Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 4 (2024)

Overview of Sql Injection Types, Defense Tools and Prevention Techniques

Khyati Sanghavi[1], Krishna Samdani[2*]

[1]*Student, MBA Tech Compuuter Engineering, Mukesh Patel School Of Technology Management & Engineering, SVKM's NMIMS University, Mumbai, India.*

[2*]*Assistant Professor, Computer Engineering Department, Mukesh Patel School Of Technology Management & Engineering, SVKM's NMIMS University, Mumbai, India.*

Abstract: SQL injection is the most widely used attack and therefore remains a highly pervasive threat to the security of databases and applications on the internet. This attack utilizes weaknesses in the input fields of an application to provide for the unauthorized execution of SQL queries on the application's database. Hackers exploit these vulnerabilities thus enabling them to steal sensitive information and breach authentication systems. This review paper discusses the different types of SQLI, such as Union-based, Error-based, Blind, and Time-based SQLI, with a strong emphasis on unique characteristics as well as the precise techniques executed in each of them. A detailed analysis would also focus on tools used in the detection and exploitation of attacks, such as SQLmap, Havij, and jSQL, as well as defensive technologies such as WAFs, parameterized queries, and secure coding practices. That is, by comparing every mode of attack with the prevention techniques used, this paper can, as such, introduce a holistic review of current defense mechanisms in place to understand their strengths and weaknesses. What is more, it identifies critical challenges and research gaps in the prevention of SQLI attacks and propounds themes that should be looked into in the future to enhance security on the web and to diminish the threats that persist.

Keywords*: SQL Injection (SQLI), Cybersecurity, Web Application Security, Anomaly Detection, Machine Learning, Database Protection.*

1. Introduction

SQL Injection, or SQLI, is a type of cyber-attack that injects faulty SQL code into input areas for other purposes, making it possible for these attackers to modify or access information in a database. It typically exploits the security weakness in the application's software, especially when handling user input and consequently grants unapproved access, alteration, or even deletion of data. SQLI attacks threaten the confidentiality, integrity, and availability of data. SQLI remains one of the most common and dangerous weaknesses in web applications.

- As found in the 2023 Verizon Data Breach Investigations Report (DBIR), SQL injection attacks represented approximately 24% of all web application attacks and is also one of the most frequently employed attack vectors. SQLI is often used in combination with other forms of injection-based attacks in many reports, but it always takes first place on their lists [7].
- SQLI is still present in the list, one of the OWASP Top 10 vulnerabilities. Even though SQLI is no longer present in the OWASP update that came in 2023, it falls under the header of "Injection" type attacks, which ranks in at 3rd [8].
- According to a report by Imperva Research Labs published in 2023, their sensors have reported about 8% of the total attempts of web application attacks that are SQLI attacks. Daily cases show that thousands of SQLI attempts are targeted towards businesses, especially if they are exposed or have relatively weak security postures [9].
- The result could be a financial loss running into billions. As observed by IBM Security and Ponemon Institute, SQLI is amongst those data breach expenses were $3.5 million on an average was seen in spending on each of these due to databases compromise and subsequent loss in terms of data theft followed by restoration process [10].

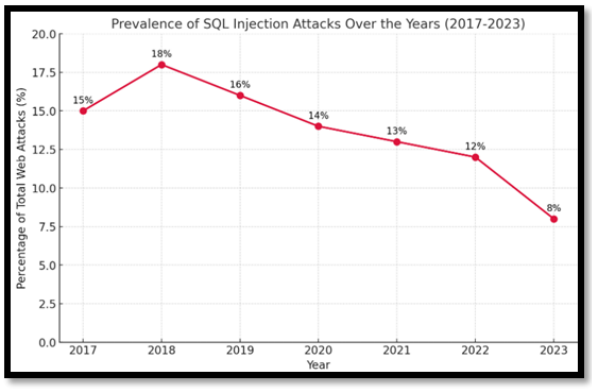

Figure 1: Graph showing the prevalence of SQL injection (SQLI) attacks as a percentage of total web application attacks from 2017 to 2023.

Therefore, with most web applications playing an important role in today's digital world wherein many services and transactions rely on them, cybersecurity is imperative and a concern for organizations and individuals. Due to this, their susceptibility to cyber threats was mainly because web applications store sensitive information in database backends.

It is a database that requires security since it contains sensitive information related to customers, finance, and intellectual property. Therefore, it involves loss of money and reputation, legal consequences, and violation of privacy. Thus, an attacker usually breaks into the database to get substantial, useful data for committing fraudulent identity theft, fraud activities, and ransomware attack. In such cases, proper implementation of controls for security will ensure the integrity, confidentiality, and availability of data.

There is an increasing number of smartphones, IoT gadgets, and laptops that multiply entry points for attackers. A new vulnerability arises every time a new device connects. Weak links in security across these endpoints may compromise an entire network or database. As more and more organizations and users become reliant on interconnected systems, there is a greater demand to have comprehensive database security measures in place to keep evolving threats at bay. For over two decades, SQL injection attacks have remained one of the most frequent and damaging threats to web application security, mainly because it injects malicious SQL code into user inputs to make manipulations and compromises of databases, thus posing a massive challenge.

The pie chart below (Figure 2) shows the distribution of various types of web attacks by frequency. SQL Injection (SQLI) accounts for the highest percentage of 24%, followed by DDoS with 22% and Cross-Site Scripting at 17%. The chart allows one to view visually the relative prevalence of each type of attack in relation to the others.

The bar graph (Figure 3) of the mean financial damage caused by each type of attack in millions of USDs shows that SQLI is ranked first with a mean damage of $3.5 million per breach. Other Injections ranked second at $3.0 million. This therefore indicates that some attacks give businesses more severe financial blows.

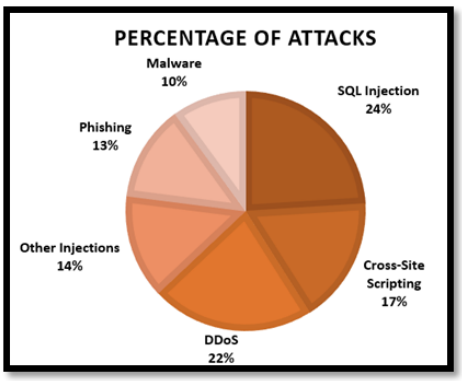

Figure 2: Representation the percentage of each attack type in terms of frequency.

