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

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

"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types

Analyze your code

Bug Blocker

The functions `memcpy`, `memmove` and `memset` can only be used for objects of trivially copyable types. This includes scalar types, arrays, and trivially copyable classes.

A class type is trivially copyable if:

- One or more of the following special member functions is trivial and the rest are deleted: copy constructor, move constructor, copy assignment operator, and move assignment operator,
- It has a trivial, non-deleted destructor,
- It has trivially copyable members and base classes,
- It has no virtual functions.

Note: a default implementation, both explicit (with `=default`) or implicit (if the special member function is omitted), is considered trivial.

Noncompliant Code Example

```
class Shape {
public:
  int x;
  int y;
  virtual ~Shape(); // This makes the class non trivially cop
};

void f(Shape *dest, Shape *source)
{
    memcpy(dest, source, sizeof Shape); // Noncompliant
}
```

Compliant Solution

```
class Shape {
public:
  int x;
  int y;
  virtual ~Shape(); // This makes the class non trivially cop
};

void f(Shape *dest, Shape *source)
{
    (*dest) = (*source);
}
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition