- 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

---

Comments should not be located at the end of lines of code

Analyze your code

⊙ Code Smell   ⊙ Minor ⓘ   🏷 convention

This rule verifies that single-line comments are not located at the ends of lines of code. The main idea behind this rule is that in order to be really readable, trailing comments would have to be properly written and formatted (correct alignment, no interference with the visual structure of the code, not too long to be visible) but most often, automatic code formatters would not handle this correctly: the code would end up less readable. Comments are far better placed on the previous empty line of code, where they will always be visible and properly formatted.

Noncompliant Code Example

```
a = b + c;    // This is a trailing comment that could be very
```

Compliant Solution

```
// This very long comment is better placed before the line of
a = b + c;
```

Exceptions

The rule will not raise an issue if the trailing comment is preceded by an opening brace, closing brace (optionally followed by a semicolon) or a preprocessor directive.

Available In:

sonarlint ⊙⊙ | **sonar**cloud ⊙ | **sonar**qube 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