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

C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code

All rules 311
Vulnerability 13
Bug 74
Security Hotspot 18
Code Smell 206
Quick Fix 14

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

XML parsers should not be vulnerable to XXE attacks

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

Freed memory should not be used

"if ... else if" constructs should end with "else" clauses

Analyze your code

Code Smell Critical based-on-misra cert

This rule applies whenever an `if` statement is followed by one or more `else if` statements; the final `else if` should be followed by an `else` statement.

The requirement for a final `else` statement is defensive programming.

The `else` statement should either take appropriate action or contain a suitable comment as to why no action is taken. This is consistent with the requirement to have a final `default` clause in a `switch` statement.

Noncompliant Code Example

```
if (x == 0) {
  doSomething();
} else if (x == 1) {
  doSomethingElse();
}
```

Compliant Solution

```
if (x == 0) {
  doSomething();
} else if (x == 1) {
  doSomethingElse();
} else {
  error();
}
```

Exceptions

When all branches of an `if-else if` end with `return`, `break` or `throw`, the code that comes after the `if` implicitly behaves as if it was in an `else` clause. This rule will therefore ignore that case.

See

- MISRA C:2004, 14.10 - All if...else if constructs shall be terminated with an else clause.
- MISRA C++:2008, 6-4-2 - All if...else if constructs shall be terminated with an else clause.
- MISRA C:2012, 15.7 - All if...else if constructs shall be terminated with an else statement
- CERT, MSC01-C. - Strive for logical completeness
- CERT, MSC57-J. - Strive for logical completeness

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition