Received: 3 March 2023 | Revised: 12 November 2023 | Accepted: 5 January 2024

DOI: 10.1002/spy2.370

SURVEY PAPER

WILEY

Analysis of SQL injection attacks in the cloud and in WEB applications

Animesh Kumar | **Sandip Dutta** | **Prashant Pranav**

Department of Computer Science & Engineering, Birla Institute of Technology, Mesra, Ranchi, India

Correspondence
Prashant Pranav, Department of Computer Science & Engineering, Birla Institute of Technology, Mesra, Ranchi 835215, India.
Email: prashantpranav19@gmail.com

Abstract

Cloud computing has revolutionized the way IT industries work. Most modern-day companies rely on cloud services to accomplish their day-to-day tasks. From hosting websites to developing platforms and storing resources, cloud computing has tremendous use in the modern information technology industry. Although an emerging technique, it has many security challenges. In structured query language injection attacks, the attacker modifies some parts of the user query to still sensitive user information. This type of attack is challenging to detect and prevent. In this article, we have reviewed 65 research articles that address the issue of its prevention and detection in cloud and Traditional Networks, of which 11 research articles are related to general cloud attacks, and the rest of the 54 research articles are specifically on web security. Our result shows that Random Forest has an accuracy of 99.8% and a Precision rate of 99.9%, and the worst-performing model is Multi-Layer Perceptron (MLP) in the SQLIA Model. For recall value, Random Forest performs best while TensorFlow Linear Classifier performs worst. F1 score is best in Random Forest, while MLP is the most diminutive performer.

KEYWORDS

cloud based SQL attacks, cloud security, machine learning, network security, SQL, SQL injection attacks, web based SQL attacks

1 | INTRODUCTION

A SQL Injection Attack (SQLIAs) is a cyber-attack where malicious SQL code is inserted into a vulnerable application's input fields to manipulate the application's database. This malicious technique exploits vulnerabilities in a software application's interaction with a database. It allows an attacker to manipulate the SQL queries executed by the application, potentially gaining unauthorized access to sensitive data, altering data, or even taking control of the entire database server. It is a significant security threat responsible for numerous high-profile data breaches. It occurs when an application does not properly validate user input before incorporating it into SQL queries. Severe security vulnerability can lead

Abbreviations: ACO, Ant Colony Optimization; BoW, Bag-of-words Model; CC, Cloud Computing; CCSD, Cloud Computing SQLIA Detection; CNN, Convolutional Neural Network; CODDLE, Deep Learning-Based Intrusion Detection; CSPs, Cloud Service Providers; DT, Decision Trees; GMSA, Gathering Multiple Signatures Approach; HDLN, Hybrid Deep Learning Network; KNN, K-Nearest Neighbors; LSTM, Long Short-Term Memory-Based SQLIA Detection; MLP, Multilayer Perceptron; NB, Naïve Bayes; PSO, Particle Swarm Optimization; RF, Random Forest; RNN, Recurrent Neural Networks; SQL, Structured Query Language; SQLIA, SQL Injection Attacks; SVM, Support Vector Machine; XSS, Cross Scripting attacks.

to unauthorized access, data theft, data manipulation, and even complete control over a database. The Attackers similarly manipulate user input data to execute arbitrary SQL code on a vulnerable web application. This results in data breaches, unauthorized access, or complete system compromise. It allows attackers to edit, modify, and delete the records present in the database.

SQL Injection (SQLi) attacks remain a prevalent and serious threat in cloud environments, just as in traditional on-premises setups. SQL Injection vulnerabilities can be particularly dangerous in cloud computing due to the shared responsibility model, where cloud service providers (CSPs) handle the infrastructure's security. The shared responsibility model in cloud computing dictates that CSPs secure the underlying infrastructure, including the physical data centres, networks, and hypervisors. This division of responsibilities highlights the critical role of secure coding practices in preventing SQL Injection attacks in cloud environments.

Here's a simplified overview of how an SQL Injection attack works:

- User Input: The attacker provides malicious input, such as specially crafted SQL code, via input fields, URL parameters, or cookies.
- Lack of Sanitization: The application does not adequately validate or sanitize the input, allowing the attacker's SQL code to be directly inserted into the SQL query.
- Malicious SQL Query: The attacker's input is treated as part of the SQL query, altering its intended behavior. This can lead to unauthorized access, data leakage, or data manipulation.

In a Web Application, an SQL Injection attack occurs when a malicious actor exploits vulnerabilities in the application's input fields to execute unauthorized SQL queries against the underlying database. This attack targets web applications that interact with databases, such as login forms, search boxes, comment sections, or any other area where users can input data. SQLIAs are a significant threat to modern web applications. Web-based SQL Injection attacks are prevalent and dangerous because they can lead to unauthorized access, data theft, data manipulation, and, in some cases, even full control of the application or the database server. Here's an overview of SQL Injection in web-based attacks:

SQL Injection Working in Web-Based Attacks are as follows:

- User Input Handling: Web applications often take user input through web forms, URL parameters, cookies, or other means.
- Lack of Input Validation: If the application doesn't properly validate and sanitize this input before constructing SQL queries, it may concatenate the user input directly into SQL statements.
- Malicious Input: An attacker can provide carefully crafted input that includes SQL code, such as a single-quote (′) or UNION statement, to manipulate the SQL query.
- SQL Query Manipulation: The attacker's input becomes part of the SQL query executed by the application. This can lead to unintended query behavior, allowing the attacker to extract, modify, or delete data.

In cloud computing environments, where multiple clients share the same infrastructure, SQLIA can spread quickly and affect numerous clients simultaneously. Some of the threats to cloud computing are discussed below:

Insecure Application Programming Interfaces (APIs): CSPs offer services to their clients and give them access to such services through application programming interfaces. Any unauthorized update to APIs causes a slew of security risks.

Denial of Service (DoS)[1]: Denial of service attacks are defined as attacks that disrupt legitimate consumers' access to services. Distributed DoS attacks are a far more significant threat to cloud service providers.

Abuse of Cloud Resources: CSPs give consumers enormous hardware resources in IaaS. Customers could use this functionality to conduct malicious activities on substantial hardware resources.

VM Escape[2]: This experimental method lets the virtual machine attack its host. The attacker executes malicious code on the virtual machine, attempting to seize control of the operating system and indirectly breaking the Hypervisor.

***Hyper*-Jacking**[3]: The attacker compromises the hypervisor using this technique. The operating system for virtual machines is their primary aim. They establish a false hypervisor on top of the primary hypervisor when they obtain access. The attacker gains direct access to the original hypervisor.

VM Side-Channel Attack[4]: By abusing the target VM's shared hardware with its co-resident VM, the attacker attempts to obtain sensitive information. The attacker examines electromagnetic signals, timing, electricity supply, and other factors to target the victim system. The Prime + Probe method is a side-channel attack to get data from a co-resident VM with the same cache memory.

XSS Attacks[5]: Malicious scripts are added to website code. The attacker sends malicious code web applications using a web browser to another user. The attacker mainly uses the URL encoding method. Hexadecimal codes are frequently used in XSS attacks. Similar to XSS, the impact of SQL injection is more on data integrity and security rather than direct performance degradation. Aliero et al.[6] reviewed for preventing and detecting SQLIAs. The study revealed that with a lack of proper tools and methods, the attacker takes advantage of server-side SQLIAs. AL-Maliki et al.[7] reviewed to improve the security of client-side attacks. Database's SQL Queries are discussed to distinguish between normal and malicious. Faghihi et al.[8] proposed a malware detector based on application class modeling.

SQL Injection attacks are also possible on 6G Communications. Rahman et al.[9] proposed a Deep learning method in the Internet of Things to counter SQL Injection, malware, DNS poisoning, brute force attacks and XSS attacks. Prateek et al.[10] discussed about quantum 6G technology based on IoT and IoV smart applications. Network architecture was proposed for secure and efficient Internet-of-Everything. Prateek et al.[11] proposed an authentication scheme for smart metering infrastructure in a smart grid. The result indicates that it performs reasonably well and enables unconditional security for the data exchanged between entities of smart grid infrastructure.

Figure 1 below shows the attacker bypassing the CSP's firewall to attack the Database server. A hybrid cloud is displayed, consisting of many interconnected networks. All networking devices are connected to a router, CSP's firewall, and so on. Attack enters the CSP's through outside networks using routers and injects malicious codes into database servers.

