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

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...

- Track uses of "FIXME" tags — Code Smell
- 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

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

Analyze your code

Code Smell Major design suspicious

Having two `cases` in a `match` statement or two branches in an `if` chain with the same implementation is at best duplicate code, and at worst a coding error. If the same logic is truly needed for both instances, then in an `if` chain they should be combined, or for a `match`, one should fall through to the other.

Noncompliant Code Example

```
value match {
  case 1 =>
    doFirstThing
    doSomething
  case 2 =>
    doSomethingDifferent
  case 3 => // Noncompliant; duplicates case 1's implem
    doFirstThing
    doSomething
  case _ =>
    doTheRest
}

if (a >= 0 && a < 10) {
  doFirstThing
  doTheThing
}
else if (a >= 10 && a < 20) {
  doTheOtherThing
}
else if (a >= 20 && a < 50) {
  doFirstThing
  doTheThing  // Noncompliant; duplicates first conditi
}
else {
  doTheRest
}
```

Exceptions

Blocks in an `if` chain that contain a single line of code are ignored, as are blocks in a `match` statement that contain a single line of code.

```
if(a == 1) {
  doSomething  //no issue, usually this is done on purp
} else if (a == 2) {
  doSomethingElse
} else {
```

Method names should comply with a naming convention

Code Smell

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

```
  doSomething
}
```

But this exception does not apply to `if` chains without `else`-s, or to `match`-es without default clauses when all branches have the same single line of code. In case of `if` chains with `else`-s, or of `match`-es with default clauses, rule {rule:scala:S3923} raises a bug.

```
if(a == 1) {
  doSomething  //Noncompliant, this might have been don
} else if (a == 2) {
  doSomething
}
```

Available In:

sonarlint | sonarcloud | sonarqube