Secrets
SAP ABAP
APEX Apex
C C
C C++
CloudFormation
COBOL COBOL
C# C#
CSS
Flex
GO Go
HTML
Java
JS JavaScript
Kotlin
Kubernetes
Objective C
php PHP
PL/I PL/I
PL/SQL PL/SQL
Python
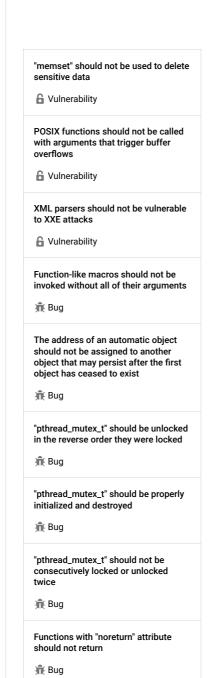RPG RPG
Ruby
Scala
Swift
Terraform
Text
TS TypeScript
T-SQL
VB VB.NET
VB6 VB6
XML XML

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

Closed resources should not be accessed

Analyze your code

🐛 Bug   ⊘ Blocker ⍰   🏷 symbolic-execution cert

Using the value of a pointer to a `FILE` object after the associated file is closed is undefined behavior.

Noncompliant Code Example

```
void fun() {
  FILE * pFile;
  pFile = fopen(fileName, "w");

  if (condition) {
    fclose(pFile);
    // ...
  }

  fclose(pFile); // Noncompliant, the file has already been c
}
```

Compliant Solution

```
void fun() {
  FILE * pFile;
  pFile = fopen(fileName, "w");

  if (condition) {
    // ...
  }

  fclose(pFile);
}
```

See

- CERT, FIO46-C. - Do not access a closed file

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