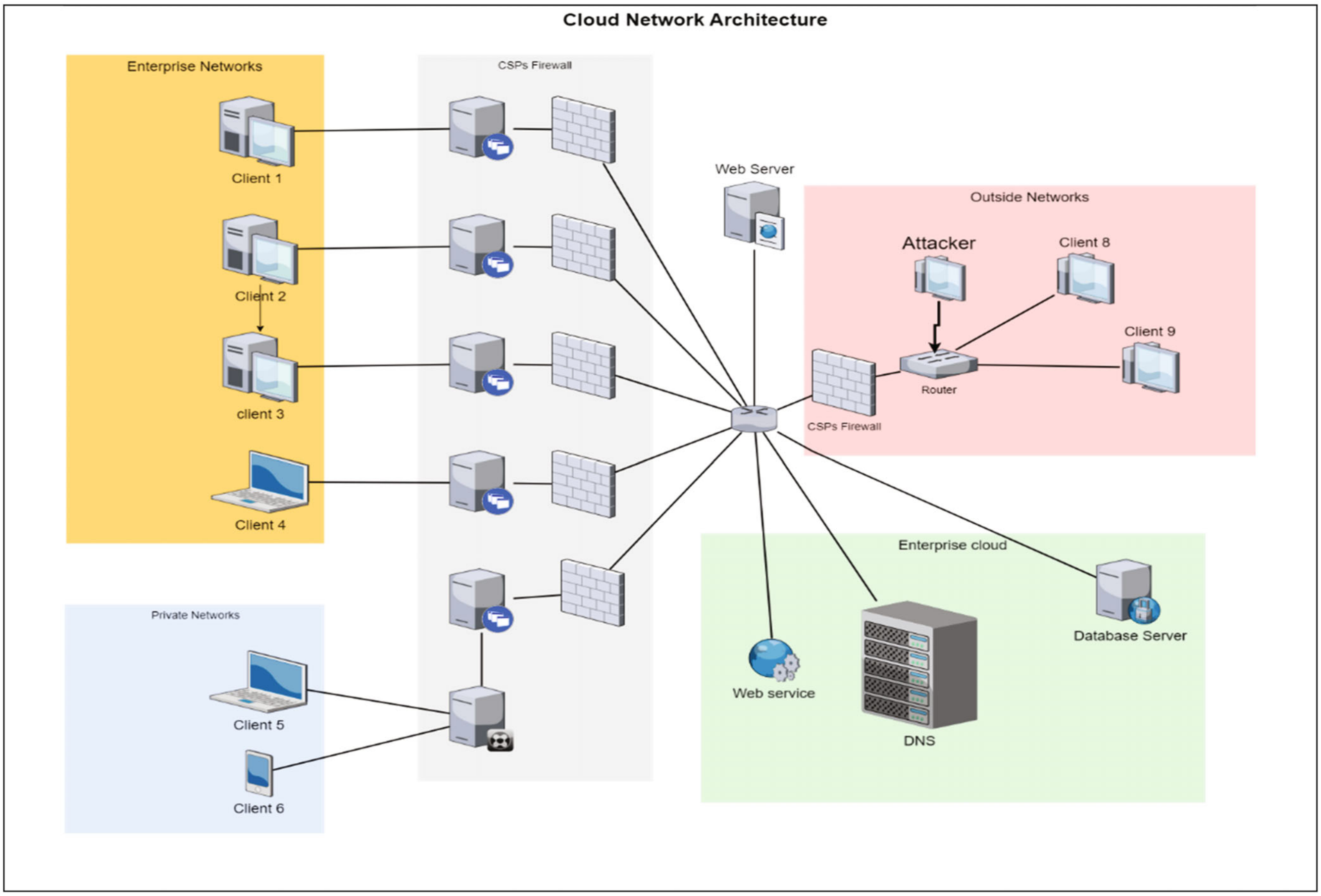

FIGURE 1 Cloud network architecture breached by attacker.

We have focused on SQL injection attacks among many prevailing possible attacks in a cloud environment. It identifies a target cloud computing service that uses SQL-based databases. Examine the target's website or application to identify potential vulnerabilities. The attacker crafts a malicious SQL statement using specialized tools or manually. The attacker sends the prepared SQL statement to the target application or database through the cloud infrastructure. The target application or database executes the malicious SQL statement, potentially allowing the attacker to bypass authentication, access sensitive data, or even take control of the entire system. The attacker can then extract, modify, or delete sensitive data, disrupt the system, or use it as a stepping stone for further attacks.

1.1 | Objective of the research

The main objective of conducting a thorough analysis of research conducted on SQLIAs in the cloud and web-based application are discussed below.

- Most organizations rely heavily on cloud-based services to store and retrieve their confidential data.
- SQLIAs manipulate Queries and attack the Cloud server. So, knowing about the state-of-the-art mechanism to prevent SQLIAs is very relevant.
- Machine Learning is a growing field; almost every sector has its application today. ML models can also detect and prevent SQLIAs with varying Accuracy and Precision. So, we have tried to analyze the Accuracy and Precision of different ML Models used in the detection and prevention of SQLIAs for different Datasets.
  Figure 2 below shows the SQLIAs in cloud web applications.

Figure 3 below explains the working model of SQLIA attack in the cloud, and the attacker compromises the cloud servers using the web application. An attacker generally uses a cloud client system to inject malicious SQL commands into the master database servers.

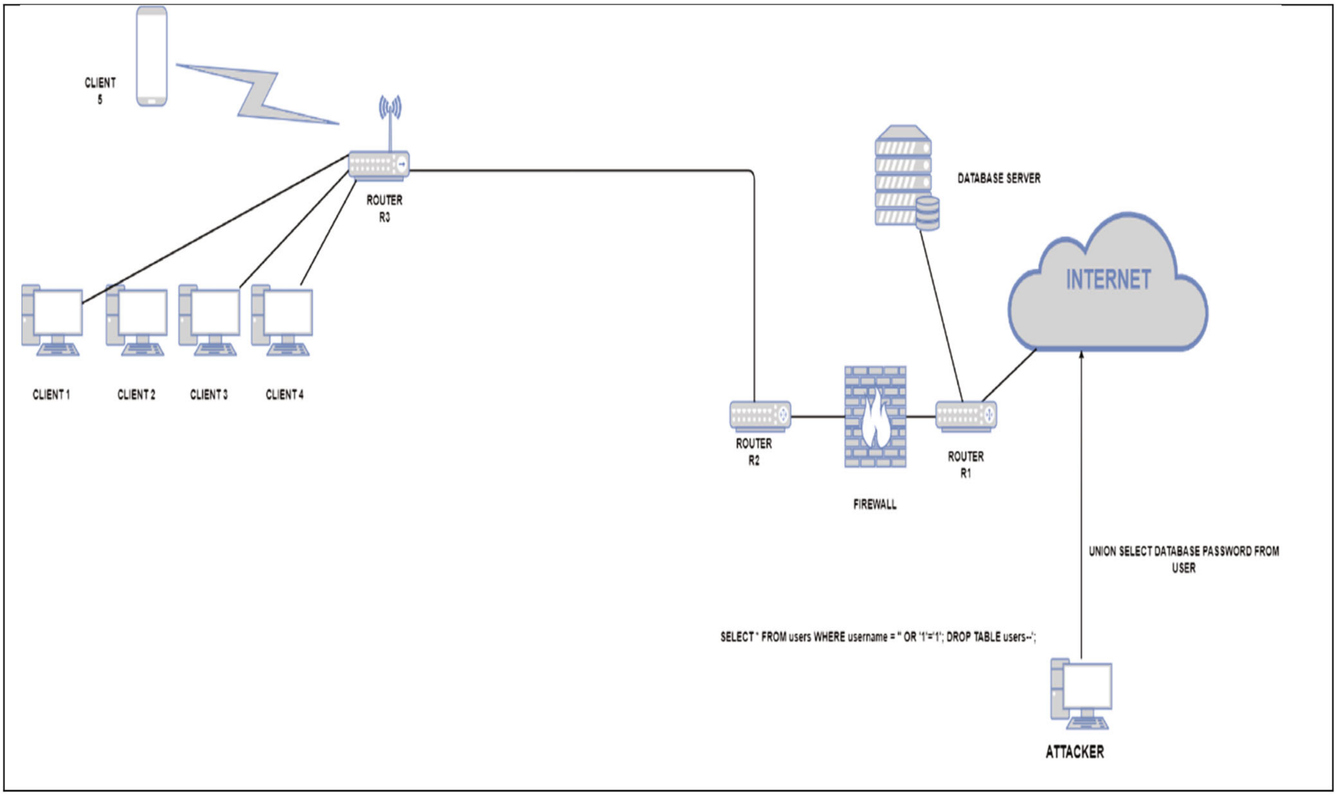
CLIENT 5
ROUTER R3
CLIENT 1
CLIENT 2
CLIENT 3
CLIENT 4
DATABASE SERVER
INTERNET
ROUTER R2
FIREWALL
ROUTER R1
UNION SELECT DATABASE PASSWORD FROM USER
SELECT * FROM users WHERE username = '' OR '1'='1'; DROP TABLE users--';
ATTACKER

FIGURE 2 SQL Injection Attacks in web-based attack.

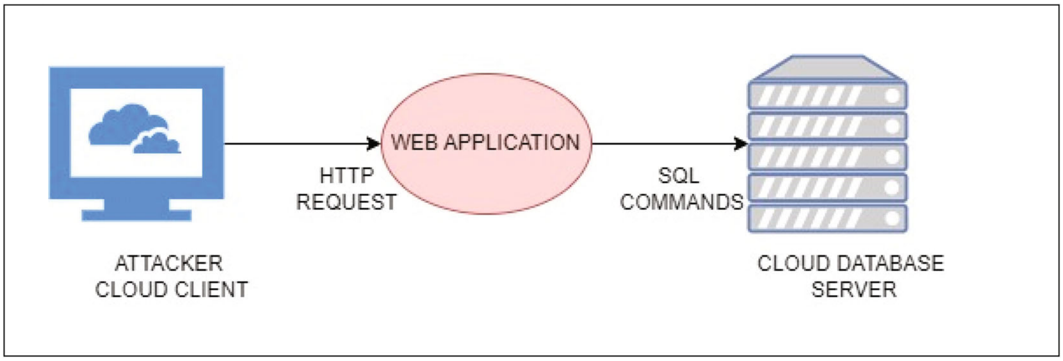

FIGURE 3 SQL Injection Attacks using web applications in the cloud database.

1.2 | Goal of the research

The main goal of this research article is summarized below:

SQLIAs are one of the most hidden attacks and are very difficult to detect. Attacker easily compromises the security of CSPs. The other factor that makes SQLIAs attack one of the most sophisticated attacks in cloud and web applications are as follows: Unauthorized Data Access: SQL injection allows attackers to execute arbitrary SQL queries on a database, potentially gaining access to sensitive data they should not have access to. This can include personal information, passwords, financial records, and more. Data Manipulation: Attackers can use SQL injection to modify, delete, or insert data into a database, potentially causing data corruption, loss, or unintended changes to the system's functionality. Authentication Bypass: SQL injection can be used to bypass authentication mechanisms. For example, an attacker might inject SQL code that allows them to log in as an administrator without providing the correct credentials. Application Compromise: If a web application is vulnerable to SQL injection, attackers can take control of its behavior, potentially using it to distribute malware, deface the website, or perform other malicious activities. Information Leakage: SQL injection can reveal sensitive information about the database structure, which can help attackers refine their attacks or find other vulnerabilities in the system. Business Impact: SQL injection attacks can have a significant impact on a business, including financial losses, reputation damage, and legal consequences. Breaches of sensitive data can result in fines and lawsuits. Persistence: Once an attacker gains access through SQL injection, they may establish a persistent presence in the system, exploiting it over time. Difficult Detection: SQL injection attacks can be challenging to detect because they often look like legitimate database queries. This can allow attackers to remain undetected for an extended period. Widespread Vulnerability: SQL injection is widespread, and many web applications remain vulnerable. Attackers can exploit known vulnerabilities without needing to discover new ones.

Good quality peer-reviewed Research articles from 2017 to 2023 were mainly considered during our research work.

1.3 | Contribution of the research

The main contribution of this research article is summarized below:

- We examine the existing tools and techniques for detecting and preventing SQLIAs.
- We investigate the existing proposed method of previous researchers in SQLIAs to detect their Accuracy, Precision, F1 Score, and Recall Value.
- Traditional detection and prevention method for SQLIAs have their shortcomings. It encourages us to write a research article on SQLIAs over the cloud and web applications.
- To the best of our knowledge, no traditional research article on SQLIAs covers cloud attacks and web security simultaneously.

