

- 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

"#pragma pack" should be used correctly

Analyze your code

Bug Major suspicious

#pragma pack is a non standard extension used to specify the packing alignment for structure, union and class members.

It is useful to

- remove padding and decrease the size of objects
- align members to better fit optimal cpu alignment

However, the pragma pack directives need to be correctly defined to work properly.

This rule raises an issue if:

- the specified packing value is incorrect: it can only be 1, 2, 4, 8, or 16
- a parameter is ill-formed
- the pop variant of this #pragma is called with both arguments identifier and value: such a call is undefined behavior
- a #pragma pack(push...) is performed but there is not corresponding use of #pragma pack(pop...)
- a #pragma pack(pop...) is performed but there is not corresponding use of #pragma pack(push...)
- a #pragma pack is in effect across several files: this becomes too complex, and could easily lead to undefined behavior, the same structure having different layout when seen from different translation units

## Noncompliant Code Example

```
#pragma pack(5) // Noncompliant, value is invalid
#pragma pack(2+2) // Noncompliant, value should be a literal

#pragma pack(4)
#include "myFile.h" // Noncompliant, the specified alignment is not supported

struct T {
    int i;
    short j;
    double k;
};

#pragma pack(push, r1, 16)
#pragma pack(pop, r1, 4) // Noncompliant, call to pop with too many arguments

#pragma pack(push, r2, 16)
#pragma pack(pop, r3) // Noncompliant, call to pop with no matching push

#pragma pack(push, 8) // Noncompliant, unmatched push
```

## Compliant Solution

```
#include "myFile.h"

#pragma pack(4)

struct T {
    int i;
    short j;
    double k;
};
```

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

```
#pragma pack(push, r1, 16)
#pragma pack(pop, r1)

#pragma pack(push, r2, 16)
#pragma pack(pop, r2)

#pragma pack(push, 8)
#pragma pack(pop)
```

Available In:

sonarcloud



sonarqube



Developer Edition