

- 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

"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

Object declarations should contain no more than 2 levels of pointer indirection

Analyze your code

Code Smell Critical based-on-misra brain-overload pitfall

While they are extraordinarily useful, pointers are not the most intuitive concept in the world. Pointers to pointers are even harder to understand and use correctly. And with each additional level of indirection, pointer variables become more difficult to use correctly. Therefore pointer declarators should be limited to no more than two levels of nesting.

## Noncompliant Code Example

```
typedef int * INTPTR;
struct s {
    int ** s1;
    int *** s2; // Noncompliant
};

struct s ** ps1;
struct s *** ps2; // Noncompliant

int ** ( *pfunc1)();
int ** ( **pfunc2)();
int ** (**pfunc3)(); // Noncompliant
int *** ( **pfunc4)(); // Noncompliant

void function( int ** par1,
               int *** par2, // Noncompliant
               INTPTR * par3,
               int * par4[],
               int ** par5[]) // Noncompliant
{
    int ** ptr1;
    int *** ptr2; // Noncompliant
    INTPTR * ptr3;
    int * ptr4[ 10 ];
    int ** ptr5[ 10 ]; //Noncompliant
}
```

## Compliant Solution

```
typedef int * INTPTR;
struct s {
    int ** s1;
    int ** s2;
};

struct s ** ps1;
struct s ** ps2;

int ** (*pfunc1)();
int ** (**pfunc2)();
int ** (**pfunc3)();
int ** (**pfunc4)();

void function( int ** par1,
               int ** par2,
               INTPTR * par3,
               int * par4[],
               int * par5[])
{
```

<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>

```
int ** ptr1;
int ** ptr2;
INTPTR * ptr3;
int * ptr4[ 10 ];
int * ptr5[ 10 ];

}
```

See

- MISRA C:2004, 17.5 - The declaration of objects should contain no more than 2 levels of pointer indirection
- MISRA C++:2008, 5-0-19 - The declaration of objects shall contain no more than two levels of pointer indirection
- MISRA C:2012, 18.5 - Declarations should contain no more than two levels of pointer nesting

Available In:

sonarcloud



sonarqube



Developer Edition