2 | TYPES OF SQL INJECTION ATTACK

Some of the most common types of SQL Injection attacks are discussed below.

It is mainly subdivided into three types, that is, Web-based, Data Transmission, and Domain Name Server-based attacks. Figure 4 below explains the types of SQL injection attacks.

Data Transmission attacks are subdivided into Time-based blind and Bool-based. Tautology and Inequality are parts of Bool-based blind. Domain Name Server consists of Execute SQL Statement and Stored Procedure. The executable function is the extension of the Execute SQL Statement. Executable stored is an extension of the Stored Procedure. Some of the SQL Types are listed below.

- **Boolean-Based Blind SQL Injection**: During this attack, the attacker sends malicious queries that force the application to produce true or false conditions. By analyzing the application's responses, the attacker can infer whether the injected query returned true or false, helping them extract sensitive data or manipulate the database.
- **Error-based SQL injection** leverages error messages generated by the database to gather information about the underlying schema and data. Attackers deliberately inject erroneous queries, causing the application to display detailed error messages that expose crucial information. In union query-based SQL injection, the attacker appends their own crafted queries to existing ones using the UNION operator. This technique allows the attacker to combine the results of their queries with the original query's output, potentially revealing sensitive data. Error-based attacks are categorized as both abnormal character and abnormal command.
- **Time-based blind SQL injection** is used when the application's response doesn't directly disclose information. The attacker inserts time-delaying statements into the queries to observe if the application's response time changes, indicating a successful injection. Piggybacked queries involve the attacker injecting multiple queries in a single input field, exploiting situations where the application allows the execution of various queries sequentially. This enables the attacker to perform several operations in a single request. When a web application relies on stored procedures for database operations, an attacker can perform a stored procedure SQL injection. By injecting malicious code into the procedure parameters, the attacker can execute unauthorized actions or gain unauthorized access to the database.

2.1 | Summary of SQLIA and its types

- We conducted a comprehensive security analysis of SQL injection attacks, a pervasive and persistent threat to database-driven web applications.

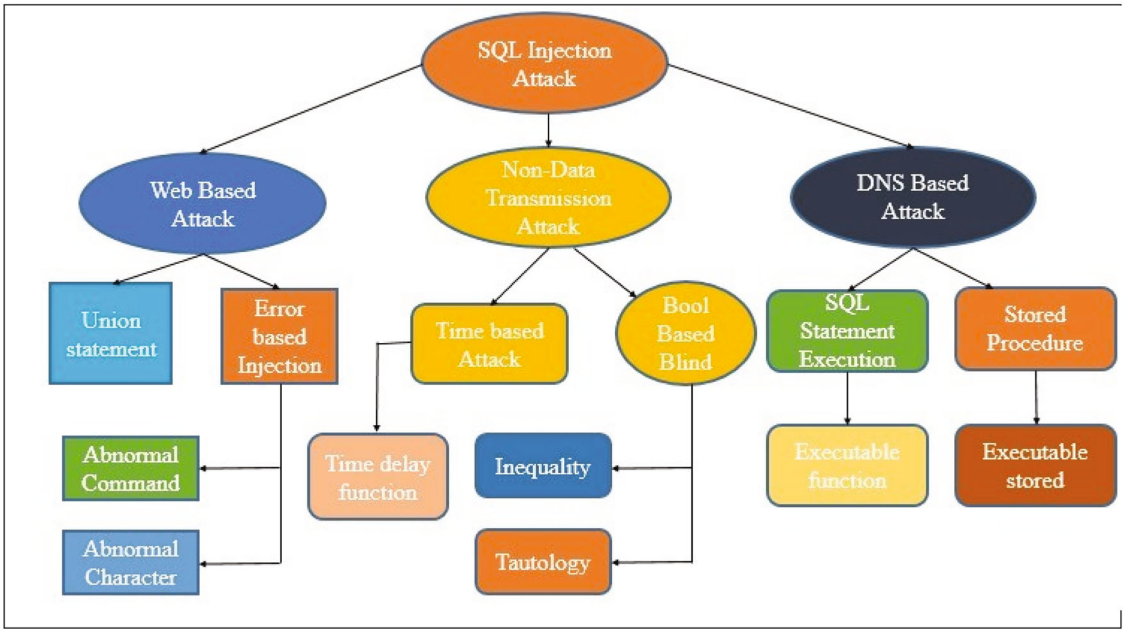

FIGURE 4 Types of SQL Injection Attacks with their subtypes.

- Using case studies, detection methodologies, and mitigation strategies, we highlighted the evolving nature of SQL injection, including its potential to lead to data breaches, unauthorized access, and system compromise.
- Our findings underscored the critical importance of proactive security measures, including input validation, parameterized queries, and access controls, to mitigate SQL injection risks.
- We discussed emerging attack trends, such as time-based blind injection, and emphasized the need for continuous security education, advanced mitigation tools like Web Application Firewalls (WAFs), and vigilant monitoring.
- This analysis serves as a comprehensive resource for organizations and developers seeking to safeguard their systems against the ever-present and evolving threat of SQL injection.

2.2 | Consequences of SQLIA

SQL injection attacks pose significant risks to both individuals and organizations, including data breaches, financial losses, reputational damage, and legal consequences.

Some of the potential consequences of a successful SQL injection attack are discussed below.

- Data exposure: Attackers can access, retrieve, and steal sensitive data from the database, such as user credentials, personal information, financial records, or intellectual property. Confidential business data and customer information can be compromised, leading to privacy violations and potential legal consequences.
- Data modification: Attackers can manipulate or modify data in the database. This can lead to data corruption, unauthorized changes, or fraudulent transactions. Attackers may sometimes alter data to cover their tracks or disrupt business operations.
- Data Deletion: Attackers may delete or destroy data in the database, causing data loss and potentially disrupting business operations. This can lead to financial losses and damage to an organization's reputation.
- Application disruption: SQL injection attacks can disrupt the normal operation of web applications, rendering them temporarily or permanently unavailable. Downtime can result in lost revenue, damage to user trust, and the cost of remediation.
- Financial losses: Organizations may incur financial losses due to fraud, legal expenses, regulatory fines, and costs associated with restoring systems and data. Stolen financial information can be used for unauthorized transactions or identity theft.
- Reputation damage: Security breaches resulting from SQL injection attacks can damage an organization's reputation, losing customer trust and loyalty. Negative publicity and media coverage can further harm the brand's image.
- Legal and Regulatory Consequences: Organizations may face legal actions, regulatory penalties, and compliance violations depending on the nature of the data exposed or manipulated. Data breach notification requirements may also apply, leading to additional costs and reputational damage.
- Operational disruption: Remediation efforts to fix vulnerabilities and recover from SQL injection attacks can disrupt normal business operations. Organizations may need to allocate significant resources to investigate, mitigate, and prevent future attacks.
- Compromised user accounts: User accounts may be compromised through SQL injection attacks, leading to unauthorized access, identity theft, or account takeover. Users may suffer financial losses and privacy violations.

Figure 5 below explains that SQLIAs are mainly solved by Etiological, Symptomatic, and Hybrid Techniques. Policy Enforcement, Instruction set randomization (ISR), and parse tree validation are three sub-parts of etiological methods. Training and Taint Tracking are sub-parts of the Symptomatic approach.

Table 1 below explains the comparative analysis of existing proposed methods to counter SQLIAs using five categories of SQLIAs.

2.3 | Classification of SQL Injection Attack

Some of the classifications of SQLIAs are discussed below:

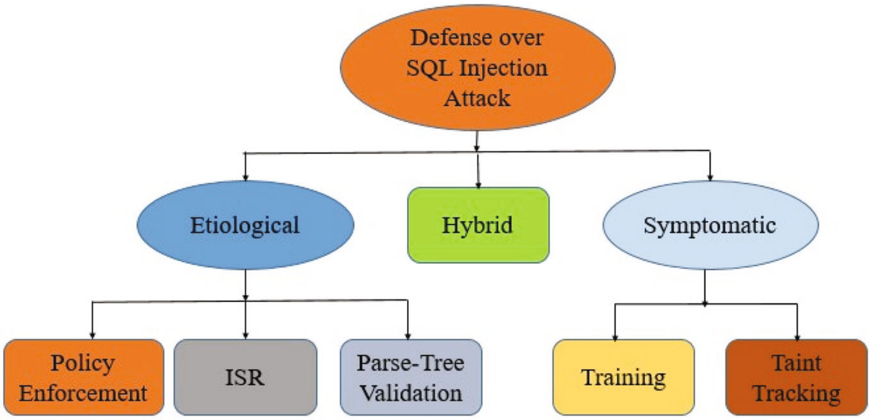

FIGURE 5 Defensive method to solve SQL Injection Attacks.

TABLE 1 Defense methods of SQL Injection Attacks.

Parse tree validation	Policy enforcement	ISR	Taint tracking	Training	Hybrid
Lu et al.[12]	Alotaibi et al.[13]	Patidar et al.[14]	Liu et al.[15]	Alghawazi et al.[16]	Saini et al.[17]
Sheth et al.[18]	Okesola et al.[19]	Labib et al.[20]	Marashdih et al.[21]	Mehta et al.[22]	Henry et al.[23]
Pruzinec et al.[24]	Irungu et al.[25]	Fratty et al.[26]	Logozzo et al.[27]	Singh et al.[28]	Crespo-Martínez et al.[29]
Fu et al.[30]	Philip et al.[31]	Barsellotti et al.[32]	Mallissery et al.[33]	Lu et al.[12]	Alotaibi et al.[13]
Guan et al.[34]	Nasrullayev et al.[35]	Muhammad et al.[36]	Brintha et al.[37]	Al Badri et al. [36]	Lu et al.[12] (2023)

- **SQL Tautology Attack**: During this attack, the injected code uses the OR conditional operator, and the query always estimates TRUE. It generally bypasses the authentication step, and data extraction is done using the WHERE section of SQL queries. All the database table rows are open to unapproved users.

  Vulnerable Input: In a web application or software that interacts with a database, user input is often used in SQL queries without proper validation.

  Injection of Tautology: The attacker injects a tautology into the input fields. A tautology is a logical expression that is always true, such as “1 = 1” or “true” in SQL syntax.

  Manipulating the Query: When the application processes the user input and constructs the SQL query, the injected tautology causes a condition to always evaluate as true, regardless of the original query’s intent.

