


-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  **Swift**
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML




Swift static code analysis

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

All rules 119

 Vulnerability 3

 Bug 14

 Security Hotspot 3

 Code Smell 99

Tags

Search by name...




Hard-coded credentials are security-sensitive

 Security Hotspot

Methods and field names should not be the same or differ only by capitalization

 Code Smell


Cipher algorithms should be robust

 Vulnerability


Using weak hashing algorithms is security-sensitive

 Security Hotspot


Cognitive Complexity of functions should not be too high

 Code Smell

"try!" should not be used

 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

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

Analyze your code

 Bug  Major  unused pitfall

A switch and a chain of if/else if statements is evaluated from top to bottom. At most, only one branch will be executed: the first one with a condition that evaluates to true.

Therefore, duplicating a condition automatically leads to dead code. Usually, this is due to a copy/paste error. At best, it's simply dead code and at worst, it's a bug that is likely to induce further bugs as the code is maintained, and obviously it could lead to unexpected behavior.

Noncompliant Code Example

```
if param == 1 {
    openWindow()
} else if param == 2 {
    closeWindow()
} else if param == 1 { // Noncompliant
    moveWindowToTheBackground()
}

switch i {
case 1:
    //...
case 3:
    //...
case 1: // Noncompliant
    //...
default:
    // ...
}
```

Compliant Solution

```
if param == 1 {
    openWindow()
} else if param == 2 {
    closeWindow()
} else if param == 3 {
    moveWindowToTheBackground()
}

switch i {
case 1:
    //...
case 3:
    //...
default:
    // ...
}
```











Available In:

sonarlint

sonarcloud

sonarqube

Developer Edition

 Bug
Return values from functions without side effects should not be ignored  Bug
Related "if/else if" statements and "cases" in a "switch" 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
Loops with at most one iteration should be refactored  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