

- 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

Loops with at most one iteration should be refactored

Analyze your code

Bug Major

A loop with at most one iteration is equivalent to the use of an `if` statement to conditionally execute one piece of code. No developer expects to find such a use of a loop statement. If the initial intention of the author was really to conditionally execute one piece of code, an `if` statement should be used instead.

At worst that was not the initial intention of the author and so the body of the loop should be fixed to use the nested `return`, `break` or `throw` statements in a more appropriate way.

Noncompliant Code Example

```
for (int i = 0; i < 10; i++) { // noncompliant, loop only executes once
    printf("i is %d", i);
    break;
}
...
for (int i = 0; i < 10; i++) { // noncompliant, loop only executes once
    if (i == x) {
        break;
    } else {
        printf("i is %d", i);
        return;
    }
}
```

Compliant Solution

```
for (int i = 0; i < 10; i++) {
    printf("i is %d", i);
}
...
for (int i = 0; i < 10; i++) {
    if (i == x) {
        break;
    } else {
        printf("i is %d", i);
    }
}
```

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