




 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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags

Search by name...

argument type

Bug

"Serializable" inner classes of non-serializable classes should be "static"

Bug

The non-serializable super class of a "Serializable" class should have a no-argument constructor

Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored

Bug

"equals(Object obj)" and "hashCode()" should be overridden in pairs

Bug

Disclosing fingerprints from web application technologies is security-sensitive

Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Security Hotspot

Delivering code in production with debug features activated is security-sensitive

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

Silly equality checks should not be made

Analyze your code

Bug

Major

cert unused

Comparisons of dissimilar types will always return false. The comparison and all its dependent code can simply be removed. This includes:

- comparing an object with null
- comparing an object with an unrelated primitive (E.G. a string with an int)
- comparing unrelated classes
- comparing an unrelated class and interface
- comparing unrelated interface types
- comparing an array to a non-array
- comparing two arrays

Specifically in the case of arrays, since arrays don't override `Object.equals()`, calling `equals` on two arrays is the same as comparing their addresses. This means that `array1.equals(array2)` is equivalent to `array1==array2`.

However, some developers might expect `Array.equals(Object obj)` to do more than a simple memory address comparison, comparing for instance the size and content of the two arrays. Instead, the `==` operator or `Arrays.equals(array1, array2)` should always be used with arrays.

Noncompliant Code Example

```
interface KitchenTool { ... };
interface Plant {...}

public class Spatula implements KitchenTool { ... }
public class Tree implements Plant { ... }
//...





Spatula spatula = new Spatula();
KitchenTool tool = spatula;
KitchenTool [] tools = {tool};

Tree tree = new Tree();
Plant plant = tree;
Tree [] trees = {tree};

if (spatula.equals(tree)) { // Noncompliant; unrelated class
    // ...
}
else if (spatula.equals(plant)) { // Noncompliant; unrelated
    // ...
}
else if (tool.equals(plant)) { // Noncompliant; unrelated in
    // ...
}
else if (tool.equals(tools)) { // Noncompliant; array & non-
    // ...
}
else if (trees.equals(tools)) { // Noncompliant; incompatibl
    // ...
}
```

https://rules.sonarsource.com/java/RSPEC-2159

1/2

<div>Creating cookies without the "secure" flag is security-sensitive</div> <div> Security Hotspot</div>
<div>Using hardcoded IP addresses is security-sensitive</div> <div> Security Hotspot</div>
<div>'serialVersionUID' field should not be set to '0L' in records</div> <div> Code Smell</div>
<div>Permitted types of a sealed class should be omitted if they are declared in the same file</div> <div></div>

```
}
else if (tree.equals(null)) { // Noncompliant
    // ...
}
```

See

- [CERT, EXP02-J](#) - Do not use the Object.equals() method to compare two arrays

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

© 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](#)