



**ABAP** 

Apex

C C

0 C++

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

ΑII 311 6 Vulnerability (13) rules

₩ Bug (74)

Security Hotspot

18

⊗ Code 206 Smell

O Quick 14

Analyze your code

Tags

Resources should be closed

Search by name.

"memset" should not be used to delete sensitive data

6 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

"pthread\_mutex\_t" should be properly initialized and destroyed

# Bua

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

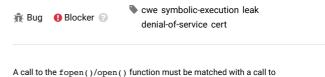
# Bug

Functions with "noreturn" attribute should not return

₩ Bua

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

🖷 Bug



 ${\tt fclose()/close.}\ Otherwise, you \ run\ the\ risk\ of\ using\ up\ all\ the\ OS's\ file\ handles,$ which could lock up not just your program but potentially everything on the box.

## Noncompliant Code Example

```
int fun() {
 FILE *f = fopen("file", "r");
  if (f == NULL) {
   return -1;
  // ...
 return 0; // Noncompliant, file f has not been closed
}
```

## **Compliant Solution**

```
int fun() {
 FILE *f = fopen("file", "r");
  if (f == NULL) {
   return -1;
  // ...
  fclose(f):
  return 0;
```

- MITRE, CWE-459 Incomplete Cleanup
- MITRE, CWE-772 Missing Release of Resource after Effective Lifetime
- CERT, FI004-J. Release resources when they are no longer needed
- CERT, FIO42-C. Close files when they are no longer needed
- Try With Resources

Available In:

sonarlint ⊖ | sonarcloud ☆ | sonarqube |

Developer

© 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

Stack allocated memory and nonowned memory should not be freed

R
Bug

Closed resources should not be
accessed
Bug

Dynamically allocated memory should
be released
Bug

Freed memory should not be used