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

The value of a complex expression should only be cast to a type that is narrower and of the same signedness as the underlying type of the expression

Analyze your code

Code Smell   Major   based-on-misra

If a cast is to be used on any complex expression, the type of cast that may be applied is severely restricted. As explained in MISRA C 2004, section 6.10, conversions on complex expressions are often a source of confusion and it is therefore wise to be cautious. In order to comply with these rules, it may be necessary to use a temporary variable and introduce an extra statement.

Noncompliant Code Example

```
... (float32_t)(f64a + f64b)
... (float64_t)(f32a + f32b) // Noncompliant
... (float64_t)f32a
... (float64_t)(s32a / s32b) // Noncompliant
... (float64_t)(s32a > s32b) // Noncompliant
... (float64_t)s32a / (float32_t)s32b
... (uint32_t)(u16a + u16b) // Noncompliant
... (uint32_t)u16a + u16b
... (uint32_t)u16a + (uint32_t)u16b
... (int16_t)(s32a - 12345)
... (uint8_t)(u16a * u16b)
... (uint16_t)(u8a * u8b) // Noncompliant
... (int16_t)(s32a * s32b)
... (int32_t)(s16a * s16b) // Noncompliant
... (uint16_t)(f64a + f64b) // Noncompliant
... (float32_t)(u16a + u16b) // Noncompliant
... (float64_t)foo1(u16a + u16b)
... (int32_t)buf16a[u16a + u16b]
```

See

- MISRA C:2004, 10.3 - The value of a complex expression of integer type may only be cast to a type that is narrower and of the same signedness as the underlying type of the expression.
- MISRA C:2004, 10.4 - The value of a complex expression of floating type may only be cast to a narrower floating type.

See Also

- MISRA C:2004, section 6.10

Available In:

sonarcloud | sonarqube Developer Edition

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

Printf-style format strings should not lead to unexpected behavior at runtime

Bug

Recursion should not be infinite

Bug

Resources should be closed

Bug

Hard-coded credentials are security-sensitive

Security Hotspot

"goto" should jump to labels declared later in the same function

Code Smell

Only standard forms of the "defined" directive should be used

Code Smell

Switch labels should not be nested inside non-switch blocks

Code Smell