



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



ΑII

rules

578

C++ static code analysis

6 Vulnerability (13)

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

R Bug (111)

"memset" should not be used to delete "std::bit_cast" should be used to sensitive data reinterpret binary representation instead of "std::memcpy"

since-c++20 bad-practice pitfall

• Security

Tags

Hotspot

Analyze your code

Search by name...

Quick 68 Fix

std::bit_cast is one of the standard functions working with the binary representation. Together with other bit-level functions, it is defined in the <bits> header introduced by C++20.

⊗ Code (436)

std::bit_cast standardizes the diverse and sub-optimal approaches of reinterpreting a value as being of a different type of the same length preserving its binary representation.

Before C++20 the correct way to reinterpret a value was a call to std::memcpy, copying the exact binary representation from a variable of one type into a variable of another. Although canonical, the use of std::memcpy might still be confusing, it is verbose, and it might introduce performance overhead if the compiler does not recognize the idiom and does not remove the function call.

In contrast, std::bit_cast clearly states the intent and is guaranteed to map to an optimal implementation.

This rule reports the uses of std::memcpy that can be replaced by std::bit_cast.

Noncompliant Code Example

static_assert(sizeof(float) == sizeof(uint32_t)); float src = 1.0f; uint32_t dst; std::memcpy(&dst, &src, sizeof(float)); // Noncompliant: verb

Compliant Solution

float src = 1.0f; auto dst = std::bit cast<uint32 t>(src); // Compliant

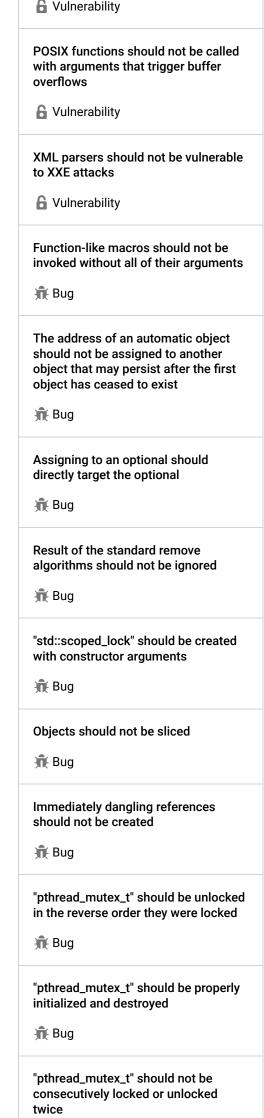
See

Other common patterns predating C++20:

- {rule:cpp:S3630} replacing std::reinterpret_cast with std::bit_cast.
- {rule:cpp:S871} replacing C-style cast with std::bit_cast.

Available In:

sonarlint in sonarcloud color 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