sonar RULES

Products

C static code analysis

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

All rules 311 | Vulnerability 13 | Bug 74 | Security Hotspot 18 | Code Smell 206 | Quick Fix 14

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

"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
Bug

Functions with "noreturn" attribute should not return
Bug

"memcmp" should only be called with pointers to trivially copyable types with no padding
Bug

"switch" statements should not be nested

Analyze your code

Code Smell | Critical | pitfall

Nested `switch` structures are difficult to understand because you can easily confuse the cases of an inner `switch` as belonging to an outer statement. Therefore nested `switch` statements should be avoided.

Specifically, you should structure your code to avoid the need for nested `switch` statements, but if you cannot, then consider moving the inner `switch` to another function.

Noncompliant Code Example

```
void func(int n, int m) {

  switch (n) {
    case 1:
      // ...
    case 2:
      // ...
    case 3:
      switch (m) {  // Noncompliant
    case 4:  // Bad indentation makes this particularly hard
      // ...
    case 5:
      // ...
    case 6:
      // ...
    }
    case 4:
      // ...
    default:
      // ...
  }
}
```

Compliant Solution

Stack allocated memory and non-owned memory should not be freed

Bug

Closed resources should not be accessed

Bug

Dynamically allocated memory should be released

Bug

Freed memory should not be used

```
void func(int n, int m) {

  switch (n) {
    case 1:
      // ...
    case 2:
      // ...
    case 3:
      int m2 = handle_m(m);
    case 4:
      // ...
    default:
      // ...
  }
}
```

Available In:

sonarlint sonarcloud sonarqube Developer Edition