C++ static code analysis

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

Insecure functions should not be used

Analyze your code

Vulnerability Critical cwe sans-top25 owasp cert

When using typical C functions, it’s up to the developer to make sure the size of the buffer to be written to is large enough to avoid buffer overflows. Buffer overflows can cause the program to crash at a minimum. At worst, a carefully crafted overflow can cause malicious code to be executed.

This rule reports use of the following insecure functions, for which knowing the required size is not generally possible: `gets()` and `getpw()`.

In such cases. The only way to prevent buffer overflow while using these functions would be to control the execution context of the application.

It is much safer to secure the application from within and to use an alternate, secure function which allows you to define the maximum number of characters to be written to the buffer:

- `fgets` or `gets_s`
- `getpwuid`

Noncompliant Code Example

```
gets(str); // Noncompliant; `str` buffer size is not checked
```

Compliant Solution

```
gets_s(str, sizeof(str)); // Prevent overflows by enforcing a
```

See

- OWASP Top 10 2017 Category A9 - Using Components with Known Vulnerabilities
- MITRE, CWE-676 - Use of Potentially Dangerous Function
- MITRE, CWE-119 - Improper Restriction of Operations within the Bounds of a Memory Buffer
- SANS Top 25 - Risky Resource Management

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