



ABAP

Apex

C C

C++

CloudFormation

COBOL

C#

3 CSS

 \mathbb{X} Flex

-GO Go

5 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

ΑII 311 6 Vulnerability (13) rules

₩ Bug (74)

Security Hotspot

18

⊗ Code 206 Smell

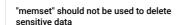
O Quick 14

Analyze your code

Tags

Conditionally executed code should

Search by name.



6 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

"pthread_mutex_t" should be properly initialized and destroyed

Bua

"pthread_mutex_t" should not be consecutively locked or unlocked

Bug

Functions with "noreturn" attribute should not return

₩ Bua

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

🖷 Bug

we based-on-misra cert unused suspicious 👬 Bug 🔷 Major 🚱 pitfall

Conditional expressions which are always true or false can lead to dead code. Such code is always buggy and should never be used in production.

Noncompliant Code Example

be reachable

```
a = false;
if (a) { // Noncompliant
 doSomething(); // never executed
if (!a || b) { // Noncompliant; "!a" is always "true", "b" is
  doSomething();
  doSomethingElse(); // never executed
```

- MISRA C:2004, 13.7 Boolean operations whose results are invariant shall not be permitted.
- MISRA C:2012, 14.3 Controlling expressions shall not be invariant
- MITRE, CWE-570 Expression is Always False
- MITRE, CWE-571 Expression is Always True
- CERT, MSC12-C. Detect and remove code that has no effect or is never executed

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube

Developer

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy

Stack allocated memory and nonowned memory should not be freed

R
Bug

Closed resources should not be
accessed
Bug

Dynamically allocated memory should
be released
Bug

Freed memory should not be used