A systematic review of detection and prevention techniques of SQL injection attacks

Article *in* Information Security Journal A Global Perspective · October 2021

DOI: 10.1080/19393555.2021.1995537

4 authors:

Mohammed Nasereddin
Institute of Theoretical and Applied Informatics, Polish Academy of Sciences
3 PUBLICATIONS 8 CITATIONS
SEE PROFILE

Ashaar Alkhamaiseh
Princess Sumaya University for Technology
3 PUBLICATIONS 11 CITATIONS
SEE PROFILE

Malik Qasaimeh
Jordan University of Science and Technology
43 PUBLICATIONS 208 CITATIONS
SEE PROFILE

Raad Al-Qassas
Princess Sumaya University for Technology
24 PUBLICATIONS 161 CITATIONS
SEE PROFILE

Information Security Journal: A Global Perspective

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiss20

A systematic review of detection and prevention techniques of SQL injection attacks

Mohammed Nasereddin, Ashaar ALKhamaiseh, Malik Qasaimeh & Raad Al-Qassas

To cite this article: Mohammed Nasereddin, Ashaar ALKhamaiseh, Malik Qasaimeh & Raad Al-Qassas (2021): A systematic review of detection and prevention techniques of SQL injection attacks, Information Security Journal: A Global Perspective, DOI: 10.1080/19393555.2021.1995537

To link to this article: https://doi.org/10.1080/19393555.2021.1995537

Published online: 27 Oct 2021.

INFORMATION SECURITY JOURNAL: A GLOBAL PERSPECTIVE
https://doi.org/10.1080/19393555.2021.1995537

A systematic review of detection and prevention techniques of SQL injection attacks

Mohammed Nasereddin[a], Ashaar ALKhamaiseh[a], Malik Qasaimeh[b], and Raad Al-Qassas[a]

[a]Department of Computer Science, Princess Sumaya University for Technology, Amman, Jordan; [b]Department of Software Engineering, Jordan University of Science and Technology, Irbid, Jordan

ABSTRACT
SQL injection is a type of database-targeted attack for data-driven applications. It is performed by inserting malicious code in the SQL query to alter and modify its meaning, enabling the attacker to retrieve sensitive data or to access the database. Many techniques have been improved and proposed to detect and mitigate these types of attacks. This paper provides a systematic review for a pool of 60 papers on web applications' SQL injection detection methods. The pool was selected using a developed searching and filtering methodology for the existing literature based on scholar databases (IEEE, ScienceDirect, and Springer) with the aim to provide specific answering for several research questions in the area of SQL injection detection. This provides a basis for the design and use of effective SQL injection detection methods.

KEYWORDS
Web-application attacks; database attacks; SQL injection; detection; mitigation; prevention

1. Introduction

Large amounts of sensitive data of companies and organizations are stored in special databases inside the organization's servers or somewhere abroad. Protecting these data from unauthorized access is extremely important to organizations in various sectors since any leak in these data will have a huge impact on the privacy of the users as well as the reputation and financial situation of the institution. Consequently, data comprise one of the most important assets that must be preserved and protected; thus, databases have become important treasuries and pillars of modern organizations.

Structured Query Language (SQL) is the standard language used to access and manipulate these databases, as it allows the database administrator to perform multiple operations on data such as storing, retrieving, creating, updating, or deleting, etc. Data is stored in databases in tables that have relations between each other. This approach is called relational database (RDB), where the relationship between data points is clearly defined, and the relationship between tables and field types is also clarified under the name of the schema, which facilitates searching between these relationships (Győrödi et al., 2015). The most popular examples of RDB are Oracle, MySQL, and Microsoft SQL Server.

On the other hand, new database structures called non-relational databases (NoSQL databases) have been developed. Its most popular examples are MongoDB, Redis, and Apache. This type of structure was launched to fix the vulnerabilities affecting traditional RDB especially in organizations that rely on big data and long-range geographical distribution (e.g., electronic commerce and social networks) (Matallah et al., 2021). The data in these structures is stored through a new model that is optimized for the type of data it stores so that query languages other than SQL can be used. Data stored in these databases must be protected, maintained safely, and accessible only by authorized users.

SQL databases are mostly used in web applications to store data for sites, users and are handled through queries within specific commands. Visitors' activities are translated into SQL commands to modify and update their data so that they don't interact directly with these databases. Besides, attackers have exploited this advantage and targeted databases to get access to data. One of the most common attacks on databases is SQL injection (Lee et al., 2012). The

CONTACT Mohammed Nasereddin mohamedsgn8020@gmail.com Department of Computer Science, Princess Sumaya University for Technology, Amman, Jordan

first appearance of SQL injection vulnerabilities dates to 1998 when Jeff Forristal was writing about how to infiltrate Windows NT server. This was the beginning of the discovery of this kind of threat.

The massive development in web application attacks has called for intensified efforts by specialists to confront them and mitigate their effects, and they have devoted a great deal to protecting user privacy and securing transmitted data that is linked between various parties (clients and servers) over the Internet. Statistics have shown that the most prevalent web applications attacks in recent years are SQL Injection and Cross-Site Scripting (XSS) of which SQL Injection had the largest share. Some studies estimated that they account for 27% of all attacks (Positive Technologies, 2019). Furthermore, an analytical study during the period Nov/2017 – Mar/2019 indicated that they account for 65% (Bonner, 2019) (B. B. Gupta et al., 2015).

XSS is a type of web application injection attack, where the end-user is exploited by injecting malicious scripts into trusted web applications that are sent to the user as a browser-side script. These attacks are common in web applications in which user inputs are frequently taken and not validated. The user's browser cannot determine that the script is malicious because it is coming from a trusted web application. These attacks can have catastrophic effects on users' information such as stealing passwords, credit cards information, session identifications embedded in cookies, and other sensitive user B. B. Gupta et al. (2015), (2018)).

Web applications use backend databases to store, transmit, and retrieve the data, which is a magnet for attackers as it may contain sensitive data. Attackers use different methods and techniques that exploit vulnerabilities in databases programming. SQL injection can be considered as a database intrusion attack (Schults, 2020), based on intruding into the database by inserting SQL characters that alert the SQL statements, resulting in a change in the logic of the SQL query (Schults, 2020).

This paper systematically reviews many research papers that proposed techniques, algorithms, and frameworks for detect and mitigate the different types of SQL injection attacks on web applications, by selecting a specific issue, defining the scope of the review, selecting scholar databases (IEEE, ScienceDirect, and Springer) to search for the related research papers. Moreover, it classified the proposed approaches, and analyze their performance. The remainder of this paper is as follows: Section 2 presents an overview of SQL injection. Section 3 presents the research methods used in this paper. Section 4 shows the results and analysis of the reviewed papers. Finally, section 5 sums up the conclusion.

2. SQL injection overview

SQL injection is a technique that is executed by attackers to target the databases, exploiting vulnerabilities in web applications. This vulnerability results from weaknesses in filtering variables (Sadeghian et al., 2013a), which allow the attacker to access, retrieve, modify, or delete user data using illegitimate methods.

2.1. Types of SQL injection attacks

SQL injection attacks can be classified into different categories based on multiple factors, but the most comprehensive and general factors are the intent of the attacker and the injection mechanism (Halfond et al., 2006). The intent here means the goal that the attacker aims to achieve from the attack, such as modifying data and changing information, or learning the schema of a database. The mechanism of the injection is determined according to the classification of vulnerability in the web application and the path the attacker takes. The most common types are listed below (Lee et al., 2012) (Halfond et al., 2006) (Sadeghian et al., 2013b).