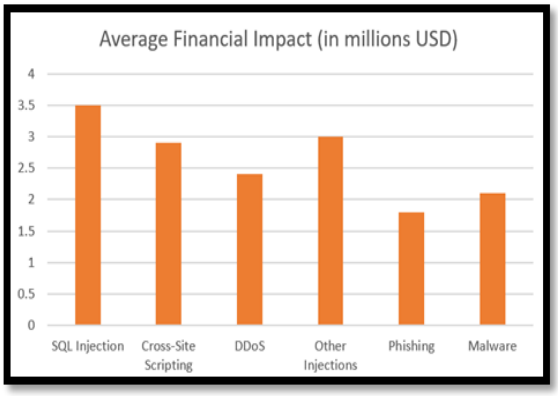

Figure 3: Graph shows the average financial impact of each type of attack, measured in millions of USD.

1.1 Steps to Perform SQLI

SQLI is injection of malicious code through SQL to exploit flaws in the interaction between an application and its database. It provides a general overview of the general structure and steps as follows:

1. Identify Input Points: Identify places where user inputs occur, such as login forms or URLs that interact with the database.
2. Create Malicious Payloads: Use SQL syntax to manipulate the query.
3. Vulnerability Test: Test if the application executes the payloads without validation; if it does, then it must be vulnerable.
4. Extract Data: Using the weakness extract data using UNION, subqueries, or Boolean-based conditions.
5. Bypass Authentication or Execute Commands: Access restricted areas or perform database commands using SQLI if applicable.

1.2 Effects of SQL Injection Attacks

SQLI exploits vulnerabilities in the web application code to allow hackers to tamper with the back-end databases, injecting malicious SQL queries. Such attacks may have a huge impact and financial implications-which may include:

- ***Data breaches***: Attacker might steal confidential information, such as customer data, financial records, and proprietary business information.
- ***System compromise***: Expert attackers can create unauthorized access to the underlying operating system. This can also lead to an entirely controlled web server and all other systems that are connected to the same network.
- ***Data manipulation***: Hackers can manipulate, delete, or otherwise alter database records, cause business disruption, and potentially irreparable damage.

SQLI vulnerabilities can cause even worse problems other than just data breaches. Once attackers inject SQL to gain unauthorized access to the underlying operating system from the database, they can also run most inappropriate scripts to transform the server into a "bot" that becomes part of a bigger botnet. These botnets can then be used for DDoS attacks, wherein the compromised server is utilized with many others to flood a target with excessive traffic and disrupt its operations. This attack chain also illustrates the risks of SQLI, because such a threat can have many influences that go well beyond the primary issue of data access as it can affect the whole network.

Wide dependence on web applications for critical services increases the impact of SQLI attacks. OWASP has consistently ranked SQLI at the top level of security threats to web applications, and this ranks very high for different global purposes [1].

1.3 General Steps for Securing Web Applications

Now, to minimize the risks caused by SQL Injection attacks, organizations must implement strong security practices in developing and maintaining their web applications.

In order to safeguard a company's web applications against attacks by SQL Injection or similar vulnerabilities, the application should have a solid layer of security measures. For example, input validation and sanitization are key: User input should then adhere to predetermined patterns through input restrictions by accepting only specific characters allowed- alphanumeric-only characters in input fields-and through application of character encoding so no attack could enter bad SQL statements into fields. For example, in email validation, accept only the characters which match the format of an email and reject those characters which do not match the pattern defined.

Secondly, the parameterized queries and prepared statements are very important for making database interactions secure. The method is by separating the SQL code from user input, thus preventing the attacker from modifying the structure of the query. Some languages, such as Java and Python, have an option to use bound parameters rather than directly inputting information from users within the SQL statements. For example, if preparing a Python statement, this will ensure the username variable is not interpreted as part of the SQL structure but rather as data that is resistant to the injection attacks.

Input validation and prepared statements are best practices that support secure coding. Avoid the use of dynamic SQL where you directly input unfiltered user input and avoid the usage of error messages as much as possible, which is the primary rule in secure coding. A generic error message should mask the details of the system or database an attacker might utilize. For example, a response to "An error occurred" masks the SQL errors for a user while logging details that an internal auditor can use for analysis.

Finally, periodic vulnerability and penetration testing will detect and correct weaknesses before the attacker can exploit them. Scanning tools for known vulnerabilities include OWASP ZAP and Nessus. While manual penetration testing provides a broader view of possible entry and misconfigurations, frequent testing with access controls reviews ensure that applications remain unbreachable against emerging risks. The above best practices reduce the opportunity to be attacked by SQL injection attacks and ensure sensitive information is not compromised.

If these practices are diligently followed, the probability of successful SQL Injection attacks decreases, and sensitive data is protected from compromise [2].

1.4 Methodology

The methodology for the paper is based on reviewing various types of SQL Injection (SQLI) attacks and assessing the effectiveness of various detection and prevention mechanisms. Here is a brief breakdown:

- The study involves a comprehensive literature review of existing SQLI techniques, detection tools, and defensive strategies. This provides foundational knowledge on types of attacks (Union-based, Blind, Time-based) and insights into security flaws in applications.
- There is classification of SQLI based on technique, attacker goal, and exploited vulnerability. The following section will then come up with a taxonomy on how the nature of attacks vary and their impacts accordingly.
- The approach is to make a review of available detection approaches, like signature-based, anomaly-based, behavior-based, and intrusion detection systems. Every type of tool that has been used as part of detection is analyzed and taken into consideration in regard to its effectiveness and applicability in real-world instances.
- This paper discusses major prevention techniques, including input validation, prepared statements, web application firewalls, and secure coding practices. A comparative analysis of their pros and cons is done.
- This paper researches into more advanced machine learning-based solutions such as hybrid ANN-SVM models, deep neural networks, and grammar-based detection techniques. How these mechanisms work, the benefits they provide, and specific use cases for preventing SQLI attacks are explained.

- The SQL and NoSQL databases comparison makes it possible to show vulnerabilities and protective measures in two systems, which will clearly determine which system is better against SQLI attacks.

