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

Freed memory should not be used

Analyze your code

Bug | Blocker | cwe symbolic-execution cert

Once a block of memory has been `freed`, it becomes available for other memory requests. Whether it's re-used immediately, some time later, or not at all is random, and may vary based on load. Because of that randomness, tests may pass when running locally, but the odds are that such code will fail spectacularly in production by returning strange values, executing unexpected code, or causing a program crash.

Noncompliant Code Example

```
char *cp = malloc(sizeof(char)*10);

// ...
free(cp);

cp[9] = 0;  // Noncompliant
```

See

- MITRE, CWE-416 - Use After Free
- CERT, MEM30-C. - Do not access freed memory
- CERT, MEM50-CPP. - Do not access freed memory
- CERT, EXP54-CPP. - Do not access an object outside of its lifetime

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