

-  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

Functions/methods should not have too many lines

Analyze your code

Code Smell Major brain-overload

A function that grows too large tends to aggregate too many responsibilities.

Such functions inevitably become harder to understand and therefore harder to maintain.

Above a specific threshold, it is strongly advised to refactor into smaller functions which focus on well-defined tasks.

Those smaller functions will not only be easier to understand, but also probably easier to test.

Available In:

sonarlint | sonarcloud | 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](#)

|   |
|---|
|  Bug   |
| <b>"std::move" and "std::forward" should not be confused</b><br> Bug   |
| <b>A call to "wait()" on a "std::condition_variable" should have a condition</b><br> Bug                                     |
| <b>A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast</b><br> Bug |
| <b>Functions with "noreturn" attribute should not return</b><br> Bug   |
| <b>RAII objects should not be temporary</b><br> Bug  |
| <b>"memcmp" should only be called with pointers to trivially copyable types with no padding</b><br> Bug                    |
| <b>"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types</b><br> Bug           |
| <b>"std::auto_ptr" should not be used</b><br> Bug  |
| <b>Destructors should be "noexcept"</b><br> Bug  |