2.1.1. Order of injection

Depending on the order of the injection, attacks can be divided into two main types: 1st, and 2nd order. 1st order injection method is the classical method of attack, while 2nd order injection differs in that a malicious user can insert a query fragment into a request (which is not inherently susceptible to injection), and then execute the inserted SQL in a second query vulnerable to SQL injection. Some types of 1st order SQL injection attack include:

Tautologies: This is the simplest type, which aims to bypass authentication by injecting a conditional statement using "OR" operator into the intact SQL query so that this condition always gives a true value.

Piggy-Backed Query: No modification is needed for the original SQL query, and this attack adds a malicious SQL query to the original one. At execution, both queries are executed, thereby launching the attack.

Stored Procedures: A set of procedures that are already stored within the database can be exploited as a vulnerability when the attack is executed by the attacker, thereby gaining control over the database.

2.1.2. Server response

This type of attack depends on how the server responds when the change occurs, and what will appear. The two most important species included in this classification are described below.

Illegal/incorrect queries injection: This attack allows the attacker to get information about the schema of the database of the web application, and thus leads to other types of attacks. It relies on the response of the database when it replies with an error message. This type of message reveals the vulnerabilities in the database to the attackers.

Time-based blind injection: It sends a set of true or false questions to the database and then determines the result according to the application response. It is almost identical to classical SQL injection, but differs in how data is retrieved from the database. It is considered more difficult because the database does not output the data to the webpage.

Another attack classification can be used depending on the Data Extraction Channel. It is divided into two main types: In-Band SQL injection, and Out-of-Band SQL injection. The first type is when the attacker can carry out the attack and collect the results through the same communication channel, and it is the most common of the two types. One of the popular methods of this type is error-based SQL injection. In the second type, the attacker cannot use the same channel for the previous two operations.

All these types of SQL injection attacks and more are being recognized by developers and system administrators. Therefore, many filters and different methods have been developed to prevent or at least mitigate the impact of these attacks.

2.2. Types of SQL injection detection methods

The researchers relied on several approaches in filter user inputs to identify and detect SQL injection attacks. Such methods are (Lee et al., 2012):

Static analysis: this approach identifies syntactic and grammatical errors in user input. It only focuses on validating user input to detect injection attacks. However, it cannot detect malicious input with the correct syntax (Lee et al., 2012).

Instruction-set Randomization: this can be considered as a branch of static analysis. It inserts random values into the SQL query statement of a web application and checks for volatility to detect SQL injection attacks.

Dynamic analysis: this approach scans the response from the web application by sending each input to the target and then receiving the response.

Combined analysis: a hybrid static and dynamic method that uses the benefits of both techniques to detect SQL injection attacks.

Machine learning: used to generate SQL queries in web applications, which are learned to generate the detection model parameters. This model is then compared with dynamic SQL queries to search for any inconsistencies (Lee et al., 2012).

3. Research methodology

This systematic review includes papers on detection methods and techniques of SQL injection attack published during the period 2009 to 2021, using the following searching engines: IEEE, ScienceDirect (Elsevier), and Springer. The filtering process and the number of resultant papers after each level are shown in Figure 1.

3.1. Research questions

Research papers included in this systematic review were selected to answer the following research questions.

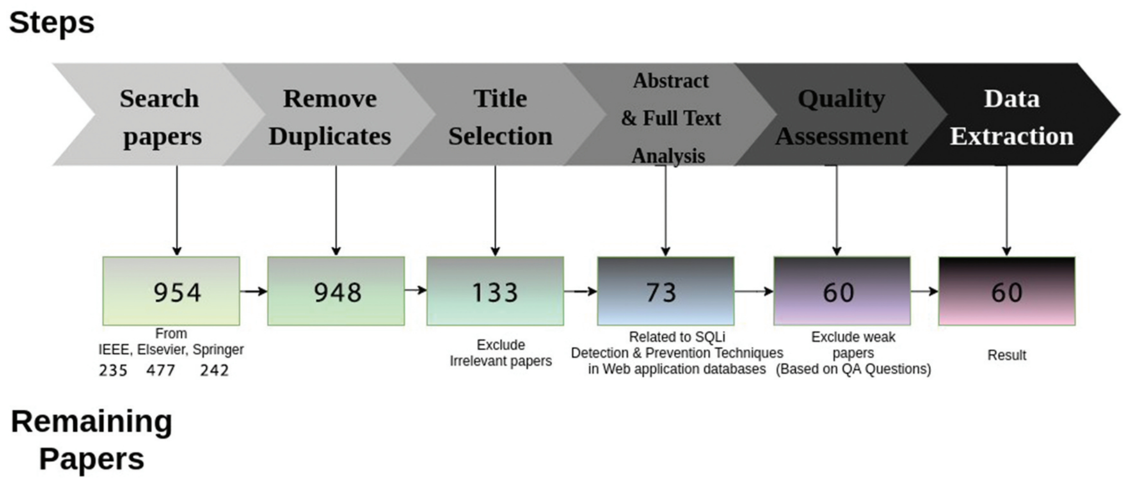
Steps
Search papers
Remove Duplicates
Title Selection
Abstract & Full Text Analysis
Quality Assessment
Data Extraction
954
948
133
73
60
60
From IEEE, Elsevier, Springer
235 477 242
Exclude Irrelevant papers
Related to SQLi Detection & Prevention Techniques in Web application databases
Exclude weak papers (Based on QA Questions)
Result
Remaining Papers

Figure 1. The filtering process and the number of resultant papers after each level.

3.1.1. RQ1: How has publication grown in the SQL injection detection field?

The purpose of this question is to examine the views of the reviewed research papers on the significance of the SQL injection attacks, regarding how these attacks are being developed and the level of severity they have reached during recent years.

3.1.2. RQ2: What types of SQL injection attacks have been discussed?

This question targets to specify the types of SQL injection attacks that can be solved by each proposed method, algorithm, or framework.

3.1.3. RQ3: What are the proposed methods to detect and mitigate these attacks?

This research question aims to explore the most common and efficient methods that have been proposed to face each type of attack.

Table 1. Searching keywords.

Keywords	Alternative keywords	Keyword combinations
SQL Injection Detection	Database -	SQL AND injection AND (detection techniques OR prevention) (SK1)SQL AND injection AND (detection techniques OR mitigation) (SK2)Database AND Injection AND (detection techniques OR prevention) (SK3)Database AND injection AND (detection techniques OR mitigation) (SK4)
techniques	Prevention, Mitigation	

3.2. Searching process

3.2.1. Searching papers

The search for the related research papers was performed on the IEEE Explore, ScienceDirect, and Springer database libraries, covering the publication period from 2009 to 2021, to include the most up-to -date and relevant publications. The keywords used in the searching process are listed in Table 1.

