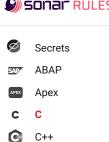
Search by name.





CloudFormation

COBOL

C#

3 CSS

 \mathbb{X} Flex

-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



C static code analysis

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

⊗ Code O Quick 14 ΑII 311 Security 18 206 6 Vulnerability (13) ₩ Bug (74) rules Hotspot Smell

Tags

"memset" should not be used to delete sensitive data 6 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 "pthread_mutex_t" should be unlocked in the reverse order they were locked "pthread_mutex_t" should be properly initialized and destroyed # Bua "pthread_mutex_t" should not be consecutively locked or unlocked # Bug Functions with "noreturn" attribute should not return

Bug

🖷 Bug

with no padding

"memcmp" should only be called with pointers to trivially copyable types

```
Methods should not be empty
                                                  Analyze your code
suspicious
There are several reasons for a method not to have a method body:
 • It is an unintentional omission, and should be fixed to prevent an unexpected
   behavior in production.
  • It is not yet, or never will be, supported. In this case an exception should be
   thrown in languages where that mechanism is available.

    The method is an intentionally-blank override. In this case a nested comment

   should explain the reason for the blank override.
Noncompliant Code Example
  void fun(int pl) {
  }
Compliant Solution
  void fun(int p1) {
    int a = doSomething(p1);
    int threshold = 42:
    if (a > threshold) {
      // ...
    }
  void fun(int p1) {
    // Intentionally unimplemented...
Exceptions
This rule doesn't raise an issue for empty class constructors or destructors. For
instance this is the only way to define user-defined default constructors.
 Available In:
                                                   Developer
 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