This structured methodology allows the study to holistically assess SQLI threats and suggest effective defensive measures.

2. Literature review

SQL Injection, also known as SQLI, is the most well-known flaw in web security. The hacker can inject detrimental SQL code into a web application because SQLI undermines availability, confidentiality, and integrity of data in a web application, and hence it falls within one of the Web's serious security threats. SQL injection vulnerabilities allow unvalidated forms of user input to be injected into SQL queries without being validated or sanitized. Some of the manipulations gain access to confidential information or by-pass authentication procedures, other than modifying existing data. It causes denial of services too. SQL injection attacks or SQLI is one of the basic types of security flaws that confront databases and web applications. But generally, it refers to the attack type based on how to exploit weaknesses in the application code by making arrangements to access, change, or delete data by manipulating SQL queries. Paper revisits the mechanics behind SQLI attacks, an impact on data availability, integrity, and secrecy, and then discusses prevention.

2.1 Effect on Confidentiality, Integrity, and Availability

The impact of SQLI attacks on the CIA triad is quite profound. Alhazmi and Alshahrani (2022) point out that SQLI attacks have catastrophic violations of confidentiality where attackers gain access to sensitive data about the users, including financial information as well as personal identifiers. There are also risks of data integrity since attackers can modify or delete entries to tamper with databases.

Moreover, SQLI attacks may have an impact on the availability of services. According to Gupta and Malhotra (2024), where DoS attacks may simply be launched by loading databases with malicious queries, thereby denying the service in order to affect business operations.

2.2 Importance of Addressing SQL Injection Vulnerabilities

Of course, today, SQLI is perceived as one of the biggest threats in terms of organization vulnerability, and thus, finding a solution to this challenge is a must. Not to mention today's impact that SQLI attacks could cause, including financial losses, reputation, and legal liabilities, the study hereby has seen estimations and calculations that will reveal even minor data breaches by SQLI attacks result in millions of dollars lost on instant finances as well as long-term brand erosion and loss of customer trust [1][3].

In addition, SQLI attacks often lead to access for further cyber-attacks that enable cyber-thieves to gain further exploits into the organization's network and even breach main systems [3]. This capability makes SQLI a foundational attack that actually enables later attacks; therefore, the urgency of remediation of these vulnerabilities is timely.

SQLI vulnerabilities often have long-range consequences in terms of severe regulatory fines, public scrutiny, and even increased business partner watchfulness. For instance, if an organization fails to protect sensitive information via SQLI attacks, it may possibly be violating its legal and contractual obligations in regard to the protection of data, based on regulations like GDPR or HIPAA, which require protection against unauthorized access. Therefore, organisations must continue strengthening their security measures with time, as threats grow.

To balance the prevention with detection, SQLI risks should be managed through a proactive approach involving prevention mechanisms along with the responsive strategies of detection [4]. The developers of the applications should focus on the development of secure applications but should also conduct regular vulnerability assessments and penetration testing that may help identify any probable SQLI flaws before attackers exploit them. This vigilance would give organizations a strategic advantage as regards the emergence of new threats and allow them to adjust their defenses accordingly.

The organizations must have a security-aware culture among its members. The developers and the staff must be trained regarding the risks posed by SQLI vulnerabilities and best practices to prevent the same. It has been noted that the majority portion of SQLI vulnerabilities are caused due to ignorance or a lack of knowledge about safe coding [2]. The reason is that the organizations considerably reduce their vulnerability to such attacks after equipping teams with relevant skills and tools to detect and mitigate SQLI risks.

As such, SQL Injection vulnerabilities, to a great extent, are not something organizations would ever consider to address from the financial viewpoint right away. In fact, it is a question of the foundations upon which lies the integrity of the organization, in staying compliant, and the ability to sustain anti-cyber threats over time. Therefore, the way to effectively protect organizational assets is through prevention, detection, and education in this kind of SQLI threat management.

3. Classifications and Taxonomy of SQL Injection Attacks

SQLI attacks are categorized in various ways: technique aims from the point of view of attackers, and vulnerabilities. Here's a taxonomy based on sources:

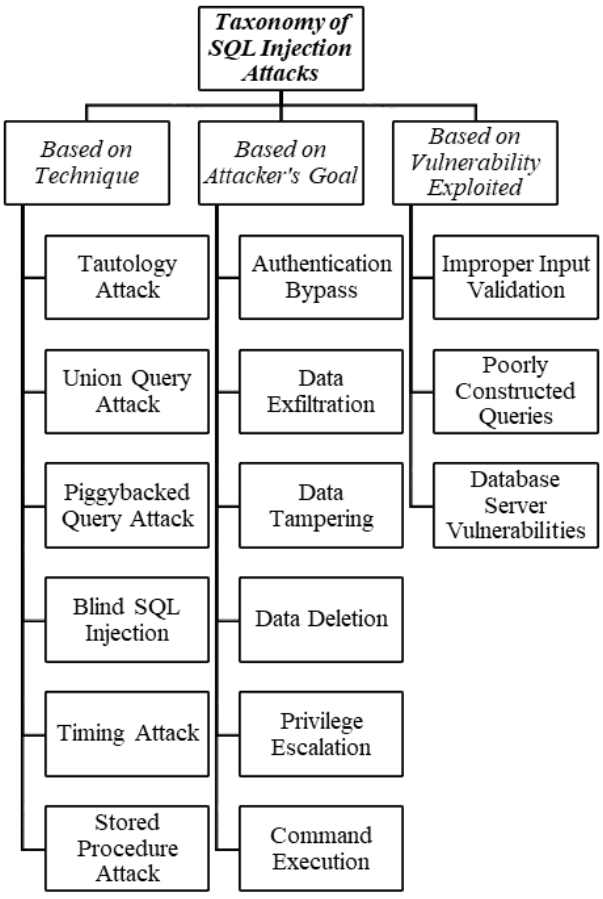

Figure 4: Taxonomy of SQL Injection Attacks: Categorization Based on Technique, Attacker's Goal, and Vulnerability Exploited

3.1 Based on Technique

Tautology Attacks [1], [2], [4], [6]: This is the exploitation of SQL syntax to create conditions that always evaluate to true, which easily allows the attackers to by-pass the authentication or other security checks. Example, An attacker might insert admin' OR '1'='1 into a username field, turning an SQL query to one that always evaluates to true, for example, SELECT * FROM users WHERE username = 'admin' OR '1'='1'.

