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

"memcmp" 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

## Functions without parameters should be declared with parameter type "void"

Analyze your code

Code Smell | Critical | based-on-misra cert pitfall

There is a real, functional difference between a function with an empty parameter list and one with an explicitly `void` parameter list: It is possible to pass parameters to a function with an empty list; the compiler won't complain. That is not the case for a function with a `void` list. Thus, it is possible, and even easy to invoke empty-list functions incorrectly without knowing it, and thereby introduce the kind of subtle bug that can be very difficult to track down.

### Noncompliant Code Example

```
void myfunc ();  // Noncompliant

//...

void otherFunc() {
  int a = 4;
  //...
  myfunc(a); // Compiler allows this
}
```

### Compliant Solution

```
void myfunc ( void );

//...

void otherFunc() {
  int a = 4;
  //...
  myfunc(a); // Compiler error!
}
```

### See

- MISRA C:2004, 16.5 - Functions with no parameters shall be declared with parameter type void
- CERT, DCL20-C. - Explicitly specify void when a function accepts no arguments

Available In:

sonarcloud | sonarqube Developer Edition

**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