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

---

Enumeration members should comply with a naming convention

Analyze your code

⚙ Code Smell   ✅ Minor ❓   🏷 convention

Shared coding conventions allow teams to collaborate efficiently. This rule checks that all enumeration member names match a provided regular expression.

Noncompliant Code Example

With default provided regular expression: `^[a-z][a-zA-Z0-9]*$`

```
enum SomeEnumeration {
    case SomeMember  // Non-Compliant
}
```

Compliant Solution

```
enum SomeEnumeration {
    case someMember
}
```

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