CRASHTEST **SECURITY**

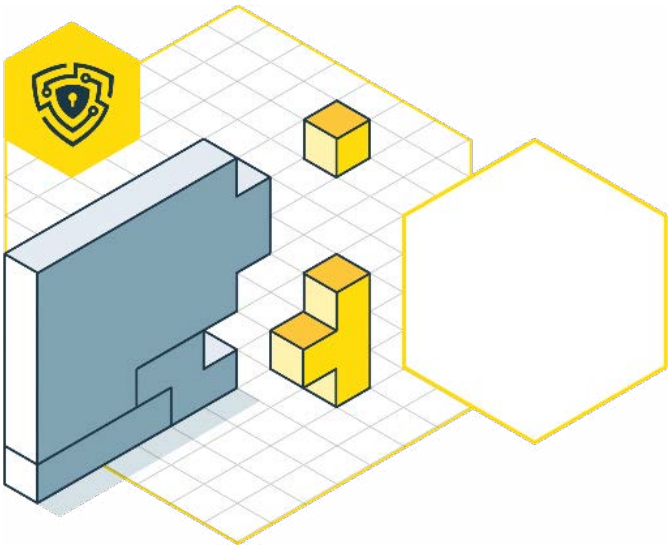
2022

GUIDE FOR PREVENTING SQL INJECTIONS

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

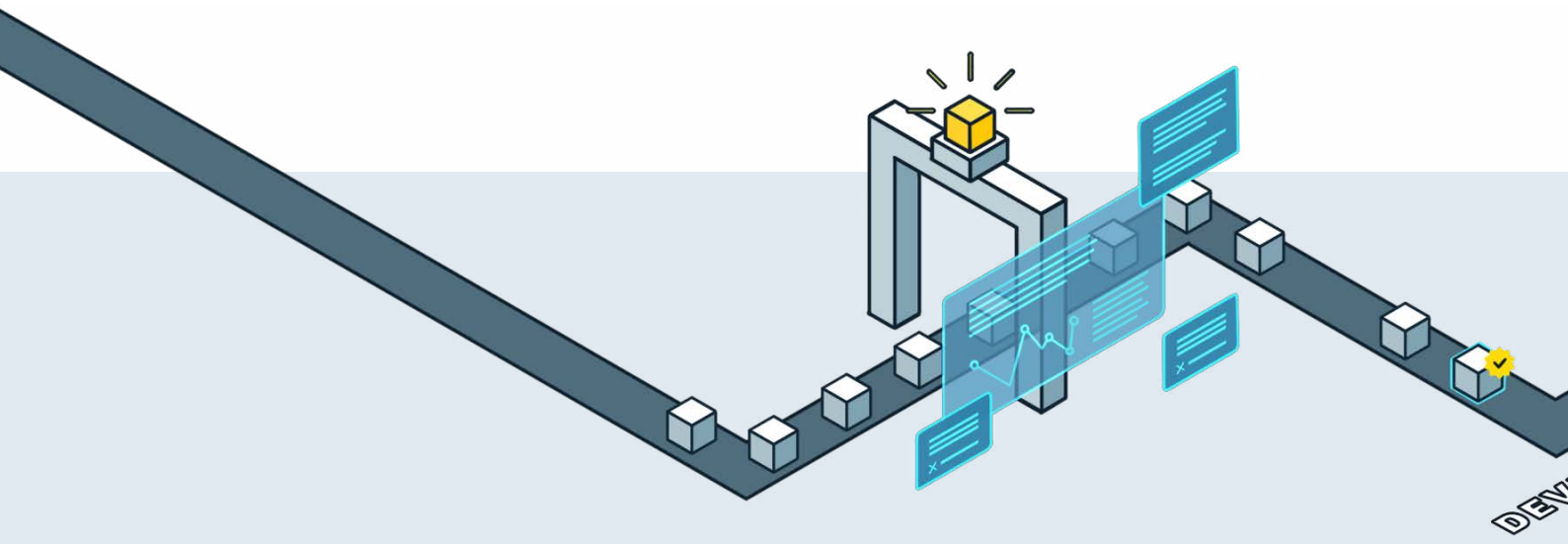
GUIDE FOR THE SQLI PREVENTION

Table of Contents



INTRODUCTION TO THIS GUIDE

By interfering with the SQL queries that the application makes to the backend database, the hackers can retrieve confidential information, examine the database or subvert the application's logic, affecting the website's availability and credibility. This guide discusses an SQLi injection attack, the types of such attacks, and learning approaches to detect and prevent SQL injection attacks in modern web applications.



WHAT IS SQL INJECTION?

SQL (Structured Query Language) injection is a code injection technique that allows the hacker to send malicious SQL queries to a web application's backend database. The SQL injection vulnerability occurs when the web page asks for a user input but accepts a SQL statement that the database can execute. The adversary can access data that the application is not built to display using these malicious SQL statements, including server configuration, user lists, and sensitive company data.

