




 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


 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
 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
 Vulnerability


"wait(...)" should be used instead of "Thread.sleep(...)" when a lock is held

Analyze your code

 Bug

 Blocker

 ?

 multi-threading performance cert

If `Thread.sleep(...)` is called when the current thread holds a lock, it could lead to performance and scalability issues, or even worse to deadlocks because the execution of the thread holding the lock is frozen. It's better to call `wait(...)` on the monitor object to temporarily release the lock and allow other threads to run.

Noncompliant Code Example

```
public void doSomething(){
    synchronized(monitor) {
        while(notReady()){
            Thread.sleep(200);
        }
        process();
    }
    ...
}
```

Compliant Solution





```
public void doSomething(){
    synchronized(monitor) {
        while(notReady()){
            monitor.wait(200);
        }
        process();
    }
    ...
}
```

See

- [CERT, LCK09-J](#). - Do not perform operations that can block while holding a lock

Available In:

sonarlint  | sonarcloud  | sonarqube 

<div>Server hostnames should be verified during SSL/TLS connections</div> <div> Vulnerability</div>
<div>Insecure temporary file creation methods should not be used</div> <div> Vulnerability</div>
<div>Passwords should not be stored in plain-text or with a fast hashing algorithm</div> <div> Vulnerability</div>
<div>Server certificates should be verified during SSL/TLS connections</div> <div> Vulnerability</div>