S

SQL Injection: How SQLi Works, Prevention, Testing, and Response

SQL injection occurs when untrusted data changes the structure of a database query. Parameter binding keeps code and data separate and is the primary defense.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
3 min

SQL injection (SQLi) is a vulnerability in which untrusted input changes the structure or meaning of a database query. It commonly occurs when an application builds SQL by concatenating request values, headers, stored data, or other attacker-controlled strings into a command.

The impact can include authentication bypass, disclosure or alteration of records, deletion, administrative database actions, and—in some configurations—access beyond the database. This is a form of code injection: untrusted input is treated as instructions instead of data.

How SQL injection happens

A vulnerable application mixes SQL code and data in one dynamically constructed string. The database cannot know which part the developer intended as data, so attacker-supplied syntax can alter the query. The input may come from forms and URLs, but also JSON, cookies, API fields,

filenames, message queues, imported files, or previously stored database values.

Common SQLi forms

- **Error-based:** database errors reveal query or schema information.
- **Union or result manipulation:** an attacker changes which rows or columns are returned.
- **Blind boolean or time-based:** true/false behavior or response delay reveals information without visible database output.
- **Second-order injection:** hostile data is stored safely at first but later concatenated into another query.
- **Out-of-band:** the database is induced to communicate through another channel when its features and permissions allow it.

Primary defense: parameterized queries

Prepared statements with bound parameters define the SQL structure first and send values separately. The database then treats the value as data even when it contains SQL-like characters. OWASP recommends prepared statements as its first defense in the SQL Injection Prevention Cheat Sheet.

Parameter binding normally applies to values, not identifiers such as table names, column names, or sort direction. When those must vary, map a limited application-controlled choice to a known-safe identifier. Do not copy arbitrary user input into the query.

Defense in depth

- avoid raw dynamic queries in application and ORM code;
- ensure stored procedures do not concatenate and execute untrusted strings;
- validate type, range, length, and business rules in addition to parameterization;
- give the application database account only required permissions and separate read/write roles;

- return generic errors to users while retaining protected diagnostic logs;
- use WAF rules and monitoring as supplemental detection, not a replacement for safe queries;
- protect database credentials and restrict network access to approved application paths.

Escaping every special character is database- and context-dependent and is more fragile than parameterization. Blocking words such as `SELECT` also causes false positives and is easy to bypass.

Safe testing

Test only applications you own or are explicitly authorized to assess. Combine code review, static analysis, dependency and framework checks, unit tests for query construction, and dynamic testing in a controlled environment. Verify data flows into raw queries, stored procedures, report builders, search filters, and administrative tools.

Incident response

1. preserve web, WAF, application, identity, and database logs;
2. contain the vulnerable endpoint without destroying evidence;
3. replace unsafe query construction with parameterization and review similar code paths;
4. determine what data and database actions were accessible or used;
5. rotate exposed credentials and keys, restore altered data from verified sources, and meet notification obligations;
6. monitor for persistence, created accounts, data exports, and follow-on access.

SQL injection FAQ

Does an ORM prevent SQLi?

It helps when using safe parameter APIs. Raw queries and unsafe interpolation can still be injectable.

Can a WAF stop SQL injection?

It may block known patterns, but only fixing unsafe query construction addresses the vulnerability reliably.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com