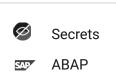
O Quick 68 Fix





APEX Apex

C++

CloudFormation

COBOL COBOL

C# C#

3 CSS

⊠ Flex

€60 Go

5 HTML

近 Java

Js JavaScript

Kotlin

Kubernetes

Objective C

PHP

PL/I

PL/SQL

Python

RPG RPG

Ruby

Scala

Swift

Terraform

Text

Ts TypeScript

T-SQL

VB VB.NET

VB6 VB6

XML XML



C++ static code analysis

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

Tags

"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 Assigning to an optional should directly target the optional 📆 Bug Result of the standard remove algorithms should not be ignored 📆 Bug "std::scoped_lock" should be created with constructor arguments

🖷 Bug

📆 Bug

📆 Bug

📆 Bug

📆 Bug

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

"::" operator should be used to Analyze your code access global variables and **functions** pitfall While it is possible to access a global variable or function without using the :: operator, it can be considered to be misleading because it might imply to the readers of your code that this is a local or class variable/function and not a global one. Being explicit also allows more freedom in naming local variables without the chance of clashing with global names. **Noncompliant Code Example** int a = 10;int main() int b = a;// Noncompliant **Compliant Solution** int a = 10;int main() { // Compliant int b = ::a; Available In: sonarlint 😁 | sonarcloud 🙆 | sonarqube | Developer Edition

⊗ Code (436)

Search by name...

© 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