



ABAP

Apex

С

C++

CloudFormation

COBOL

C#

CSS

Flex

Go =GO

HTML 5

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



Objective C static code analysis

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

ΑII 315 rules

6 Vulnerability 10

R Bug (75)

o Security Hotspot

⊗ Code (212)

O Quick 13 Fix

Tags

Search by name...

"memset" should not be used to delete sensitive data

Vulnerability

POSIX functions should not be called with arguments that trigger buffer overflows

■ Vulnerability

Function-like macros should not be invoked without all of their arguments

📆 Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist

📆 Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked

📆 Bug

"pthread_mutex_t" should be properly initialized and destroyed

📆 Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice

📆 Bug

Functions with "noreturn" attribute should not return

📆 Bug

"memcmp" should only be called with pointers to trivially copyable types with no padding

🖷 Bug

Stack allocated memory and nonowned memory should not be freed

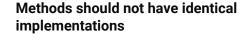
📆 Bug

Closed resources should not be accessed

📆 Bug

Dynamically allocated memory should be released

📆 Bug



Analyze your code

confusing duplicate suspicious

When two methods have the same implementation, either it was a mistake something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, one implementation should invoke the

This rule raises an exception when two methods implemented inside the class definition share the same implementation.

Noncompliant Code Example

```
class Point {
  int x;
  int y;
// ....
public:
  void setX(int v) {
    if (v \ge 0 \&\& v < MAX_X) {
      x = v;
      return;
    }
    error();
  void setY(int v) { // Noncompliant
    if (v \ge 0 \&\& v < MAX_X) {
      x = v;
      return;
    }
    error();
};
```

Compliant Solution

```
class Point {
 int x;
 int y;
// ....
public:
void setX(int v) {
    if (v >= 0 \&\& v < MAX X) {
      x = v;
      return;
    }
    error();
  void setY(int v) {
    if (v \ge 0 \& v < MAX_X) {
      return;
    error();
```

Freed memory should not be used 📆 Bug Memory locations should not be released more than once 📆 Bug Memory access should be explicitly bounded to prevent buffer overflows 📆 Bug Printf-style format strings should not lead to unexpected behavior at runtime 📆 Bug Recursion should not be infinite 📆 Bug Resources should be closed 📆 Bug Hard-coded credentials are securitysensitive Security Hotspot "goto" should jump to labels declared later in the same function Code Smell Only standard forms of the "defined" directive should be used Code Smell Switch labels should not be nested inside non-switch blocks

Code Smell

Exceptions

Empty methods, methods with the same name (overload) and methods with only one statement are ignored.

Available In:

sonarcloud of 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