

- 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

"memcpy" should only be called with pointers to trivially copyable types with no padding

Bug

"enum" members other than the first one should not be explicitly initialized unless all members are explicitly initialized

Analyze your code

Code Smell Major based-on-misra

If an enumerator list is given with no explicit initialization of members, then C/C++ allocates a sequence of integers starting at zero for the first element and increasing by one for each subsequent element.

An explicit initialization of the first element, as permitted by this rule, forces the allocation of integers to start at the given value. When adopting this approach it is essential to ensure that the initialization value used is small enough that no subsequent value in the list will exceed the `int` storage used by enumeration constants.

Explicit initialization of all items in the list, which is also permissible, prevents the mixing of automatic and manual allocation, which is error prone.

However, it is then the responsibility of the developer to ensure that all values are in the required range, and that values are not unintentionally duplicated.

### Noncompliant Code Example

```
enum color { red = 3, blue, green, yellow = 5 }; // Noncompliant
```

### Compliant Solution

```
enum color { red = 3, blue = 4, green = 5, yellow = 5 }; // Compliant
```

### See

- MISRA C:2004, 9.3 - In an enumerator list, the "=" construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.
- MISRA C++:2008, 8-5-3 - In an enumerator list, the "=" construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.

Available In:

sonarlint sonarcloud sonarqube Developer Edition

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