



_{АРЕХ} Арех

ABAP

c c

C++

CloudFormation

COBOL COBOL

C# C#

E CSS

X Flex

GO Go

∃ HTML

🖺 Java

JS JavaScript

Kotlin

Kubernetes

© Objective C

PHP

PL/I PL/I

PL/SQL PL/SQL

🦆 Python

RPG RPG

Ruby

Scala

Swift

Terraform

Text

тs TypeScript

T-SQL

VB VB.NET

VB6 VB6

XML XML



Objective C static code analysis

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

All 315 rules

6 Vulnerability 10

R Bug 75

Security Hotspot

18

⊗ Code 212

O Quick 13 Fix

Analyze your code

Tags

Variables should not be shadowed

✓ Search by name...

"memset" should not be used to delete sensitive data

6 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

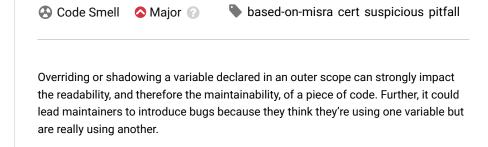
🕦 Bug

Closed resources should not be accessed

📆 Bug

Dynamically allocated memory should be released

📆 Bug



Noncompliant Code Example

```
class Foo
{
  public:
    void doSomething();

private:
    int myField;
};

void Foo::doSomething()
{
    int myField = 0; // Noncompliant
    // ...
}
```

```
void f(int x, bool b) {
  int y = 4;
  if (b) {
    int x = 7; // Noncompliant
    int y = 9; // Noncompliant
    // ...
}
```

Compliant Solution

```
class Foo
{
  public:
    void doSomething();

private:
    int myField;
};

void Foo::doSomething()
{
    int myInternalField = 0; // Compliant
    // ...
}
```

```
void f(int x, bool b) {
  int y = 4;
  if (b) {
    int z = 7; // Better yet: Use meaningful names
    int w = 9;
    // ...
}
```

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

It is common in a constructor to have constructor arguments shadowing the fields that they will initialize. This pattern avoids the need to select new names for the constructor arguments, and will not be reported by this rule:

```
class Point{
public:
   Point(int x, int y) : x(x), y(y) {} // Compliant by excepti
private:
   int x;
   int y;
};
```

See

- MISRA C:2004, 5.2 Identifiers in an inner scope shall not use the same name as an identifier in an outer scope, and therefore hide that identifier
- MISRA C++:2008, 2-10-2 Identifiers declared in an inner scope shall not hide an identifier declared in an outer scope
- MISRA C:2012, 5.3 An identifier declared in an inner scope shall not hide an identifier declared in an outer scope
- CERT, DCL01-C. Do not reuse variable names in subscopes
- CERT, DCL51-J. Do not shadow or obscure identifiers in subscopes

Available In:

sonarcloud \delta | 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