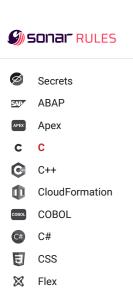
O Quick 14



-GO Go

5 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



sensitive data

6 Vulnerability

♠ Vulnerability

to XXE attacks

Vulnerability

👬 Bug

👬 Bug

Bua

Bug

Bug

🖷 Bug

should not return

with no padding

overflows

C static code analysis

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

ΑII 311 Security 6 Vulnerability (13) ₩ Bug (74) rules Hotspot

"memset" should not be used to delete

POSIX functions should not be called

XML parsers should not be vulnerable

Function-like macros should not be

The address of an automatic object

object that may persist after the first

"pthread_mutex_t" should be unlocked

in the reverse order they were locked

"pthread_mutex_t" should be properly

"pthread_mutex_t" should not be consecutively locked or unlocked

Functions with "noreturn" attribute

"memcmp" should only be called with

pointers to trivially copyable types

initialized and destroyed

should not be assigned to another

object has ceased to exist

invoked without all of their arguments

with arguments that trigger buffer

Pointer and reference local variables should be "const" if the

Tags

corresponding object is not

18

Analyze your code

Search by name.

A Code Smell

modified





based-on-misra bad-practice

This rule leads to greater precision in the definition of local variables by making the developer intention about modifying the variable explicit. The const qualification shall be applied to the object pointed to, not to the pointer, since it is the object itself that is being protected.

0

⊗ Code

Smell

206

Noncompliant Code Example

```
std::string& getString();
void myfunc()
  std::string& s = getString(); // Noncompliant
  if (s.size()) {
  std::cout << s;
}
```

Compliant Solution

```
std::string& getString();
void myfunc () {
 const std::string& x = getString();
  if (s.size()) {
   std::cout << s:
  }
}
```

• MISRA C:2012, 8.13 - A pointer should point to a const-qualified type whenever possible

Available In:

sonarlint ⊖ | sonarcloud ☆ | sonarqube

© 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

Stack allocated memory and nonowned memory should not be freed

R
Bug

Closed resources should not be
accessed
Bug

Dynamically allocated memory should
be released
Bug

Freed memory should not be used