



ABAP

Apex

С

C++

CloudFormation

COBOL

C#

CSS

Flex

Go =GO

5 HTML

JavaScript

Java

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

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

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

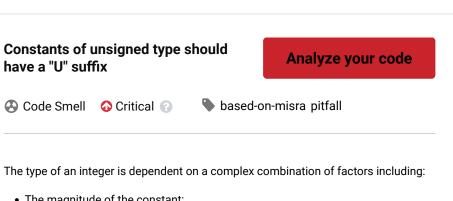
"pthread_mutex_t" should be unlocked in the reverse order they were locked

📆 Bug

"pthread_mutex_t" should be properly initialized and destroyed

📆 Bug

"pthread_mutex_t" should not be consecutively locked or unlocked



Search by name...

- · The magnitude of the constant;
- The implemented sizes of the integer types;
- · The presence of any suffixes;
- The number base in which the value is expressed (i.e. decimal, octal or hexadecimal).

For example, the value 0x8000 is of type unsigned int in a 16-bit environment, but of type (signed) int in a 32-bit environment.

- Any value with a "U" suffix is of unsigned type;
- An unsuffixed decimal value less than 2^31 is of signed type.

But:

- An unsuffixed hexadecimal value greater than or equal to 2^15 may be of signed or unsigned type;
- For C90, an unsuffixed decimal value greater than or equal to 2^31 may be of signed or unsigned type.

In C++, if an overload set includes candidates for an unsigned int and an int, then the overload that would be matched by 0x8000 is therefore dependent on the implemented integer size. Adding a "U" suffix to the value specifies that it is unsigned.

See

- MISRA C:2004, 10.6 A "U" suffix shall be applied to all constants of unsigned
- MISRA C++:2008, 2-13-3 A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.
- MISRA C:2012, 7.2 A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type.

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

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