C++ static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C++ code

"std::filesystem::path" should be used to represent a file path

Analyze your code

Code Smell | Major | performance since-c++17 clumsy

Since C++17, the class `std::filesystem::path` can be used to store a file path. Compared to a regular string, it offers several advantages:

- Having a dedicated type makes the intention clear
- This class stores the path with an encoding that is appropriate to the OS where the program runs
- It provides several functions that make it more convenient to manipulate than a `string` (for instance `operator/` for concatenations)
- It provides a normalized way to specify the path, easing the portability of the code (on Windows and Linux, the native way is equivalent to the normalized way, which reduces overhead).

This rule raises an issue when the same `string` is converted several times to a `path` because it indicates that a single path object could have been used in all occurrences. Additionally, it can also be more efficient, since a conversion from `string` to `path` may require a change of encoding and a memory allocation.

Noncompliant Code Example

```
std::string getUserData();
namespace fs = std::filesystem;
void f() {
  std::string const filePath = getUserData();
  if (fs::exists(filePath)) {
    logTime(fs::last_write_time(filePath)); // Noncompliant
  }
}
```

Compliant Solution

```
std::string getUserData();
namespace fs = std::filesystem;
void f() {
  fs::path const filePath = getUserData();
  if (fs::exists(filePath)) {
    logTime(fs::last_write_time(filePath)); // Compliant
  }
}
```

Available In:
sonarlint | sonarcloud | sonarqube Developer Edition

Bug

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

Bug

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

Bug

"std::auto_ptr" should not be used

Bug

Destructors should be "noexcept"

Bug