it requires no derivation knowledge during the initial search and has very few parameters—it has been extensively used for a wide range of optimization challenges. Additionally, it is straightforward, user-friendly, adaptable, scalable, and has the extraordinary capacity to combine exploration and exploitation in a way that promotes enabling convergence during the search process.

In[68], the author employed two different binary iterations of the Gray-Wolf method to select the best features from the dataset. The best datasets that were produced were subjected to a variety of machine-learning techniques. The test results show that the proposed SQL injection detector achieves 99.68% accuracy, 99.40% precision, and 98.72% sensitivity. The proposed method enhances attack detection systems by selecting 20% of the most valuable information.

Table 3. Summary of Bio-inspired Algorithm

Algorithm	Refer. No.	Author(s)	year	Overview	Application
Genetic Algorithm (GA)	[55]	John Holland	1975	Genetic algorithms (GAs) are heuristic optimization methods that draw inspiration from the phenomenon of natural selection. These algorithms aim to identify optimal solutions by emulating the evolutionary process, which involves selection, crossover, and mutation.	In computer science, GA is among the most significant algorithms. It can be useful in many areas, such as robotics, economics, scheduling design, and marketing.
Artificial Bee Colony	[58]	Karaboga	2005	This algorithm was proposed by Karaboga for	The ABC algorithm is extensively utilized in

Algorithm (ABCA)				optimizing numerical problems. The algorithm simulates the intelligent foraging conduct of honey bee swarms. It is a very undemanding, strong, and population-based stochastic optimization algorithm.	optimization challenges, encompassing numerical optimization, engineering optimization, and neural network optimization, owing to its efficacy and straightforwardness.
Cuckoo Search Algorithm (CSA)	[60]	Xin-She Yang and Suash Deb	2009	The (CSA) is a powerful metaheuristic method for optimization It belongs to the family of nature-inspired algorithms, a family of stochastic optimization methods based on imitating certain biological or social processes commonly found in the natural world. These methods have gained a lot of popularity in recent years due to their ability to deal with large, complex, and dynamic real-world optimization problems. Although they do not ensure finding the global optimal solution, in most cases, they can find more accurate solutions to many problems than the classical mathematical optimization methods.	Due to its efficacy in resolving intricate issues, the CSA has been extensively utilized in various domains, including engineering optimization, scheduling challenges, and machine learning. The technique under consideration exhibits a diverse array of applications, encompassing the optimization of network architectures and the resolution of economic dispatch challenges within power systems.
Bat Algorithm (BT)	[62]	Xin-She Yang	2010	Yang suggested the Bat algorithm in 2010. The basic Bat algorithm is bio-inspired. It's based on the characteristics of bat bio-sonar or echolocation. Bats in the wild emit ultrasonic waves into the environment to aid in hunting or navigation.	To solve optimization problems, the Bat Algorithm (BA) utilizes Continuous Optimization, Combinatorial Optimization and Scheduling, Inverse Problems and Parameter Estimation, Image Processing, Fuzzy Logic, and Other Applications.
Ant Colony Optimization Algorithm (ACO)	[65]	Marco Dorigo	1992	The algorithm described above is a probabilistic methodology employed to resolve computational issues by locating optimal routes within graphs, hence yielding solutions of reduced length. In the field of network routing, scheduling, and other optimization activities, Ant Colony Optimization (ACO) utilizes the principles of natural efficiency to improve performance. It is influenced by the behaviour	Applications of ant colony optimization (ACO) concerning hydrology and hydrogeology include irrigation water allocation, urban drainage network design, groundwater long-term monitoring, reservoir optimization, watershed management, coastal aquifer management, hydraulic parameters stimulation, and water distribution systems.

				of ants, which adeptly navigate the most direct path to access food sources.	
Grey wolf optimization algorithm (GWO)	[67]	Seyedali Mirjalili	2014	The (GWO) algorithm is a novel meta-heuristic, inspired by the social hunting behaviour of grey wolves.	The application of (GWO) is Engineering Design Problems, Robotic and path planning, and Image processing.

