sonar RULES   Products

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- 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

All rules 578 | Vulnerability 13 | Bug 111 | Security Hotspot 18 | Code Smell 436 | Quick Fix 68

Tags | Search by name...

"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 twice

Macros used in preprocessor directives should be defined before use

Analyze your code

Bug   Major   based-on-misra preprocessor

An attempt to use an undefined identifier may elicit a warning from the preprocessor. Or it may not; the preprocessor may simply assume that the undefined token has a value of 0.

Therefore macro identifiers should not be used in preprocessor directives until after they have been defined, and this limited usage should be enforced with the use of definition tests.

Noncompliant Code Example

```
#if x > 0  /* x assumed to be zero if not defined */
#include SOMETHING_IMPORTANT
#endif

#ifdef y  /* Okay; y is not evaluated */
#if y > 0 /* Okay; y must be defined to reach this point */
...
#endif
#endif
```

Compliant Solution

```
#define x 10
...
#if x > 0
#include SOMETHING_IMPORTANT
#endif

#if defined ( y ) && ( y > 0 )  /* more compact form, same re
...
#endif
```

See

- MISRA C:2004, 19.11 - All macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef preprocessor directives and the defined() operator.
- MISRA C:2012, 20.9 - All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation

Available In:
sonarlint sonarcloud sonarqube Developer Edition

Bug

"std::move" and "std::forward" should not be confused

Bug

A call to "wait()" on a "std::condition_variable" should have a condition

Bug

A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast

Bug

Functions with "noreturn" attribute should not return

Bug

RAII objects should not be temporary

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

Bug

Destructors should be "noexcept"

Bug