sonar RULES

Products

- 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

Flexible array members should not be declared

Analyze your code

Code Smell | Critical | based-on-misra suspicious

Flexible array members are most likely to be used in conjunction with dynamic memory allocation.

The presence of flexible array members modifies the behaviour of the `sizeof` operator in ways that might not be expected by a programmer. The assignment of a structure that contains a flexible array member to another structure of the same type may not behave in the expected manner as it copies only those elements up to but not including the start of the flexible array member.

Noncompliant Code Example

```
#include <stdlib.h>
struct s
{
  uint16_t len;
  uint32_t data[ ]; // Noncompliant - flexible array member
} str;

struct s *copy ( struct s *s1 )
{
  struct s *s2 = malloc ( sizeof ( struct s ) + ( s1->len * s
  /* Omit malloc ( ) return check for brevity */
  *s2 = *s1; /* Only copies s1->len */
  return s2;
}
```

See

- MISRA C:2012, 18.7 - Flexible array members shall not be declared.

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