

- 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 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

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

Bug

"case" ranges should not be empty

Analyze your code

Bug Major ? gnu

The GNU compiler extension that allows cases to be specified with ranges will only recognize ranges specified from a smaller value to a larger value. Flip the order and the range will evaluate as empty.

Noncompliant Code Example

```
switch (i) {
  case 0:
    //...
    break;
  case 1 ... 2:
    //...
    break;
  case 5 ... 3: // Noncompliant
    //...
    break;
```

Compliant Solution

```
switch (i) {
  case 0:
    //...
    break;
  case 1 ... 2:
    //...
    break;
  case 3 ... 5
    //...
    break;
```

Available In:

sonarlint

sonarcloud

sonarqube

Developer Edition

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