**Journal of Information Systems Engineering and Management**
2025, 10(35s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

**Research Article**

# The Implications for a Hybrid Detection Technique Against Malicious SQL Attacks on Web Applications

Sarajaldeen Akram Bahjat Arif[1], Dr. Sharyar Wani[2]
[1] *Student Kulliyyah of Information and Communication Technology, International Islamic University Malaysia (IIUM)*
[2] *Asst. Prof. Kulliyyah of Information and Communication Technology, International Islamic University Malaysia (IIUM)*

**ARTICLE INFO**

Received: 18 Dec 2024

Revised: 10 Feb 2025

Accepted: 28 Feb 2025

**ABSTRACT**

Today, most web applications are vulnerable to SQL-injection attacks. Malicious inputs by unauthorized attackers causing the deletion, modification, or retrieval of confidential data from remote database which creates huge losses of money and even affect the work of commercial vendors and financial companies. Therefore, it is essential to develop a new technique to authenticate access to database related to web applications and prevent SQL injection vulnerabilities. But the large number of available prevention techniques make the selection of the best solution a big challenge, because not every technique fit all types of web application, hence a one technique for all is another issue and a difficult task. Accordingly, the aim of this study is to identify the latest SQL injection attacks based on user's inputs in web application associated with remote server database, and to develop a new method based on dynamic detection technique to prevent SQL injection attacks. The methodology is based on JavaScript and PHP languages for developing a new technique called DetectCombined capable of filtering queries using parameterized queries to protect against SQL injection which is a safe method. It is a code with double shield protection that prevents unauthorized extraction or damaging the remote database in the server side due to malicious SQL injection. The proposed DetectCombined is an innovated technique that execute a protection code based on a sequence of three stages: filtration-validation-history, this technique produces a robust protection code that distinguish between safe SQL commands and malicious ones, and reinforce the memory of detection procedure by saving previous SQL attacks in special tables in the remote database, regardless of the types of users whether a general user of admin. The outcome of this study will add to the body of knowledge the most important and recent proposed solutions to mitigate SQL injection attack, in particular those based on machine learning algorithm.

**Keywords:** SQL injection, Malicious Attack, SQL detection, Machine Learning.

## INTRODUCTION

Today, web security and user privacy in database-driven web applications are extremely difficult tasks against malicious users and web attackers. SQL-injection is one of the most hazardous cyber-attacks nowadays, causing large losses of money and confidential data to commercial vendors and financial organizations (Kareem et al., 2021). A successful web application attack allows a malicious attacker to delete, edit, or retrieve critical data without being authenticated by the server and using hacked rights to access a remote database (Rai & Nagpal, 2019). The number of web application attacks is rapidly increasing. The availability of massive volumes of data on the internet motivates hackers to launch novel forms of assaults. Extensive web application security research has been conducted in this field. Structured Query Language Injection (SQLI) is the most deadly online application attack. This attack is a serious threat to web applications (Ines et al., 2020). The majority of SQL injection attacks are linked to unauthenticated password filling in order to retrieve crucial information related to authorized users whose data is kept in a remote database. As a result, the SQL Injection attack allows malicious individuals to read such information from the remote database. While secured systems will only allow access to data that is publicly available. However, a badly built system is vulnerable to malicious SQL injection, which allows external users to penetrate the database via the password entry field or hijack users' passwords and use them to access the database (Madhusudhan & Ahsan, 2022). This can be accomplished by inserting a

SQL statement injection query into the password field of the login interface page (Nithya et al., 2013). The lack of security guarantee in online apps is the source of the majority of complaints. It has been discovered that a lack of information security and the impact of malicious assaults on the performance of online apps have an impact on their users (Gogoi et al., 2021). It is clear that the problem of lack of security in web applications eases SQL injection risks and increases security, which is a major difficulty for practically all web developers nowadays (Kareem et al., 2021; Hadabi et al., 2022). As a result, this article will classify current SQL injection methods and compare them in order to develop a new technique for avoiding these assaults on the client side. Furthermore, this study will emphasize the shortcomings and performance of each solution and will seek to prevent them in the recommended solution by proposing a new technique to prevent SQL injection attacks.

