THE USA JOURNALS
THE AMERICAN JOURNAL OF ENGINEERING AND TECHNOLOGY (ISSN – 2689-0984)
VOLUME 06 ISSUE10

PUBLISHED DATE: - 02-10-2024
DOI: - https://doi.org/10.37547/tajet/Volume06Issue10-03 **PAGE NO.: - 17-22**

RESEARCH ARTICLE **Open Access**

METHODS FOR PREVENTING SQL INJECTION IN IDENTITY AND ACCESS MANAGEMENT (IAM) SYSTEMS

Asha Seshagiri
Software Development Engineer 3 at Expedia, Austin Texas, USA

Abstract
This paper discusses methods for preventing SQL (Structured Query Language) injections in identity and access control (IAM) systems. SQL injections represent one of the most serious threats to web security, allowing attackers to gain unauthorized access to and modify data. The main security methods include filtering input data, using prepared statements and parameterization, implementing stored procedures, restricting access rights, and regularly updating software. Effective privilege management and database activity monitoring also play a key role in preventing attacks. The introduction of these measures helps protect confidential information, ensures reliable authentication and authorization, and maintains data integrity. The paper highlights the importance of an integrated approach to database security in the face of growing cyber threats.

Keywords SQL injection, programming, identity and access management systems, IAM.

INTRODUCTION

This topic remains highly relevant. Password and payment information leaks from databases, significant disruptions in the operation of web applications—these are some of the consequences of SQL injections. Even companies that prioritize cybersecurity, as well as their partners and clients, fall victim to attackers.

A vivid example is the attack on the IT company Kaseya in July 2021. On July 2, 2021, several managed service providers (MSPs) and their clients fell victim to a ransomware attack carried out by the REvil group, causing massive downtime for more than 1000 companies. Cybercriminals bypassed authentication, uploaded a payload to the VSA server, and used SQL injection to deploy malicious updates. As a result, more than 36,000 service providers could not access Kaseya's flagship VSA service for at least four days. The downtime led to significant losses [1].

In June 2023, over 200 organizations faced data breaches due to security flaws discovered about six weeks earlier in a popular file transfer program. This attack was triggered by an SQL injection vulnerability in the MOVEit file transfer program from Progress Software. Attackers gained access to files and credentials. Among the victims were schools in the USA, universities worldwide, and companies such as BBC, Boots, and British Airways. In total, 17-20 million people were affected by the breaches [2].

Therefore, the aim of this work is to examine methods for preventing SQL injections in identity and access management (IAM) systems.

1. General Characteristics of SQL Injections

SQL injection (SQLi) is one of the critical

vulnerabilities in web security, allowing attackers to interfere with queries that a web application sends to its database. This interference provides attackers with unauthorized access to data that is usually protected. Attackers can gain access to other users' information or other data accessible to the application. In some cases, the attacker can modify or delete data, causing long-term changes in the content or behavior of the application.

SQL injection vulnerabilities can be identified manually through systematic testing of each entry point into the application. Some common methods include:

- Introducing a single quote (') to identify errors or other anomalies.

- Using SQL-specific syntax to assess the underlying value of the entry point and identify differences in the application's responses.

- Applying logical conditions, such as OR 1=1 and OR 1=2, to detect differences in the application's responses.

- Using payloads designed to trigger time delays during the execution of an SQL query, which helps identify differences in response times.

- Employing out-of-band network interactions (OAST) intended to trigger interactions within the SQL query and track any resulting activities.

Additionally, most SQL injection vulnerabilities can be quickly and effectively detected using automated tools such as Burp Scanner [3]. The main consequences of an attack are reflected in Table 1.

Table 1. Consequences of Cyberattacks Caused by SQL Injection Vulnerabilities

Name	Description
Confidentiality	Since SQL databases often contain sensitive information, maintaining this confidentiality is one of the main concerns related to SQL injection vulnerabilities.
Authentication	If username and password checks are based on unreliable SQL commands, an attacker can gain access to the system under someone else's account without knowing the actual password.
Authorization	If authorization rights information is stored in the SQL database, successful exploitation of an SQL injection vulnerability can allow an attacker to alter this information.
Integrity	Just as confidential data can be accessed, it can also be altered or even deleted through an SQL injection attack.

Figure 1 further summarizes the types of SQL injections.

