

- 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

"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

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

## Control should not be transferred into a complex logic block using a "goto" or a "switch" statement

Analyze your code

Code Smell Blocker lock-in cert misra-c++2008 pitfall

Having a `switch` and its cases wholly encompassed by a control structure such as a `try`, `@try`, `catch`, `@catch`, or a loop is perfectly acceptable. (`try` and `catch` are used hereafter to refer to both variants.) It is also acceptable to have a `goto` and its target label wholly encompassed in a control structure.

What is not acceptable is using a `goto` or `case` to suddenly jump into the body of a `try`, `catch`, Objective-C `@finally`, or loop structure. Tangling labels or `switch` blocks with other control structures results in code that is difficult, if not impossible to understand. More importantly, when it compiles (some of these constructs won't compile under ISO-conformant compilers), it can lead to unexpected results. Therefore this usage should be strictly avoided.

This C++ code sample, which is also applicable to Objective-C if `try` and `catch` are converted to `@try` and `@catch`, demonstrates jumping into a `switch` and into a `try` and `catch` :

### Noncompliant Code Example









```
void f ( int32_t i )
{
    if ( 10 == i )
    {
        goto Label_10; // Noncompliant; goto transfers control into try block
    }

    if ( 11 == i )
    {
        goto Label_11; // Noncompliant; goto transfers control into catch block
    }

    switch ( i )
    {
        case 1:
            try
            {
                Label_10:
                case 2: // Noncompliant; switch transfers control into try block
                    // Action
                    break;
            }
            catch ( ... )
            {
                Label_11:
                case 3: // Noncompliant; switch transfers control into catch block
                    // Action
                    break;
            }
            break;
        default:
        {
            // Default Action
            break;
        }
    }
}
```

### Compliant Solution

```
void f ( int32_t i )
{
    switch ( i )
    {
        case 1:
        case 2:
            // Action
            break;
        case 3:
            // Action
            break;
        case 10:
```

<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
<b>The right-hand operands of &amp;&amp; and    should not contain side effects</b>  Code Smell

```
default:
{
    // Default Action
    break;
}

try
{
    if ( 2 == i || 10 == i)
    {
        // Action
    }
}
catch ( ... )
{
    if (3 == i || 11 == i)
    {
        // Action
    }
}
}
```

See

- MISRA C++:2008, 15-0-3 - Control shall not be transferred into a try or catch block using goto or switch statement
- [CERT, MSC20-C.](#) - Do not use a switch statement to transfer control into a complex block

Available In:

sonarcloud



sonarqube

 Developer Edition