## THE CHALLENGES OF WEB SECURITY

One of the key issues of SQL injection is allowing users to customize their own text and symbols without many constraints, which may cause the user to detest the input and abandon the application (Yadav & Kumar, 2022). Because SQL statements are merely text, it is straightforward to dynamically update SQL statements using a tiny bit of computer code to present the user with specified data (Yadav & Kumar, 2022). As a result, insufficient input filtration and validation of input forms in dynamic online applications is a major issue that web developers are still working to solve (Dasmohapatra & Priyadarshini, 2022). Furthermore, a web developer may make a mistake when implementing validation code for certain parameters, such as SQL, making the online application particularly vulnerable (Tafa & Resulaj, 2021). In this regard, Raniah (2019) identified the problems and challenges that most web applications face when dealing with SQL injection attacks and securing the applications, such as tracing the inputs from the first entry point to the SQL statement, securing existing web applications at a low cost, securing saved procedures as well as dynamic SQL statements, and methods and techniques for eliminating and modifying specific malicious inputs to modify them as needed.

A review of the literature reveals that several studies have been conducted in the past to address this issue, with the majority of detection techniques and algorithms focusing on mitigating this SQL injection attack either by preventing it at an early stage (on the client side) or detecting it when it occurs (on the server side). However, all of these strategies and algorithms are still ineffective, and there is a small window for hackers to infiltrate the database on the distant server via the web application, which the technique described in this study should fill. Hackers have continued to utilize innovative approaches to attack the core security services, such as authentication, confidentiality, integrity, authorization, and integrity, even in recent years (Deepa et al., 2018). According to OWASP (2018) and SANS (2020), SQL injection vulnerabilities continue to be the most deadly and widespread web assaults. As a result, despite the diversity of strategies and algorithms provided in the past, and new techniques are emerging, algorithms to deal with SQL injection attacks are more important than ever (Ines et al., 2020). While the enormous variety of available prevention approaches makes selecting the best solution a tough effort, because not every algorithm and technique fits all sorts of online applications, having one technique that works for all is another issue and a challenging undertaking. Furthermore, the majority of existing SQL injection countermeasures used either syntax-based detection methods or a list of predefined rules to detect SQL injection, making them vulnerable to advanced and sophisticated attacks because attackers create new ways to evade detection by leveraging prior knowledge (Yazeed, 2021). As a result, this study will seek to answer this later problem by providing a basic overview of all known approaches and algorithms and then building on them to produce a new one suite all online apps, which will be the finest recently presented solutions for this problem.

## SQL INJECTION ATTACKS

Today SQL injection (SLQI) is a form of malicious attack that targets popular types of web applications and increases the susceptibility of web application access. SQLI exists in specific online applications that do not filter the input data. Thus, these websites exploit a high level of vulnerability, allowing a malicious attacker to introduce malicious code, such as SQL commands, through input fields such as login and password (Statista, 2019). A hacker is someone who uses computer, networking, or other technical skills to solve a technological problem or to commit malicious activities (Deepa et al., 2018). Web applications are heavily used in today's culture, whether for shopping or financial transactions. It is vital to protect the security of these internet programs. The majority of the transaction or consumer information is stored in the backend databases for these web apps. SQL injection attacks are one of the drawbacks of these web apps (D'silva et al., 2017). Furthermore, if the adversary acquires the session id, the web application sessions are susceptible to session hijacking attempts. Online applications are especially vulnerable to session hijacking attacks due to the diversity of mechanisms available to acquire session/HTTP cookies.

Although numerous ways for protecting databases from SQL injection attacks have been proposed, there is no single answer or approach for preventing these types of cyber-attacks. As a result, SQL injection has become a major threat for database-driven websites around the world. When a cybercriminal runs a SQL query and sends it to a database, the data is sent from the client to the server through the input data. In data-plane input, the SQL command is typically used in place of the password or login credentials. This allows the cybercriminal to run their own SQL commands. Furthermore, if a SQL injection attack is successful, confidential information can be read, stolen, edited, added, updated, or removed. Cyber attackers can also use the database to do administrative activities such as shutting down the computer, recovering content from any file, and even delivering commands to the operating system (Jemal et al., 2020). Cybercriminals' techniques are evolving in lockstep with company technology and security solutions. Cybercrime cost businesses around the world $2.7 billion in 2018, and research conducted by infographic from Raconteur (2019) shows into the average damage caused by cyberattacks on various industries as shown in Figure 1.