Union Query Attacks [1], [2], [6]: These attacks use the UNION operator to join results obtained from legitimate queries with evil ones, offering the attacker a path into other table data. Example, If the search query is run as SELECT * FROM products WHERE name = 'input_name', then attacker can use input_name' UNION SELECT username, password FROM users--, that leads to leakage of private data.

Piggybacked Query Attacks [1], [2], [4]: Malicious SQL statements are appended to innocent queries using semicolons (;) so that the attacker can run multiple commands in a single request. Example; An attacker may input feedback'; DROP TABLE feedback; -- in a feedback form, resulting in the erasure of all the table in the feedback after submitting his/her comment.

Blind SQL Injection [1], [2], [4], [6]: Blind SQL injection attackers do not get immediate feedback from their applications but deduce information concerning the structure and content of the database by analyzing statements and the application behavior. Example, an attacker could enter input' AND (SELECT SUBSTRING (password,1,1) FROM users WHERE username='admin') = 'a' and decide if the first character of the password is 'a' by observing the application's reaction.

Timing Attacks [1], [2]: In timing attacks, the application's response time reveals secret information about the database. Such attacks cause delays based on true/false conditions and allow an attacker to decide if the data exists or does not exist. Example, An attacker could input' IF (SELECT COUNT (*) FROM users) > 0 WAITFOR DELAY '00:00:05' -- to check if the response time is longer, which would indicate the condition was true.

Stored Procedure Attacks [1], [5], [6]: The stored procedure attack is one form of attack that takes advantages of vulnerabilities that exist within the stored procedure. Stored procedures are precompiled SQL statements which are contained inside a database and may later be executed. For instance, A stored procedure does not validate input coming to it. Therefore, an attacker will be capable of designing attacking parameters and invoking the process that makes unauthorized data and even further modification of the same possible.

3.2 Based on Attacker's Goal

Authentication Bypass [1], [4], [6]: The goal is to obtain unauthorized access by bypassing some form of authentication mechanism. These are typically tautology or union query attacks.

Data Exfiltration: The goal is to retrieve sensitive information, such as user credentials or private data. This is mostly a union queries or a blind SQL injection.

Data Tampering [1]: The goal is to update records that are already available in the database which may change core information or confuse end-users.

Data Deletion [1], [4]: The goal is to delete data from the application and hence its users will be brought out of action. This can be achieved by malicious functionality in a query that is piggybacked onto an innocuous transaction.

Privilege Escalation [1]: The goal is to gain elevated levels of privilege within the database system; then to exploit these privileges and do things outside the original level of access.

Denial of Service (DoS) [1]: The goal is to overload a database through too many queries so that such a database is not accessible for genuine users.

Command Execution [1], [2]: The goal is that the attack will make use of OS commands for the exploitation of vulnerabilities of the database, and the attackers will obtain the access or perform harmful operations of the system at a deeper level.

3.3 Based on Vulnerability Exploited

Improper Input Validation [1], [2], [5], [6]: An absence of proper sanitization of user inputs can lead to SQL injections. In applications failing to validate or sanitize an input, malicious SQL statements are processed.

Poorly Constructed Queries [5], [6]: Use of dynamic SQL queries that directly concatenate user inputs into a constructed SQL statement without proper input escaping or whitelisting render it vulnerable to manipulation.

Database Server Vulnerabilities [1], [2], [4], [5]: The actual exploit is based on weaknesses in the database server software itself, for example, implementation flaws of database functions or configuration errors.

4. Detection Methods and Tools

Detecting SQLI attacks involves identifying suspicious patterns in user inputs, network traffic, or database logs. Here's a table summarizing the various SQL injection detection methods and tools, along with their overviews, mechanisms, advantages, and disadvantages:

Table 1: Inference for SOL injection detection methods and tools.

Detection Method/Tool	Overview	Mechanism	Advantages	Disadvantages
Signature-Based Detection [1], [4], [5], [6]	Relies on a database of known SQLI attack patterns.	Detects specific keywords or patterns in incoming requests and blocks or alerts when matches are found.	High accuracy for known threats, low false positive rates.	Limited against novel or unknown attacks, can be bypassed by modifying payloads.
Anomaly-Based Detection [1], [5], [6]	Establishes a baseline of normal behavior for applications and databases.	Monitors deviations from the baseline, such as unusual query structures or access patterns.	Identifies unknown attacks by detecting unusual patterns.	Higher false positive rates; requires continuous adjustment of baseline.
Behaviour-Based Detection	Analyzes user actions and interactions to spot suspicious activities.	Tracks user behavior and compares it to expected norms to raise alerts for unusual actions, such as repeated failed logins.	Contextual analysis of user behavior helps identify subtle SQLI attempts.	Resource-intensive for data analysis; challenging to differentiate between legitimate and suspicious actions.
Log Analysis [5]	Examines server logs for evidence of SQLI attacks.	Reviews logs for unusual query structures, error messages, and access attempts to sensitive data.	Provides insights into past interactions; can identify attack patterns over time.	Time-consuming and may require advanced tools; reactive rather than proactive.
Intrusion Detection Systems (IDS) [1], [5]	Monitors network traffic for malicious activities, including SQLI attempts.	Uses a mix of detection techniques (signature, anomaly, behaviour) to identify and respond to potential attacks.	Real-time monitoring and alerting enable rapid responses to detected threats.	May produce false positives; signature-based systems can miss novel attack techniques.
Web Application Firewalls (WAFs) [5]	Protects web applications from various attacks, including SQLI.	Filters incoming traffic based on rules, analyzing HTTP requests and blocking potentially malicious inputs.	Tailored protection for web applications; adaptable through rule updates.	Requires proper configuration; overly strict rules may lead to false positives.

Penetration Testing [1], [2], [4]	Simulates real-world attacks to identify vulnerabilities in web applications.	Ethical hackers use various tools and techniques to find and exploit vulnerabilities; helps organizations remediate weaknesses.	Provides a comprehensive understanding of security posture; identifies areas for improvement.	Resource-intensive and requires specialized skills; results can be time-sensitive.

