

-  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

A field should not duplicate the name of its containing class

Analyze your code

Code Smell Major brain-overload

It's confusing to have a class member with the same name (case differences aside) as its enclosing class. This is particularly so when you consider the common practice of naming a class instance for the class itself.

Best practice dictates that any field or member with the same name as the enclosing class be renamed to be more descriptive of the particular aspect of the class it represents or holds.

Noncompliant Code Example

```
public class Foo {
    private var foo : String

    public func getFoo() -> String {
        return foo
    }

    //...

}

var foo = Foo()
foo.getFoo() // what does this return?
```

Compliant Solution

```
public class Foo {
    private var name : String

    public func getName() -> String {
        return name
    }

    //...

}

var foo = Foo();
foo.getName()
```

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>