6. Discussion

As cybersecurity threats become more sophisticated, academic circles have paid close attention to the study of SQL injection (SQLI) attacks. These attacks, which use flaws in web applications to compromise database dependability, represent a persistent and extensive threat to information security. Most people agree that SQLI can violate data integrity, circumvent authentication systems, and jeopardize confidentiality. It is essential to know every approach to prevent or reduce attacks. This work's main contribution is the thorough description of using bio-inspired algorithms to counter SQL injection attacks is one of the main contributions of this work. The authors investigate how many bio-inspired algorithms, such as swarm-based and genetic algorithms (GA), can improve security against SQL injection attacks. These bio-inspired algorithms are a useful tool for defending against SQL injection attacks and can improve the accuracy, recall, and F1 scores of identifying dangerous payloads. Furthermore, the paper demonstrates the ability to construct sophisticated, adaptive defences es against SQL injection attacks by fusing machine learning techniques with bio-inspired techniques. This combination may result in more intricate algorithms that can change in response to attack trends. Such a persistent threat would strengthen cybersecurity. The writers underline the need for collaboration between academics and businesses to promote the exchange of knowledge and resources, which can expedite the development of these new defences.

7. Conclusion and Future Directions

An overview of bio-inspired algorithms' application to SQL injection attacks (SQLIAs) closes this paper. It emphasizes how dangerous SQLIAs are still to web applications, as well as how advanced detection and prevention methods are required. The survey shows how effective machine learning techniques are for identifying and mitigating these risks. The bio-inspired technique can effectively identify and compare SQL injection attempts. However, there are three specific applications of bio-inspired methods currently available. Finding a new kind of SQL injection attack comes first, followed by choosing supplementary features. We then fine-tune the settings of the machine-learning algorithms. Furthermore, integrating these algorithms with machine learning methods improves defensive systems' resistance and flexibility to changing attack plans. Future work should focus on developing flexible algorithms that can adapt in response to attack strategies and promoting industry-academy collaboration to leverage shared resources and expertise.

Acknowledgment

The author would like to thank the anonymous reviewers for their efforts.

References

[1] J. H. B. Johny, W. A. F. B. Nordin, N. M. B. Lahapi, and Y.-B. Leau, “SQL Injection prevention in web application: a review,” in *Advances in Cyber Security: Third International Conference,*

ACeS 2021, Penang, Malaysia, August 24–25, 2021, Revised Selected Papers 3, Springer, 2021, pp. 568–585.

[2] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of sql injection attack using machine learning techniques: a systematic literature review," *J. Cybersecurity Priv.*, vol. 2, no. 4, pp. 764–777, 2022.

[3] J. Strickland, "Web Operating Systems Work." Accessed: Feb. 26, 2024. [Online]. Available: https://computer.howstuffworks.com/web-operating-system.htm

[4] Foundation, "Owasp top ten." Accessed: Apr. 20, 2024. [Online]. Available: https://owasp.org/www- project- top- ten/

[5] M. A. Hussain, H. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, and A. Ibrahim, "DNS Protection against Spoofing and Poisoning Attacks," in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, 2016, pp. 1308–1312. doi: 10.1109/ICISCE.2016.279.

[6] G. Deepa, P. S. Thilagam, F. A. Khan, A. Praseed, A. R. Pais, and N. Palsetia, "Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications," *Int. J. Inf. Secur.*, vol. 17, no. 1, pp. 105–120, 2018, doi: 10.1007/s10207-016-0359-4.

[7] Y. Pan *et al.*, "Detecting web attacks with end-to-end deep learning," *J. Internet Serv. Appl.*, vol. 10, no. 1, p. 16, 2019, doi: 10.1186/s13174-019-0115-x.

[8] W. Zhang *et al.*, "Deep neural network-based SQL injection detection method," *Secur. Commun. Networks*, vol. 2022, 2022.

[9] 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.

[10] Y. Fang, J. Peng, L. Liu, and C. Huang, "WOVSQLI: Detection of SQL Injection Behaviors Using Word Vector and LSTM," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, in ICCSP 2018. New York, NY, USA: Association for Computing Machinery, 2018, pp. 170–174. doi: 10.1145/3199478.3199503.

[11] Q. Li, F. Wang, J. Wang, and W. Li, "LSTM-Based SQL Injection Detection Method for Intelligent Transportation System," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4182–4191, 2019, doi: 10.1109/TVT.2019.2893675.

[12] D. Chen, Q. Yan, C. Wu, and J. Zhao, "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning," *J. Phys. Conf. Ser.*, vol. 1757, no. 1, 2021, doi: 10.1088/1742-6596/1757/1/012055.