5. Advanced ML Techniques for SQL Injection Detection and Prevention

Several solutions have been designed to counter SQL injection attacks, presenting various mechanisms, advantages, and limitations. In the next subsection, we briefly discuss some of the most significant solutions through references given by the sources, hence mentioning their mechanism and advantages. At the same time, we stress that with time, hybrid techniques are more in demand that enhance the capability of detection and prevention from SQLI attacks.

5.1 Hybrid ANN-SVM [1]

Mechanism: In this hybrid approach, it utilizes the pattern recognition characteristic of ANNs and the classification power of SVMs in the process of recognizing SQLI attacks. ANNs learn about big amounts of data to detect intricate patterns, whereas SVMs classify the data points based upon their decision boundaries.

Advantages:

- High Precision: The hybrid of ANN with SVM would therefore provide high precision and recall rates. This hybrid system will be more accurate at differentiating malicious and benign queries as compared to a single model.
- Flexibility: ANNs can be trained over time with new information, while SVMs adapt quickly to classify new threats, thus ensuring continued effectiveness against SQLI.
- Complete Coverage: ANNs are very sensitive to identifying minute patterns, whereas SVM handles specific classifications that give the best possible detection of a variety of types of SQLI.

Use case: This hybrid ANN-SVM would be helpful for organizations which deal with vast amounts of data and where the complexity of pattern along with accuracy in classification is inherent.

5.2 Grammar Pattern Recognition and Access Behaviour Mining [11]

Mechanism: This solution examines the SQL query grammar and structure with constant observation of user access patterns. It will detect anomalies in the already set SQL grammar rules as well as arbitrary access behaviors that may turn out to be the signature of SQLI attacks.

Advantages:

- Grammar-Based Detection: The system focus on SQL syntax properly recognizes anomalies even during advanced SQLI attempts.
- User Behavior Insights: By monitoring access, the system can identify anomalies that might signify unauthorized access and even data exfiltration attempts.
- Advanced Security Context: It entails grammar and behavioral analysis to give a holistic view of the structure of the query and user intent.

Use Case: It would be apt for applications wherein regular user interactions entail that detection accuracy may improve with the analysis of patterns of behavior and syntax of the query.

5.3 Deep Neural Network Detection [1]

Mechanism: DNNs are trained on large data sets and learn to identify complex query patterns and SQLI attempts. DNNs can pick up deep features from queries that improve detection as they would be able to detect hidden associations in data.

Advantages:

- Feature Extraction: The DNN is very good at picking even minute details present in the SQL queries and thus is very effective against advanced SQLI attacks.
- Reducing False Positives: DNNs may cut down the false alarms since they understand deeper query relationships and would distinguish the benign queries much more accurately.
- Scalability: DNNs are suitable for large-scale environments where SQLI patterns evolve continuously, thus providing long-term adaptability.

Use Case: DNN-based systems are ideal for organizations involved in high-volume data and require an accurate, low-latency detection mechanism.

5.4 Proxy Filtering and SQL Injection Prevention and Auditing (SQLIPA) Techniques [2]

Mechanism: Intrusion using proxy filtering monitors and analyzes network traffic for any harmful request, while using SQLIPA techniques inspect requests for existing signs of SQLI. With them, it is implemented at the two different security layers.

Advantages:

- Layered Security: Since analysis of traffic and query inspection take place simultaneously, SQLI attacks are detected before reaching the database.
- Real-Time Analysis: Proxy filtering offers immediate alert to any suspicious requests and reduces damage in cases of intrusion.
- Proactive Prevention: Through SQLIPA methods, since SQL inputs undergo analysis, the SQLIPA methods can prevent malicious content that may lurk in the queries from reaching the application.

Use Case: The methodology is aptly suited for organizations that, regarding real-time detection and prevention, have high traffic cases.

6. Benefits Hybrid Techniques for SQLI Detection and Prevention

Hybrid techniques [1] for SQLI detection and prevention seem to protect better against SQL injections since they take the best out of multiple techniques. Here are some benefits:

Hybrid solutions, such as the combination of ANN-SVM, increase precision because of complementary strengths of pattern recognition with high precision in classification. This means higher detection accuracy since more than one technique examine every query from a different angle.

Hybrid systems also decrease false positives by cross-validating results from different techniques. For instance, an anomaly that is detected by ANN might be confirmed by SVM to ensure that genuine threats are flagged only.

Improvisation against Emerging Threats: Hybrid systems can include novel detection methods and can track SQLI tactic shift. Since multiple algorithms are in play, hybrid systems prove much more resilient against emerging attack patterns with time.

Better Coverage of the Threat Detection: Hybrid systems realize the potential of numerous techniques being used along with one another. These systems may even detect simple tautology-based SQLI attacks as well as complex, timing-based ones. This sure enough provides better protection in a scenario of emerging threats.

Cost-effectiveness: Hybrids, as complex as they are, save money in the long run on the incidents' cost of security through the prevention of SQLI attacks; this creates savings that grow over time as accuracy and decreased false positives increase into efforts in response to remediation of the incidents.

In fact, hybrid approaches outweigh the disadvantages of stand-alone detection techniques due to strong tools against SQLI attacks. Organizations can select a technique that is suitable for their protection needs and capitalise on the strengths of pattern recognition, behavioral analysis, and real-time filtering to protect their applications and databases effectively.

7. Using AI & ML in Detecting SQLI Attacks

The impact of AI as well as the recent technological impacts on the detection and prevention of SQL injection attacks is greatly important, as both attacker and defender techniques evolve. As in the case of SQL injection, an attack is related to exploiting weaknesses in the application. An attacker introduces some malicious SQL code which modifies databases by using any existing SQL statement. So this is a serious danger regarding data integrity and its confidentiality. This calls for adequate mechanisms of detection as well as prevention.

