



ABAP

Apex

C

C++

CloudFormation

COBOL

C#

CSS

Flex

Go =GO

HTML 5

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

"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 Objects should not be sliced 📆 Bug Immediately dangling references should not be created 📆 Bug "pthread_mutex_t" should be unlocked in the reverse order they were locked

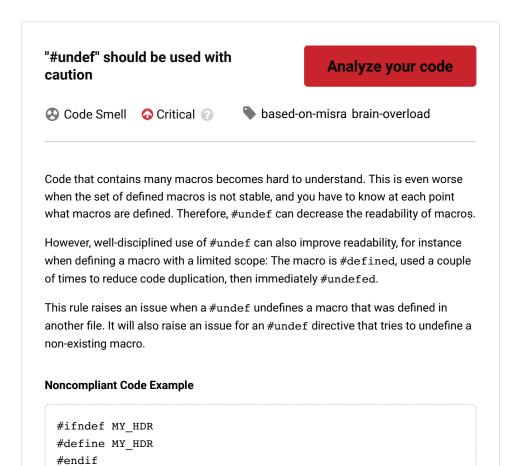
📆 Bug

📆 Bug

"pthread_mutex_t" should be properly

"pthread_mutex_t" should not be consecutively locked or unlocked

initialized and destroyed



/* Noncompliant */

Search by name...

Compliant Solution

#undef MY_HDR

#define LEVEL(i) int const i = #i LEVEL(Debug); LEVEL(Warning); LEVEL(Error); #undef LEVEL

See

- MISRA C:2004, 19.6 #undef shall not be used.
- MISRA C++:2008, 16-0-3 #undef shall not be used.
- MISRA C:2012, 20.5 #undef should not be used

Available In:

sonarlint 😊 | sonarcloud 🙆 | sonarqube | Developer 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 |