TYPES OF SQL INJECTIONS

SQL injection is the most common type of web application attack and is divided into various categories based on the techniques used to extract data. Types of SQL injections include:

IN-BAND SQL INJECTION ATTACKS

This type of SQL injection occurs when the hacker uses the same communication channel to send malicious SQL database queries and collect results. In-band SQL is considered the simplest and most prevalent form of SQL attack and falls into two main categories:

1. Error-based in-band SQL Injection

In this type of attack, the hacker obtains information about the configuration of the database from error messages generated by the server. In addition, attackers can enumerate the entire database engine using malicious input that exposes information about its version and structure.

2. Union-based SQL Injection

The attacker relies on the UNION operator to combine a genuine SQL query with the supplied malicious SELECT statements of the same structure to get a single HTTP response. The information to be accessed is part of this HTTP response. Therefore, the hacker can extend the results of an original query to extract further information about the database.

OUT-OF-BAND SQL INJECTION

In this SQLi attack, the malicious user cannot gather information from the same channel used to launch the attack. Instead, they rely on available functions to exfiltrate data through outbound channels. Such functions include connection establishment functions and file operations. The success of this attack depends on whether certain database server features are enabled, such as its ability to initiate an outbound DNS/HTTP request for data transfer, making the attack less prevalent.

INFERENTIAL/BLIND SQL INJECTION

In this SQL injection attack, the application does not transfer data from the database to the attacker. As a result, hackers have to learn about the server's structure by observing its behavior and response. These attacks are much slower since successful attacks require continuous observation of HTTP response patterns. Blind SQL injection attacks are typically categorized into:

1. Boolean/content-based attacks

The attacker crafts malicious SQL statements that ask the database TRUE or FALSE questions, then validating whether these queries modify the information within the server's HTTP response. The attacker then makes inferences about the server's structure and configuration by determining if each message generates a TRUE or FALSE statement.

2. Time-based SQL Injection attacks

The attacker crafts a query that forces the database server to wait for a set period before sending the response. This is followed by sending a malicious SQL payload and observing whether the server responds instantly or after a delay. Attackers use the response to infer whether a statement used is TRUE or FALSE, thus enumerating the database without relying on data in its response.

WHAT IS THE SEVERITY LEVEL OF SQL INJECTION ATTACKS?

The impacts and severity of an SQL injection attack depend on the security countermeasures in place and the attacker's skill. The **Online Web Application Security Project (OWASP)** recognizes injection attacks as one of the most common and far-reaching attacks, consistently ranking them as one of the top 10 application security risks. The attack is mainly found in application frameworks with legacy functional interfaces and is a common vulnerability of ASP and PHP web applications.

By injecting malicious commands into the data plane input, attackers influence how the application executes SQL commands while obtaining access to information stored and processed by the application. As a result, such attacks have severe effects, including:

Identity spoofing

Attackers use SQL injections to collect credentials of an application's users in the database. They can then impersonate the database user and carry out privileges and tasks according to the victim.

Repudiation issues

SQL injection vulnerabilities allow attackers access to data stored in a database server. As a result, they can alter the data in the database resulting in repudiation errors such as **mismatched balance** and **null transactions**, affecting the application's data credibility.

Unauthorized data access

SQLi vulnerability lets adversaries select and output database table names and column names, thereby allowing them to access sensitive intellectual data, personal data, or the database structure, granting them complete control of the application.

Chained attacks

In applications where the OS relies on the database server, attackers can use SQLi attacks as the initial attack vector to orchestrate further attacks within the application.

HOW TO IDENTIFY SQL INJECTION VULNERABILITY WITH CRASHTEST SECURITY?

The Crashtest Security Suite includes an automated scanning and penetration testing solution that helps reduce the risk of being hacked through SQLi vulnerabilities. The software establishes **systematic tests** across each phase of the SDLC to uncover and mitigate SQL injection attack surfaces. Crashtest Security is built to find injection vulnerabilities in data-driven applications to eliminate blind spots before attackers can leverage them to feed malicious SQL commands to the backend database.

Crashtest Security's vulnerability scanner relies on test cases developed using SQL payloads built to generate specific error messages. In addition, the vulnerability testing tool includes an extensive list of queries and variables that enable security and QA teams to identify vulnerabilities of many SQLi attack vectors.

The solution also blends Blind SQL injection with **machine learning techniques** and **application security mechanisms** (DAST) to emulate different attack patterns used in exploiting the vulnerability. This helps to uncover various approaches

The Crashtest Security Suite natively integrates with all modern web development stacks, enabling teams to instantly scan web applications and APIs for SQL injection vulnerabilities. The quick security assessment compares the website against **OWASP's top 10 benchmarks** to ensure the website is free of security gaps that are commonly used for SQL injection.

SQL INJECTION PREVENTION TECHNIQUES

Though prevention techniques differ with the use case, here are some common ways to prevent SQL injection attacks:

- Enforce server and client-side input validation
- Utilize parameterized queries
- Implement stored procedures
- Enforce the law of least privilege
- Enable Object Relational Mapping (ORM)
- Escape user input
- Use a Web Application Firewall

ENFORCE SERVER AND CLIENT-SIDE INPUT VALIDATION

Enforcing adequate measures to ensure users only submit the allowed input type is recommended. Whenever there is a user-controlled input interface, policies should define the accepted length, style, input format, and other characteristics used to identify valid input. This should also be supplemented with whitelists for the input type that passes validation checks and enforces maximum control on the strings that can be accepted through input fields.

