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...

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

Server hostnames should be verified during SSL/TLS connections
Vulnerability

Insecure temporary file creation methods should not be used
Vulnerability

Passwords should not be stored in

Methods "wait(...)", "notify()" and "notifyAll()" should not be called on Thread instances

Analyze your code

Bug Blocker multi-threading

The methods `wait(...)`, `notify()` and `notifyAll()` are available on a `Thread` instance, but only because all classes in Java extend `Object` and therefore automatically inherit those methods. But there are two very good reasons for not calling them on a `Thread`:

- Internally, the JVM relies on these methods to change the state of the Thread (`BLOCKED`, `WAITING`, ...), so calling them will corrupt the behavior of the JVM.
- It is not clear (perhaps even to the original coder) what is really expected. For instance, it is waiting for the execution of the Thread to suspended, or is it the acquisition of the object monitor that is waited for?

Noncompliant Code Example

```
Thread myThread = new Thread(new RunnableJob());
...
myThread.wait(2000);
```

Available In:
sonarlint sonarcloud sonarqube

plain-text or with a fast hashing algorithm

Vulnerability

Server certificates should be verified during SSL/TLS connections

Vulnerability

Persistent entities should not be used as arguments of "@RequestMapping" methods

Vulnerability

"HttpSecurity" URL patterns should be correctly ordered

Vulnerability