

- 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

"self" should only be used when required

Analyze your code

Code Smell Minor clumsy

The use of `self` is optional except when in closure expressions, and when it's needed to distinguish between property names and arguments. For the sake of brevity, `self` should be omitted when it's not strictly required.

Noncompliant Code Example

```
class Car {
    var color: Int

    init(color: Int) {
        self.color = color
    }

    func fade() {
        self.color-- // Noncompliant
    }
}
```

Compliant Solution

```
class Car {
    var color: Int

    init(color: Int) {
        self.color = color
    }

    func fade() {
        color--
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition

 Bug
<div>Return values from functions without side effects should not be ignored</div> <div> Bug</div>
<div>Related "if/else if" statements and "cases" in a "switch" should not have the same condition</div> <div> Bug</div>
<div>Identical expressions should not be used on both sides of a binary operator</div> <div> Bug</div>
<div>All code should be reachable</div> <div> Bug</div>
<div>Loops with at most one iteration should be refactored</div> <div> Bug</div>
<div>"IBInspectable" should be used correctly</div> <div> Code Smell</div>
<div>Functions should not have identical implementations</div> <div> Code Smell</div>
<div>Ternary operators should not be nested</div> <div> Code Smell</div>
<div>Closure expressions should not be nested too deeply</div> <div> Code Smell</div>
<div>Backticks should not be used around</div>