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

Memory locations should not be released more than once

Analyze your code

Bug Blocker cwe symbolic-execution

Using `free(...)` or `delete` releases the reservation on a memory location, making it immediately available for another purpose. So releasing the same memory location twice can lead to corrupting the program's memory.

A best practice to avoid this bug calls for setting just-freed pointers to `NULL`, and always null-testing before a `free` or `delete`.

Noncompliant Code Example

```
void doSomething(int size) {
  char *cp = (char *) malloc(sizeof(char) * size);

  // ...
  if (condition) {
    free(cp);
  }

  free(cp);  // Noncompliant
}
```

Compliant Solution

```
void doSomething(int size) {
  char *cp = (char *) malloc(sizeof(char) * size);

  // ...
  if (condition) {
    free(cp);
    cp = NULL; // This will prewent freeing the same memory a
  }

  free(cp); // This is OK: if the memory was freed in the if-
  cp = NULL; // This will prevent freeing the same memory aga
}
```

See

- MITRE, CWE-415 - Double Free
- OWASP, Doubly freeing memory

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.

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

Privacy Policy