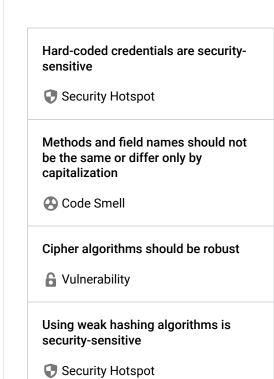


- Secrets
- SAP ABAP
- Apex Apex
- **c** C
- C++
- CloudFormation
- COBOL COBOL
- C# C#
- **S** CSS
- X Flex
- **=GO** Go
- HTML
- 🐇 Java
- Js JavaScript
- Kotlin
- Kubernetes
- **6** Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TS TypeScript
- T-SQL
- VB VB.NET
- VB6 VB6
- XML XML



## **Swift static code analysis**

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



Cognitive Complexity of functions

Code Smell

"try!" should not be used

should not be too high

Code Smell

String literals should not be duplicated

Code Smell

Functions and closures should not be empty

Code Smell

Collection elements should not be replaced unconditionally

📆 Bug

Collection sizes comparisons should make sense

📆 Bug

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

🛊 Bug

Infix operators that end with "=" should update their left operands

📆 Bug

Precedence and associativity of standard operators should not be changed

📆 Bug

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

Tags

Analyze your code

Search by name..

Having two cases in the same switch statement or branches in the same if structure 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 they should be combined.

## Noncompliant Code Example

```
switch i {
  case 1:
    doFirstThing()
    doSomething()
  case 2:
    doSomethingDifferent()
  case 3: // Noncompliant; duplicates case 1's implementati
    doFirstThing()
    doSomething()
  default:
    doTheRest()
if a >= 0 && a < 10 {
  doFirstThing()
  doTheThing()
} else if a >= 10 && a < 20 {
  doTheOtherThing()
} else if a >= 20 && a < 50 {
  doFirstThing()
                    // Noncompliant; duplicates first condi
  doTheThing()
} else {
  doTheRest()
}
```

## Exceptions

case labels that declare variables cannot have multiple patterns. Therefore this situation is ignored.

```
switch a {
    case .STR_CASE(let x):
        print(x)
    case .INT_CASE(let x):
        print(x)
    default:
        print("default")
}
```

Blocks in an if chain that contain a single line of code are ignored, as are blocks in a switch statement that contain a single line of code with or without a following break.

Return values from functions without side effects should not be ignored

Rug Bug

Related "if/else if" statements and "cases" in a "switch" should not have the same condition

Rug Bug

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

Rug Bug

All code should be reachable

Rug Bug

Loops with at most one iteration should be refactored

Rug Bug

"IBInspectable" should be used correctly

Code Smell

Functions should not have identical implementations

Code Smell

Ternary operators should not be nested

Code Smell

Closure expressions should not be nested too deeply

Code Smell

Backticks should not be used around symbol names

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

```
if a >= 0 && a < 10 {
                         //Noncompliant, this might have be
  doTheThing()
} else if a >= 10 && a < 20 {
  doTheThing()
```

Available In:

}

sonarlint sonarcloud sonarqube Developer Edition

© 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