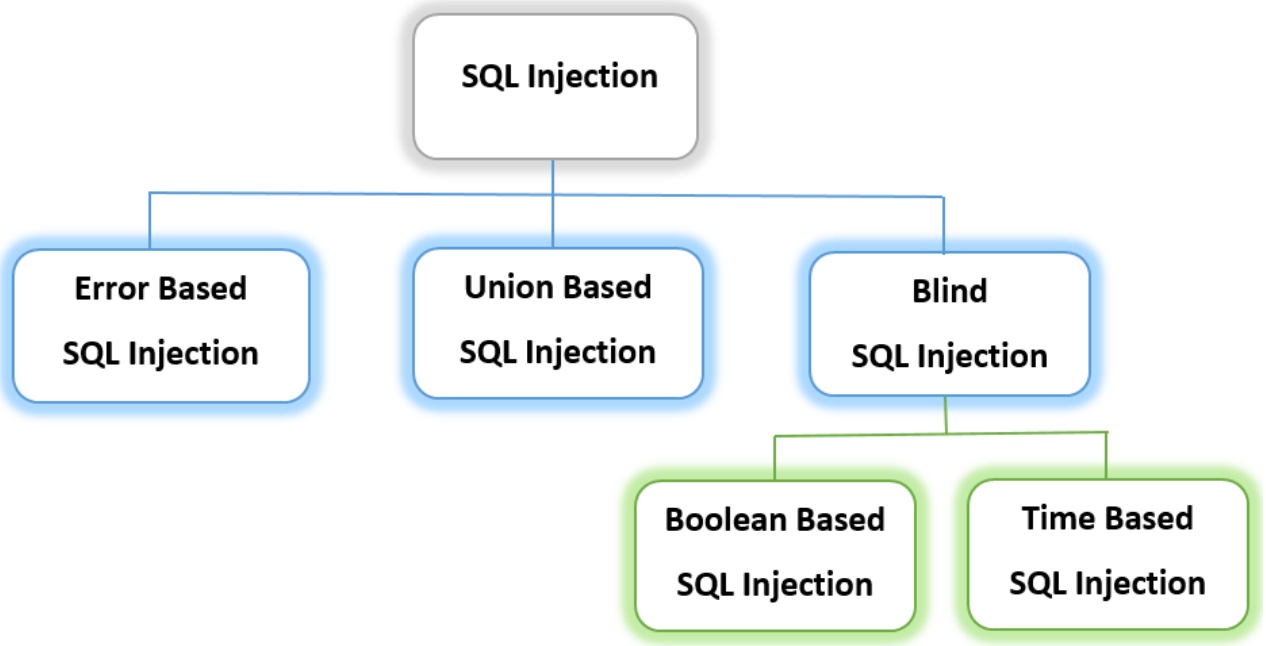

Fig.1. Types of SQL Injections [5]

From Figure 1, the following types of SQL injections are identified:

1. Error-Based SQL Injections: Error-based SQL injections allow attackers to extract information about the database structure through error messages generated by the database server. In certain situations, using this method alone, an attacker can gain access to the entire database structure.

2. Union-Based SQL Injections: Union-based SQL injections use the SQL UNION operator to combine the results of multiple SELECT queries into a single data set, which is then returned in the HTTP response.

3. Blind Boolean-Based SQL Injections: Boolean-based SQL injection works by sending an SQL query that forces the application to return different responses based on whether the query returns TRUE or FALSE.

4. Blind Time-Based SQL Injections: Time-based SQL injection works by sending a query that causes the database to wait a specified amount of time before responding. The response time allows the attacker to determine if the result is TRUE or FALSE. Depending on the result, the HTTP response will either be delayed or returned immediately. Even if no data is returned, the attacker can determine the truth of the query. However, this attack is slow, especially in large databases, as the attacker iterates through characters one by one [5].

2. General Characteristics of IAM

Identity and Access Management (IAM) is a crucial discipline in cybersecurity aimed at ensuring proper management of user access to digital resources. IAM systems are designed to prevent unauthorized access and ensure that each user is granted only the permissions necessary to perform

their duties.

The goal of IAM is to prevent hacker activities by allowing authorized users to perform their tasks securely and efficiently. IAM systems typically include four key components: identity lifecycle management, access control, authentication and authorization, and identity governance.

IAM initiatives contribute to cybersecurity, business process optimization, and regulatory compliance. IAM systems help centralize access management in the context of digital transformation, supporting secure access for various types of users to diverse resources. They simplify IT management and network administration, ensure compliance with regulatory requirements, and enhance data security by protecting against credential-based attacks [6,7]. Table 2 lists the main advantages of using IAM.

Table 2. Advantages of Using IAM

Advantage	Description
Unified Access to All Resources	Single Sign-On (SSO) simplifies the authentication process by allowing users to access multiple resources using a single set of credentials. This significantly reduces the number of required passwords and simplifies account management.
Centralized Management	IAM systems ensure precise assignment of privileges, allowing the granting of appropriate access rights to the right users. This enhances both efficiency and security.
Enhanced Security	Through centralized access control and strengthened authentication measures, IAM systems significantly increase the overall security level of the enterprise, preventing unauthorized access.

