O Quick Fix





- 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

Security

Hotspot

Tags

18

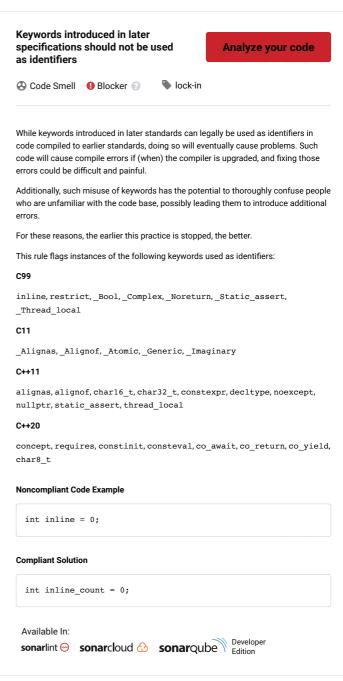
"memset" should not be used to delete sensitive data	
⑥ Vulnerability	
POSIX functions should not be called with arguments that trigger buffer overflows	
⑥ Vulnerability	
XML parsers should not be vulnerable to XXE attacks	
⑥ Vulnerability	
Function-like macros should not be invoked without all of their 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	
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	

"pthread_mutex_t" should be unlocked

in the reverse order they were locked

"pthread_mutex_t" should be properly

Bug



⊗ Code

Smell

436

Search by name.

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, ${\tt SONARSOURCE, SONARLINT, SONARQUBE\ and\ SONARCLOUD\ are\ trademarks\ of\ }$ SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy

initialized and destroyed

in Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice

in Bug

"std::move" and "std::forward" should not be confused

in Bug

A call to "wait()" on a "std::condition_variable" should have a