3.2.2. Selection execution

A large number of research papers have been collected from each search engine through the used search method. All the extracted papers were filtered during multiple levels: creating a CSV file to remove duplicate papers, selection based on the title, selection based on abstract, and selection based on quality assessment questions and full-text analysis. These levels are explained as follows:

Level 1. Removing Duplications: All papers resulting from the searching on the three engines totaled 954 papers, using the strings mentioned in Table 1, including 235 papers from IEEE, 477 papers from ScienceDirect, and 242 papers from Springer. To remove any duplications among these papers, a CSV file was used to filter the duplications, removing 6 duplicate papers.

Level 2. Title Selection: In this level, papers from level 1 were filtered based on evaluating their titles and determining how relevant are the titles to the research questions. The first author nominated papers extracted from the IEEE engine and

Table 2. Quality assessment criteria.

Q.N.	Question
Q1	Does the title of the paper clarify the idea of the research?
Q2	Does the abstract include the proposed approach?/Does the abstract explain the objectives of the research?
Q3	Does the writer follow the systematic/standard arrangement of the research paper?
Q4	Does the paper use a mechanism, technique, or methodology?
Q5	Does the paper specify the type of the SQL injection attack?
Q6	Does the proposed approach succeed in detecting/mitigating SQL injection attacks?
Q7	Are the results/evaluations shown conclusively in the paper?

the second author nominated those extracted from ScienceDirect and Springer. This resulted in 133 papers.

Level 3. Abstract & Full-Text Analysis Selection: The 133 papers were then filtered again by reading their abstracts and how relevant they are to the research questions. Surveys about SQL injection attacks (Kumar & Pateriya, 2012), papers that just introduced evaluation or comparison between already existing techniques (Tajpour et al., 2010) (Tajpour & zade Shooshtari, 2010), or those that didn't propose any specific mitigation techniques (Wu, 2010) in their abstracts were removed. This resulted in 73 research papers.

Level 4. Quality Assessment Questions: Finally, each of the 73 papers was assessed based on the quality assessment questions (described below). Papers with a score lower than 3.5 were dropped, so 13 papers were removed.

3.2.3. Quality assessment and extracting information

Quality assessment targets to evaluate the papers to determine their level of quality. Each paper must pass through 7 assessment questions with one point for each question. Based on the cumulative score, the quality level of the paper is determined. Table 2 shows the quality assessment criteria developed for this purpose.

The papers that scored 3.5 points and more are listed with their details in Table 3.

4. Results and analysis

4.1. Publication growth in SQL injection detection field (RQ1)

The answer to this question is related to exploring the motivation behind each approach and how security concerns have become important in developing secure web applications through recent years. This is indicated by the number of publications in the field of SQL injection and web application attacks detection during the period 2009–2021. We found that the number of articles that discuss the topic of SQL injection attacks increased during the first three years from 2009 to 2011, as shown in Figure 2, while the big jump was in the year 2016. The amount of publication growth in this field is shown in Figure 2.

4.2. Types of SQL injection attacks (RQ2)

Answering this question is based on classifying the type of attack that each research paper discussed and was supposed to solve. This classification is presented in Table 4. Types of SQL injection attacks discussed in the reviewed papers can be classified into the following categories: Order of injection, Server response, Combined, In-band and Out-band, and General.

As shown in Figure 3. The distribution of these categories discussed by the reviewed papers showed the results as follows:

General: 50.00% of the reviewed papers either proposed techniques to confront SQL injection attacks in general or didn't specify the type of SQL injection attack that they were supposed to mitigate, so we categorized them in the General category. For example, approaches proposed in papers P8, P9, P10, P13, P14, and P55 discussed SQL injection attacks without customization.

Order of injection: this includes 1st order attacks (such as tautology or alternate encoding), or 2nd order attacks. For example, P1 proposed a method to detect 2nd order attacks by building new sets of SQL instruction based on randomizing the SQL keywords included in the web application. P6 proposed a new technique that uses regular expressions and finite automata to detect tautology attacks. this approach has the advantage of considering languages other than English. This category represents 7.00% of the total attacks mentioned in the reviewed papers.

Server response: this category contains server response attacks that also account for 3.00%. All these categories and their rates are shown in Figure 3. P51 dealt with all types of attacks, including blind-based ones.

Table 3. Result of quality assessment.

Paper No.	SRC	Q1	Q2	Q3	Q4	Q5	Q6	Q7	100%
P1 = (Ping, 2017)	IEEE	1	1	1	1	1	1	1	100%
P2 = (Katole et al., 2018)	IEEE	1	1	1	1	1	1	1	100%
P3 = (Appiah et al., 2017)	IEEE	1	1	1	0.5	1	1	1	92.9%
P4 = (Karuparthi & Zhou, 2016)	IEEE	1	1	1	1	1	1	0	85.7%
P5 = (Pinzón et al., 2010)	IEEE	1	1	1	1	1	1	1	100%
P6 = (Qbea'h et al., 2016)	IEEE	1	1	1	1	1	1	1	100%
P7 = (Ghafarian, 2017)	IEEE	1	1	1	1	1	1	0.5	92.8%
P8 = (Ping et al., 2016)	IEEE	1	1	0.5	1	0.5	1	1	85.7%
P9 = (Prabakar et al., 2013)	IEEE	1	1	1	1	0.5	1	0.5	85.7%
P10 = (Oosawa & Matsuda, 2014)	IEEE	1	1	1	1	0.5	1	1	92.9%
P11 = (Shanmughaneethi et al., 2009)	IEEE	1	1	0.5	1	1	1	0	78.6%
P12 = (Li et al., 2019)	IEEE	1	1	1	1	1	1	1	100%
P13 = (Chenyu & Fan, 2016)	IEEE	1	1	1	1	1	1	1	100%
P14 = (Matsuda et al., 2011)	IEEE	1	0.5	1	1	1	1	1	92.9%
P15 = (Buja et al., 2014)	IEEE	1	1	1	1	0.5	0.5	1	85.7%
P16 = (Srivastava, 2014)	IEEE	1	1	1	1	1	1	0.5	92.9%
P17 = (Xie et al., 2019)	IEEE	1	1	1	1	0.5	1	1	92.9%
P18 = (Jiao et al., 2012)	IEEE	1	1	1	1	1	1	0	85.7%
P19 = (Xiao et al., 2017)	IEEE	1	1	1	1	1	1	1	100%
P20 = (Pomeroy & Tan, 2011)	IEEE	0.5	0.5	0.5	0.5	0.5	1	0.5	57.1%
P21 = (He et al., 2015)	IEEE	1	0.5	1	1	1	0.5	0	71.4%
P22 = (Hanmanthu et al., 2015)	IEEE	1	1	1	1	1	1	1	100%
P23 = (Umar et al., 2018)	IEEE	1	1	1	1	0.5	1	0	78.6%
P24 = (Upadhyay & Khilari, 2016)	IEEE	1	1	1	1	1	1	1	100%
P25 = (Xue & He, 2011)	IEEE	1	1	1	1	1	1	0	85.7%
P26 = (Uwagbole et al., 2017)	IEEE	1	1	1	1	1	1	1	100%
P27 = (Wu & Chan, 2012)	IEEE	1	1	1	1	0.5	0.5	1	85.7%
P53 = (Gu et al., 2019)	IEEE	1	1	1	1	1	1	1	100%
P54 = (Tripathy et al., 2020)	IEEE	1	1	1	1	0.5	1	1	92.9%
P55 = (Kuroki et al., 2020)	IEEE	1	1	0.5	1	1	1	1	92.9%
P56 = (Hlaing & Khaing, 2020)	IEEE	1	1	1	1	1	1	1	100%
P57 = (Jana & Maity, 2020)	IEEE	1	1	1	1	0.5	1	1	92.9%
P28 = (Balasundaram & Ramaraj, 2012)	ScienceDirect	1	0.5	0	1	0.5	1	1	71.4%
P29 = (Kar et al., 2016)	ScienceDirect	1	1	1	1	1	1	1	100%
P30 = (Lee et al., 2012)	ScienceDirect	1	1	1	1	1	1	1	100%
P31 = (Natarajan & Subramani, 2012)	ScienceDirect	1	1	0	1	1	1	1	85.7%
P32 = (Kim & Lee, 2014)	ScienceDirect	1	1	1	1	1	1	1	100%
P33 = (Pinzon et al., 2013)	ScienceDirect	1	1	1	1	1	1	1	100%
P34 = (Jang & Choi, 2014)	ScienceDirect	1	1	1	1	1	1	1	100%
P35 = (Mitropoulos & Spinellis, 2009)	ScienceDirect	1	1	1	1	1	1	1	100%
P36 = (McWhirter et al., 2018)	ScienceDirect	1	1	1	1	0.5	1	1	92.9%
P37 = (Patel & Shekokar, 2015)	ScienceDirect	1	1	1	1	1	1	1	100%
P58 = (Tang et al., 2020)	ScienceDirect	1	1	1	1	0	1	1	85.7%
P38 = (Narayanan et al., 2011)	Springer	1	1	0	1	1	1	1	85.7%
P39 = (Doshi et al., 2014)	Springer	1	0.5	1	0.5	0.5	0.5	0.5	64.3%
P40 = (Singh et al., 2015)	Springer	1	0.5	1	1	1	1	1	92.9%
P41 = (Selvamani & Kannan, 2011)	Springer	1	1	1	1	0.5	1	1	92.9%
P42 = (Choraś et al., 2013)	Springer	1	0.5	1	1	1	1	1	92.9%
P43 = (Kar et al., 2015)	Springer	1	1	1	1	1	1	1	100%
P44 = (Hidhaya & Geetha, 2012)	Springer	1	1	1	1	0.5	1	1	92.9%
P45 = (Raj & Sherly, 2017)	Springer	1	0.5	0.5	1	1	1	1	85.7%
P46 = (Maheswari & Anita, 2016)	Springer	1	1	1	1	1	1	1	100%
P47 = (Huang et al., 2017)	Springer	1	0.5	1	1	0.5	0.5	1	78.6%
P48 = (Sadalkar et al., 2011)	Springer	1	1	0.5	1	1	1	1	92.9%
P49 = (Feng et al., 2019)	Springer	1	1	0.5	0.5	0.5	0.5	1	71.4%
P50 = (Perkins et al., 2016)	Springer	1	0.5	1	1	0.5	1	1	85.7%
P51 = (Joseph & Jevitha, 2015)	Springer	1	1	1	1	1	1	1	100%
P52 = (Pinzón et al., 2009)	Springer	1	1	1	1	0.5	1	1	92.9%
P59 = (Aliero et al., 2020)	Springer	1	1	1	1	1	1	1	100%
P60 = (Abikoye et al., 2020)	Springer	1	1	1	1	1	1	1	100%

