

- 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 **578**

Vulnerability **13**

Bug **111**

Security Hotspot **18**

Code Smell **436**

Quick Fix **68**

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

Assigning to an optional should directly target the optional

Bug

Result of the standard remove algorithms should not be ignored

Bug

"std::scoped_lock" should be created with constructor arguments

Bug

Objects should not be sliced

Bug

Immediately dangling references should not be created

Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked

Bug

"pthread_mutex_t" should be properly

Resources should be closed

Analyze your code

Bug

Blocker

cwe symbolic-execution leak denial-of-service cert

A call to the `fopen()`/`open()` function must be matched with a call to `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;
}
```

See

- [MITRE, CWE-459](#) - Incomplete Cleanup
- [MITRE, CWE-772](#) - Missing Release of Resource after Effective Lifetime
- [CERT, FIO04-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 Edition

initialized and destroyed

 Bug

"pthread_mutex_t" should not be
consecutively locked or unlocked
twice

 Bug

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

 Bug

A call to "wait()" on a
"std::condition_variable" should have a