

**ABAP** 

Apex

С

C++

CloudFormation

COBOL

C#

CSS

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



## C++ static code analysis

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

All 578 6 Vulnerability (13) rules

**R** Bug (111)

nested

o Security Hotspot

⊗ Code (436)

O Quick 68 Fix

Analyze your code

Tags

Try-catch blocks should not be

Search by name...

error-handling confusing

"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

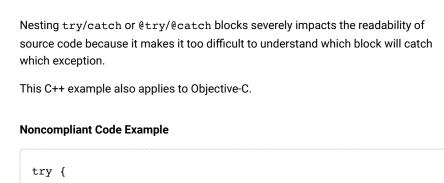
"pthread\_mutex\_t" should be unlocked in the reverse order they were locked

📆 Bug

"pthread\_mutex\_t" should be properly initialized and destroyed

📆 Bug

"pthread\_mutex\_t" should not be consecutively locked or unlocked



```
// Noncompliant
 try {
    doSomething();
 } catch (RuntimeException e) {
    /* Ignore */
 doSomethingElse();
} catch (Exception e) {
```

```
try {
  dedicatedMethod();
                                             // Compliant
 doSomethingElse();
} catch (Exception e) {
  /* ... */
/* ... */
private void dedicatedMethod() {
                                             // Compliant
    doSomething();
  } catch (RuntimeException e) {
    /* Ignore */
}
```

Available In:

**Compliant Solution** 

sonarlint 😊 | sonarcloud 🟡 | sonarqube Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONAROUBE 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
<b>∰</b> 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
<b>ਜ਼ਿ</b> 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
<b>n</b> Bug
Destructors should be "noexcept"
🖟 Bug