

-  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



Objective C static code analysis


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

All rules 315

 Vulnerability 10

 Bug 75

 Security Hotspot 18

 Code Smell 212

 Quick Fix 13

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

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

 Bug

Stack allocated memory and non-owned memory should not be freed

 Bug

Closed resources should not be accessed

 Bug

Dynamically allocated memory should be released

 Bug

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 security-sensitive
 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

Function names should be used either as a call with a parameter list or with the "&" operator

Analyze your code

 Code Smell  Critical   based-on-misra suspicious

Using a "bald" function name is likely a bug. Rather than testing the return value of a function with a void parameter list, it implicitly retrieves the address of that function in memory. If that's truly what's intended, then it should be made explicit with the use of the & (address-of) operator. If it's not, then a parameter list (even an empty one) should be added after the function name.

Noncompliant Code Example

```
int func(void) {
    // ...
}

void f2(int a, int b) {
    // ...
    if (func) { // Noncompliant - tests that the memory address
        //...
    }
    // ...
}
```

Compliant Solution

```
void f2(int a, int b) {
    // ...
    if (func()) { // tests that the return value of func() > 0
        //...
    }
    // ...
}
```

Exceptions

Callback functions are a common occurrence and are usually not passed with a preceding &. There is however little ambiguity so this rule ignores function identifiers when used as a parameter of a function call.

```
void foo() {
    // ...
}

registerEvent(AnEvent, foo);
```

See

- MISRA C:2004, 16.9 - A function identifier shall only be used with either a preceding &, or with a parenthesized parameter list, which may be empty.
- MISRA C++:2008, 8-4-4 - A function identifier shall only be used to call the function or it shall be preceded by &.

Available In:   Developer Edition