

- 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

"static" should not be used for the size of an array parameter

Analyze your code

Code Smell Critical based-on-misra pitfall

Theoretically, the use of the `static` keyword on the size of an array parameter means you can assume that only arrays of at least that size will be passed as arguments to the function. I.e. a function parameter of `int my_array[static 10]` means that `my_array` will always be *at least* 10 elements long. If it is not, the behavior is undefined.

In practice, the use of `static` on the size of an array parameter means the compiler might issue a warning if a noncompliant array is passed to the function - a warning that might or might not be ignored. Therefore, in practice the use of `static` on an array parameter's size merely lends a false sense of security, and `static` should not be used in this context.

Note that for some compiler/processor combinations, more efficient code can be generated when `static` is used, but these combinations are limited, and the benefit does not outweigh the cost.

### Noncompliant Code Example

```
int total (int size, int my_array[static 10]) {...}
```

### Compliant Solution

```
int total (int size, int my_array[10]) {...}
```

### See

- MISRA C:2012, 17.6 - The declaration of an array parameter shall not contain the `static` keyword between the `[]`

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>