sonar RULES

Products

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML

Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Code Smell

Assertions should be complete
Code Smell

Tests should include assertions
Code Smell

Silly bit operations should not be performed
Code Smell

Child class fields should not shadow parent class fields
Code Smell

JUnit test cases should call super methods
Code Smell

TestCases should contain tests
Code Smell

Short-circuit logic should be used in boolean contexts
Code Smell

Methods and field names should not be the same or differ only by capitalization
Code Smell

Switch cases should end with an unconditional "break" statement
Code Smell

"switch" statements should not contain non-case labels
Code Smell

Future keywords should not be used as names
Code Smell

Thread suspensions should not be

"PreparedStatement" and "ResultSet" methods should be called with valid indices

Analyze your code

Bug Blocker sql

The parameters in a `PreparedStatement` are indexed beginning at 1, not 0, so using any "set" method of a `PreparedStatement` with a number less than 1 is a bug, as is using an index higher than the number of parameters. The same indexing style also applies to `ResultSet`.

Noncompliant Code Example

```
PreparedStatement ps = con.prepareStatement("SELECT fname, l
ps.setDate(0, date);  // Noncompliant
ps.setDouble(3, salary);  // Noncompliant

ResultSet rs = ps.executeQuery();
while (rs.next()) {
  String fname = rs.getString(0);  // Noncompliant
  // ...
}
```

Compliant Solution

```
PreparedStatement ps = con.prepareStatement("SELECT fname, l
ps.setDate(1, date);
ps.setDouble(2, salary);

ResultSet rs = ps.executeQuery();
while (rs.next()) {
  String fname = rs.getString(1);
  // ...
}
```

Available In:

sonarlint sonarcloud sonarqube

Thread suspensions should not be vulnerable to Denial of Service attacks

Vulnerability

A new session should be created during user authentication

Vulnerability

JWT should be signed and verified with strong cipher algorithms

Vulnerability

Cipher algorithms should be robust

Vulnerability

Encryption algorithms should be used with secure mode and padding