


































-  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 **311**

 Vulnerability **13**

 Bug **74**

 Security Hotspot **18**

 Code Smell **206**


 Quick Fix **14**

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

"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

 Bug

Functions with "noreturn" attribute should not return

 Bug

"memcpy" should only be called with pointers to trivially copyable types with no padding

 Bug

The "<stdlib.h>" functions "bsearch" and "qsort" should not be used

Analyze your code

 Bug  Major  based-on-misra unpredictable

The identifiers `bsearch` and `qsort` shall not be used and no macro with one of these names shall be expanded.

These two functions take as arguments a caller-defined comparison function. If the comparison function does not behave consistently when comparing elements, or if it modifies any of the elements, the behavior is undefined.

Note: the unspecified behavior, which relates to the treatment of elements that compare as equal, can be avoided by ensuring that the comparison function never returns 0. When two elements are otherwise equal, the comparison function could return a value that indicates their relative order in the initial array.

Further, the implementation of `qsort` is likely to be recursive and will therefore place unknown demands on stack resources. This is of concern in embedded systems because the stack is likely to have a fixed, often small, size.

See

- MISRA C:2012, 21.9 - The library functions `bsearch` and `qsort` of `<stdlib.h>` shall not be used.

Available In:

   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](#)

Stack allocated memory and non-owned memory should not be freed

 Bug

Closed resources should not be accessed

 Bug

Dynamically allocated memory should be released

 Bug

Freed memory should not be used