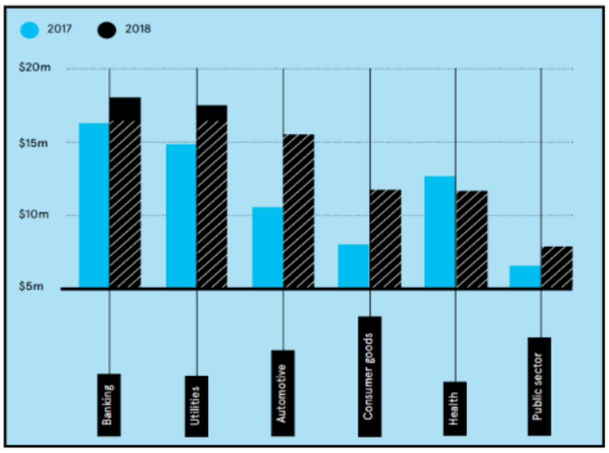


Figure-1: The average annual cost of cybercrimes by industry (Raconteur, 2019)

Another indicator on the value of risk on business is shown in Figure 2. It is found that 77% comes from direct cyberattacks in the future, while 23% from indirect cyberattacks. The issues of SQLI this figure will continue to rise for many years in the future. Vendors pay to developers in order to protect their business from these kinds of attacks, more money spent on upgrades and repairs of current systems, and the costs associated with lost clients and a tarnished brand are among the losses.

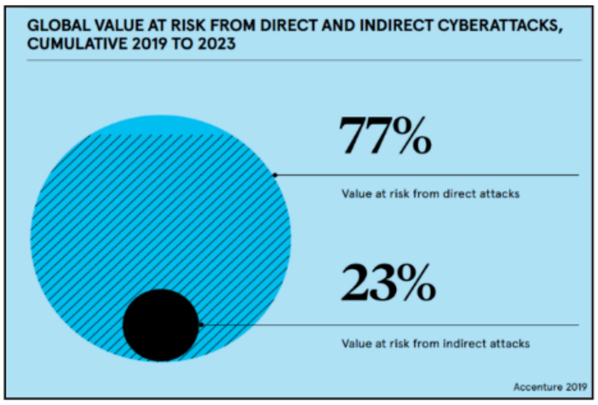


Figure-2: Global value of risks due to cyberattacks worldwide (Accenture, 2019)

The examination of the literature suggests that a SQLI attack is a common security breach that attacks the database of an online application. With the number of methods for exploiting SQLIA vulnerabilities in online applications rising all the time, there is no one-size-fits-all solution or tactic. As a result, multiple SQLI procedures have to be devised and improved in order to mitigate the possible threats posed by these various attack approaches. The majority of these methods, however, have yet to be investigated, and they are still only theories that must be executed, assessed, and limited. Furthermore, the majority of existing SQL injection countermeasures used either syntax-based detection methods or a set of predefined rules to detect SQL injection, making them vulnerable to advanced and sophisticated attacks because attackers create new ways to evade detection by leveraging prior knowledge. Although semantic-based qualities can aid in identification, no studies to our knowledge have focused on extracting semantic features from SQL statements (Yazeed, 2021).

## DETECTION TECHNIQUES

The most mature and extensively used type of intrusion detection system (IDS) is signature-based IDS. And Intrusion Detection Systems as suggested by Gupta and Sharma (2022) were one of the tactics used by certain researchers to counter SQLI. However, because signature-based IDSs are static, they cannot detect new types of attacks, and the attacker can easily avoid detection by changing the appearance of the attack (Panigrahi et al., 2022). While IDSs alone cannot protect against all types of SQLI, a combination of optimal web server configuration and the usage of parameterized queries during the coding phase can. Several SQLI detection algorithms have been proposed in the literature, but none of them consider SQLI in stored procedures. Although the SQLI method is the same for both stored procedures and application layer programs, due to their limited programmability, as well as the technique's usability and deployment ability (Fang et al., 2018), the same detection technique could not be utilized to stored procedures. In certain studies, stored procedures are mentioned as a solution to SQLIs. Because stored procedures are saved in the database, the solutions they provide cannot be utilized to protect the stored procedures themselves. Ke et al. (2006), for example, proposed a novel way to protecting stored processes from SQLI assaults. Our solution combines static application code analysis with runtime validation to prevent such attacks. In the static section, they develop a stored procedure parser, which we use to instrument the necessary statements in order to compare the original SQL statement structure to that integrating user inputs for any SQL statement that depends on user inputs. The implementation of this technology can be automated and used only when necessary.

