

-  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
"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

"goto" statements should not be used to jump into blocks

Analyze your code

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

Use of `goto` can lead to programs that are extremely difficult to comprehend and analyse, and possibly to unspecified behavior.

Unfortunately, removing `goto` from some code can lead to a rewritten version that is even more difficult to understand than the original. Therefore, limited use of `goto` is sometimes advised.

However, the use of `goto` to jump into or out of a sub-block of code, such as into the body of a `for` loop is never acceptable, because it is extremely difficult to understand and will likely yield results other than what is intended.

Noncompliant Code Example

```
void f1 (int a) {
    if (a <=0) {
        goto L2; // Noncompliant; jumps into a different block
    }

    if (a == 0) {
        {
            goto L1; // Compliant
        }
        goto L2; // Noncompliant; jumps into a block
    }

L1:
    for (int i = 0; i < a; i++) {
L2:
        //... Should only have come here with a >=0. Loop is inf
    }
}
```

Compliant Solution

```
void f1 (int a) {
    if (a <=0) {
        // ...
    }

    if (a == 0) {
        {
            goto L1; // Compliant
        }
    }

L1:
    for (int i = 0; i < a; i++) {
L2:
        //...
    }
}
```

See

- MISRA C++:2008, 6-6-1 - Any label referenced by a `goto` statement shall be declared in the same block, or in a block enclosing the `goto` statement
- MISRA C:2012, 15.3 - Any label referenced by a `goto` statement shall be declared in the same block, or in a block enclosing the `goto` statement

<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>

Available In:

 Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

[Privacy Policy](#)