7.1 Use of ML: Defender's Perspective

From the defender's point of view, AI and ML have emerged as the most powerful tools for enhancing the detection of SQL injection attacks. A large number of researches have been conducted to demonstrate the effectiveness of ML algorithms in vulnerability detection. For instance, Thalji highlights the importance of AI-based security measures, emphasizing that deep learning models can effectively detect SQL injection attempts by analyzing patterns in database queries [19]. Similarly, Alghawazi et al. provide a systematic review of machine learning techniques that have shown promising results in SQL injection detection, indicating a growing body of research focused on this area [13]. Moreover, Abdulmalik discusses an improved detection model that utilizes semantic feature extraction to enhance the identification of SQL injection attacks, showcasing the potential of advanced ML techniques in this domain [12].

Recent advancements in deep learning have further revolutionized SQL injection detection. Zhao et al. propose a deep-learning-based method that leverages complex neural networks to identify injection attacks, demonstrating the capability of these models to handle large datasets and complex patterns [20]. Additionally, Chen et al. introduce a support vector machine (SVM) approach that compares parse trees to detect anomalies indicative of SQL injection, illustrating the diverse methodologies being employed to combat these threats [14]. The integration of grammar pattern recognition and access behavior mining, as discussed by Gao et al., also highlights innovative strategies that utilize AI to enhance detection capabilities [15].

7.2 Use of ML: Attacker's Perspective

Attackers are continuously evolving their techniques to bypass these advanced defenses. As SQL injection remains one of the most prevalent vulnerabilities, attackers exploit weaknesses in input validation and parameterization [18]. This cat-and-mouse dynamic necessitates ongoing research and development of more sophisticated detection and prevention systems. For instance, Habibi emphasizes the need for comprehensive monitoring of database access to mitigate the risks associated with SQL injection attacks [16]. Furthermore, the use of automated testing and input generation techniques illustrates how defenders can proactively identify vulnerabilities before they can be exploited [17].

In conclusion, the interplay between AI technologies and SQL injection attack prevention is a critical area of research. While defenders leverage machine learning and deep learning techniques to enhance detection capabilities, attackers continuously adapt their strategies to exploit vulnerabilities. This ongoing evolution underscores the necessity for robust security measures and innovative approaches to safeguard against SQL injection threats.

7.3 Advantages of Using AI & ML in Detecting SQLI Attacks

Deep learning models are able to achieve higher accuracy than traditional rule-based techniques for the detection of SQL injection attacks. By learning large datasets consisting of benign and malicious queries, machine learning algorithms are able to identify subtle patterns and anomalies characteristic of an attack as bypassing conventional security measures.

Hybrid approaches, such as the one combining ANN with SVM, are found to be well performing in achieving a precision and recall value of more than 99%, and hence higher than the other ML approaches.

Machine learning models are particularly quite malleable to the ever-innovating malware threats that exist with SQL injection attacks. When new patterns of attacks are proposed, newer information could be fed into the model, and it becomes capable of identifying attacks that have never been seen before. This will ensure that the bad guys - the attackers - are kept at bay as they always come up with new ways of circumventing security measures.

Machine learning-based detection systems can automate the process of identifying and responding to SQL injection attacks, thus reducing the need for manual intervention. These systems also work with heavy loads of data and traffic and therefore would be best for the protection of web applications with large user activity.

7.4 Disadvantages of Using AI & ML in Detecting SQLI Attacks

The performance of machine learning models depends much on the quality and representativeness of the training data. A model would result in a poor performance if the training data is incomplete, biased, or includes incorrect labels. Source mentions that getting high-quality datasets that include known SQL injection samples and regular requests is crucial to acquiring accurate results in the detection.

Training complex machine learning models consumes significant computational resources, such as processors, memory, and storage space. According to the source, some algorithms are now very demanding when hybrid approaches are used; thus, most require a lot of data and higher computational power than traditional machine learning methods.

Although the machine learning could produce several successful predictions, it is not 100% foolproof. The possibility of false positives is always present, meaning that legitimate queries are flagged as attacks; the possibility of false negatives, where malicious queries go undetected. False positives can lead to blocking valid user requests, which in turn might result in a bad experience for users and could develop into any form of business-related operations into disruptions. False negatives could mean that successful attacks may compromise the integrity of data. It is very crucial that both false negatives and positives are minimized within the machine learning-based system for the proper detection of SQL injection attacks.

8. Overview of Common SQL Injection Prevention Solutions

Knowing all the variations of strengths and weaknesses that come with several SQL injection (SQLI) prevention techniques would help develop a holistic, effective security plan. Here is some detailed overview of the commonly used solutions, along with their advantages, disadvantages, challenges, and limitations:

8.1 PHP Data Object (PDO) and Prepared Statements [4]

Advantages:

- Improved Security: It keeps the logic of the SQL query separated from user input, thereby making SQL injection impossible to be injected.
- Precompiles SQL Queries: It disables SQLI attacks because SQL statements are prepared before the SQL statement is executed.
- Database Abstraction Layer: This layer promotes portability because it's easier to translate code from one database to the other.

Disadvantages:

- Less Flexible Flexibility: It does not allow SQL identifiers, phrases, or keywords to be used as an argument within prepared statements; hence, this reduces flexibility.

Challenges:

- Training Needs: The developers who are not aware of PDO and prepared statements need more training before they can utilize these.

- Implementation Risk: Failure in the implementation may let some residual vulnerabilities nullify the security benefit actually desired to be achieved.

Limitation:

- Not a Silver Bullet: The advantage is purely dependent on the implementation throughout the application.
- Incompleteness in Prevention of Attacks: Does not prevent all SQLI attacks if implemented in a wrong or varying manner across all queries.

8.2 Input Validation and Keyword Filtering [1], [2], [5], [6]

Advantages:

- Ease of Use: It is one approach to making inputs secure that may repel very simple attempts at SQLI.
- Counter Effectiveness Against Popular Methods: Catches most forms of injections that rely on the existence of some keyword or special character.

Disadvantages:

- Evasion Susceptibility: Susceptible to evasion attempts such as encoding or obfuscation.

Challenges:

- Keyword Selection: It requires careful selection of keywords and characters for filtering in such a way that it does not interfere with the legitimate user input.
- False Positives: Chances of declining valid inputs that contain words or characters blocked.

Limitations:

- Prevention Scope: Restricted to known attack patterns and is not adaptive to new, sophisticated approaches.
- Neglecting Root Causes: It fails to handle the root cause of SQLI vulnerabilities because of misconstruction of queries.

8.3 A-Gap Weighted String Subsection Algorithm and Support Vector Machines (SVM) [1], [4], [11]

Advantages:

- Dynamic Classification: Classifies and blocks SQLI attacks based on the pattern of query strings using machine learning.
- Efficient Detection: Detects both known and unknown malware due to matching patterns and payloads.

Disadvantages:

- Computational Cost: Many SVM methods are computationally expensive and inefficient with large datasets.
- Training Data Requirements: It requires an enormously large dataset to work effectively for classification.

Challenges:

- Large Training Set: It involves building a large training database that contains all possible SQLI attacks.
- Accuracy-Cost Tradeoff: It focuses on optimizing both accuracy and the cost of computation.

Limitations:

- Training Quality: It depends on good quality and diversified training data so that it can work effectively and achieve high accuracy.
- Unfamiliarity with Patterns: It will not be able to identify SQLI patterns that deviate significantly from those in its training set.

8.4 EQA - Encoding Query Based Lightweight Algorithm [4]

Advantages:

- Lightweight and Efficient: This plugin is mainly designed against some SQLI, which includes Tautology, Piggybacked, and Comment attacks.
- Improves performance. Because it reduces HTTP request/response time as well as reduces the size of queries.

Disadvantages:

- Measuring Performance Routine: It infrequently captures performance measurements.
- Processing Overhead: For high-volume application, it will consume more processing.

Challenges:

- Integration Requirements: Viability depends on integration with various database systems and query structures.
- Substantially Not Interfering with Legitimate User Queries: This requirement is true for the encoding not to interfere with the legitimate user queries.

Limitations:

- Subset Coverage: It is mainly effective against a minimal range of SQLI attacks leaving certain vulnerabilities unaddressed.
- Incomplete coverage: Not completely covered up against all possible SQLI attack vectors.

Here is a table matching each SQL injection solution with the types of SQLI attacks it best prevents. Solutions appear in columns while attack types are in rows. An "✔" for each solution means it prevents some type of attack, and "✗" for an attack means it's generally ineffective against that type.

Tabe 2: Inference of existing systems

SQLI Attack Type	**PDO & Prepared Statements**	**Input Validation & Keyword Filtering**	**A-Gap Weighted SVM Algorithm**	**Encoding Query-Based Algorithm (EQA)**
Tautology Attacks	✔ - Prepares statements, preventing logic manipulation	✔ - Filters dangerous keywords like "OR"	✗ - Not focused on tautology patterns	✗ - Does not directly address logical alterations
Union Query Attacks	✔ - Isolates input, preventing query modification	✗ - May miss complex UNION patterns	✔ - Classifies query structures	✗ - Encoding may not prevent union combination
Piggybacked Query Attacks	✗ - May not block appended SQL commands	✔ - Blocks extra SQL keywords like "DROP"	✔ - Detects unusual chained queries	✔ - Encodes parts of query to prevent chaining

Blind SQL Injection	✔ - Separates inputs, preventing SQLI manipulation	✗ - Limited effectiveness on blind attacks	✔ - Analyzes responses for abnormal patterns	✗ - Encoding doesn't affect timing-based attacks
Timing Attacks	✗ - Doesn't address timing manipulations	✗ - Filtering doesn't impact response times	✔ - Can detect timing discrepancies	✗ - Encoding does not control response timings
Stored Procedure Attacks	✗ - Needs manual validation in stored procedures	✔ - Can filter dangerous commands	✗ - Limited if stored procedures are not covered	✔ - Encodes query elements, though not specific to stored procedures

9. Comparison of various Database Management System for Security Features

Security is among the topmost aspects of modern database management systems, as sophisticated data breaches and cyber threats have increased. In comparing the different databases currently available, such as SQL and NoSQL, represented by systems like Firebase, one can notice how these two kinds of databases are far from similar in their security architecture and features.

SQL-based databases, for example, MySQL and PostgreSQL, generally enforce strict security, which may include authentication, access control, and encryption of data. These databases often utilize clearly defined roles and permissions to implement access controls, which can be highly granular in order to restrict the activities that a user can undertake depending on the role they play [22]. Further, SQL databases generally offer sophisticated encryption standards for data both at rest and in motion, thus enhancing the confidentiality and integrity of the data [27]. For instance, Yenioğlu discussed the need for encryption at backup in SQL Server and how encryption algorithms may be used to safeguard sensitive information [27].

NoSQL databases, however, have a different paradigm for security. Zugaj stated that most NoSQL systems lack some of the security features, particularly those relating to access control and logging, which are crucial to forensic analysis [28]. There usually is a trade-off involving performance and security because such databases, by their design, can be flexible- perhaps at the cost of other things such as security because they favor scalability and rapid speed [25]. For example, Firebase does have some innate security measures, such as built-in authentication for users, data validation rules, etc. However, it still does not eliminate the need to implement layers of security that can mitigate common vulnerabilities and attacks such as injection attack [21].

Fahd et al. mention a comparative study on vulnerabilities of NoSQL databases identifying them to be insecure since they are susceptible to attacks through injection due to the use of flexible query languages [23]. Its vulnerability increases with a reason that there is no one standardized security protocol set over NoSQL systems and cause inconsistent security postures for NoSQL systems [26]. Since most NoSQL databases have limited or no access control mechanism at all, it goes without saying that there's a very huge likelihood that the unauthorized use of such information happens due to not being handled appropriately.

With this, NoSQL databases may be further secured by having additional security frameworks. An example is the cloud-based protection system proposed by Ghazi et al., wherein authentication and fine-grained authorization mechanisms are specifically designed for document-oriented NoSQL databases [24]. This would therefore address the security flaws in NoSQL systems by establishing a more organized security framework.

While SQL databases generally provide a mature, comprehensive security framework with well-established protocols for authentication, access control, and encryption, NoSQL databases like Firebase present unique

challenges. Their flexible architectures create vulnerabilities that require additional security measures. A multi-layered approach to security must be implemented while carefully weighing the particular needs of an organization with the security features each database system has to offer.

10. Future Work

Some potential research work in SQLI prevention and detection can be focused on these later directions:

Hybrid Model Improvement: Hybrid approaches, as already demonstrated in the hybrid of ANN with SVM, are quite effective. Other machine learning combinations, such as ensemble methods, might allow achieving even better accuracy and coverage in the detection of SQLIs.