Existing techniques such as filtering, penetration testing, information-flow analysis, and defensive coding can also detect and mitigate a fraction of SQLI vulnerabilities. However, it appears that filtering (input validation) is the most efficient and straightforward technique. Although SQLI detection techniques that use input validation are prone to a significant number of false positives, there is no guarantee of no false negatives. Input validation is one of the most effective SQLI prevention methods. Checking for single quotes and dashes and manually escaping them is a simple example of input validation. This may be easily avoided by using the ASCII form of these characters, such as CHAR (0x27) for single quotes. Cook and Rai (2005) define safe query objects. Ntagwabira and Kang (2010) proposed a method for identifying and avoiding SQLI attacks by looking for changes in the intended outcome of the query induced by user inputs in the same context. They advised employing Query tokenization, which the QueryParser function performs, to identify SQLI assaults. A hacker's SQLI input should most likely include a space, single quotes, or double dashes.

Separately tokenizing the initial enquiry and a query with injection should be part of the SQLI solution. Tokenization is accomplished by recognizing all strings preceding each sign, as well as a space, a single quote mark, or two dashes. Following the formation of the tokens, they are combined to form an array, with each token serving as an element. The lengths of two arrays returned by the first query and a query with injection are compared to determine whether or not there is injection. As a result, access to data can be permitted or prohibited depending on whether the array lengths are the same or different. The insufficient input validation detection technique will allow code to be executed without being properly validated. This method has been successful in several SQLIs in preventing hackers from exploiting inadequate input validation, which endangers the target database and allows the attacker to easily launch attacks using malicious SQL code (Statista, 2019). Table-1 summarizes the assessment and reveals the detection strategies and capabilities. The symbol denotes systems that correctly detect all types of malicious assaults within a specific kind. The symbol denotes strategies that are incapable of detecting all attacks of that type.

Table-1: A comparison between SQLI detection techniques based on the types of attacks.

| | SAFE LI | SQ L-IDS | Swaddl er | SQL Preve nt | SQLr an | SQLI PA | DIWe Da | Tautolo gy Checke r | Removi ng SQL query | SQLCh eck | SQL Gua rd |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tautolog ies | × | √ | × | √ | √ | √ | × | √ | √ | √ | √ |

Journal of Information Systems Engineering and Management
2025, 10(35s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Piggy-backed | √ | √ | × | √ | √ | × | × | × | √ | √ | √ |
| Illegal/ Incorrec | √ | √ | × | √ | √ | × | × | × | √ | √ | √ |
| Union | √ | √ | × | √ | √ | × | × | × | √ | √ | √ |
| Stored Procedu | √ | √ | × | √ | × | × | × | × | √ | × | × |
| Inferenc e | √ | √ | × | √ | √ | × | √ | × | √ | √ | √ |
| Alternat e | √ | √ | × | √ | √ | × | × | × | √ | √ | √ |

## RESEARCH METHODOLOGY

SQL statements are primarily text, and this text is the weak point of data validation throughout the routine procedure, as well as transferring this text to the remote database on the server to extract some information or perform other tasks the user desires when logging into his/her account. In this chapter, the researcher's devised technique will demonstrate how data validation prevents the access of any type of malicious code using a double shield to prevent unauthorized extraction or damage to the database. The initial stage in our technique is to validate the prototype website's input fields. Website prototypes are usually interactive examples of a website that are created early in the project's lifespan. They are used to test the efficacy of our technology and to show how the detection process works and detects when a person attempts to access the distant database.

