



SAP ABAP

Apex Apex

**C** C

C++

CloudFormation

COBOL COBOL

C# C#

**E** CSS

X Flex

**€** Go

**5** HTML

👙 Java

Js JavaScript

Kotlin

Kubernetes

Objective C

PP PHP

PL/I

PL/SQL

Python

RPG RPG

Ruby

**S**cala

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 578 rules

6 Vulnerability 13

**R** Bug (111)

Security Hotspot

⊗ Code 436

Quick 68 Fix

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

Parameters in an overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments

Analyze your code

Code Smell



cppcoreguidelines misra-c++2008 pitfall

Overriding the default parameter value inherited from a parent class will lead to unexpected results when the child class is referenced from a pointer to the parent class.

## **Noncompliant Code Example**

```
enum E_ShapeColor {E_RED, E_GREEN, E_BLUE};

class Shape
{
  public:
    virtual void draw(E_ShapeColor color = E_RED) const
    {
        ...
    }
  };
  class Rectangle : public Shape
{
  public:
    virtual void draw(E_ShapeColor color = E_BLUE) const overri
    {
        ...
    }
  };

int main() {
    Shape *shape = new Rectangle{};
    shape->draw(); // unexpectedly calls Rectangle::draw(RED)
}
```

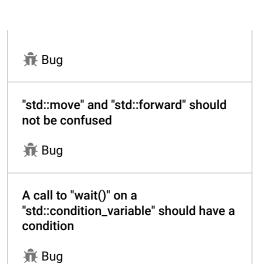
## **Compliant Solution**

```
enum E_ShapeColor {E_RED, E_GREEN, E_BLUE};

class Shape
{
  public:
    virtual void draw(E_ShapeColor color = E_RED) const
    {
        ...
    }
};

class Rectangle : public Shape
{
  public:
    virtual void draw(E_ShapeColor color) const override
    // OR: virtual void draw(E_ShapeColor color = E_RED) const
    {
        ...
    }
};

int main() {
```



A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic\_cast



Functions with "noreturn" attribute should not return

Rug Bug

RAII objects should not be temporary

📆 Bug

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

📆 Bug

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

Rug Bug

"std::auto\_ptr" should not be used

🕀 Bug

Destructors should be "noexcept"

📆 Bug

```
Shape *shape = new Rectangle{};
shape->draw(); // expectedly calls Rectangle::draw(RED)
```

## See

• MISRA C++ 2008, 8-3-1 - Parameters in a overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments.

Available In:

sonarlint sonarcloud sonarqube Developer Edition

 $@\ 2008-2022\ Sonar Source\ S.A.,\ Switzerland.\ All\ content\ is\ copyright\ protected.\ SONAR,$ 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