Combined: the fourth category includes approaches that mitigate attacks of both error-based and any type of the 1st order. For instance, P2 proposed a method of removing parameters from the original query and then comparing it with the modified one. P3's technique uses a fingerprint combined with pattern matching to detect attacks of the 1st order during the programming and query phases. The Combined category represents 38.00% of all attacks discussed.

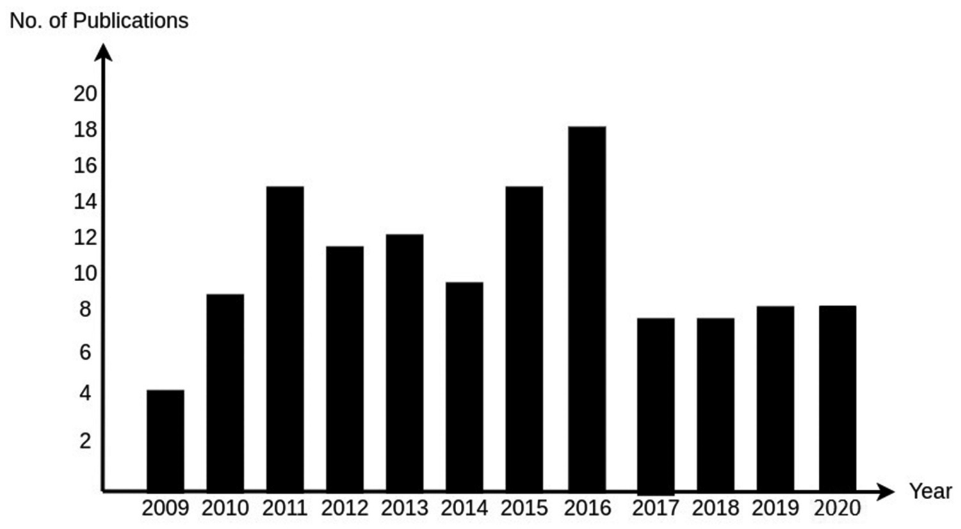

Figure 2. The growth of publication in SQL injection Field.

In-band/out of band: papers that discuss in-band and out-band attacks are classified in a single category and represent 2.00% of total reviewed papers. Only one paper in this category, P44, tried to mitigate data extraction channel attacks such as in-band and out-band attacks.

4.3. SQL injection detection techniques (RQ3)

The reviewed papers have followed different detection techniques that can be classified into four categories: Static, Dynamic, Combined, and Machine learning. Table 5 lists these techniques.

4.3.1. Static technique

We found that 44.00% of the proposed techniques are of the static type, as detection of SQL injection attacks is done during the development phase of the web application. These approaches are mentioned in P2, P3, P6, P8, P14, P15, P16, P18, P20, P56, and others. P18 proposed a secure approach described as a defensive level of SQL injection middleware that inserted between the authentication system and the database to detect any SQL injection attack.

4.3.2. Dynamic technique

This technique represents 13.00% of all the proposed techniques. It relies on detecting any injections during the run-time before being sent to the database server. For example, this technique was followed by P39 to develop a dynamic network filter using java programming language to detect SQL injection attacks.

4.3.3. Combined technique

13.00% of approaches belong to the category of using both static and dynamic methods. P48 used a hybrid model to detect SQL injection attacks in PHP environments.

4.3.4. Machine learning technique

Many approaches have used machine learning as a detection technique. Such approaches are proposed in P12 and P54, which detect SQL injection attacks using an adaptive deep forest-based ADP method. This improves the structure of the deep forest and integrates it with the AdaBoost algorithm. Approaches using machine learning represent about 30.00% of all proposed techniques. These categories and their distributions are shown in Figure 4 and Table 5.

5. Conclusion

This systematic review analyzed a total of 60 research papers, finding that most approaches deployed “the static technique” (44.00%) for detecting SQL injection attacks. Most papers didn’t specify the type of SQL injection attacks they dealt with (50.00%). Attacks of both 1st order and error-based types were the second

Table 4. SQL injection attacks classification.

