C++ static code analysis

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

"pthread_mutex_t" should not be consecutively locked or unlocked twice

Analyze your code

Bug Blocker symbolic-execution multi-threading

Mutexes are synchronization primitives that allow to manage concurrency.

- *non recursive mutexes* are targeted by this rule. They can be locked/unlocked only once. Any locking/unlocking sequence that contains two consecutive identical operations leads to an undefined behaviour.
- *recursive mutexes* are not target by this rule. They can be locked several times and unlocked several times as long as the number of locks/unlocks is the same.

This rule raises an issue when a `pthread_mutex_t` is locked or unlocked several times in a row. We assume that all `pthread_mutex_t` are non-recursive (this is the most common case).

Noncompliant Code Example

```
pthread_mutex_t mtx1;

void bad1(void)
{
  pthread_mutex_lock(&mtx1);
  pthread_mutex_lock(&mtx1);
}

void bad2(void)
{
  pthread_mutex_unlock(&mtx1);
  pthread_mutex_unlock(&mtx1);
}
```

Compliant Solution

```
pthread_mutex_t mtx1;

void ok(void)
{
  pthread_mutex_lock(&mtx1);
  pthread_mutex_unlock(&mtx1);
}
```

See

- The Open Group pthread_mutex_init, pthread_mutex_destroy

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