


































-  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

Pre-defined macros should not be defined, redefined or undefined

Analyze your code

 Code Smell

 Critical

 based-on-misra preprocessor suspicious

The standard, predefined macros, such as `__FILE__` and `__LINE__`, are primarily intended for use by the implementation, and changing them could result in undefined behavior.

This rule checks that the following predefined macros are not defined, undefined, or redefined: `assert`, `errno`, `__FILE__`, `__LINE__`, `__TIME__`, `__DATE__`, `__TIMESTAMP__`, `__COUNTER__`, `__INCLUDE_LEVEL__`, `__BASE_FILE__`, and `__Pragma`.

Noncompliant Code Example

```
#undef __LINE__
```

See

- MISRA C:2004, 20.1 - Reserved identifiers, macros and functions in the standard library shall not be defined, redefined, or undefined
- MISRA C++:2008, 17-0-1 - Reserved identifiers, macros and functions in the standard library shall not be defined, redefined, or undefined
- MISRA C:2012, 21.1 - `#define` and `#undef` shall not be used on a reserved identifier or reserved macro name

Available In:

sonarlint  | **sonarcloud**  | **sonarqube**  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