## RESULTS AND FINDINGS

The review of literatures showed that when a web developer allows the users to enter data without restriction, a malicious code maybe injected using SQL commands to harm the remote database in many ways. It is very common that web developers neglect this point and let the users to input their own values for different purposes such as login names or search for certain values in the database. The developed technique in this study will show how data validation prevent the access of any kind of malicious code with double shield to prevent unauthorized extraction or damaging the database.The technique proposed (DetecCombined) successfully tested a prototype webpage filtering and validation to input fields and for new or previous SQL injection attempts of malicious parameter to the database as text include SQL commands that can be used incorrectly to access the data stored in all rows (records) of tables in the remote database. The webpages used to enter personal data of users and admins could be shielded through DetecCombined code to protect a web application from any compromise to the security of the database in the server side. For this purpose a front-end interface has been designed of DetecCombined application, which is the layer the user will use to enter the username and password, as well as see, and interact with the database through buttons, images, interactive elements, and text. In the front-end interface, HTML was deployed as well as JavaScript. Our method will filter the entered username and password through a temporary variable to check first if there is any parameter or text that can be used incorrectly to access the data stored in all rows (records) in the remote database. This step is essential in *DetectCombined* technique to prevent malicious users to inject SQL commands into an SQL statement through an intermediate variable via web page input as well initiate a shield in client side to protect a web application from any compromise to the security of the database in the server side. The flowchart in Figure 3 shows the flow of search steps as an admin in red color path.

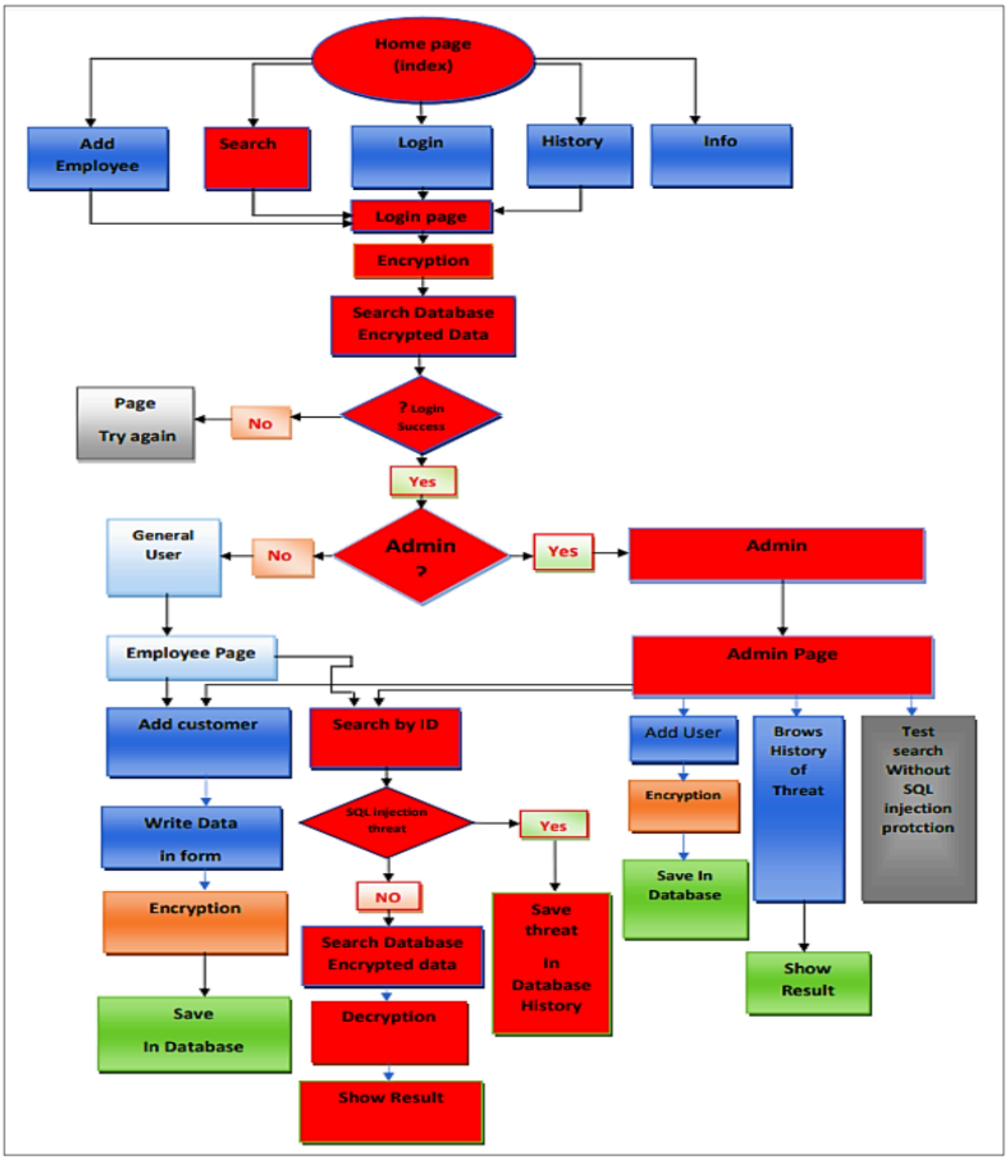
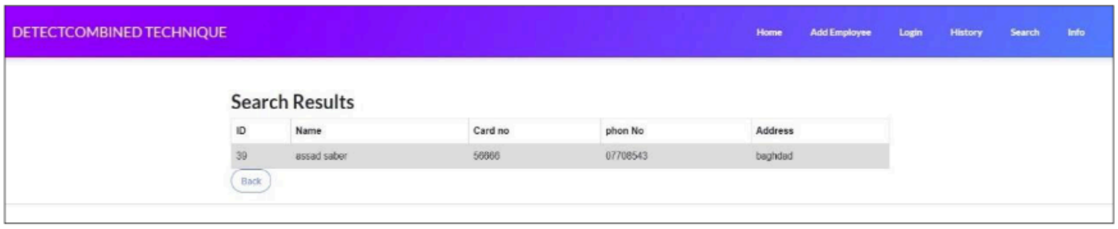