[13] S. Abaimov and G. Bianchi, "A survey on the application of deep learning for code injection detection," *Array*, vol. 11, p. 100077, 2021, doi: https://doi.org/10.1016/j.array.2021.100077.

[14] S. Son, K. S. McKinley, and V. Shmatikov, "Diglossia: detecting code injection attacks with precision and efficiency," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer \& Communications Security*, in CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 1181–1192. doi: 10.1145/2508859.2516696.

[15] R. Yan, X. Xiao, G. Hu, S. Peng, and Y. Jiang, "New deep learning method to detect code injection attacks on hybrid applications," *J. Syst. Softw.*, vol. 137, pp. 67–77, 2018, doi: https://doi.org/10.1016/j.jss.2017.11.001.

[16] J. Jain, "Artificial intelligence in the cyber security environment," *Artif. Intell. Data Min. Approaches Secur. Fram.*, pp. 101–117, 2021.

[17] Z. Marashdeh, K. Suwais, and M. Alia, "A survey on sql injection attack: Detection and challenges," in *2021 International Conference on Information Technology (ICIT)*, IEEE, 2021, pp. 957–962.

[18] M. Hasan, Z. Balbahaith, and M. Tarique, "Detection of SQL injection attacks: a machine learning approach," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, IEEE, 2019, pp. 1–6.

[19] 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.

[20] 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.

[21] H. Alyasiri, J. A. Clark, A. Malik, and R. de Fréin, "Grammatical evolution for detecting cyberattacks in Internet of Things environments," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 2021, pp. 1–6.

[22] 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.

[23] M. A. Hussain, H. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, and A. Ibrahim, "DNS Protection against Spoofing and Poisoning Attacks," in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, 2016, pp. 1308–1312. doi: 10.1109/ICISCE.2016.279.

[24] 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

[25] W. G. Halfond, J. Viegas, and A. Orso, "A classification of SQL-injection attacks and countermeasures," in *Proceedings of the IEEE international symposium on secure software engineering*, IEEE Piscataway, NJ, 2006, pp. 13–15.

[26] M. A. Hussain, Z. Alaa Hussien, Z. A. Abduljabbar, S. Abdulridha Hussain, and M. A. Al Sibahee, "Boost Secure Sockets Layer against Man-in-the-Middle Sniffing Attack via SCPK," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 295–300. doi: 10.1109/ICOASE.2018.8548813.

[27] M. A. Hussain, H. Jin, Z. A. Hussien, Z. A. Abduljabbar, S. H. Abbdal, and A. Ibrahim, "ARP Enhancement to Stateful Protocol by Registering ARP Request," in *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, 2016, pp. 31–35. doi: 10.1109/ICNISC.2016.017.

[28] M. A. Hussain *et al.*, "Provably throttling SQLI using an enciphering query and secure matching," *Egypt. Informatics J.*, vol. 23, no. 4, pp. 145–162, 2022.

[29] T. Muhammad and H. Ghafory, "SQL Injection Attack Detection Using Machine Learning Algorithm," *Mesopotamian J. CyberSecurity*, vol. 2022, pp. 5–17, 2022, doi: 10.58496/MJCS/2022/002.

[30] M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks," *Inf. Secur. J.*, vol. 32, no. 4, pp. 252–265, 2023, doi: 10.1080/19393555.2021.1995537.

[31] V. Abdullayev and D. A. S. Chauhan, "SQL Injection Attack: Quick View," *Mesopotamian J. Cyber Secur.*, vol. 2023, pp. 30–34, 2023, doi: 10.58496/mjcs/2023/006.

[32] H. R. Y. Wimukthi, H. Kottegoda, D. Andaraweera, and P. Palihena, "A comprehensive review of methods for SQL injection attack detection and prevention," *Int. J. Sci. Res. Sci. Technol. IJSRST*, 2022.

[33] M. A. Hussain *et al.*, "Web application database protection from SQLIA using permutation encoding," *ACM Int. Conf. Proceeding Ser.*, no. March, pp. 13–21, 2021, doi: 10.1145/3459955.3460594.

[34] J. Abirami, R. Devakunchari, and C. Valliyammai, "A top web security vulnerability SQL injection attack - Survey," *ICoAC 2015 - 7th Int. Conf. Adv. Comput.*, 2016, doi: 10.1109/ICoAC.2015.7562806.

