

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

o Security **⊗** Code (436) Quick 68 Fix ΑII 578 6 Vulnerability (13) **R** Bug (111) rules Hotspot Tags Search by name... "memset" should not be used to delete sensitive data "#include" paths should be portable Analyze your code Vulnerability pitfall

practice, it slightly differs in different systems.

ignored on Windows but considered on Unix).

This rule raises an issue when:

**Noncompliant Code Example** 

Available In:

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 Assigning to an optional should directly target the optional 📆 Bug Result of the standard remove algorithms should not be ignored 📆 Bug

with constructor arguments

Objects should not be sliced

Immediately dangling references

should not be created

initialized and destroyed

"pthread\_mutex\_t" should not be consecutively locked or unlocked

📆 Bug

📆 Bug

📆 Bug

📆 Bug

📆 Bug

© 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** 

The way an #include directive finds an actual file is implementation-defined, and in

• The case of the file in the #include directive does not match the case of the

• The file name in the #include directive contains trailing spaces (they would be

#include "Foo.h" // Noncompliant if the actual file name is

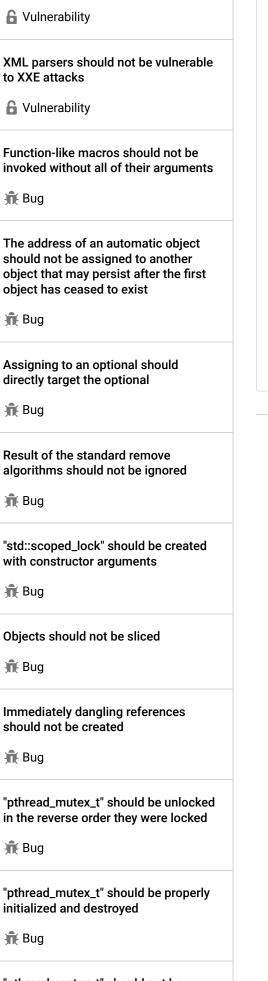
#include "bar.h " // Noncompliant, trailing space

sonarlint 😁 | sonarcloud 🙆 | sonarqube | Developer

file on the disk (the inclusion would not work on a case-sensitive OS),

Therefore, a good practice is to identify the files to include in the most

straightforward way possible to reduce the risk of inconsistent behaviors.



I
🖟 Bug
"std::move" and "std::forward" should not be confused
<b>∰</b> 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
<b>ਜ਼ਿ</b> 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
<b>n</b> Bug
Destructors should be "noexcept"
🖟 Bug