



ABAP

- _{АРЕХ} Арех
- **c** C
- C++
- CloudFormation
- COBOL COBOL
- C# C#
- **E** CSS
- X Flex
- **GO** Go
- **HTML**
- 🐇 Java
- Js JavaScript
- Kotlin
- Kubernetes
- **6** Objective C
- PHP PHP
- PL/I
- PL/SQL PL/SQL
- Python
- RPG RPG
- Ruby
- Scala
- Swift
- **Terraform**
- **Text**
- тs TypeScript
- T-SQL
- VB VB.NET
- VB6 VB6
- XML XML

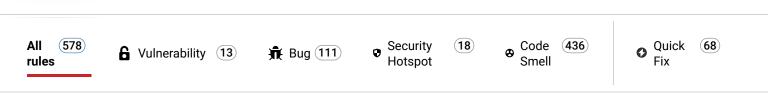


C++ static code analysis

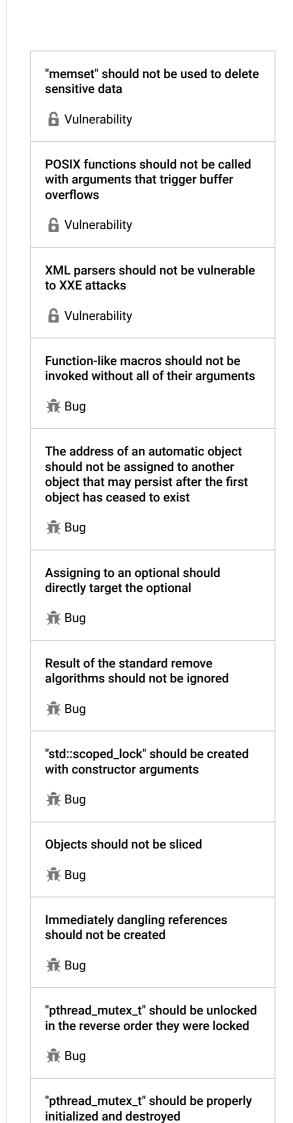
Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C++ code

See

Available In:



Tags



📆 Bug

"pthread_mutex_t" should not be consecutively locked or unlocked

```
Analyze your code
Threads should not be detached
☼ Code Smell ♥ Minor ②
                                cppcoreguidelines since-c++11 clumsy
Sometimes, you might want to make a thread run indefinitely in the background by
not binding it to its creation scope. Even though calling detach() on an
std::thread or std::jthread object would satisfy this need, it is not the easiest
way to do it: there will be no direct way to monitor and communicate with the
detached thread, the std::thread or std::jthread object is no longer
associated to any thread.
An easier alternative to satisfy this need is giving the thread a global scope. This way
the thread will run as long as the program does. The thread will not be bound to any
scope. It is also possible to do it by giving the std::thread or std::jthread a
scope that is big enough for your use case. For example, the program's main
function
Noncompliant Code Example
  void backgroundTask();
  void startBackgroundTask(){
    // Assume you want the thread to run after the end of start
    std::jthread backgroundThread(backgroundTask);
    backgroundThread.detach(); // Noncompliant
Compliant Solution
  void backgroundTask();
  std::jthread backgroundThread;
  void startBackgroundTask(){
    // Assume you want the thread to run after the end of start
    backgroundThread = std::move(std::jthread{backgroundTask});
```

Search by name...

© 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

• C++ Core Guidelines CP.26 - Don't detach() a thread

sonarlint 😔 | sonarcloud 🙆 | sonarqube | Developer

I
🖟 Bug
"std::move" and "std::forward" should not be confused
∰ 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
ਜ਼ਿ 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
n Bug
Destructors should be "noexcept"
🖟 Bug