



<sub>АРЕХ</sub> Арех

ABAP

**c** c

C++

CloudFormation

COBOL COBOL

C# C#

**E** CSS

**⊠** Flex

**€** Go

5 HTML

🎒 Java

Js JavaScript

Kotlin

Kubernetes

Objective C

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

8 Code 436

Quick 68 Fix

Tags

✓ Search by name...

"memset" should not be used to delete sensitive data

**6** 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

<table-of-contents> 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

Use type-erased "coroutine\_handle" when applicable

Analyze your code

Code Smell

O Minor 🕝

Quick

bad-practice since-c++20

The implementation of the await\_suspend method accepts the handle to the suspended coroutine as the parameter. This parameter can be defined with either specific promise type coroutine\_handle<PromiseType> or type erased coroutine\_handle<>. The former allows await\_suspend to access the promise of the coroutine; however, it ties the implementation to a particular type. In contrast, using coroutine\_handle<> increases the reusability of the code because this parameter type supports all promise types.

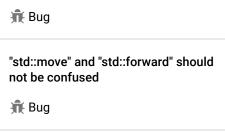
This rule raises an issue for the implementation of await\_suspend that accepts handles to a specific promise type and yet does not use that information.

## Noncompliant Code Example

```
struct Awaiter1
   Event& event;
   bool await_suspend(std::coroutine_handle<Promise> current)
     return event.register_callback([current] {
              current.resume();
            });
};
struct Awaiter2
   Event& event;
   bool await_suspend(std::coroutine_handle<PromiseA> current
     return event.register_callback([current] {
              current.resume();
            });
   bool await_suspend(std::coroutine_handle<PromiseB> current
     return event.register callback([current] {
              current.resume();
            });
};
struct Awaiter3
   Event& event;
   template<typename PromiseType>
   bool await_suspend(std::coroutine_handle<PromiseType> curr
     return event.register_callback([current] {
              current.resume();
            });
};
```

## **Compliant Solution**

```
struct Awaiter // Instead of each of Awaiter1, Awaiter2, Awai
```



A call to "wait()" on a "std::condition\_variable" should have a condition

📆 Bug

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

📆 Bug

Functions with "noreturn" attribute should not return

📆 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

```
Event& event;
   /* ... */
   bool await_suspend(std::coroutine_handle<> current) {
     return event.register_callback([current] {
              current.resume();
            });
   }
};
struct AwaiterUsingPromise
  /* ... */
  void await_suspend(std::coroutine_handle<Promise> current)
    auto wokeUpTime = std::chrono::system_clock::now() + std:
    current.promise().executor().schedule_at(wokeUpTime, curr
};
```

Available In:

sonarlint sonarcloud sonarqube Developer Edition

© 2008-2022 SonarSource 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