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

Only standard forms of the "defined" directive should be used

Analyze your code

Code Smell | Blocker | based-on-misra | bad-practice

The `defined` preprocessing directive is used in the context of `#if` and `#elif` expressions to see whether a given identifier has been defined as a macro. It returns a value of 0 (false) or 1 (true), and has two valid forms, `defined IDENTIFIER` and `defined ( IDENTIFIER )`. Since it is essentially a macro existence check, it cannot take expressions as arguments.

Note that since

```
#if defined AN_IDENTIFIER
```

is equivalent to

```
#ifdef AN_IDENTIFIER
```

`defined` is most useful when there are multiple arguments to check, E.G.

```
#if defined AAA || defined BBB
```

Noncompliant Code Example

```
#if defined ( X > Y ) // Noncompliant; expressions not allowe
```

Compliant Solution

```
#if defined X && defined Y && X > Y
```

See

- MISRA C:2004, 19.14 - The defined preprocessor operator shall only be used in one of the two standard forms.
- MISRA C++:2008, 16-1-1 - The defined preprocessor operator shall only be used in one of the two standard forms.

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