Paper No.	Attack Classification
P1 = (Ping, 2017)	2nd order (order of injection)
P2 = (Katole et al., 2018)	1st order -error based (combined)
P3 = (Appiah et al., 2017)	Order of injection-error based (combined)
P4 = (Karuparthi & Zhou, 2016)	1st order & error based (combined)
P5 = (Pinzón et al., 2010)	General
P6 = (Qbea'h et al., 2016)	Tautology & alternate encoding (order of injection)
P7 = (Ghafarian, 2017)	1st order & error based (combined)
P8 = (Ping et al., 2016)	General
P9 = (Prabakar et al., 2013)	General
P10 = (Oosawa & Matsuda, 2014)	General
P11 = (Shanmughaneethi et al., 2009)	1st order & error based (combined)
P12 = (Li et al., 2019)	1st order & server response (combined)
P13 = (Chenyu & Fan, 2016)	General
P14 = (Matsuda et al., 2011)	General
P15 = (Buja et al., 2014)	General
P16 = (Srivastava, 2014)	1st order & server response (combined)
P17 = (Xie et al., 2019)	General
P18 = (Jiao et al., 2012)	General
P19 = (Xiao et al., 2017)	Order of injection & server response (combined)
P20 = (Pomeroy & Tan, 2011)	General
P21 = (He et al., 2015)	General
P22 = (Hanmanthu et al., 2015)	1st order & server response (combined)
P23 = (Umar et al., 2018)	General
P24 = (Upadhyay & Khilari, 2016)	1st order & server response (combined)
P25 = (Xue & He, 2011)	Tautology, encoding & error based (combined)
P26 = (Uwagbole et al., 2017)	General
P27 = (Wu & Chan, 2012)	General
P53 = (Gu et al., 2019)	Server response
P54 = (Tripathy et al., 2020)	Server response & out of band (combined)
P55 = (Kuroki et al., 2020)	General
P56 = (Hlaing & Khaing, 2020)	1st order (order of injection)
P57 = (Jana & Maity, 2020)	General
P28 = (Balasundaram & Ramaraj, 2012)	Tautology, stored procedure & error based (combined)
P29 = (Kar et al., 2016)	General
P30 = (Lee et al., 2012)	1st order & error based (combined)
P31 = (Natarajan & Subramani, 2012)	General
P32 = (Kim & Lee, 2014)	1st order & error based (combined)
P33 = (Pinzon et al., 2013)	1st order & error based (combined)
P34 = (Jang & Choi, 2014)	General
P35 = (Mitropoulos & Spinellis, 2009)	General
P36 = (McWhirter et al., 2018)	General
P37 = (Patel & Shekokar, 2015)	General
P58 = (Tang et al., 2020)	General
P38 = (Narayanan et al., 2011)	1st order (order of injection)
P39 = (Doshi et al., 2014)	General
P40 = (Singh et al., 2015)	1st order & error based (combined)
P41 = (Selvamani & Kannan, 2011)	General
P42 = (Choraś et al., 2013)	1st order & server response (combined)
P43 = (Kar et al., 2015)	General
P44 = (Hidhaya & Geetha, 2012)	In-band & out of band
P45 = (Raj & Sherly, 2017)	1st order & server response (combined)
P46 = (Maheswari & Anita, 2016)	1st order & error based (combined)
P47 = (Huang et al., 2017)	General
P48 = (Sadalkar et al., 2011)	1st order & error based (combined)
P49 = (Feng et al., 2019)	General
P50 = (Perkins et al., 2016)	General
P51 = (Joseph & Jevitha, 2015)	Blind based (server response)
P52 = (Pinzón et al., 2009)	General
P59 = (Aliero et al., 2020)	Order of injection & server response (combined)
P60 = (Abikoye et al., 2020)	1st order & server response (combined)

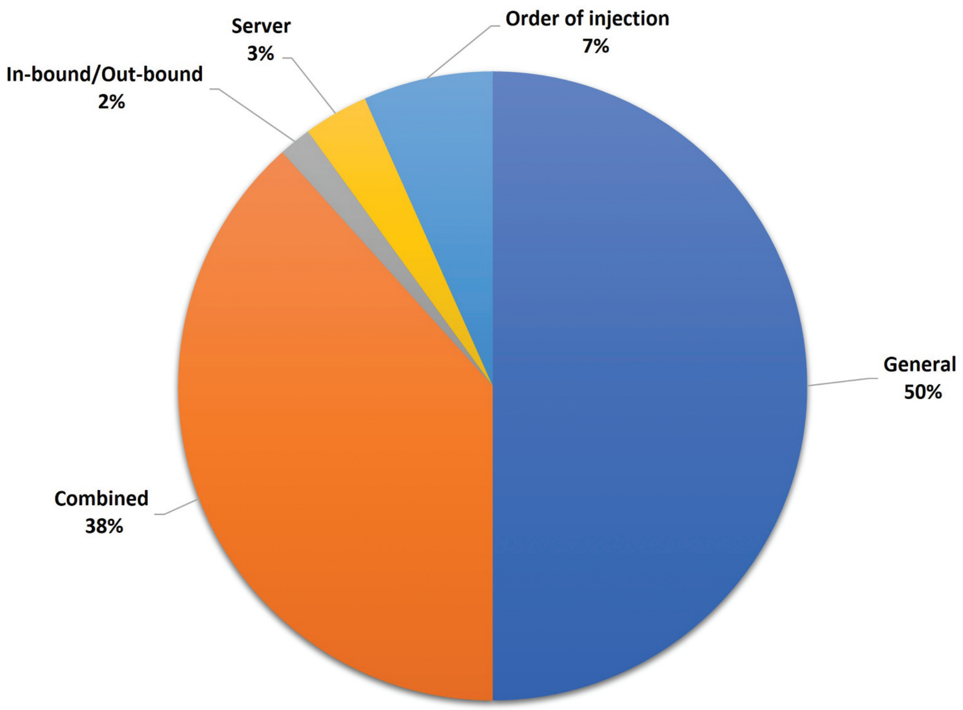

Figure 3. Distribution of types of SQL injection attacks.

Table 5. SQL injection detection techniques classification.

Paper No.	Detection Technique
P1 = (Ping, 2017)	Dynamic
P2 = (Katole et al., 2018)	Static "Parameter Filtering"
P3 = (Appiah et al., 2017)	Static "Parameter Filtering"
P4 = (Karuparthi & Zhou, 2016)	Dynamic
P5 = (Pinzón et al., 2010)	Machine Learning
P6 = (Qbea'h et al., 2016)	Static
P7 = (Ghafarian, 2017)	Combined
P8 = (Ping et al., 2016)	Static "Instruction-Set Randomization"
P9 = (Prabakar et al., 2013)	Combined
P10 = (Oosawa & Matsuda, 2014)	Machine Learning
P11 = (Shanmughaneethi et al., 2009)	Combined
P12 = (Li et al., 2019)	Machine Learning
P13 = (Chenyu & Fan, 2016)	Dynamic
P14 = (Matsuda et al., 2011)	Static
P15 = (Buja et al., 2014)	Static "Parameter Filtering"
P16 = (Srivastava, 2014)	Static
P17 = (Xie et al., 2019)	Machine Learning
P18 = (Jiao et al., 2012)	Static "Defense Mechanism"
P19 = (Xiao et al., 2017)	Machine Learning
P20 = (Pomeroy & Tan, 2011)	Static
P21 = (He et al., 2015)	Static
P22 = (Hanmanthu et al., 2015)	Machine Learning
P23 = (Umar et al., 2018)	Static
P24 = (Upadhyay & Khilari, 2016)	Static "Parameter Filtering"
P25 = (Xue & He, 2011)	Static
P26 = (Uwagbole et al., 2017)	Machine Learning
P27 = (Wu & Chan, 2012)	Machine Learning
P53 = (Gu et al., 2019)	Dynamic
P54 = (Tripathy et al., 2020)	Machine Learning
P55 = (Kuroki et al., 2020)	Combined
P56 = (Hlaing & Khaing, 2020)	Static
P57 = (Jana & Maity, 2020)	Static
P28 = (Balasundaram & Ramaraj, 2012)	Static
P29 = (Kar et al., 2016)	Machine Learning
P30 = (Lee et al., 2012)	Combined
P31 = (Natarajan & Subramani, 2012)	Dynamic
P32 = (Kim & Lee, 2014)	Machine Learning

(*Continued*)

Table 5. (Continued).

Paper No.	Detection Technique
P33 = (Pinzon et al., 2013)	Machine Learning
P34 = (Jang & Choi, 2014)	Combined
P35 = (Mitropoulos & Spinellis, 2009)	Machine Learning
P36 = (McWhirter et al., 2018)	Machine Learning
P37 = (Patel & Shekokar, 2015)	Static "Parameter Filtering"
P58 = (Tang et al., 2020)	Machine Learning
P38 = (Narayanan et al., 2011)	Static
P39 = (Doshi et al., 2014)	Dynamic
P40 = (Singh et al., 2015)	Machine Learning
P41 = (Selvamani & Kannan, 2011)	Dynamic
P42 = (Choraś et al., 2013)	Static
P43 = (Kar et al., 2015)	Machine Learning
P44 = (Hidhaya & Geetha, 2012)	Static
P45 = (Raj & Sherly, 2017)	Static "Parameter Filtering"
P46 = (Maheswari & Anita, 2016)	Machine Learning
P47 = (Huang et al., 2017)	Static
P48 = (Sadalkar et al., 2011)	Combined
P49 = (Feng et al., 2019)	Combined
P50 = (Perkins et al., 2016)	Static "Instruction-Set Randomization"
P51 = (Joseph & Jevitha, 2015)	Static
P52 = (Pinzón et al., 2009)	Static
P59 = (Aliero et al., 2020)	Dynamic
P60 = (Abikoye et al., 2020)	Static

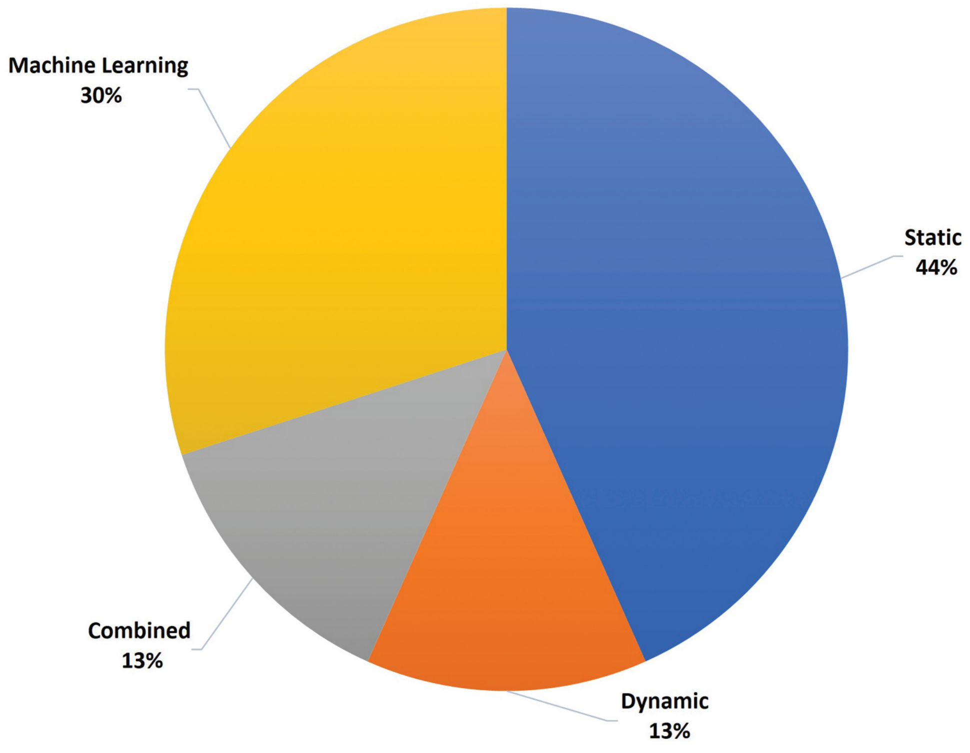

Figure 4. Distribution of SQL injection detection techniques.

most mitigated attacks (38.00%). In this paper, we presented the main detection and prevention techniques that are commonly used in the field of SQL injection, and classify the common types of SQL injection that are used to attack web applications and databases. The research papers that have been reviewed are bounded in the period 2009–2021.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Mohammed Nasereddin http://orcid.org/0000-0002-3740-9518
Ashaar ALKhamaiseh http://orcid.org/0000-0002-1551-5758

References

Abikoye, O. C., Abubakar, A., Dokoro, A. H., Akande, O. N., & Kayode, A. A. (2020). A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *EURASIP Journal on Information Security*, *(2020*(1), 1–14. https://doi.org/10.1186/s13635-020-00113-y

Aliero, M. S., Ghani, I., Qureshi, K. N., & Rohani, M. F. A. (2020). An algorithm for detecting SQL injection vulnerability using black-box testing. *Journal of Ambient Intelligence and Humanized Computing*, *11*(1), 249–266. https://doi.org/10.1007/s12652-019-01235-z

Appiah, B., Opoku-Mensah, E., & Qin, Z. (2017). SQL injection attack detection using fingerprints and pattern matching technique. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 583–587). IEEE, Beijing, China.

Balasundaram, I., & Ramaraj, E. (2012). An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching. *Procedia Engineering*, *30*(1), 183–190. https://doi.org/10.1016/j.proeng.2012.01.850

Bonner, M., (2019). *Web Attacks and Gaming Abuse*. (Akamai Web Attacks and Gaming Abuse Report, vol. 5, no. 3, pp. 2-4). Akamai. https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-web-attacks-and-gaming-abuse-report-2019.pdf

Buja, G., Abd Jalil, K. B., Ali, F. B. H. M., & Rahman, T. F. A. (2014). Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack. In *2014 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE)* (pp. 60–64). IEEE.

Chenyu, M., & Fan, G. (2016). Defending SQL injection attacks based-on intention-oriented detection. In *2016 11th International Conference on Computer Science & Education (ICCSE)* (pp. 939–944). IEEE, Nagoya, Japan.

Choraś, M., Kozik, R., Puchalski, D., & Hołubowicz, W. (2013). Correlation approach for SQL injection attacks detection. In *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions* (pp. 177–185). Springer, Ostrava, Czech Republic.

Doshi, J. C., Christian, M., & Trivedi, B. H. (2014). SQL FILTER–SQL Injection prevention and logging using dynamic network filter. In *International symposium on security in computing and communication* (pp. 400–406). Springer.

Feng, K., Gu, X., Peng, W., & Yang, D. (2019). Moving target defense in preventing sql injection. In *International Conference on Artificial Intelligence and Security* (pp. 25–34). Springer, New York, NY, USA.

Ghafarian, A. (2017). A hybrid method for detection and prevention of SQL injection attacks. In *2017 Computing Conference* (pp. 833–838). IEEE, London, UK.

Gu, H., Zhang, J., Liu, T., Hu, M., Zhou, J., Wei, T., & Chen, M. (2019). DIAVA: A traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data. *IEEE Transactions on Reliability*, *69*(1), 188–202. https://doi.org/10.1109/TR.2019.2925415

Gupta, B. B., Gupta, S., Gangwar, S., Kumar, M., & Meena, P. K. (2015). Cross-site scripting (XSS) abuse and defense: Exploitation on several testing bed environments and its defense. *Journal of Information Privacy and Security*, *11*(2), 118–136. https://doi.org/10.1080/15536548.2015.1044865

Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). Hunting for DOM-Based XSS vulnerabilities in mobile cloud-based online social network. *Future Generation Computer Systems*, *79*, 319–336. https://doi.org/10.1016/j.future.2017.05.038

Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015). A comparative study: MongoDB vs. MySQL. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1–6). IEEE, Oradea, Romania.

Halfond, W. G., Viegas, J., & Orso, A. (2006). A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE international symposium on secure software engineering* (pp. 13–15). IEEE.

Hanmanthu, B., Ram, B. R., & Niranjan, P. (2015). SQL Injection Attack prevention based on decision tree classification. In *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)* (pp. 1–5). IEEE, Karpagam College of Engineering, Coimbatore, Tamil NADU, India.

He, P., Lv, Y., Cai, J., Yi, Y., & Dai, Z. (2015). Study and design of database protection system for Sql attacks. In *2015 IEEE International Conference on Communication Software and Networks (ICCSN)* (pp. 378–383). IEEE, Chengdu, China.

Hidhaya, S. F., & Geetha, A. (2012). Intrusion protection against SQL injection and cross site scripting attacks using a reverse proxy. In *International Conference on Security in Computer Networks and Distributed Systems* (pp. 252–263). Springer, Trivandrum, India.

Hlaing, Z. C. S. S., & Khaing, M. (2020). A detection and prevention technique on sql injection attacks. In *2020 IEEE Conference on Computer Applications (ICCA)* (pp. 1–6). IEEE, Yangon, Myanmar.

Huang, Y., Fu, C., Chen, X., Guo, H., He, X., Li, J., & Liu, Z. (2017). A mutation approach of detecting SQL injection vulnerabilities. In *International Conference on Cloud Computing and Security* (pp. 175–188). Springer, Nanjing, China.

Jana, A., & Maity, D. (2020). Code-based Analysis Approach to Detect and Prevent SQL Injection Attacks. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1–6). IEEE, IIT Kharagapur, West Bengal, India.

Jang, Y. S., & Choi, J. Y. (2014). Detecting SQL injection attacks using query result size. *Computers & Security*, *44*, 104–118. https://doi.org/10.1016/j.cose.2014.04.007

Jiao, G., Xu, C. M., & Maohua, J. (2012). SQLIMW: A new mechanism against SQL-Injection. In *2012 International Conference on Computer Science and Service System* (pp. 1178–1180). IEEE, Nanjing, China.

Joseph, S., & Jevitha, K. P. (2015). An automata based approach for the prevention of nosql injections. In *International Symposium on Security in Computing and Communication* (pp. 538–546). Springer.

Kar, D., Panigrahi, S., & Sundararajan, S. (2015). SQLiDDS: SQL injection detection using query transformation and document similarity. *In International Conference on Distributed Computing and Internet Technology* (pp. 377–390). Springer, Bhubaneswar, India.

Kar, D., Panigrahi, S., & Sundararajan, S. (2016). SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. *Computers & Security*, *60*, 206–225. https://doi.org/10.1016/j.cose.2016.04.005

Karuparthi, R. P., & Zhou, B. (2016). Enhanced approach to detection of SQL injection attack. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 466–469). IEEE, Anaheim, CA, USA.

Katole, R. A., Sherekar, S. S., & Thakare, V. M. (2018). Detection of SQL injection attacks by removing the parameter values of SQL query. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)* (pp. 736–741). IEEE, Coimbatore, India.

Kim, M. Y., & Lee, D. H. (2014). Data-mining based SQL injection attack detection using internal query trees. *Expert Systems with Applications*, *41*(11), 5416–5430. https://doi.org/10.1016/j.eswa.2014.02.041

Kumar, P., & Pateriya, R. K. (2012). A survey on SQL injection attacks, detection and prevention techniques. In *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)* (pp. 1–5). IEEE, Coimbatore, India.

Kuroki, K., Kanemoto, Y., Aoki, K., Noguchi, Y., & Nishigaki, M. (2020). Attack Intention Estimation Based on Syntax Analysis and Dynamic Analysis for SQL Injection. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1510–1515). IEEE, Madrid, Spain.

Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. *Mathematical and Computer Modelling*, *55*(1–2), 58–68. https://doi.org/10.1016/j.mcm.2011.01.050

Li, Q., Li, W., Wang, J., & Cheng, M. (2019). A SQL injection detection method based on adaptive deep forest. *IEEE Access*, *7*, 145385–145394. https://doi.org/10.1109/ACCESS.2019.2944951

Maheswari, K. G., & Anita, R. (2016). An intelligent detection system for SQL attacks on web IDS in a real-time application. In *Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC–16')* (pp. 93–99). Springer, Anchorage, AK, USA.

Matallah, H., Belalem, G., & Bouamrane, K. (2021). Comparative study between the MySQL relational database and the MongoDB NoSQL database. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *13*(3), 38–63. https://doi.org/10.4018/IJSSCI.2021070104

Matsuda, T., Koizumi, D., Sonoda, M., & Hirasawa, S. (2011). On predictive errors of SQL injection attack detection by the feature of the single character. In *2011 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1722–1727). IEEE.

McWhirter, P. R., Kifayat, K., Shi, Q., & Askwith, B. (2018). SQL injection attack classification through the feature extraction of SQL query strings using a Gap-Weighted string subsequence Kernel. *Journal of Information Security and Applications*, *40*, 199–216. https://doi.org/10.1016/j.jisa.2018.04.001

Mitropoulos, D., & Spinellis, D. (2009). SDriver: Location-specific signatures prevent SQL injection attacks. *Computers & Security*, *28*(3–4), 121–129. https://doi.org/10.1016/j.cose.2008.09.005

Narayanan, S. N., Pais, A. R., & Mohandas, R. (2011). Detection and prevention of sql injection attacks using semantic equivalence. In *International conference on information processing* (pp. 103–112). Springer, Bangalore, India.

Natarajan, K., & Subramani, S. (2012). Generation of SQL-injection free secure algorithm to detect and prevent SQL-injection attacks. *Procedia Technology*, *4*, 790–796. https://doi.org/10.1016/j.protcy.2012.05.129

Oosawa, T., & Matsuda, T. (2014). SQL injection attack detection method using the approximation function of zeta distribution. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 819–824). IEEE, San Diego, CA, USA.

Patel, N., & Shekokar, N. (2015). Implementation of pattern matching algorithm to defend SQLIA. *Procedia Computer Science*, *45*, 453–459. https://doi.org/10.1016/j.procs.2015.03.078

Perkins, J., Eikenberry, J., Coglio, A., Willenson, D., Sidiroglou-Douskos, S., & Rinard, M. (2016). AutoRand: Automatic keyword randomization to prevent injection attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 37–57). Springer, San Sebastián, Spain.

Ping, C. (2017). A second-order SQL injection detection method. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (pp. 1792–1796). IEEE, Chengdu, China.

Ping, C., Jinshuang, W., Lin, P., & Han, Y. (2016). Research and implementation of SQL injection prevention method based on ISR. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)* (pp. 1153–1156). IEEE, Chengdu, China.

Pinzón, C., De Paz, J. F., Bajo, J., Herrero, Á., & Corchado, E. (2010). AIIDA-SQL: An adaptive intelligent intrusion detector agent for detecting SQL injection attacks. In *2010 10th International Conference on Hybrid Intelligent Systems* (pp. 73–78). IEEE, Atlanta, GA, USA.

Pinzón, C., De Paz, Y., & Bajo, J. (2009). A multiagent based strategy for detecting attacks in databases in a distributed mode. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)* (pp. 180–188). Springer.

Pinzon, C. I., De Paz, J. F., Herrero, A., Corchado, E., Bajo, J., & Corchado, J. M. (2013). idMAS-SQL: Intrusion detection based on MAS to detect and block SQL injection through data mining. *Information Sciences*, *231*, 15–31. https://doi.org/10.1016/j.ins.2011.06.020

Pomeroy, A., & Tan, Q. (2011). Effective SQL injection attack reconstruction using network recording. In *2011 IEEE 11th International Conference on Computer and Information Technology* (pp. 552–556). IEEE, Paphos, Cyprus.

Positive Technologies. (2019). *Attacks on Web Applications: 2018 in Review*. https://www.ptsecurity.com/ww-en/analytics/web-application-attacks-2019/

Prabakar, M. A., KarthiKeyan, M., & Marimuthu, K. (2013). An efficient technique for preventing SQL injection attack using pattern matching algorithm. In *2013 IEEE international conference on emerging trends in computing, communication and nanotechnology (ICECCN)* (pp. 503–506). IEEE, Tirunelveli, India.

Qbea'h, M., Alshraideh, M., & Sabri, K. E. (2016). Detecting and preventing SQL injection attacks: A formal approach. In *2016 Cybersecurity and Cyberforensics Conference (CCC)* (pp. 123–129). IEEE, Amman, Jordan.

Raj, S. N., & Sherly, E. (2017). An SQL injection defensive mechanism using reverse insertion technique. In *International Conference on Next Generation Computing Technologies* (pp. 335–346). Springer, Dehradun, India.

Sadalkar, K., Mohandas, R., & Pais, A. R. (2011). Model based hybrid approach to prevent SQL injection attacks in PHP. In *International Conference on Security Aspects in Information Technology* (pp. 3–15). Springer, Haldia, India.

Sadeghian, A., Zamani, M., & Ibrahim, S. (2013a). SQL injection is still alive: A study on SQL injection signature evasion techniques. In *2013 International Conference on Informatics and Creative Multimedia* (pp. 265–268). IEEE, Kuala Lumpur, Malaysia.

Sadeghian, A., Zamani, M., & Manaf, A. A. (2013b). A taxonomy of SQL injection detection and prevention techniques. In *2013 international conference on informatics and creative multimedia* (pp. 53–56). IEEE, Kuala Lumpur, Malaysia.

Schults, C. (2020, December 15). *SQL injection, explained: What it is and how to prevent it*. Paris France. https://blog.sqreen.com/sql-injection-explained/

Selvamani, K., & Kannan, A. (2011). A Novel Approach for Prevention of SQL Injection Attacks Using Cryptography and Access Control Policies. In *International Conference on Power Electronics and Instrumentation Engineering* (pp. 26–33). Springer, Nagpur, India.

Shanmughaneethi, S. V., Shyni, S. C. E., & Swamynathan, S. (2009). Sbsqlid: Securing web applications with service based sql injection detection. In *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies* (pp. 702–704). IEEE, Bangalore, India.

Singh, G., Kant, D., Gangwar, U., & Singh, A. P. (2015). Sql injection detection and correction using machine learning techniques. In *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1* (pp. 435–442). Springer.

Srivastava, M. (2014). Algorithm to prevent back end database against SQL injection attacks. *In 2014 International conference on computing for sustainable global development (INDIACom)* (pp. 754–757). IEEE, New Delhi, India.

Tajpour, A., Massrum, M., & Heydari, M. Z. (2010). Comparison of SQL injection detection and prevention techniques. In *2010 2nd International Conference on Education Technology and Computer* (pp. V5–174). IEEE, Shanghai, China.

Tajpour, A., & zade Shooshtari, M. J. (2010). *Evaluation of SQL injection detection and prevention techniques*. In *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks* (pp. 216–221). IEEE, Liverpool, UK.

Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. *Knowledge-Based Systems*, *190*, 105528. https://doi.org/10.1016/j.knosys.2020.105528

Tripathy, D., Gohil, R., & Halabi, T. (2020). Detecting SQL injection attacks in cloud SaaS using machine learning. In *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)* (pp. 145–150). IEEE, Baltimore, MD, USA.

Umar, K., Sultan, A. B., Zulzalil, H., Admodisastro, N., & Abdullah, M. T. (2018). Formulation of SQL Injection Vulnerability Detection as Grammar Reachability Problem. In *2018 International Conference on Information and Communication Technology for the Muslim World (ICT4M)* (pp. 179–184). IEEE, Kuala Lumpur, Malaysia.

Upadhyay, U., & Khilari, G. (2016). SQL injection avoidance for protected database with ASCII using SNORT and HONEYPOT. In *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)* (pp. 596–599). IEEE, Ramanathapuram, India.

Uwagbole, S. O., Buchanan, W. J., & Fan, L. (2017). Applied machine learning predictive analytics to SQL injection attack detection and prevention. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (pp. 1087–1090). IEEE.

Wu, B. (2010). SQL Injection Defense Mechanisms for IIS+ ASP+ MSSQL Web Applications. In *International Conference on Forensics in Telecommunications, Information, and Multimedia* (pp. 271–276). Springer, Shanghai, China.

Wu, X. R., & Chan, P. P. (2012). SQL injection attacks detection in adversarial environments by k-centers. In *2012 International Conference on Machine Learning and Cybernetics* (Vol.1, pp. 406–410). IEEE, Xi'an, China

Xiao, Z., Zhou, Z., Yang, W., & Deng, C. (2017). An approach for SQL injection detection based on behavior and response analysis. In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)* (pp. 1437–1442). IEEE, Guangzhou, China.

Xie, X., Ren, C., Fu, Y., Xu, J., & Guo, J. (2019). Sql injection detection for web applications based on elastic-pooling cnn. *IEEE Access*, *7*, 151475–151481. https://doi.org/10.1109/ACCESS.2019.2947527

Xue, Q., & He, P. (2011). On defense and detection of SQL server injection attack. In *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 1–4). IEEE, Wuhan, China.