Figure-3: The flowchart of search for data as normal user (search/admin/level1)

The *DetectCombined* technique also allow for doing a search as an admin (Level 1) using the ID page. The same procedure is be applied for non-admin user which imply the detection any SQL injection threat before showing any result, in case there is any suspicion it will save threat attack in database history with specific message so that to swiftly detect the same attack in the future, and if any the search result will show (0) as shown in Figure 4. But if there is no threat the *DetectCombined* will search the database safely and encrypted the data then do decryption and show the search result without encryption so that the user can understand, i.e., ID, Name, Card Number, Phone Number, Address. An example of this procedure is shown in Figure 4.

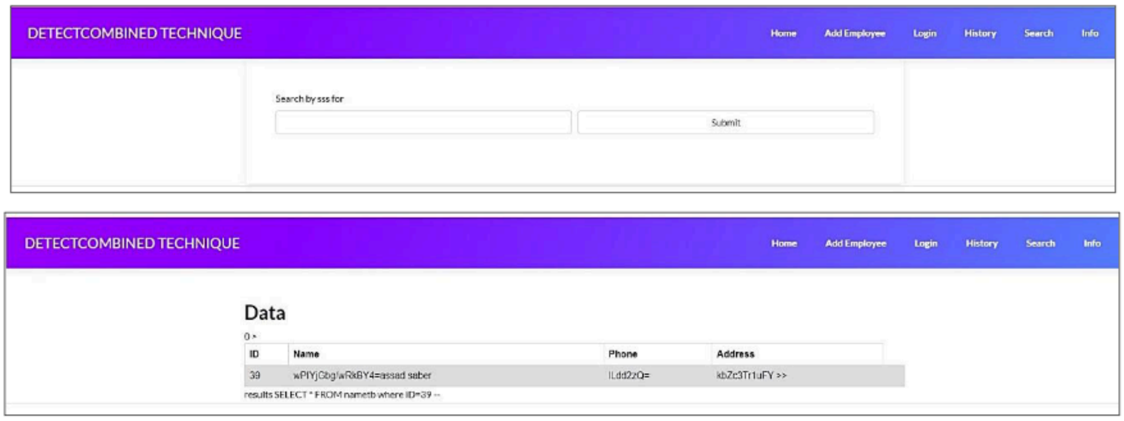


Search Results

| ID | Name | Card no | phon No | Address |
|---|---|---|---|---|
| 39 | assad saber | 56866 | 07708543 | baghdad |

Figure-4: The output of valid and clear entry

We can do a test search by using the ID and then the search will begin without protection from SQL injection, the attacks will be shown on the history stored in the database as shown in Figure 5.