Integration with Real-time Monitoring Systems: SQLI defense systems can be incorporated with real-time data processing and monitoring that make it possible to react instantaneously to the possible attacks, thus lowering the extent of successful exploits.

Usage of emerging strategies of SQLI: In reality, SQLI techniques are constantly upgraded by attackers. Future research may be towards designing intelligent systems that can learn and change their strategy based on the changing threats without considerable manual re-configuration processes.

Scalability and Performance Optimization: Implementations such as deep learning and highly complex hybrid models are computationally expensive. Thus, the future work is to be optimized at deployment in the real world, in such a way that it balances security with system performance.

SQLI has been an ever-living challenge in cybersecurity; therefore, further research and innovation promise to enhance the capability to protect data from such attacks.

11. Conclusion

SQL injection attacks are one of the major security threats to web applications and databases. They attack the sensitive data within the targeted system or database through openings in the code of applications, either for manipulative purposes or information extraction. The several varieties of SQLI, such as union-based, blind, and error-based injections, share only one characteristic: the possibilities hackers use to break into a system are numerous.

Moreover, to safeguard organizations from those attacks, there has to be comprehensive preventive measures that include the use of WAFs, secure coding, parameterized queries, and scanning for vulnerabilities always. Advanced detection tools including anomaly-based and machine learning-based systems elaborate much more on providing protection by detecting unusual patterns in user inputs and query behaviors. With strong preventive and detection measures, organizations are likely to minimize the risks of SQLI attacks while strengthening data and systems' protection.

References

[1] W. B. Demilie and F. G. Deriba, "Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques," Journal of Big Data, vol. 9, Art. no. 124, Dec. 2022.
[2] R. Shobana and Suriakala, "A Thorough Study On SQL Injection Attack-Detection And Prevention Techniques And Research Issues," Journal Name, 2021.
[3] H. Gupta et al., "Impact of SQL Injection in Database Security," in 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dec. 11–12, 2019, Amity University Dubai, UAE.
[4] F. Q. Kareem et al., "SQL Injection Attacks Prevention System Technology: Review," Asian Journal of Research in Computer Science, vol. 10, no. 3, pp. 13-32, 2021.
[5] V. Abdullayev and A. S. Chauhan, "SQL Injection Attack: Quick View," Azerbaijan State Oil and Industry University, Azerbaijan, and ABES Engineering College, Ghaziabad, India, Jan. 10, 2023, Accepted Feb. 11, 2023.

[6] Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," in 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, Nov. 2020, pp. 1-4.
[7] Verizon. (2023). 2023 Data Breach Investigations Report (DBIR). Verizon Enterprise Solutions.
[8] OWASP. (2023). OWASP Top 10: 2023. Open Web Application Security Project.
[9] Imperva. (2023). Web Application Attack Report. Imperva Research Labs.
[10] IBM Security & Ponemon Institute. (2023). Cost of a Data Breach Report. IBM Security.
[11] H. Gao, J. Zhu, L. Liu, J. Xu, Y. Wu and A. Liu, "Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining," 2019 IEEE International Conference on Energy Internet (ICEI), Nanjing, China, 2019, pp. 493-498, doi: 10.1109/ICEI.2019.00093.
[12] Abdulmalik, Y. (2021). An improved sql injection attack detection model using machine learning techniques. International Journal of Innovative Computing, 11(1), 53-57.
[13] Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of sql injection attack using machine learning techniques: a systematic literature review. Journal of Cybersecurity and Privacy, 2(4), 764-777.
[14] Chen, Z., Guo, M., & Zhou, L. (2018). Research on sql injection detection technology based on svm. Matec Web of Conferences, 173, 01004.
[15] Gao, H., Zhu, J., Liu, L., Xu, J., Wu, Y., & Liu, A. (2019). Detecting sql injection attacks using grammar pattern recognition and access behavior mining.
[16] Habibi, Z. (2020). A review on the comparable analyses of sql injection: attacks, vulnerabilities, and their detection techniques. International Journal of Advanced Academic Studies, 2(1), 06-10.
[17] Harefa, J., Prajena, G., Alexander, A., Muhamad, A., Dewa, E., & Yuliandry, S. (2021). Sea waf: the prevention of sql injection attacks on web applications. Advances in Science Technology and Engineering Systems Journal, 6(2), 405-411.
[18] Kumar, B. and Anaswara, P. (2018). Vulnerability detection and prevention of sql injection. International Journal of Engineering & Technology, 7(2.31), 16.
[19] Thalji, N. (2023). Ae-net: novel autoencoder-based deep features for sql injection attack detection. Ieee Access, 11, 135507-135516.
[20] Zhao, C., Si, S., Tu, T., Shi, Y., & Qin, S. (2022). Deep-learning based injection attacks detection method for http. Mathematics, 10(16), 2914.
[21] Ahmad, K., Alam, M., & Udzir, N. (2019). Security of nosql database against intruders. Recent Patents on Engineering, 13(1), 5-12.
[22] Devara, S. and Azad, C. (2023). Improved database security using cryptography with genetic operators. Sn Computer Science, 4(5).
[23] Fahd, K., Venkatraman, S., & Hammeed, F. (2019). A comparative study of nosql system vulnerabilities with big data. International Journal of Managing Information Technology, 11(4), 1-19.
[24] Ghazi, Y., Masood, R., Rauf, A., Shibli, M., & Hassan, O. (2016). Db-secaas: a cloud-based protection system for document-oriented nosql databases. Eurasip Journal on Information Security, 2016(1).
[25] Hauger, W. and Olivier, M. (2018). Nosql databases: forensic attribution implications. Saiee Africa Research Journal, 109(2), 119-132.
[26] Saleh, M. (2019). Security testing tool for nosql systems. Journal of King Abdulaziz University-Computing and Information Technology Sciences, 8(1), 85-93.
[27] Yenioğlu, Z. (2022). Backup encryption and encrypted backup operation performance in sql server. International Journal of Pure and Applied Sciences, 8(2), 380-385.
[28] Zugaj, W. (2019). Analysis of standard security features for selected nosql systems. American Journal of Information Science and Technology, 3(2), 41.