

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- 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

Constants of unsigned type should have a "U" suffix

Analyze your code

Code Smell Critical based-on-misra pitfall

The type of an integer is dependent on a complex combination of factors including:

- The magnitude of the constant;
- The implemented sizes of the integer types;
- The presence of any suffixes;
- The number base in which the value is expressed (i.e. decimal, octal or hexadecimal).

For example, the value 0x8000 is of type `unsigned int` in a 16-bit environment, but of type (signed) `int` in a 32-bit environment.

Note:

- Any value with a "U" suffix is of unsigned type;
- An unsuffixed decimal value less than 2^31 is of signed type.

But:

- An unsuffixed hexadecimal value greater than or equal to 2^15 may be of signed or unsigned type;
- For C90, an unsuffixed decimal value greater than or equal to 2^31 may be of signed or unsigned type.

In C++, if an overload set includes candidates for an `unsigned int` and an `int`, then the overload that would be matched by 0x8000 is therefore dependent on the implemented integer size. Adding a "U" suffix to the value specifies that it is unsigned.

See

- MISRA C:2004, 10.6 - A "U" suffix shall be applied to all constants of unsigned type.
- MISRA C++:2008, 2-13-3 - A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.
- MISRA C:2012, 7.2 - A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type.

Available In:

sonarcloud | sonarqube Developer Edition

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