[35] J. R. Khan, S. A. Farooqui, and A. A. Siddiqui, "A Survey on SQL Injection Attacks Types & their Prevention Techniques," *J. Indep. Stud. Res. Comput.*, vol. 21, no. 2, pp. 10–13, 2023, doi: 10.31645/jisrc.23.21.2.1.

[36] W. B. Demilie and F. G. Deriba, "Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques," *J. Big Data*, vol. 9, no. 1, 2022, doi: 10.1186/s40537-022-00678-0.

[37] K. Elshazly, Y. Fouad, M. Saleh, and A. Sewisy, "A survey of SQL injection attack detection and prevention," *J. Comput. Commun.*, vol. 2014, 2014.

[38] Z. C. S. S. Hlaing and M. Khaing, "A detection and prevention technique on sql injection attacks," in *2020 IEEE Conference on Computer Applications (ICCA)*, IEEE, 2020, pp. 1–6.
[39] P. Suri, "DATA PROTECTION : SQL INJECTION PREVENTION," no. 01, pp. 2716–2732, 2024.
[40] D. Chen, Q. Yan, C. Wu, and J. Zhao, "Sql injection attack detection and prevention techniques using deep learning," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 12055.
[41] T. Jones-Low, "Security benefits are provided by using stored procedures to access data." Accessed: Apr. 22, 2024. [Online]. Available: https://stackoverflow.com/questions/421553/what-security-benefits-are-provided-by-using-stored-procedures-to-access-data
[42] K. Ahmad and M. Karim, "A method to prevent SQL injection attack using an improved parameterized stored procedure," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, 2021.
[43] A. Goyal and P. Matta, "Beyond the Basics: A Study of Advanced Techniques for Detecting and Preventing SQL Injection Attacks," in *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, 2023, pp. 628–631. doi: 10.1109/ICOSEC58147.2023.10276077.
[44] R. Johari and P. Sharma, "A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection," in *2012 international conference on communication systems and network technologies*, IEEE, 2012, pp. 453–458.
[45] R. Johari and P. Sharma, "A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection," *Proc. - Int. Conf. Commun. Syst. Netw. Technol. CSNT 2012*, pp. 453–458, 2012, doi: 10.1109/CSNT.2012.104.
[46] Z. S. Alwan and M. F. Younis, "Detection and prevention of SQL injection attack: a survey," *Int. J. Comput. Sci. Mob. Comput.*, vol. 6, no. 8, pp. 5–17, 2017.
[47] B. Brindavathi, A. Karrothu, and C. Anilkumar, "An Analysis of AI-based SQL Injection (SQLi) Attack Detection," *Proc. 2023 2nd Int. Conf. Augment. Intell. Sustain. Syst. ICAISS 2023*, no. Icaiss, pp. 31–35, 2023, doi: 10.1109/ICAISS58487.2023.10250505.
[48] M. S. Husain, "Nature inspired approach for intrusion detection systems," *Des. Anal. Secur. Protoc. Commun.*, pp. 171–182, 2020.
[49] S. Roy, S. Biswas, and S. S. Chaudhuri, "Nature-inspired swarm intelligence and its applications," *Int. J. Mod. Educ. Comput. Sci.*, vol. 6, no. 12, p. 55, 2014.
[50] A. Darwish, "Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications," *Futur. Comput. Informatics J.*, vol. 3, no. 2, pp. 231–246, 2018, doi: https://doi.org/10.1016/j.fcij.2018.06.001.
[51] H. Tavakoli and B. D. Barkdoll, "Sustainability-based optimization algorithm," *Int. J. Environ. Sci. Technol.*, vol. 17, no. 3, pp. 1537–1550, 2020, doi: 10.1007/s13762-019-02535-9.
[52] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in *2007 IEEE congress on evolutionary computation*, Ieee, 2007, pp. 4661–4667.
[53] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 12, pp. 4831–4845, 2012, doi: 10.1016/j.cnsns.2012.05.010.
[54] R. C. Jeyavim Sherin and K. Parkavi, "Investigations on Bio-Inspired Algorithm for Network Intrusion Detection – A Review," *Int. J. Comput. Networks Appl.*, vol. 9, no. 4, pp. 399–423, 2022, doi: 10.22247/ijcna/2022/214503.
[55] John H. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1975.
[56] J. Guo, Y. Li, and Z. Tu, "Research on System of Genetic Algorithm-Based SQL Injection Test Data," in *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, IEEE, 2023, pp. 717–722.
[57] A. S. Jaradat, A. Nasayreh, Q. Al-Na'amneh, H. Gharaibeh, and R. E. Al Mamlook, "Genetic Optimization Techniques for Enhancing Web Attacks Classification in Machine Learning," in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, IEEE, 2023, pp. 130–136.

[58] D. Karaboğa, "AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION,"2005.[Online]. Available: https://api.semanticscholar.org/CorpusID:8215393

[59] G. H. Varazdin, *Icwi Ac 2021 Genomic Data Analysis : Conceptual Framework for the Application of Artificial Intelligence in Personalized*, no. November. 2021.

[60] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009 World congress on nature & biologically inspired computing (NaBIC)*, Ieee, 2009, pp. 210–214.

[61] S. Venkatramulu and C. V Guru Rao, "CSES: Cuckoo Search Based Exploratory Scale to Defend Input-Type Validation Vulnerabilities of HTTP Requests," in *Proceedings of the Second International Conference on Computational Intelligence and Informatics: ICCII 2017*, Springer, 2018, pp. 245–256.

[62] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, 2010, pp. 65–74.

[63] X. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, 2012.

[64] 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.

[65] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politec. di Milano*, 1992.

[66] K. Baptista, A. M. Bernardino, and E. M. Bernardino, "Detecting SQL Injection Vulnerabilities Using Nature-inspired Algorithms," in *International Conference on Computational Science*, Springer, 2022, pp. 451–457.

[67] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[68] B. Arasteh, B. Aghaei, B. Farzad, K. Arasteh, F. Kiani, and M. Torkamanian-Afshar, "Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms," *Neural Comput. Appl.*, pp. 1–22, 2024.

Journal of Basrah Researches (Sciences) **50**(1), 340 (2024)
DOI: https://doi.org/10.56714/bjrs.50.1.27

دراسة استقصائية حول استخدام الخوارزمية المستوحاة من الطبيعة للحماية من هجمات حقن قواعد البيانات

زينب حيدر جودي

قسم الحاسوب -كلية علوم الحاسوب والرياضيات -جامعة الكوفة, النجف, العراق

معلومات البحث

الاستلام 4 ايار 2024
القبول 18 حزيران 2024
النشر 30 حزيران 2024

الكلمات المفتاحية

1-خوارزمية مستوحاة من الحيوية، 2-نظرة عامة على خلفية حقن SQL، 3-تقنيات الكشف عن حقن SQL والوقاية.

Citation: Zainab H. Jody, J. Basrah Res. (Sci.) 50(1), 340 (2024). DOI:https://doi.org/10.56714/bjrs.50.1.27

الملخص

توفر هجمات حقن SQL، أو هجمات SQLI، تهديدات كبيرة لأمن التطبيقات عبر الإنترنت. إنها تستفيد من نقاط الضعف في أنظمة قواعد البيانات ويمكن أن تؤدي إلى الوصول غير المصرح به إلى البيانات الحساسة وتسويتها. تبحث هذه الدراسة في العديد من الخوارزميات المستوحاة من العمليات الحيوية لمعالجة هذه الهجمات، وتقييم تطبيقاتها وإمكاناتها لتعزيز تدابير الأمن السيبراني. تسلط الملاحظة الضوء على أهمية الدراسة المستمرة وتحسين خوارزميات الكشف، لا سيما تلك التي تستفيد من القدرة على التكيف المتأصلة في التقنيات المستوحاة من الحياة والتعلم الآلي. من الضروري استخدام هذه الاستراتيجيات المتقدمة للحماية بشكل فعال من هجمات حقن SQL. وهذا يتطلب التعاون بين الأوساط الأكاديمية والصناعة. ولضمان الاعتمادية والجدارة بالثقة لأنظمتنا الرقمية، يجب علينا تعديل استراتيجياتنا الدفاعية لمواكبة التهديدات السيبرانية دائمة التطور. ويسلط التقرير الضوء على الحاجة الملحة إلى حلول أمنية تتسم بالمرونة والقابلية للتكيف للدفاع ضد التهديدات السيبرانية المتغيرة باستمرار.

***Corresponding author email:** zainabh.alfaham@student.uokufa.edu.iq

N: 1817-2695 (Print); 2411-524X (Online)
line at: https://jou.jobrs.edu.iq