  Unauthorized Access or Manipulation: Depending on the context and the specific query being targeted, the attacker can use the tautology to gain unauthorized access to data, bypass authentication, or manipulate the query to achieve their malicious goals.

  However, Guardium prevents this type of attack.

- **Arbitrary String Pattern**: An arbitrary string pattern in SQL refers to a flexible or generic pattern that matches or filters strings based on certain criteria. The term “arbitrary” means that you’re not specifying a fixed or exact string but rather a pattern that can match various strings that adhere to a particular structure or format. The attacker adds characters in between the queries. They use this concept to decode the signature detection of ‘or, ‘Union’ characters, etc.

  /* to start, */ to end the comment line in C language.

  Multiline in C syntax code is used to make SQL injection attacks unpredictable.

- **Group Concatenate String**: ‘concat’ or ‘Group_Concat’ are used in this type of SQL injection attack. The attacker does not require ‘like,’ ‘or’ characters.

  For example, Select Group_Concat (Roll number, password2) From student 1.

  Select Concat (Roll number, Password2) From student 2.

- **Stored Procedure**: is a precompiled collection of one or more SQL statements or commands that are stored in the database and can be executed as a single unit. Stored procedures are typically used to encapsulate a set of operations or business logic that can be repeatedly performed on a database. They offer several advantages, including code reusability, improved performance, and enhanced security.

\Privilege escalation is used in these types of attacks. Xp_CmdShell, drop Table, Shut-Down, etc., are mainly used. DoS and Random command attacks come under this category.

- **Alternate Encoding**: SQL injection commands are inserted into the database using encoding methods. ASCII (), DEC (), UNHEX (), BASE64(), BIN(), etc., are some of the signatures of SQL injection. Delay, drop, update, shutdown, execute, etc., are some commands that require scanning for SQL injection attacks. "Wait" signature must be followed by "delay" whereas "into" followed by the "insert" signature. To prevent this, we use UTF-8 representation in XML. The command used to detect the signature is "&#."
- **Piggybacked Queries:** SQL typically refers to a technique where an attacker injects additional SQL queries into an existing query to execute multiple queries in a single request. This technique is commonly used in SQL injection attacks to perform malicious actions within the context of an initial legitimate query.

  An attacker injects additional queries into the Grade Central site's original query to add, modify, or delete student accounts.

2.4 | SQL Injection Attack tools

Here are some common SQL Injection Attack Tools explained below.

- **Havji**: It is an automated tool developed by *ITSecTeam* to discover and attack SQL Injection vulnerabilities on a web page. Multiple databases and SQL injection methods of all kinds are supported. It can also fingerprint and extract a database's entirety. It also has access to execute operating system commands and the file system. The user interface is simple to use. Even amateur hackers will find it simple, thanks to automated settings.
- SQLMap is one of the most popular and powerful open-source SQL Injection tools. It automates the process of detecting and exploiting SQL Injection vulnerabilities in web applications. It can work with various database management systems (DBMS) such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
- SQLninja is an open-source SQL Injection tool designed to automate the process of exploiting SQL Injection flaws in web applications. It can be used for educational purposes to demonstrate the impact of SQL Injection.
- BSQL Hacker is a commercial tool that provides automated SQL Injection exploitation capabilities. It allows security professionals to test the security of web applications and databases.
- JSQL is an open-source tool for detecting and exploiting SQL Injection vulnerabilities in Java web applications. It can be used to test Java-based web applications for SQL Injection flaws.
- SQL Inject Me is a Firefox browser extension designed for security testing. It helps identify SQL Injection vulnerabilities by sending various SQL Injection payloads to web forms and checking for potential vulnerabilities.
- BeEF: While primarily a Browser Exploitation Framework, BeEF can be used to test web applications for vulnerabilities, including SQL Injection, by exploiting client-side vulnerabilities in browsers.
- AQUATONE is a reconnaissance tool that can help identify web application vulnerabilities, including SQL Injection. It can be used to discover potentially vulnerable web applications for further testing.
- ZAP (OWASP Zed Attack Proxy): SQL Injection tool, ZAP is an open-source security testing tool that includes automated scanners for various web application vulnerabilities, including SQL Injection. It is a part of the Open Worldwide Application Security Project (OWASP) Project.
- W3af: The Web application attack and Audit Framework (W3af) is an open-source web application security testing tool that includes modules for detecting and exploiting SQL Injection vulnerabilities.

2.5 | Existing mechanism of defense over SQLIA

- **Parse-Tree Validation**: It is an effective method for detecting DSL code injection attacks. SQL checks are preferred in this section. Java Scripts are generally used in this. It is based on an etiological method. SQLGuard and SQLcheck are two mechanisms for the protection of SQL injection attacks.

- **Policy Enforcement:** XSS and CSRF attacks are stopped using this method. Policies are designed based on pattern matching, syntax-oriented, and JavaScript extension. Security features deployed on the server side. It is based on an etiological approach. Content security policy, NoForge, BrowserShield, JCSRF, CsFire, SOMA, Noxes, and NoForge are some of the essential mechanisms in policy enforcement.
- **Instruction Set Randomization** (**ISR**): These are first applied to prevent binary code injection attacks. Randomization algorithms are used in this approach. Randomized keys are also used. It is based on a deterministic approach. SMASK, SQLrand, Noncespaces, XJS, etc., are some of the best mechanisms of ISR. SQLrand integrates proxy with database server while Noncespaces and XJS need modification on both server and client-side. Comes under the etiological category.
- **Taint Tracking:** Suspicious data, programs set in a web form, and traces of programs come under taint tracking schemes. It is based on the symptomatic technique. Perl and Ruby programming languages use the taint tracking method. They refuse to run the vulnerable code for SQL injection attacks.
- **Hybrid Approach**: Diglossia, SIF, Hails, and Aeolus are essential mechanisms for preventing SQL injection attacks. It is the combination of both etiological and symptomatic approaches.

3 | SQLIA IN CLOUD COMPUTING

SQL injection is a critical security vulnerability that affects cloud computing environments. Cloud computing uses remote servers and services to store, manage, and process data and applications. Cloud-based applications and databases are susceptible to SQL injection attacks. In multi-tenant cloud environments, multiple customers share the same underlying infrastructure. If one customer's application is vulnerable to SQL injection, it potentially impacts other customers' data and applications if the attacker gains unauthorized access to the shared resources. Cloud applications use NoSQL databases that are not based on traditional relational models and may have different security considerations. NoSQL databases can still be vulnerable to injection attacks if proper input validation and sanitization are not applied. Literature reviews are as follows.

3.1 | Literature review

Wirz et al.[38] implemented a cloud-based system that uses Apache Kafka and Spark streaming to detect and process high-traffic inputs in HTTP communications. They used signature-based detection to identify unusual input patterns and reduce false alarms in SQL injection attacks. This system's architecture is designed to handle many inputs efficiently and scale as needed. Their results were impressive, as they detected 89.38% of SQL injection attacks and 97.5% of Cross-scripting attacks. They also improved the system to detect and prevent other cyber-attack types, such as Cross-Site and OS Command injection attacks. In summary, their implementation demonstrated the potential of cloud-based systems to detect and prevent various types of cyber-attacks in real-time.

Tripathy et al.[39] tackled the challenges of SQL injection attacks (SQLIA) in Software-as-a-Service (SaaS) cloud environments by using machine learning techniques. They explored several classifiers to find the best solution, including random forest, adaboost, Deep Neural Network, Decision tree, and Tensorflow. The results showed that the random forest classifier achieved an excellent accuracy rate of 99.8%. Future work suggests a collection of the different datasets from other domains to improve the algorithm for combating SQL attacks in the cloud environment.

Rongzhou[40] presented a filtering method as a defense mechanism against cyber-attacks, particularly SQL injection and Cross-site scripting (XSS) attacks. The proposed technique involves filtering the input parameters to identify and eliminate sensitive characters that could be used in these attacks. The article also explains the design of the sensitive character library. IBM Security AppScan is used for SQL scanning.

Shunmugapriya[41] utilized the Twofish encryption technique to enhance the security of clients' data from SQL injection attacks (SQLIA). The article includes a functional block diagram of the Twofish algorithm and thoroughly explains its working mode. Furthermore, the article discusses crypt analysis and the performance of the Twofish encryption method. The results suggest that Twofish can robustly protect against SQLIA and other cyber threats.

Singh et al.[42] proposed a hybrid approach to tackle SQL injection attacks by combining static, dynamic, and runtime detection methods with a preventive approach. The proposed model comprises three main components: client, logic access, and data server. The article explains the proposed algorithm, Query Access and Denial System, which utilizes Python. The algorithm aims to prevent SQL injection attacks by detecting and denying unauthorized access to the data server. The configuration of the attacker's server is designed using a master–slave approach, where the master server includes WASP, Apache, and MySQL. In contrast, the slave server creates an SQL injection attack environment. The study's results showed an 80%–90% accuracy rate, which can further improve firewall security. In future work, the authors suggested improving the utility rate of the proposed approach.

Ranjan[43] explored different types of malware attacks, such as SQLIA, Cross-site scripting (XSS), command infusion, etc., in the cloud environment. The article explains the SQL generation process in the cloud and presents a data flow diagram to illustrate the scenario of SQLIA. The study results show that the proposed algorithm can increase the cloud's access speed. Still, there is a need for further improvement in the algorithm to enhance the cloud's protection against various malware attacks.

Web application security from SQL injection attacks, XSS attacks, and so on, Bishit et al.[44] proposed SECUREWEB. It is based on proxy-based source code analyzer SECUREEYE and SECURESOLUTION for auto-patching of threats. The secure web has three stages: monitoring, injection, and attack. A secure solution consists of the prevention stage. The flowchart of the proposed tool SECUREWEB is explained. Boolean-based blind SQL injection and MongoDB NoSQL injection attack scenarios were taken and solved. This model also takes care of phishing attacks in web applications. The architecture of the secure solution is designed for the software-as-a-service (SaaS) model.

Web-based application threats in cloud computing, especially SQLIA, are highlighted by Ponmuthukalaiselvi.[45] Two fish encryption method is used for saving clients' data. The functional block diagram of two fish is described using multiple rotations of S-boxes. Cryptanalysis of the algorithm is also mentioned. The proposed algorithm is explained through a data flow diagram. ASP.net is used for the web application. The result shows that the proposed algorithm prevents and protects the data from advanced SQL injection attacks in cloud computing.

Wu et al.[46] proposed a CCSD model for cloud computing SQLIA detection. This model can be deployed to any cloud environment. The architecture of CCSD is explained. The model compares the parse tree structure to the user's request and works on both offline and online phases. The result shows it has an accuracy of over 99%, the false positive rate is less than 4%, and less time-consuming compared to the traditional approach. The CCSD approach is compared with static and dynamic analysis, taint tracking, and so on. In the future, CCSD can be enhanced by adding NoSQL databases.

Yassin et al.[47] proposed the SQLIIDaaS model, especially for SaaS providers. This model uses the concept of SQL query and HTTP request mapping. The intrusion detection system does the quality of SQL detection—integration of proposed work framework in Amazon web service. The article ignores the IaaS and PaaS models of cloud computing. Implementation of the model is shown using HTTP Threat, SQL threat, correlation Module, and detection module. There is scope for improvement in the proposed algorithm for tenant's angle by ensuring the cloud model's on-demand service and pay-as-you-go characteristics.

Rajeh[48] proposed a three-tier technique for solving SQL attacks in cloud systems. Client logic, access, and data server are three-tier architecture models of the proposed method. Detection and mitigation are in the first stage. Authentication, coupling, cohesion, logs, and audits occur in the second phase. The third phase is purely based on the database server. The proposed method applies to web applications. This technique is deployed on proxy servers where they filter out infected queries. The proposed algorithm can be improved by countering any high-level SQL injection attack in the future.

Wang[49] proposed an SQLIA detection technique in a cloud computing environment based on dynamic taint analysis and input filtering of SQL statements. Lexical analysis of the input string takes place in the first stage. Tree rules are further added to keep out infected strings. Parsing algorithms are explained using a flow chart. Tools like SQLMap, SQLninja, and SQLPoison are used. The result shows an improvement in the accuracy rate for SQL injection attacks.

4 | SQLIA IN WEB APPLICATION

SQL over web applications refers to using SQL to interact with databases from within a web application. Web applications often require access to databases to store, retrieve, and manipulate data dynamically. SQL is the standard language for managing relational databases, and it allows web developers to perform various database operations, such as querying, inserting, updating, and deleting data. Literature reviews are mentioned below.

Ibarra-Fiallos et al.[50] proposed a validation filter based on OWASP stinger against common injection attacks in web-based applications. All types of SQL injection attacks are discussed, along with web-based attacks. The article ignores URL parameters in web applications and tests the sample size in web applications. An accuracy of 98.4% is achieved with an average processing time of 50 ms performed using extra resources. An aggregate report with 1000 concurrent threads is presented. The proposed filter can be improved by adding a URL parameter—testing of a sample size of the web application for improvement in proposed algorithms.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{1}$$

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FN} + \text{TN} + \text{FP})} \times 100, \tag{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3}$$

$$\text{F} = 2.\frac{\text{precision.recall}}{\text{precision} + \text{recall}}, \tag{4}$$

$$\text{DE} = \frac{\text{attacks} - \text{successattacks}}{\text{defensive}} \times 100\%. \tag{5}$$

True Positive (TP) means an attack exists, and the request is stopped. True Negative (TN) means the attack does not exist, and the request is not stopped. False Negative (FN) represents an attack that does not exist, but the request is stopped, and False Positive (FN) represents an attack that existed but is not stopped. DE is the Defensive Efficiency rate with which the filter managed to stop common injection attacks.

Zheng et al.[51] proposed the SQL generation Method. Query samples for real-time databases are used for model training and testing—the novel method of malicious query detection to counter noise exploitation attacks. Algorithms like Cloning attacks, Differential attacks, Pattern Mining of malicious SQL Queries, and SQL generation are successfully implemented. However, the author lacks to solve the re-learning of malicious query classification and detection—91% accuracy rate achieved against cloning attacks by software simulations. The accuracy of malicious query classification and precision between malicious and normal queries is shown.

$$\text{GI}_\text{m} = \sum_{\text{K}=1}^{\text{K}} \sum_{\text{k}' \neq \text{k}} p_{\text{mk}}(1 - p_{\text{mk}'}) = 1 - \sum_{\text{k}=1}^{\text{K}} p_{\text{mk}}^2, \tag{6}$$

where GI is the Gini coefficient, K is the number of classes, p_{mk} is the weight of class k in mode m.

$$\text{VIM}_\text{j} = \frac{\text{VIM}_\text{j}}{\sum_{\text{i}=1}^{\text{m}} \text{VIM}_\text{j}}, \tag{7}$$

where VIM is the variable measures importance based on the Gini coefficient.

Haifeng Gu et al.[52] proposed a novel technique named DIAVA based on a traffic-based framework that can send an early warning to tenants. It can successfully identify all types of SQL injection attacks. The article explains Boolean-based blind SQLIA, error-based SQL, UNION query-based SQLIA, and Time-based SQLIA with their features and Descriptions. Data Extraction using Identifier strings and GPU-based Dictionary attack algorithms are appropriately implemented. DIAVA framework explained both using front-end and back-end. The proposed model includes traffic analysis, behavior analysis, data extraction, and dictionary attack analysis.

Li et al.[53] proposed an adaptive deep forest algorithm to solve SQL injection attacks. AdaBoost algorithms based on deep forest models work on each network layer. This model is better than traditional classical machine learning techniques. The result shows higher efficiency. Thirty detection features for SQL injection attacks using the tabular form and overall procedure of adaptive deep forest are presented.

Abaimov et al.[54] author proposed CODDLE, an intrusion detection system based on a deep neural network to counter web-based attacks. It improves the detection rate to 95% from 75%. Precision rate of 99%, along with a recall value of 92%. Transformation of the original set of data into an encoded pattern. Identifies the hidden layers of neural network and maximum batch size with minimum training cycle shown in the result section. Statistical analysis and evaluation of the model are presented in the experimental setup.

Appelt et al.[55] proposed ML-Driven to test the attack detection capability of web application firewalls with a significant focus on SQL injection attacks. Performance improvement, compared with two different ML-Classifiers. Tree decomposition, ML-driven SQL attack generation, and off springs generation are proposed algorithms explained in the article. The author performed the Wilcoxon test, comparing random Tree and random forest classifiers. The proposed method is compared with state-of-the-art previous techniques. Effectiveness and efficiency can be improved in the proposed algorithm. Future work suggests resampling of data, a penalty-based training approach, and training of weighted samples can be done for the improvement of the proposed algorithm.

Alnabulsi et al.[56] proposed the GMSA technique to defend against code injection attacks. It provides better performance and results as compared to other articles. The accuracy level reaches 99.45%. False positives are also low in this method. Code injection attacks like SQL injection, cross-site script attacks (XSS), shell injection, and file inclusion attacks are broadly debated. Proposed algorithm compared with C 4.5 decision tree, Support Vector Machine, Random Tree, and Random Forest. This technique is also able to detect XSS attacks. Eleven different datasets were created to check the performance of the proposed algorithm.

Liu et al.[57] designed a web second-order threat detection algorithm. It reduces the time complexity of detecting web security. SQL and XSS attacks are mainly considered in this article. The proposed algorithm is presented by using a flowchart. Python and PHP are used in the proposed algorithm. For scanning, ZAP software is used. SQL Testing results are displayed using screenshots. There is a scope for improvement in the proposed algorithm by adding the kinds and qualities of attack vectors.

Mitropoulos et al.[58] proposed models for the critical weaknesses of code injection attacks. Defense mechanisms are also analyzed broadly. Approximately 41 previous research articles are analyzed. Policy-Tree method, Taint tracking, and ISR are some approach methods discussed. Approaches like Parse-Tree validation, Policy enforcement, ISR, Taint tracking, Training, and Hybrid Methods to stop SQL and XSS and other code injection attacks. SQLGuard, SQLcheck, SQLrand, and smask, and so on, are some popular mechanisms to counter SQLIA. There is future scope for new, practical countermeasures against injection attacks.

$$\mathrm{PPV} = \frac{\mathrm{SE} \times \mathrm{PR}}{\mathrm{SE} \times \mathrm{PR} + (1 - \mathrm{SP}\ ) \times (1 - \mathrm{PR})}, \tag{8}$$

$$\mathrm{NPV} = \frac{\mathrm{SP} \times \quad (1 - \mathrm{PR})}{(1 - \mathrm{SE}) \times \mathrm{PR} + \mathrm{SP} \quad \times (1 - \mathrm{PR})}, \tag{9}$$

where Positive Prediction Value (PPV): Probability that a reported attack is real and Negative prediction Value (NPV): Probability that if nothing is reported, no attack occurs.

Bossi et al.[59] proposed a DetAnom mechanism to detect any injection attack. This technique also detects modifications in SQL queries. The relational schema is elaborately discussed. Pseudocode of Anomaly detection explained—test application details based on timing range from 40 seconds to 4 days shown. Network overhead and execution time overhead are presented in the result section. In the future, there is scope to improve the signature generation scheme using different operators.

Li et al.[60] proposed an LSTM SQL injection attack for an intelligent transport system. It can deal effectively with complex, massive data. It can solve the problem of overfitting due to insufficient positive samples. The model can actively generate valid positive samples. SVM, KNN, Decision Tree, NB, and RF classical machine learning methods are compared using CNN, RNN, and MLP deep learning methods. This algorithm improves the accuracy and helps detect SQL attacks. SQL attack injection attack is explained correctly using the data transmission method. The author needs to provide a detailed literature review of SQL injection attacks.

$$\mathrm{Acc} = \left(1 - \frac{\mathrm{errorsum}}{\mathrm{sum}}\right) \times 100\%. \tag{10}$$

Priyaa et al.[61] proposed the framework for detecting SQLIAs using SVM Classification model for web application. Some of the general preventive methods from SQLIAs are as follows:

- **Use parameterized queries**: Parameterized queries can help prevent SQLIA by separating the SQL code from user input data. This ensures that user input data is treated as data rather than code.

- **Validate user input**: Input validation can prevent attackers from injecting malicious code into the application. Ensure that user input data is properly validated and sanitized before it is used in SQL queries.
- **Use least privilege**: Limit the privileges of application accounts to only what is required to perform their functions. This can limit the impact of a successful SQLIA.
- **Use secure coding practices**: Use secure coding practices such as prepared statements, stored procedures, and input validation to prevent SQL injection attacks.
- **Regularly update software**: Keep the software and applications used in the cloud environment updated with the latest security patches and updates.

TABLE 2 Accuracy of methods to counter SQL Injection Attacks.

