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

"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

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

Non-empty statements should change control flow or have at least one side-effect

Analyze your code

Bug Major cwe based-on-misra unused

Any statement (other than a null statement, which means a statement containing only a semicolon `;`) which has no side effect and does not result in a change of control flow will normally indicate a programming error, and therefore should be refactored.

Noncompliant Code Example

```
int func(int a, int b) {
  int result = 0;
  a + b; // Noncompliant, no side effect.
  return result;
}
```

Compliant Solution

```
int func(int a, int b) {
  int result = a + b; // Compliant
  return result;
}
```

See

- MITRE, CWE-482 - Comparing instead of Assigning
- MISRA C:2004, 14.2 - All non-null statements shall either have at least one side-effect however executed, or cause control flow to change.

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