

Ruby

Scala Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

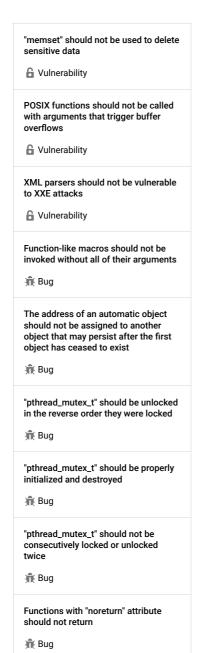
XML



C static code analysis

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

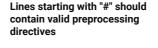




"memcmp" should only be called with

pointers to trivially copyable types with no padding

🖷 Bug



👬 Bug 🔷 Major 🕙

Tags

Analyze your code

Search by name.

Preprocessing directives (lines that start with #) can be used to conditionally include or exclude code from compilation. Malformed preprocessing directives could lead to the exclusion or inclusion of more code than was intended. Therefore all preprocessing directives should be syntactically meaningful.

based-on-misra preprocessor

Noncompliant Code Example

```
#define AAA 2
int foo(void)
  int x = 0;
#ifndef AAA
 x = 1;
#else1 /* Noncompliant */
 x = AAA;
#endif
 return x:
}
```

Compliant Solution

```
#define AAA 2
int foo(void)
  int x = 0;
  . . .
#ifndef AAA
 x = 1;
#else
  x = AAA;
#endif
 return x;
```

- MISRA C:2004, 19.16 Preprocessing directives shall be syntactically meaningful even when excluded by preprocessor.
- MISRA C++:2008, 16-0-8 If the # token appears as the first token on a line, then it shall be immediately followed by a preprocessing token.

Stack allocated memory and nonowned memory should not be freed

Rug

Closed resources should not be
accessed

Rug

Dynamically allocated memory should
be released

Bug

Freed memory should not be used

MISRA C:2012, 20.13 - A line whose first token is # shall be a valid preprocessing directive

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