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...

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 vulnerable to Denial of Service attacks

"wait" should not be called when multiple locks are held

Analyze your code

Bug Blocker multi-threading deadlock

When two locks are held simultaneously, a `wait` call only releases one of them. The other will be held until some other thread requests a lock on the awaited object. If no unrelated code tries to lock on that object, then all other threads will be locked out, resulting in a deadlock.

Noncompliant Code Example

```
synchronized (this.mon1) {  // threadB can't enter this bloc
        synchronized (this.mon2) {
                this.mon2.wait();  // Noncompliant; threadA
        }
}
```

Available In:
sonarlint sonarcloud sonarqube

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 scheme