Next, we will consider the main methods for preventing SQL injection attacks:

1. Input Data Filtering: Input filtering is a fundamental method for protecting against SQL injections. It involves detecting and removing malicious code from user-inputted data. Malicious users can use complex URLs and special characters to execute commands and gain unauthorized access to the database.

2. Database Code Restriction: In addition to input filtering, it is necessary to restrict the code available to the database. This helps prevent the execution of undesirable queries and exploration of the database. It is important to reduce functionality, use stored procedures, and apply prepared statements and parameterization.

3. Database Access Restriction: To minimize the damage from potential attacks, database access should be restricted. This includes using firewalls to filter data, limiting user access rights, and encrypting data. Access restrictions should be implemented based on the principle of least privilege, which means using the minimum necessary access rights to perform tasks.

4. Updating Applications and Databases: Regularly updating software and databases is a key aspect of protecting against SQL injections. Vulnerabilities in applications and databases are regularly discovered and published. Organizations need to keep track of vulnerability news and promptly install updates and patches.

5. Monitoring Input Data and Data Exchange:

Continuous monitoring of SQL operators and database activity allows for effective detection and prevention of attacks. Using machine learning and behavioral analysis within privileged access management (PAM) systems and security information and event management (SIEM) systems significantly enhances protection [9].

3. Implementation of Procedures

One of the most effective methods for preventing SQL injection attacks is the implementation of stored procedures. The use of stored procedures not only improves application performance but also significantly reduces the risk of SQL injection attacks. This approach allows developers to clearly define acceptable SQL operators for execution, ensuring that the application does not execute arbitrary SQL code.

The main reason stored procedures prevent SQL injections is the clear separation of data and code. Instead of directly passing SQL operators to the database, stored procedures require developers to pass only the necessary data as parameters. These parameters are then processed within the stored procedure using predefined SQL operators, eliminating the possibility of injecting malicious code.

The process of implementing stored procedures can be divided into three main stages:

1. Creating the Stored Procedure: Developing an SQL script that defines the logic and parameters of the function. This script should adhere to the best practices of SQL programming to avoid potential security threats.

2. Granting Permissions: Properly managing access by providing the necessary permissions to execute the stored procedure. This ensures that only specific users or roles can execute the procedure, reducing the risk of malicious exploitation.

3. Calling the Stored Procedure: Updating the application code to call the stored procedure using appropriate parameters. It is important to ensure that these values are sanitized and validated before being passed to the stored procedure [10].

CONCLUSION

In conclusion, preventing SQL injections in identity and access management (IAM) systems is critically important for ensuring cybersecurity. The discussed methods, such as input data filtering, the use of stored procedures, prepared statements, and parameterization, are effective means of protection against SQL injections. Limiting database access and regularly updating software further strengthen system protection. These measures ensure reliable authentication and authorization, protect sensitive information, and maintain data integrity. Implementing a comprehensive database security approach significantly reduces risks and protects IAM systems from attacker exploits.

REFERENCES

1. The attack of the Kaseya VSA ransomware. [Electronic resource] Access mode: https://en.wikipedia.org/wiki/Kaseya_VSA_ransomware_attack (accessed 06/20/2024).
2. More than 200 organizations have become victims of violations related to information technology MOVE. [Electronic resource] Access mode: https://www.axios.com/2023/07/07/moveit-hack-200-target-millions-victims (accessed 06/20/2024).
3. Types of SQL Injection (SQLi). [Electronic resource] Access mode: https://www.geeksforgeeks.org/types-of-sql-injection-sqli/ (access date 06/20/2024).
4. What is identity and access management (IAM). [Electronic resource] Access mode: https://www.ibm.com/topics/identity-

access-management (access date 06/20/2024).

5. What IAM is and what it does. [Electronic resource] Access mode: https://www.microsoft.com/en-us/security/business/security-101/what-is-identity-access-management-iam (access date 06/20/2024).
6. Top Identity and Access Management Systems | IAM | Open Source | Enterprise. [Electronic resource] Access mode: https://medium.com/@devops.ent/top-identity-and-access-management-systems-iam-open-source-enterprise-92cf66560a55 (access date 06/20/2024).
7. How to Prevent SQL Injection: 5 Key Prevention Methods. [Electronic resource] Access mode: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/
8. How to Prevent SQL Injection Attacks: Essential Tips and Best Practices. [Electronic resource] Access mode: https://www.sql-easy.com/learn/how-to-prevent-sql-injection-attacks/ (access date 06/20/2024).