- 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

Comma operator should not be used

[Analyze your code]

⊙ Code Smell   ⊘ Major ⍰   🏷 based-on-misra

The comma operator takes two expressions, executes them from left to right, and returns the result of the second one. The use of this operator is generally detrimental to the readability and reliability of code, and the same effect can be achieved by other means.

It is especially error-prone in array subscripts where it might be misinterpreted as accessing a multidimensional array. The use of a comma in this context is deprecated since C++20 (it might be repurposed in a later edition of the standard to actually access multidimensional arrays, but until then, it should really not be used).

Noncompliant Code Example

```
i = a += 2, a + b;  // Noncompliant. What's the value of i ?

a[1, 2] = 3; // Noncompliant: 1 is ignored. This is not an ac

x = a[i++, j = i + 1, j*2]; // Noncompliant. What index is us
```

Compliant Solution

```
a +=  2;
i = a + b;

j = i + 1;
x = a[j*2];
++i;
```

Exceptions

Use of comma operator is tolerated in initialization and increment expressions of `for` loops.

```
for(i = 0, j = 5; i < 6; i++, j++) { ... }
```

See

- MISRA C:2004, 12.10 - The comma operator shall not be used.
- MISRA C++:2008, 5-18-1 - The comma operator shall not be used.
- MISRA C:2012, 12.3 - The comma operator should not be used

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