



**ABAP** 

Apex

C С



CloudFormation

COBOL

C#

3 CSS

 $\mathbb{X}$ Flex

-GO Go

5 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

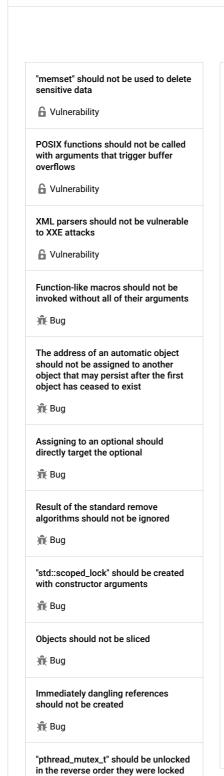
XML



# C++ static code analysis

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

⊗ Code O Quick 68 Fix ΑII 578 Security 18 436 6 Vulnerability (13) **R** Bug (111) rules Hotspot Smell



# Bug

"pthread\_mutex\_t" should be properly

### Dynamically allocated memory should be released

Tags

Analyze your code

Search by name.

cwe symbolic-execution leak 👬 Bug \phantom 🔒 Blocker 🚱 denial-of-service cert

Memory allocated dynamically with calloc(...), malloc(...), realloc(...)or  ${\tt new}$  should be released when it's not needed anymore. Failure to do so will result in a memory leak that could bring the box to its knees.

This rule raises an issue when memory is allocated and not freed in the same function. Allocated memory is ignored if a pointer to it is returned to the caller or stored in a structure that's external to the function.

#### Noncompliant Code Example

```
int fun() {
  char* name = (char *) malloc (size);
  if (!name) {
   return 1;
 // ...
 return 0; // Noncompliant, memory pointed by "name" has not
```

## **Compliant Solution**

```
int fun() {
  char* name = (char *) malloc (size);
  if (!name) {
   return 1;
  // ...
  free(name);
  return 0;
```

#### See

- MITRE, CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')
- . MEM00-C. Allocate and free memory in the same module, at the same level of
- . CERT, MEM31-C. Free dynamically allocated memory when no longer needed

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

initialized and destroyed

👬 Bug

"pthread\_mutex\_t" should not be consecutively locked or unlocked twice

<table-of-contents> Bug

"std::move" and "std::forward" should not be confused

<table-of-contents> Bug

A call to "wait()" on a "std::condition\_variable" should have a