



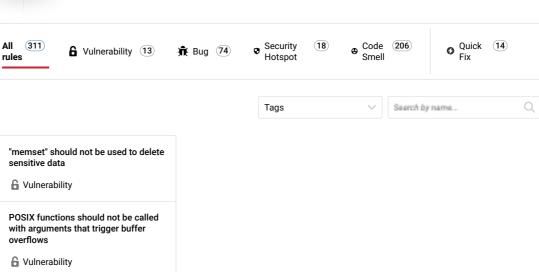
👬 Bug

👬 Bug

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



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



Stack allocated memory and nonowned memory should not be freed

🕕 Bug

Closed resources should not be accessed

₩ Bua

Dynamically allocated memory should be released

👬 Bug

Freed memory should not be used

Null pointers should not be dereferenced

Analyze your code

👚 Bug 🔷 Major 🔞



we symbolic-execution cert

A pointer to null (the 0 memory address) should never be dereferenced/accessed. Doing so will at best cause abrupt program termination, without the ability to run any cleanup processes. At worst, it could expose debugging information that would be useful to an attacker or it could allow an attacker to bypass security measures.

Noncompliant Code Example

```
char *p1 = ...;
if (p1 == NULL && *p1 == '\t') { // Noncompliant, p1 will be
char *p2 = ...;
if (p2 != NULL) {
   // ...
*p2 = '\t'; // Noncompliant; potential null-dereference
char *p3, *p4;
p3 = NULL;
// ...
p4 = p3;
*p4 = 'a'; // Noncompliant
```

Compliant Solution

```
char *p1 = ...;
if (pl != NULL && *pl == '\t') { // Compliant, *pl cannot be
// ...
}
char *p2 = ...;
if (p2 != NULL) {
   // ...
  *p2 = '\t'; // Compliant
```

See

- MITRE, CWE-476 NULL Pointer Dereference
- CERT, EXP34-C. Do not dereference null pointers
- CERT, EXP01-J. Do not use a null in a case where an object is required

Available In:

sonarlint o sonarcloud o sonarqube Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy