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

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

## Conditional operators should not be nested

Analyze your code

Code Smell | Major | confusing

Just because you *can* do something, doesn't mean you should, and that's the case with nested ternary operations. Nesting ternary operators results in the kind of code that may seem clear as day when you write it, but six months later will leave maintainers (or worse - future you) scratching their heads and cursing.

Instead, err on the side of clarity, and use another line to express the nested operation as a separate statement.

### Noncompliant Code Example

```
int max(int p1, int p2, int p3) {
  return p1 > p2 ? (p1 > p3 ? p1 : p3) : (p2 > p3 ? p2 : p3);
}
```

### Compliant Solution

```
int max(int p1, int p2, int p3) {
  if (p1 > p2) {
    return p1 > p3 ? p1 : p3;
  } else {
    return p2 > p3 ? p2 : p3;
  }
}
```

### Exceptions

For C++11 mode only, the issue is not raised for ternary operators used inside `constexpr` functions. In C++11 such functions are limited to just a return statement, so the use of a ternary operator is required in them. This restriction is lifted in later standards, and thus issues are raised.

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**