sonar RULES

Products

- 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

"std::chrono" components should be used to operate on time

Analyze your code

Code Smell   Major   since-c++20 confusing

The chrono library, introduced in C++20, provides support for calendars, time zones, and i/o formatting and parsing operations on time-related objects.

chrono is a better alternative to the C/POSIX functions that operate on time_t, tm, or timespec types. In comparison to C facilities, it provides a better integration with other components of the C++ standard library: (iostreams and format). Also, it supports compile-time computation and it is thread safe.

This rule raises an issue on any use of C/POSIX functions that can be replaced with one of the std::chrono components:

- querying for current time (time, timespec_get, clock_gettime)
- date to time-point conversion (mktime, gmtime, localtime)
- time serialization (ctime, asctime, strftime)
- time parsing (strptime)

Noncompliant Code Example

```
int currentMonth() {
  std::time_t tp;
  std::time(&tp);
  std::tm* date = std::gmtime(&tp);
  return date->tm_mon + 1;
}

std::chrono::system_clock::time_point makeSomeDay() {
  // Creates time_point corresponding to 2020-09-04
  std::tm date{};
  date.tm_year = 120;
  date.tm_mon = 8;
  date.tm_mday = 4;
  std::time_t t = std::mktime(&date); // Noncompliant
  return std::chrono::system_clock::from_time_t(t);
}

std::optional<int> yearOfTimePoint(std::chrono::system_clock:
  std::time_t t = std::chrono::system_clock::to_time_t(tp);
  std::tm* date = std::gmtime(&t); // Noncompliant
  if (!date)
    return std::nullopt;
  return date->tm_year + 1900;
}

std::string toIsoString(std::chrono::system_clock::time_point
  std::time_t t = std::chrono::system_clock::to_time_t(tp);
  std::tm* date = std::gmtime(&t);  // Noncompliant
  if (!date)
    throw InvalidDate();

  std::string buffer(100, ' ');
  std::size_t written = std::strftime(buffer.data(), buffer.s
  buffer.resize(written);
  return buffer;
}
```

Compliant Solution

```
std::chrono::month currentMonth() {
```

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

```
  using namespace std::chrono;
  auto dp = floor<days>(system_clock::now());
  return year_month_day(dp).month();
}

std::chrono::system_clock::time_point makeSomeDay() {
  using namespace std::chrono;
  return sys_days(2020y/September/4);
}

std::optional<std::chrono::year> yearOfTimePoint(std::chrono:
  using namespace std::chrono;
  year_month_day date(floor<days>(tp));
  if (!date.ok())
    return std::nullopt;
  return date.year();
}

std::string toIsoString(std::chrono::system_clock::time_point
  return std::format("{:%F}", tp);
}
```

Available In:

sonarlint sonarcloud sonarqube Developer Edition