S. no	Year	Methods	Accuracy	References
1	2021	Apache kafka and Spark Streaming	89.38	38
2	2019	SQL-Security Tool	90.00	42
3	2020	Random Forest	99.80	39
4		TensorFlow BoostedTree Classifier	99.60	39
5		AdaBoostClassifier	99.50	39
6		Decision Tree	99.50	39
7		SGD Classifier	98.60	39
8		Deep ANN	98.40	39
9		TensorFlow Linearclassifier	97.80	39
10	2017	CCSD	99.00	46
11	2019	CODDLE	94.00	54
12	2019	LSTM	91.32	60
13		SVM	90.79	60
14		KNN	89.28	60
15		NB	91.51	60
16		DT	93.57	60
17		RF	93.21	60
18		RNN	90.24	60
19		CNN	88.83	60
20		MLP	87.84	60
21		Word2vec	93.47	60
22		BoW	91.93	60
23	2018	HDLN	97.55	33
24	2020	Pattern mining and detection	94.00	37
25	2021	Effective Filter	98.40	50
26	2018	RandomTree	94.60	56
27	2018	GMSA	99.45	56
28		C4.5 decision tree + ACO	95.06	61
29		SVM + ACO	90.82	61
30		C4.5 decision tree + PSO	95.37	61
31		SVM + PSO	91.57	61
32		Proposed + SVM + PSO	95.67	61

- **Use a Web Application Firewall**: Implement a Web Application Firewall (WAF) to prevent SQLIA. A WAF can identify and block malicious requests before they reach the application.

5 | RESULTS

In this section, we have analyzed the prevailing defense techniques against SQLIA based on many parameters such as accuracy, precision, F1-score, and recall value of the algorithms. A list of proposed methods is arranged concerning their accuracy percentage, as shown in Table 2 below.

Graphical analysis of techniques and accuracy percentage is shown in Figure 6 below.

The list of all proposed methods is arranged concerning their Precision percentage, as shown in Table 3 below.

Graphical Analysis of Techniques along with Precision percentage is shown in Figure 7 below.

A list of all proposed methods is arranged concerning their Recall value percentage, as shown in Table 4 below.

Graphical analysis of techniques and recall value percentage is shown in Figure 8 below.

A list of all proposed methods is arranged concerning their F- measure percentage, as shown in Table 5 below.

Graphical analysis of proposed techniques and F-measure percentage is shown in Figure 9 below.

6 | DISCUSSION

Based on the vast analysis of literature carried out, we can summarize our findings as under:

- As far Regarding the accuracy of various models for detecting SQLIA, Random Forest, with an accuracy of almost 99.8%, outperforms other methods. The worst-performing model is Multi-Layer Perceptron (MLP).
- If we consider the precision of various models concerned for detecting SQLIA, random forest with a precision of almost 99.9% outperforms other methods. The worst performing model is again Multi-Layer Perceptron (MLP).
- The recall value of various models was analyzed, and we found that a random forest with a recall of 99.9% is best for detecting SQLIAs. The worst-performing model in this case is the TensorFlow Linear classifier.
- F1 – score of random forest (99.9%) is the highest among all other models for preventing SQLIAs. In this case, the worst-performing model is MLP.

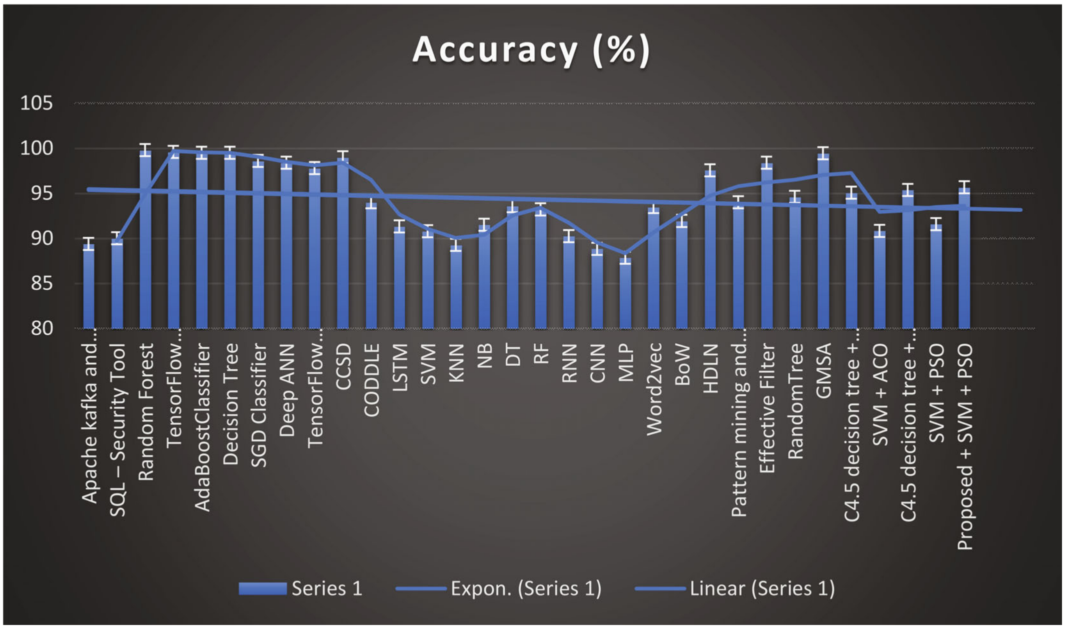

FIGURE 6 Accuracy comparison graph.

TABLE 3 Precision of methods to counter SQL Injection Attacks.

S.NO	YEAR	Methods	Precision	References
1	2020	Random Forest	99.90	42
2		TensorFlow BoostedTreeClassifier	98.90	39
3		AdaBoostClassifier	99.70	39
4		Decision Tree	99.80	39
5		SGDClassifier	98.80	39
6		Deep ANN	93.40	39
7		TensorFlow Linearclassifier	90.80	39
9	2019	CODDLE	99.00	39
10	2019	LSTM	91.32	60
11		SVM	90.77	60
12		KNN	90.30	60
13		NB	90.62	60
14		DT	92.02	60
15		RF	92.88	60
16		RNN	90.71	60
17		CNN	90.63	60
18		MLP	88.32	60
19		Word2vec	93.56	60
20		BoW	90.67	60

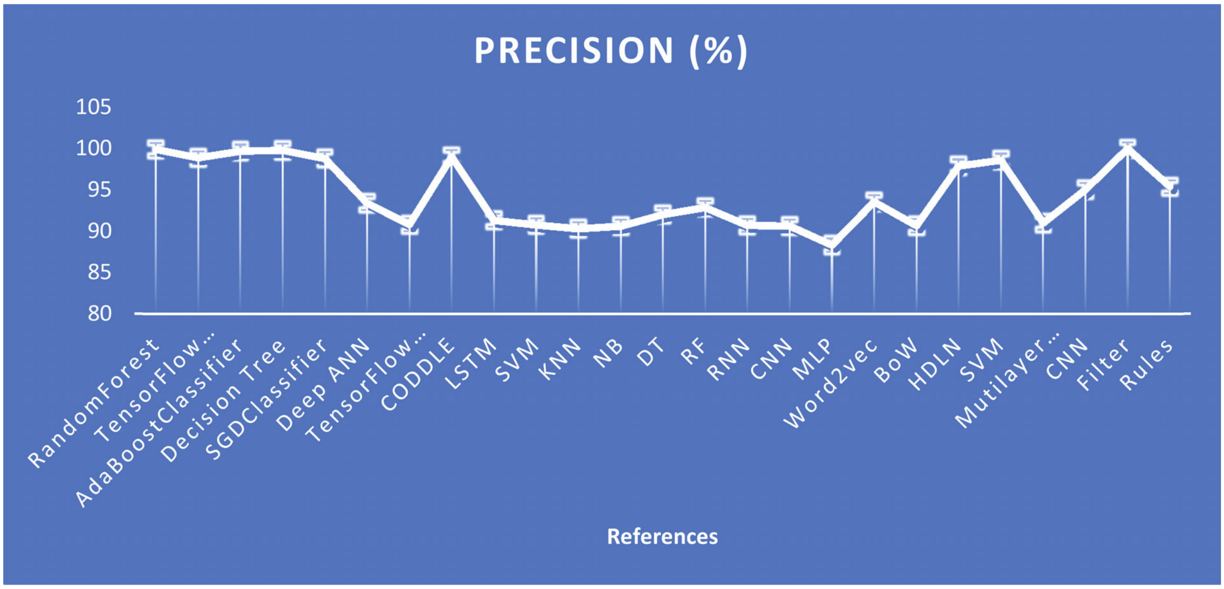

FIGURE 7 Precision comparison graph.

TABLE 4 Recall value of methods to counter SQL Injection Attacks.

S.NO	YEAR	Methods	Recall	References
1	2020	RandomForest	99.90	42
2		TensorFlow BoostedTreeClassifier	96.10	39
3		AdaBoostClassifier	99.60	39
4		Decision Tree	99.70	39
5		SGDClassifier	99.70	39
6		Deep ANN	82.00	39
7		TensorFlow Linearclassifier	75.90	39
9	2019	CODDLE	93.00	39
10	2019	LSTM	90.89	60
11		SVM	91.38	60
12		KNN	88.26	60
13		NB	91.83	60
14		DT	91.31	60
15		RF	94.43	60
16		RNN	89.40	60
17		CNN	84.01	60
18		MLP	85.78	60
19		Word2vec	92.43	60
20		BoW	92.91	60
21	2021	Effective Filter	98.40	50

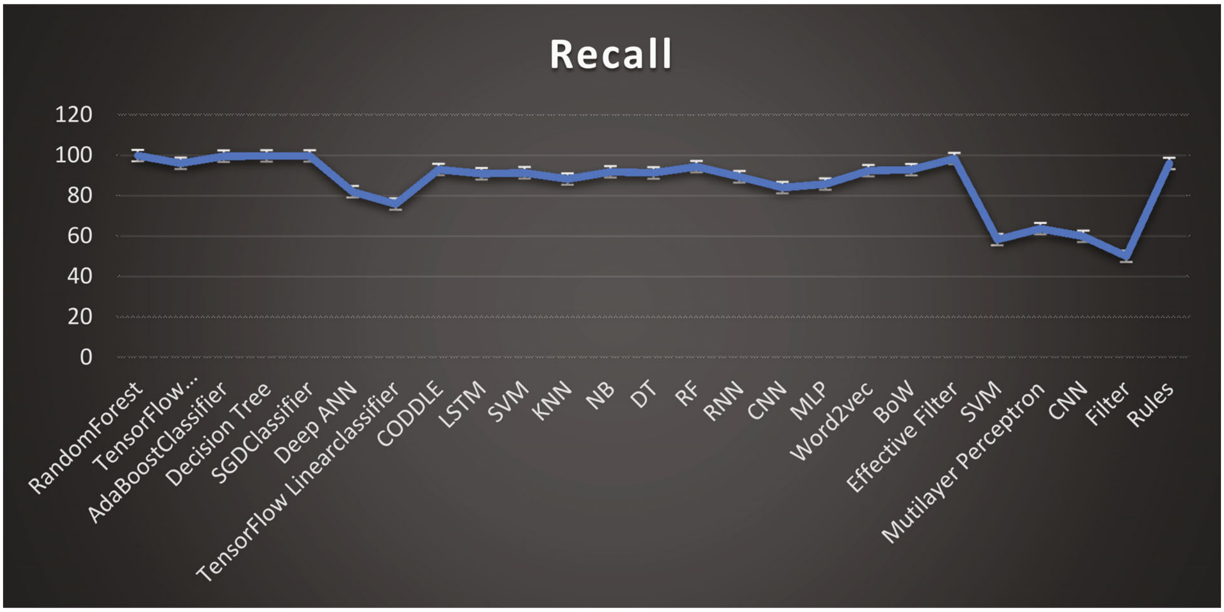

FIGURE 8 Recall value comparison graph.