Data

0 »

| ID | Name | Phone | Address |
|---|---|---|---|
| 39 | wPIYjGbg/wRkBY4=assad saber | ILdd2zQ= | kbZc3Tr1uFY >> |

results SELECT * FROM nametb where ID=39 --

Figur-5: The search result with protection against SQL injection

## 7 CONCLUSIONS

The review of literature and the results of the prototype technique (DetectCombined) reveal that malicious SQL injection attacks can be effectively blocked if all vulnerable gaps in the validation protocol for input fields are effectively filtered and encrypted before sending SQL queries to the target databases on the remote server. The various techniques found in the literature show that SQL injection is one of the highest threats to all kinds of industries, including banks, businesses, service companies, and government agencies. The study highlights the risks of SQL injection attacks on the security and confidentiality of data stored in databases. The malicious SQL commands passed in each parameter to the query later can be effectively mitigated using the method developed in this study.

The method used in this study is based on a new technique (filtration, validation, and history) that allows the protection code to distinguish between safe SQL commands and malicious ones, regardless of whether the user is a general user or an admin. It is evident that scholars in this field have emphasized that weak filtering of input fields in any web form, especially for username and password, is potentially risky, especially if the remote database stores confidential data like credit card information. Many developers neglect adding a history of previous attacks to their techniques, which can help distinguish between a real user and a malicious one in dynamic web applications such as e-commerce and banking web portals. Neglecting these points permits hackers to execute malicious code and delete certain tables belonging to the database or even the entire database, causing significant problems for organizations dealing with critical and sensitive information.

To that end, the proposed DetectCombined technique developed and tested in this study is an innovative method that executes a protection code based on a sequence of three stages: filtration-validation-history. This technique produces a robust protection code that distinguishes between safe SQL commands and malicious ones and reinforces the memory of the detection procedure by saving previous SQL attacks in special tables in the remote database, regardless of whether the user is a general user or an admin.

## 8 RECOMMENDATIONS

Based on the findings and results of this paper, the study suggests the following recommendations for web developers and vendors:

1. **Use a history record of previous SQL attacks**: This will enhance the security of webpages that include input fields and are linked to a remote database.

2. **Provide effective input validation, filtering, and encryption**: Avoid any weaknesses in the SQL server by implementing robust input validation, filtering, and encryption to discriminate against malicious parameters used for SQL injection attacks that may damage the entire data in a target database. Data sanitization and validation are crucial and should already be in place.

3. **Implement strong filtering of user input**: Weak filtering of user input can enable a hacker to insert malicious SQL code, depending on specific SQL statements, and permit the hacker to execute malicious code.

4. **Use multiple detection methods for SQL injection**: Today, no single solution is sufficient to defeat SQL injection attacks. This is due to the power and flexibility of SQL. The DetectCombined method should address all the previous gaps in this domain.

By implementing these recommendations, web developers and vendors can significantly enhance the security of their web applications and protect against SQL injection attacks, thereby safeguarding critical and sensitive information.

## REFERENCES

[1] Accenture. (2019). Global value of risks from direct and indirect cyberattacks, cumulative 2019 to 2023. Retrieved April 16, 2000. https://res.cloudinary.com/yumyoshojin/image/upload/v1/pdf/fighting-fraud-2019.pdf

[2] Cook, W. R., & Rai, S. (2005). Safe query objects: statically typed objects as remotely executable queries. In *Proceedings of the 27th International Conference on Software engineering* (pp. 97-106).

[3] Dasmohapatra, S., & Priyadarshini, S. B. B. (2022). A Comprehensive Study on SQL Injection Attacks, Their Mode, Detection and Prevention. *In Proceedings of Second Doctoral Symposium on Computational Intelligence* (pp. 617-632). Springer, Singapore.

[4] Deepa, G., Thilagam, P. S., Khan, F. A., Praseed, A., Pais, A. R., & Palsetia, N. (2018). Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. *International Journal of Information Security*, 17, 105-120.

[5] D'silva, K., Vanajakshi, J., Manjunath, K. N., & Prabhu, S. (2017). An effective method for preventing SQL injection attack and session hijacking. *In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 697-701). IEEE.

[6] Fang, Y., Peng, J., Liu, L., & Huang, C. (2018). WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM. In *Proceedings of the 2nd international conference on cryptography, security and privacy* (pp. 170-174).

