



ABAP

Apex

С

C++

CloudFormation

COBOL

C#

CSS

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

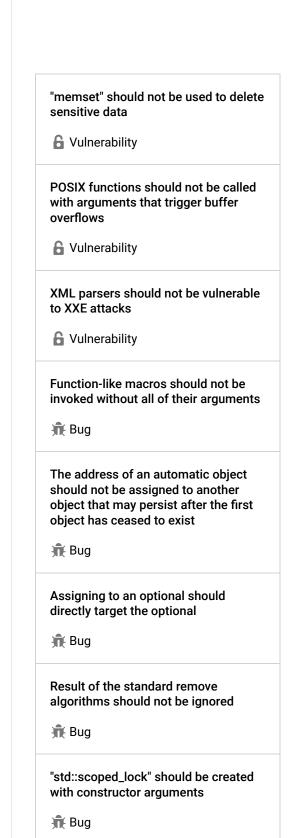


C++ static code analysis

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

• Security ⊗ Code (436) Quick 68 Fix ΑII 578 6 Vulnerability (13) **R** Bug (111) rules Hotspot

Tags



Objects should not be sliced

Immediately dangling references

"pthread_mutex_t" should be unlocked in the reverse order they were locked

"pthread_mutex_t" should be properly

"pthread_mutex_t" should not be consecutively locked or unlocked

initialized and destroyed

should not be created

📆 Bug

👬 Bug

📆 Bug

📆 Bug

"/*" and "//" should not be used Analyze your code within comments ☼ Code Smell ♥ Minor ② based-on-misra cert confusing Defining a nested single-line comment within a multi-line comment invites errors. It may lead a developer to wrongly think that the lines located after the single-line comment are not part of the comment. If a comment starting sequence, /* or //, occurs within a /* comment, is it quite likely to be caused by a missing */ comment ending sequence. If a comment starting sequence occurs within a // comment, it is probably because a region of code has been commented-out using //.

Search by name...

Noncompliant Code Example

/* some comment, end comment marker accidentally omitted // Make sure this function is called in a thread safe context Perform Critical Safety Function(X); /* this comment is non-compliant */

Exceptions

The sequence // is permitted within a // comment.

See

- CERT, MSC04-C. Use comments consistently and in a readable fashion
- MISRA C:2004, 2.3 The character sequence /* shall not be used within a
- MISRA C++:2008, 2-7-1 The character sequence /* shall not be used within a C-style comment.
- MISRA C:2012, 3.1 The character sequences /* and // shall not be used within a comment

Available In:

sonarlint sonarcloud sonarqube Develop Edition

© 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**

I
🖟 Bug
"std::move" and "std::forward" should not be confused
∰ Bug
A call to "wait()" on a "std::condition_variable" should have a condition
n Bug
A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast
ਜ਼ਿ Bug
Functions with "noreturn" attribute should not return
👬 Bug
RAII objects should not be temporary
्रे Bug
"memcmp" should only be called with pointers to trivially copyable types with no padding
🙃 Bug
"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types
🙃 Bug
"std::auto_ptr" should not be used
n Bug
Destructors should be "noexcept"
🖟 Bug