TABLE 5 F1-Score of methods to counter SQL Injection Attacks.

S.NO	YEAR	Methods	F Measure	References
1	2020	Random Forest	99.90	42
2		TensorFlow Boosted Tree Classifier	99.80	39
3		AdaBoost Classifier	99.70	39
4		Decision Tree	99.70	39
5		SGD Classifier	99.20	39
6		Deep ANN	87.30	39
7		TensorFlow Linear classifier	98.80	39
8	2019	LSTM	91.10	60
9		SVM	91.07	60
10		KNN	89.27	60
11		NB	91.22	60
12		DT	91.66	60
13		RF	93.64	60
14		RNN	90.34	60
15		CNN	87.19	60
16		MLP	87.03	60
17		Word2vec	92.99	60
18		BoW	91.78	60
19	2021	Effective Filter	99.20	50

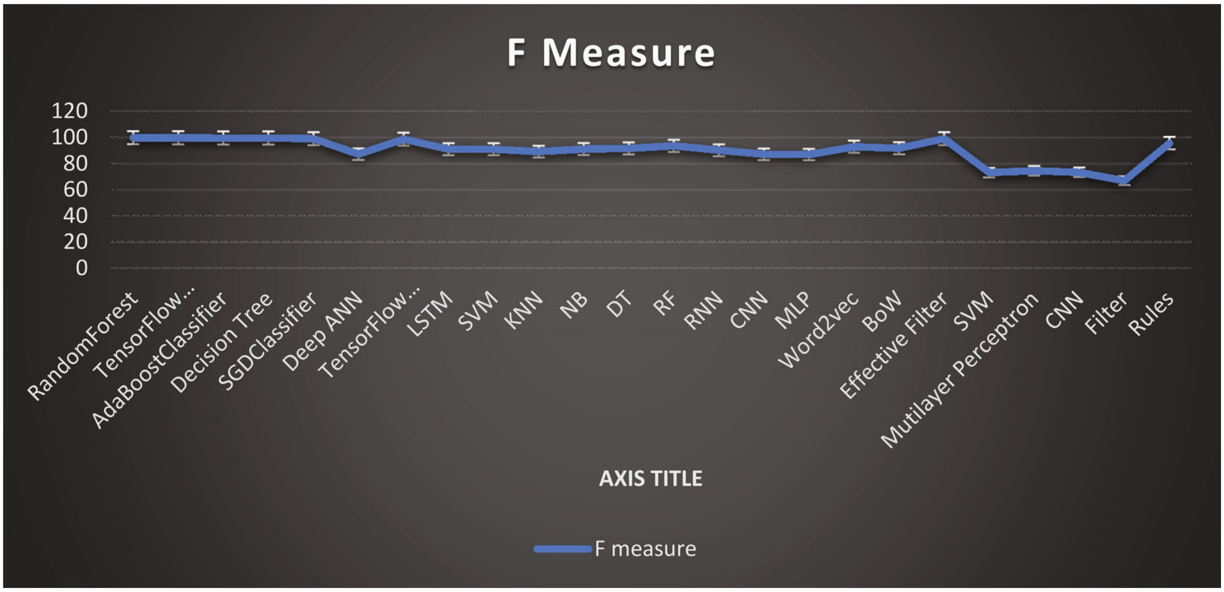

FIGURE 9 F-measure comparison graph.

We can see that Random Forest behaves best for detecting SQLIAs considering various efficiency parameters. The fact cannot be ruled out that the dataset considered for training and testing of the random forest model may be biased or small. So, a more rigorous analysis taking various new dataset can be performed using these models to predict, if the listed models behave the same in every scenario and for both the classifiers viz. binary and multi class.

7 | CONCLUSION AND FUTURE WORK

We reviewed existing literature on detecting and preventing of SQLIAs in a cloud computing environment and web applications. Among many existing techniques, ML-based techniques such as Random Forest, SVM, KNN, and so on, are popular SQLIAs detection mechanisms. The advent of Machine Learning has paved the way for many paths breaking research. Although security of devices and cloud is a challenging task, and attackers come out with new and powerful attack types to breach into a system of a server, timely detection of an attack scenarios in a standalone system or a cloud server can reduce the chances of heavy financial and personal losses to the concerned service provider or the end user. ML techniques such as various binary and multiclass classifiers such as random forest, SVM, CNN to name a few can be utilized by training them properly using various publicly available attack datasets to detect and prevent these attacks in a real time scenario. Researchers have paved the way be defining new datasets and new ML models to counter cloud attacks. The efficiency of these models is very good considering that fact they were on larger datasets.

New hybrid classifiers can be developed, including various new algorithms and trained on newly created datasets to detect SQLIAs. The work can be extended to analyze the effectiveness of the approaches on different data sets, and a new methodology can be proposed to counter SQL Injection attacks in a cloud computing environment.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Prashant Pranav https://orcid.org/0000-0002-3932-3048

REFERENCES

1. Velliangiri S, Karthikeyan P, Vinoth Kumar V. Detection of distributed denial of service attacks in cloud computing using the optimization-based deep networks. *J Exp Theor Artif Intell*. 2021;33(3):405-424. doi:10.1080/0952813X.2020.1744196
2. Shaikh AH, Meshram BB. Security issues in cloud computing. In: Balas VE, Semwal VB, Khandare A, Patil M, eds. *Intelligent Computing and Networking*. Lecture Notes in Networks and Systems. Vol 146. Springer; 2021. doi:10.1007/978-981-15-7421-4_6
3. Ahmad W, Rasool A, Javed AR, Baker T, Jalil Z. Cyber security in IoT-based cloud computing: a comprehensive survey. *Electronics*. 2022;11:16. doi:10.3390/electronics11010016
4. Albalawi A, Vassilakis V, Calinescu R. Side-channel attacks and countermeasures in cloud services and infrastructures. NOMS 2022-2022 IEEE/IFIP network operations and management symposium, Budapest, Hungary 1–4. 2022. doi:10.1109/NOMS54207.2022.9789783
5. Li X, Wang T, Zhang W, et al. An LSTM based cross-site scripting attack detection scheme for cloud computing environments. *J Cloud Comp*. 2023;12:118. doi:10.1186/s13677-023-00483-x
6. Aliero MS, Qureshi KN, Pasha MF, Ghani I, Yauri RA. Systematic review analysis on SQLIA detection and prevention approaches. *Wireless Pers Commun*. 2020;112:2297-2333. doi:10.1007/s11277-020-07151-2
7. AL-Maliki M, Jasim M. Review of SQL injection attacks: detection, to enhance the security of the website from client-side attacks. *Int J Nonlinear Anal Appl*. 2022;13(1):3773-3782. doi:10.22075/ijnaa.2022.6152
8. Faghihi F, Zulkernine M, Ding S. AIM: an android interpretable malware detector based on application class modeling. *J Inf Secur Appl*. 2023;75:103486. doi:10.1016/j.jisa.2023.103486
9. Rahman MA, Hossain MS. A deep learning assisted software defined security architecture for 6G wireless networks: IIoT perspective. *IEEE Wireless Commun*. 2022;29(2):52-59. doi:10.1109/MWC.006.2100438
10. Prateek K, Ojha NK, Altaf F, Maity S. Quantum secured 6G technology-based applications in internet of everything. *Telecommun Syst*. 2023;82:315-344. doi:10.1007/s11235-022-00979-y
11. Prateek K, Maity S, Amin R. An unconditionally secured privacy-preserving authentication scheme for smart metering infrastructure in smart grid. *IEEE Trans Netw Sci Eng*. 2022;10(2):1085-1095. doi:10.1109/TNSE.2022.3226902

12. Lu D, Fei J, Liu L. A semantic learning-based SQL injection attack detection technology. *Electronics*. 2023;12(6):1344. doi:10.3390/electronics12061344

13. Alotaibi FM, Vassilakis VG. Toward an SDN-based web application firewall: Defending against SQL injection attacks. *Fut Internet*. 2023;15(5):170. doi:10.3390/fi15050170

14. Patidar K, Tiwari DP, Sharma P, Pandagre KN, Shukla AK. An efficient approach for sending identification bit with the help of blowfish-RC6. In *handbook of Research on Advancements of Contactless Technology and Service Innovation in Library and Information Science*, pp. 268–286. IGI global. 2023. doi:10.4018/978-1-6684-7693-2.ch014

15. Liu J, He C. Online detection of SQL injection attacks based on ECA rules and dynamic taint analysis. *J Comput Appl*. 2023;43(5):1534. doi:10.11772/j.issn.1001-9081.2022040636

16. Alghawazi M, Alghazzawi D, Alarifi S. Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model. *Mathematics*. 2023;11(15):3286. doi:10.3390/math11153286

17. Saini N, Bhat Kasaragod V, Prakasha K, Das AK. A hybrid ensemble machine learning model for detecting APT attacks based on network behavior anomaly detection. *Concurr Comput Pract Exp*. 2023;35(28):e7865. doi:10.1002/cpe.7865

18. Sheth T, Anap J, Patel H, Singh N, Ramya RB. Detection of SQL injection attacks by giving apriori to Q-learning agents. In *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, 1–6. IEEE. 2023, May.

