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

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...

Security Hotspot

Searching OS commands in PATH is security-sensitive

Security Hotspot

Allowing both safe and unsafe HTTP methods is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure" flag is security-sensitive

Security Hotspot

Using hardcoded IP addresses is security-sensitive

Security Hotspot

'serialVersionUID' field should not be set to '0L' in records

Code Smell

Permitted types of a sealed class should be omitted if they are declared in the same file

Code Smell

Switch arrow labels should not use redundant keywords

Code Smell

Text blocks should not be used in complex expressions

Code Smell

Pattern Matching for "instanceof" operator should be used instead of simple "instanceof" + cast

Code Smell

Call to Mockito method "verify", "when" or "given" should be simplified

"Double.longBitsToDouble" should not be used for "int"

Analyze your code

Bug Major

`Double.longBitsToDouble` expects a 64-bit, `long` argument. Pass it a smaller value, such as an `int` and the mathematical conversion into a `double` simply won't work as anticipated because the layout of the bits will be interpreted incorrectly, as if a child were trying to use an adult's gloves.

Noncompliant Code Example

```
int i = 42;
double d = Double.longBitsToDouble(i);  // Noncompliant
```

Available In:

sonarlint sonarcloud sonarqube

Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions

Code Smell

Consecutive AssertJ "assertThat" statements should be chained

Code Smell

Chained AssertJ assertions should be simplified to the corresponding dedicated assertion

Code Smell

Exception testing via JUnit @Test