To ensure strong input validation for whitelists, developers should also use regular expressions (Regex) for structured data. Using a pre-fixed value set, such as a drop-down list, also ensures that the data supplied at the input matches the defined characteristics exactly.

Finally, developers should treat all user-supplied input as malicious, ensuring that all data received from external entities is validated, keeping internal and unauthorized parties from executing malicious SQL statements.

UTILIZE PARAMETERIZED QUERIES

A parameterized query is a pre-built SQL statement that prompts the user to supply parameters (values/variables) before the server executes it. Prepared statements allow developers to define all acceptable SQL code, making it easier for the database server to distinguish between user input data and executable code. Since a parameterized query automatically quotes the parameter‘s value, the user-supplied input does not subvert the application‘s logic, thus preventing SQL injection attacks. If a hacker enters malicious SQL commands, the intent of a query cannot be changed, thereby protecting against the execution of unwanted statements.

IMPLEMENT STORED PROCEDURES

Stored procedures allow developers to define and store prepared SQL statements within the database server so they can be invoked from the application. The developers' group, several SQL statements in a logical unit, then define the execution plan such that the queries can be used repeatedly with no extra effort. This ensures that the application only executes known SQL queries, preventing attempts at SQL injection. In addition, most development frameworks allow for the parameterization of stored SQL statements, which boosts defenses against SQLi attacks.

ENFORCE THE LAW OF LEAST PRIVILEGE

Developers should avoid connecting the API to the database server using an account with administrative privileges unless it is absolutely necessary. Attackers could manipulate the database to access the root system, thus performing escalated attacks against the OS and backend server. Additionally, each application/entity should have its database access credentials with minimum access rights to data hosted within the tables. Using Role-Based Access Controls (RBAC), the teams should determine each user's access needs and create modes to ensure only the necessary permissions are granted.

ENABLE OBJECT RELATIONAL MAPPING (ORM)

ORM frameworks seamlessly translate the results of SQL queries into code, eliminating the need to use SQL statements in application code. ORM mappers also apply parameterized statements under the hood, which limits the type of user-supplied input to be executed. While it prevents dynamic SQL queries, the ORM model does not make the site entirely immune to attacks as web development frameworks allow fragmented SQL statements for complex operations.

ESCAPE USER INPUT

As a last resort, developers should escape all user-supplied input strings and special characters before they are applied to a query. Using the appropriate escaping scheme for user input ensures that the server does not confuse the input with code, which prevents the execution of unwanted SQL commands.

USE A WEB APPLICATION FIREWALL

SQL injection attacks come in various patterns, with attack vectors not covered by prevention methods such as parameterization and user input validation. The Web Application Firewall (WAF) provides holistic protection against a wide variety of web security threats, including SQL injection. The WAF helps prevent attacks by identifying & mitigating adversaries and malicious code before they reach the vulnerable web server. With a proper WAF in place, developers, security, and QA teams can improve the application's defenses against SQLi attacks without making significant changes to the application.

BEST PRACTICES TO PREVENT SQL INJECTION ATTACKS

Validating user input and avoiding dynamic queries are central to preventing SQL injection attacks in web frameworks. The following section discusses implementing SQLi prevention measures in popular development frameworks.

Django

Django includes ***querysets*** that automatically construct parameterized queries. These sets ensure that the query parameters are defined separately from the query code. Django allows developers to execute custom SQL and raw queries in complex cases. As a best practice, it is recommended to use Django's authentication library for the ***extra()*** and ***RawSQL()*** constructs when executing arbitrary queries. Escaping input to these queries also restricts SQL input that the user can supply. Django also includes an **Object Relational Model (ORM)** that maps database tables with the same fields and implements query parameterization to eliminate the SQLi attack surface.

Laravel

Laravel ships with the **Fluent Query Builder** and the **Eloquent ORM** that aid in defining prepared statements. PHP application developers can prevent all special characters from escaping through application forms by leveraging these integrations. When hackers pass malicious SQL queries through these forms, the SQL command is escaped, and the database saves the invalid query as text.

HOW CAN THE CRASHTEST SECURITY SUITE HELP?

The **Crashtest Security Suite** enables a continuous scanning and testing process that helps teams uncover vulnerabilities within their APIs, Javascript, and web applications.

In addition, the vulnerability scanner offers actionable vulnerability reports with low false positives and negatives, thereby enabling security teams to have confidence in identifying and mediating security gaps. And to spare precious time looking for solutions on how to fix the vulnerability, we also feature remediation advice directly in the report and our wiki.

Crashtest Security's scanner fits seamlessly into different development environments, helping organizations detect and remediate SQL injection flaws throughout the CI/CD pipeline.

The tool is easy to set up and can start scanning applications in minutes, granting instant access to 15+ scanners, including a **dedicated SQLi vulnerability scanner**. Start your **free, 2-week trial** to explore how you can efficiently reduce SQLi attack vectors in your web application without adding effort overheads.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM