

# SQL Injection — Defense Overview

## Primary Defense (Correct Fix)

### Parameterized Queries / Prepared Statements

- SQL query structure is defined first
  - User input is bound as data, not concatenated
  - Input cannot modify SQL logic
- This is the **only reliable fix** for SQL Injection.
- 

## Why Parameterization Works

- SQL is parsed and compiled before input is applied
  - Special characters ( `'`, `--`, `OR` ) lose meaning
  - Input is treated as a literal value
- Result: injected payloads cannot change query behavior.
- 

## Common Implementations

- Prepared statements (language-level)
  - ORM query builders (when used correctly)
  - Bind variables / placeholders
- All follow the same principle: **code != data**.
- 

## Weak / Insufficient Protections

These **do NOT reliably prevent SQLi**:

- Input sanitization alone
  - Escaping characters
  - Blacklists / regex filter
  - Client-side validation
- They reduce noise but do not remove the root cause.

---

## Defense in Depth (Secondary)

Used **in addition to parameterization**, not instead:

- Least-privilege database accounts
  - Generic error messages
  - Logging and monitoring
  - Web application firewalls (WAF)
- 

## One-Line Rule

If user input can change SQL query logic → SQL Injection exists.

If input is always treated as data → SQL Injection is prevented.