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

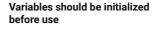**"pthread_mutex_t" should not be consecutively locked or unlocked twice**

🐞 Bug

Functions with "noreturn" attribute should not return

🐞 Bug

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

🐞 Bug

Variables should be initialized before use

[**Analyze your code**]

🐞 Bug   ⬦ Major ⓘ   🏷 cwe symbolic-execution based-on-misra

Variables should be initialized before their use to avoid unexpected behaviors due to garbage values.

Noncompliant Code Example

```
int function(int flag, int b) {
  int a;
  if (flag) {
    a = b;
  }
  return a; // Noncompliant - "a" has not been initialized in
}
```

Compliant Solution

```
int function(int flag, int b) {
  int a = 0;
  if (flag) {
    a = b;
  }
  return a;
}
```

See

- MITRE, CWE-457 - Use of Uninitialized Variable
- MISRA C:2004, 9.1 - All automatic variables shall have been assigned a value before being used.
- MISRA C++:2008, 8-5-1 - All variables shall have a defined value before they are used.

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition

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