sonar RULES

Products

- 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

## Appropriate char types should be used for character and integer values

Analyze your code

Code Smell | Minor | based-on-misra cert confusing

There are three distinct `char` types, (plain) `char`, `signed char` and `unsigned char`. `signed char` and `unsigned char` should only be used for numeric data, and plain `char` should only be used for character data. Since it is implementation-defined, the signedness of the plain `char` type should not be assumed.

**Noncompliant Code Example**

```
signed char a = 'a'; // Noncompliant, explicitly signed
unsigned char b = '\r'; // Noncompliant, explicitly unsigned
char c = 10; // Noncompliant

unsigned char d = c; // Noncompliant, d is explicitly signed
char e = a; // Noncompliant, a is explicitly signed while e i
```

**Compliant Solution**

```
char a = 'a';
char b = '\r';
unsigned char c = 10;
signed char c = 10;
```

**Exceptions**

- Since the integer value 0 is used as a sentinel for the end of a string, converting this value to char is ignored.

**See**

- MISRA C:2004, 6.1 - The plain char type shall be used only for the storage and use of character values
- MISRA C:2004, 6.2 - signed and unsigned char type shall be used only for the storage and use of number values
- MISRA C++:2008, 5-0-11 - The plain char type shall only be used for the storage and use of character values
- MISRA C++:2008, 5-0-12 - signed char and unsigned char type shall only be used for the storage and use of numeric values
- CERT, INT07-C. - Use only explicitly signed or unsigned char type for numeric values
- CERT, STR00-C. - Represent characters using an appropriate type
- CERT, STR04-C. - Use plain char for characters in the basic character set

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