


-  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

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

Analyze your code

 Code Smell  Major  based-on-misra cert

The use of increment and decrement operators in method calls or in combination with other arithmetic operators is not recommended, because:

- It can significantly impair the readability of the code.
- It introduces additional side effects into a statement, with the potential for undefined behavior.
- It is safer to use these operators in isolation from any other arithmetic operators.

Noncompliant Code Example

```
u8a = ++u8b + u8c--;  
foo = bar++ / 4;
```

Compliant Solution

The following sequence is clearer and therefore safer:

```
++u8b;  
u8a = u8b + u8c;  
u8c--;  
foo = bar / 4;  
bar++;
```

See

- MISRA C:2004, 12.1 - Limited dependence should be placed on the C operator precedence rules in expressions.
- MISRA C:2004, 12.13 - The increment (++) and decrement (--) operators should not be mixed with other operators in an expression.
- MISRA C++:2008, 5-2-10 - The increment (++) and decrement (--) operator should not be mixed with other operators in an expression.
- MISRA C:2012, 12.1 - The precedence of operators within expressions should be made explicit
- MISRA C:2012, 13.3 - A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that cause by the increment or decrement operator
- [CERT, EXP30-C](#) - Do not depend on the order of evaluation for side effects
- [CERT, EXP50-CPP](#) - Do not depend on the order of evaluation for side effects
- [CERT, EXP05-J](#) - Do not follow a write by a subsequent write or read of the same object within an expression

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>