
































-  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... 


- Nested blocks of code should not be left empty


 Code Smell
- Functions should not have too many parameters


 Code Smell
- Collapsible "if" statements should be merged


 Code Smell
- Using hardcoded IP addresses is security-sensitive


 Security Hotspot
- Multi-line comments should not be empty


 Code Smell
- Boolean checks should not be inverted


 Code Smell
- Unused local variables should be removed

 Code Smell
- Local variable and function parameter names should comply with a naming convention

 Code Smell
- Boolean literals should not be redundant

 Code Smell
- Class names should comply with a naming convention

 Code Smell
- Method names should comply with a naming convention

 Code Smell

Methods should not have identical implementations

Analyze your code

 Code Smell

 Major ?

 confusing duplicate suspicious

When two methods have the same implementation, either it was a mistake - something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, one implementation should invoke the other.

Noncompliant Code Example

```
class Box(length: Int, width: Int, height: Int) {
  def volume: Int = {
    val s = length * width
    s * height
  }

  def area: Int = {
    val s = length * width
    s * height
  }
}
```

Compliant Solution

```
class Box(length: Int, width: Int, height: Int) {
  def volume: Int = {
    val s = length * width
    s * height
  }





  def area: Int = {
    length * width
  }
}
```

Exceptions

Methods with fewer than 2 statements are ignored.

Available In:

 |  | 

Track uses of "TODO" tags  Code Smell
Track lack of copyright and license headers  Code Smell
"match" statements should not be nested  Code Smell
Control flow statements "if", "for", "while", "match" and "try" should not be nested too deeply  Code Smell
"if ... else if" constructs should end