




 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



Scala static code analysis

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

All rules 41

Bug 6

Security Hotspot 2

Code Smell 33

Tags ▾

Search by name... 🔍

Should not have exactly the same implementation

Bug

Related "if"/"else if" statements and "case" in a "match" should not have the same condition

Bug

Identical expressions should not be used on both sides of a binary operator

Bug

All code should be reachable

Bug

Variables should not be self-assigned

Bug

Useless "if(true){...}" and "if(false){...}" blocks should be removed

Bug

Methods should not have identical implementations

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

"match" expressions should not have too many "case" clauses

Code Smell

Sections of code should not be commented out

Code Smell

Unused function parameters should be removed

Code Smell

String literals should not be duplicated

Analyze your code

Code Smell

Critical ?

design

Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.

On the other hand, constants can be referenced from many places, but only need to be updated in a single place.

Noncompliant Code Example

With the default threshold of 3:

```
public def run() {
  prepare("action random1")    // Noncompliant - "action random1"
  execute("action random1")
  release("action random1")
}
```

Compliant Solution

```
public def run() {
  val action = "action random1"
  prepare(action)
  execute(action)
  release(action)
}
```

Exceptions

To prevent generating some false-positives, literals having 5 or less characters are excluded as well as literals containing only letters, digits and '.'.

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](#)

https://rules.sonarsource.com/scala/RSPEC-1192

1/2

<div>Unused "private" methods should be removed</div> <div> Code Smell</div>
<div>Track uses of "FIXME" tags</div> <div> Code Smell</div>
<div>Nested blocks of code should not be left empty</div> <div> Code Smell</div>
<div>Functions should not have too many parameters</div> <div> Code Smell</div>
<div>Collapsible "if" statements should be merged</div>