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

"/*" and "//" should not be used within comments

Analyze your code

Code Smell Minor based-on-misra cert confusing

Defining a nested single-line comment within a multi-line comment invites errors. It may lead a developer to wrongly think that the lines located after the single-line comment are not part of the comment.

If a comment starting sequence, `/*` or `//`, occurs within a `/*` comment, is it quite likely to be caused by a missing `*/` comment ending sequence.

If a comment starting sequence occurs within a `//` comment, it is probably because a region of code has been commented-out using `//`.

Noncompliant Code Example

```
/* some comment, end comment marker accidentally omitted
// Make sure this function is called in a thread safe context
Perform_Critical_Safety_Function(X);
...
/* this comment is non-compliant */
```

Exceptions

The sequence // is permitted within a // comment.

See

- CERT, MSC04-C. - Use comments consistently and in a readable fashion
- MISRA C:2004, 2.3 - The character sequence /* shall not be used within a comment.
- MISRA C++:2008, 2-7-1 - The character sequence /* shall not be used within a C-style comment.
- MISRA C:2012, 3.1 - The character sequences /* and // shall not be used within a comment

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