[7] Gogoi, B., Ahmed, T., & Dutta, A. (2021, December). Defending against SQL Injection Attacks in Web Applications using Machine Learning and Natural Language Processing. *In 2021 IEEE 18th India Council International Conference (INDICON)* (pp. 1-6). IEEE.

[8] Gupta, A., & Sharma, L. S. (2022). A Novel Approach for Detecting SQL Injection Attacks Using Snort. *Journal of The Institution of Engineers (India): Series B*, 1-9.

[9] Hadabi, A., Elsamani, E., Abdallah, A., & Elhabob, R. (2022). An Efficient Model to Detect and Prevent SQL Injection Attack. *Journal of Karary University for Engineering and Science*.

[10] Ines, J., Omar, C., Habib, H., & Adel, M. (2020). SQL Injection Attack Detection and Prevention Techniques Using Machine Learning. *International Journal of Applied Engineering Research,* 15(6), 569-580.

[11] Jemal, I., Cheikhrouhou, O., Hamam, H., & Mahfoudhi, A. (2020). SQL injection attack detection and prevention techniques using machine learning. *International Journal of Applied Engineering Research*, *15*(6), 569-580.

[12] Kareem, F. Q., Ameen, S. Y., Salih, A. A., Ahmed, D. M., Kak, S. F., Yasin, H. M., & Omar, N. (2021). SQL injection attacks prevention system technology. *Asian Journal of Research in Computer Science,* 13, 32.

[13] Ke, W., Muthuprasanna, M., & Kothari, S. (2006, April). Preventing SQL injection attacks in stored procedures. In *Australian Software Engineering Conference (ASWEC'06)* (pp. 8-pp). IEEE.

[14] Madhusudhan, R., & Ahsan, M. (2022). Prevention of SQL Injection Attacks Using Cryptography and Pattern Matching. *In International Conference on Advanced Information Networking and Applications* (pp. 624-634). Springer, Cham.

[15] Nithya, V., Regan, R., & Vijayaraghavan, J. (2013). A survey on SQL injection attacks, their detection and prevention techniques. *International Journal of Engineering and Computer Science*, 2(4), 886-905.

[16] Ntagwabira, L., & Kang, S. L. (2010, July). Use of Query Tokenization to detect and prevent SQL Injection Attacks. In *2010 3rd International Conference on Computer Science and Information Technology* (Vol. 2, pp. 438-440). IEEE.

[17] OWASP. (2018). Owasp top ten project, https://www.owasp.org/index.php/Category:OWASP Top Ten Project, 2019, accessed on March 2023.

[18] Panigrahi, R., Borah, S., Pramanik, M., Bhoi, A. K., Barsocchi, P., Nayak, S. R., & Alnumay, W. (2022). Intrusion detection in cyber–physical environment using hybrid Naïve Bayes-Decision table and multi-objective evolutionary feature selection. *Computer Communications*, 188, 133-144.
[19] Raconteur. (2019). Cybercrime is learning from business, and it's paying off. Available at: https://www.raconteur.net/technology/cybercrime-business.
[20] Rai, S., & Nagpal, B. (2019). Detection & Prevention of SQL Injection Attacks: Developments of the Decade.
[21] Raniah, A. (2019). SQL Injection Attacks: detection and prevention techniques. 3rd international conference on reliability, INFOCOM technologies and optimization (ICRITO) (Trends and Future Directions), Oct 8-10, 2014, AIIT, Amity University Uttar Pradesh, Noida, India
[22] Statista. (2019) Number of web attacks blocked daily worldwide 2015-2018. [Online]. Available: https://www.statista.com/statistics/494961/web-attacksblocked-per-day-worldwide/
[23] Tafa, I., & Resulaj, E. (2021, December). SQL Injection Attacks: Its Types and Ways to Prevent Them. *In Book of Proceedings,* 102.
[24] Yadav, A. K., & Kumar, A. (2022). String Matching Algorithm Based Filter for Preventing SQL Injection and XSS Attacks. *In Inventive Computation and Information Technologies* (pp. 793-807). Springer, Singapore.
[25] Yazeed, A. (2021). An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques. *UTM International Journal of Innovative Computing*, 11(1).