19. Okesola JO, Ogunbanwo AS, Owoade A, Olorunnisola EO, Okokpuji K. Securing web applications against SQL injection attacks-a parameterised query perspective. In *2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*, Vol. 1, 1–6. IEEE. 2023. doi:10.1109/SEB-SDG57117.2023.10124613

20. Labib RJ, Srivastava G, Lin JCW. Wireless and Mobile security in edge computing. *Security and Risk Analysis for Intelligent Edge Computing*. Springer International Publishing; 2023:193-207. doi:10.1007/978-3-031-28150-1_10

21. Marashdih AW, Zaaba ZF, Suwais K. An enhanced static taint analysis approach to detect input validation vulnerability. *J King Saud Univ Comput Inf Sci*. 2023;35(2):682-701. doi:10.1016/j.jksuci.2023.01.009

22. Mehta D, Suhagiya H, Gandhi H, Jha M, Kanani P, Kore A. SQLIML: a comprehensive analysis for SQL injection detection using multiple supervised and unsupervised learning schemes. *SN Comput Sci*. 2023;4(3):281. doi:10.1007/s42979-022-01626-8

23. Henry A, Gautam S, Khanna S, et al. Composition of hybrid deep learning model and feature optimization for intrusion detection system. *Sensors*. 2023;23(2):890. doi:10.3390/s23020890

24. Pružinec J, Quynh NA. Hakuin: optimizing blind SQL injection with probabilistic language models. In *2023 IEEE Security and Privacy Workshops (SPW)*, 384–393. IEEE. 2023. doi:10.1109/SPW59333.2023.00039

25. Irungu J, Graham S, Girma A, Kacem T. Artificial intelligence techniques for SQL injection attack detection. In *Proceedings of the 2023 8th international conference on intelligent information technology*, 38–45. 2023. doi:10.1145/3591569.3591576

26. Fratty R, Saar Y, Kumar R, Arnon S. Random routing algorithm for enhancing the cybersecurity of LEO satellite networks. *Electronics*. 2023;12(3):518. doi:10.3390/electronics12030518

27. Logozzo F, Mohamed I. How to make taint analysis precise. *Challenges of Software Verification*. Springer Nature Singapore; 2023:43-55. doi:10.1007/978-981-19-9601-6_3

28. Singh K, Kokardekar S, Khonde G, Dekate P, Badkas N, Lachure S. Cloud engineering-based on machine learning model for SQL injection attack. *In 2023 International Conference on Communication, Circuits, and Systems (IC3S)*. Vol 13. IEEE; 2023:1-6. doi:10.3390/app13074365

29. Crespo-Martínez IS, Campazas-Vega A, Guerrero-Higueras ÁM, Riego-DelCastillo V, Álvarez-Aparicio C, Fernández-Llamas C. SQL injection attack detection in network flow data. *Comput Secur*. 2023;127:103093. doi:10.1016/j.cose.2023.103093

30. Fu H, Guo C, Jiang C, Ping Y, Lv X. SDSIOT: an SQL injection attack detection and stage identification method based on outbound traffic. *Electronics*. 2023;12(11):2472. doi:10.3390/electronics12112472

31. Philip J, Rani DM, Rao PN. Application gateway with web application firewall. *AIP Conference Proceedings*. Vol 2492. AIP Publishing; 2023. doi:10.1063/5.0113195

32. Barsellotti L, De Marinis L, Cugini F, Paolucci F. FTG-net: hierarchical flow-to-traffic graph neural network for DDoS attack detection. *In 2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*. IEEE; 2023:173-178. doi:10.1109/HPSR57248.2023.10147929

33. Mallissery S, Chiang KY, Bau CA, Wu YS. Pervasive micro information flow tracking. *IEEE Trans Dependable Secure Comput*. 2023;20:4957-4975. doi:10.1109/TDSC.2023.3238547

34. Guan Y, He J, Li T, Zhao H, Ma B. SSQLi: a black-box adversarial attack method for SQL injection based on reinforcement learning. *Fut Internet*. 2023;15(4):133.

35. Nasrullayev N, Muminova S, Istamovich DK, Boltaeva M. Providing IoT security in industry 4.0 using web application firewall. in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 1788–1792. IEEE. 2023. doi:10.1109/ICESC57686.2023.10193640

36. Muhammad Z, Anwar Z, Javed AR, Saleem B, Abbas S, Gadekallu TR. Smartphone security and privacy: a survey on APTs, sensor-based attacks, Side-Channel attacks, Google play attacks, and defenses. *Dent Tech*. 2023;11(3):76. doi:10.3390/technologies11030076

37. Brintha NC, Jaswanth GS, Anusha M, Narayana JA, Venkat D. Securing banking credentials from SQL injection attacks using AES algorithm. In *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, 796–799. IEEE. 2023. doi:10.1109/ICEARS56392.2023.10085050

38. Wirz L, Tanthanathewin R, Ketphet A, Fugkeaw S. Design and development of a cloud-based IDS Using Apache Kafka and spark streaming. *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE), Bangkok, Thailand*, 1–6. 2022. doi:10.1109/JCSSE54890.2022.9836264

39. Tripathy D, Gohil R, Halabi T. *Detecting SQL injection attacks in cloud SaaS using machine learning. 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS).* Baltimore; 2020:145-150. doi:10.1109/BigDataSecurity-HPSC-IDS49724.2020.00035
40. Rongzhou L, Nanfeng X. WEB Protection Scheme Based on A Cloud Computing Platform. 2020 5th IEEE international conference on big data analytics (ICBDA), Xiamen, China, 323–328. 2020. doi:10.1109/ICBDA49040.2020.9101215
41. Shunmugapriya B, Paramasivan B. Protection against SQL injection attack in cloud computing. *IJERT*. 2020;9:502-510.
42. Singh N, Singh AK. SQL-injection vulnerabilities resolving using valid security tool in cloud. *Pertanika J Sci Technol*. 2019;27(1):159-174.
43. Ranjan I, Agnihotri RB. Ambiguity in cloud security with malware-injection attack. 2019 3rd international conference on electronics, communication and aerospace technology (ICECA), Coimbatore, India, 306–310. 2019. doi:10.1109/ICECA.2019.8821844
44. Bisht P, Pant D, Rauthan MS. Analyzing and Defending web application vulnerabilities through proposed security model in cloud computing. *J Graphic Era Univ*. 2018;6(2):183-196.
45. Ponmuthukalaiselvi P, Santha SK, Shunmugapriya MB. Defending against SQL injection attack in cloud computing.
46. Wu TY, Chen CM, Sun X, et al. A countermeasure to SQL injection attack for cloud environment. *Wireless Pers Commun*. 2017;96:5279-5293. doi:10.1007/s11277-016-3741-7
47. Yassin M, Ould-Slimane H, Talhi C, Boucheneb H. SQLIIDaaS: A SQL injection intrusion detection framework as a service for SaaS providers. 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 163–170. 2017. doi:10.1109/CSCloud.2017.27
48. Rajeh W, Abed A. A Novel Three-Tier SQLi Detection and Mitigation Scheme for Cloud Environments. 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), Palembang, Indonesia. 33–37. 2017. doi:10.1109/ICECOS.2017.8167160
49. Wang K, Hou Y. Detection method of SQL injection attack in cloud computing environment. IEEE advanced information management, communicates, electronic and automation control conference (IMCEC), Xi'an, China, 2016, 487–493. 2016. doi:10.1109/IMCEC.2016.7867260
50. Ibarra-Fiallos S, Higuera JB, Intriago-Pazmiño M, Higuera JRB, Montalvo JAS, Cubo J. Effective filter for common injection attacks in online web applications. *IEEE Access*. 2021;9:10378-10391. doi:10.1109/ACCESS.2021.3050566
51. Zheng J, Shen X. Pattern mining and detection of malicious SQL queries on anonymization mechanism. *IEEE Access*. 2021;9:15015-15027. doi:10.1109/ACCESS.2021.3052956
52. Gu H, Zhang J, Liu T, et al. DIAVA: a traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data. *IEEE Trans Reliab*. 2020;69(1):188-202. doi:10.1109/TR.2019.2925415
53. Li Q, Li W, Wang J, Cheng M. A SQL injection detection method based on adaptive deep forest. *IEEE Access*. 2019;7:145385-145394. doi:10.1109/ACCESS.2019.2944951
54. Abaimov S, Bianchi G. CODDLE: code-injection detection with deep learning. *IEEE Access*. 2019;7:128617-128627. doi:10.1109/ACCESS.2019.2939870
55. Appelt D, Nguyen CD, Panichella A, Briand LC. A machine-learning-driven evolutionary approach for testing web application firewalls. *IEEE Trans Reliabil*. 2018;67(3):733-757. doi:10.1109/TR.2018.2805763
56. Alnabulsi H, Islam R, Talukder M. GMSA: gathering multiple signatures approach to defend against code injection attacks. *IEEE Access*. 2018;6:77829-77840. doi:10.1109/ACCESS.2018.2884201
57. Liu M, Wang B. A web second-order vulnerabilities detection method. *IEEE Access*. 2018;6:70983-70988. doi:10.1109/ACCESS.2018.2881070
58. Mitropoulos D, Louridas P, Polychronakis M, Keromytis AD. Defending against web application attacks: approaches, challenges and implications. *IEEE Trans Dependable Secure Comput*. 2019;16(2):188-203. doi:10.1109/TDSC.2017.2665620
59. Bossi L, Bertino E, Hussain SR. A system for profiling and monitoring database access patterns by application programs for anomaly detection. *IEEE Trans Softw Eng*. 2017;43(5):415-431. doi:10.1109/TSE.2016.2598336
60. Li Q, Wang F, Wang J, Li W. LSTM-based SQL injection detection method for intelligent transportation system. *IEEE Trans Veh Technol*. 2019;68(5):4182-4191. doi:10.1109/TVT.2019.2893675
61. Priyaa BD, Devi MI. Fragmented query parse tree based SQL injection detection system for web applications. *In 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*. IEEE; 2016:1-5. doi:10.1109/ICCTIDE.2016.7725367

How to cite this article: Kumar A, Dutta S, Pranav P. Analysis of SQL injection attacks in the cloud and in WEB applications. *Security and Privacy*. 2024;7(3):e370. doi: 10.1002/spy2.370