

-  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

### Track lack of copyright and license headers

Analyze your code

 Code Smell  Blocker 

Each source file should start with a header stating file ownership and the license which must be used to distribute the application.

This rule must be fed with the header text that is expected at the beginning of every file.

#### Compliant Solution

```
/*
 * SonarQube, open source software quality management tool.
 * Copyright (C) 2008-2013 SonarSource
 * mailto:contact AT sonarsource DOT com
 *
 * SonarQube is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; eith
 * version 3 of the License, or (at your option) any later ve
 *
 * SonarQube is distributed in the hope that it will be usefu
 * but WITHOUT ANY WARRANTY; without even the implied warrant
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General
 * along with this program; if not, write to the Free Softwar
 * Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1
 */
```

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>  Bug
<b>A call to "wait()" on a "std::condition_variable" should have a condition</b>  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>  Bug
<b>Functions with "noreturn" attribute should not return</b>  Bug
<b>RAII objects should not be temporary</b>  Bug
<b>"memcmp" should only be called with pointers to trivially copyable types with no padding</b>  Bug
<b>"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types</b>  Bug
<b>"std::auto_ptr" should not be used</b>  Bug
<b>Destructors should be "noexcept"</b>  Bug