

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

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

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags

Search by name...

Code Smell

Nested blocks of code should not be left empty

Methods should not have too many parameters

Unused "private" fields should be removed

Collapsible "if" statements should be merged

Unused labels should be removed

Standard outputs should not be used directly to log anything

OS commands should not be vulnerable to argument injection attacks

"ActiveMQConnectionFactory" should not be vulnerable to malicious code deserialization

Logging should not be vulnerable to injection attacks

Exceptions should not be thrown from servlet methods

Return values should not be ignored when they contain the operation status code

Conditionally executed code should be reachable

Analyze your code

Bug Major cwe cert unused suspicious pitfall

Conditional expressions which are always true or false can lead to dead code. Such code is always buggy and should never be used in production.

Noncompliant Code Example

```
a = false;
if (a) { // Noncompliant
    doSomething(); // never executed
}

if (!a || b) { // Noncompliant; "a" is always "true", "b" is always "false"
    doSomething();
} else {
    doSomethingElse(); // never executed
}
```

Exceptions

This rule will not raise an issue in either of these cases:

- When the condition is a single final boolean

```
final boolean debug = false;
//...
if (debug) {
    // Print something
}
```

- When the condition is literally true or false.

```
if (true) {
    // do something
}
```

In these cases it is obvious the code is as intended.

See

- [MITRE, CWE-570](#) - Expression is Always False
- [MITRE, CWE-571](#) - Expression is Always True
- [CERT, MSC12-C](#) - Detect and remove code that has no effect or is never executed

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-2583

1/2


Repeated patterns in regular expressions should not match the empty string  Bug
AssertJ assertions "allMatch" and "doesNotContains" should also test for emptiness  Bug
Double Brace Initialization should not be used  Bug
Non-primitive fields should not be

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)