

**CSS** 

Go **GO** 

Flex

5 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** 

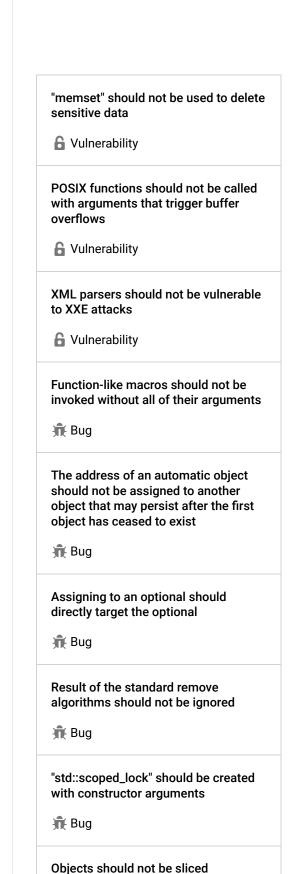


# C++ static code analysis

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

⊗ Code (436) • Security Quick 68 Fix 578 ΑII 6 Vulnerability 13 **R** Bug (111) Hotspot rules

Tags



📆 Bug

🖷 Bug

📆 Bug

📆 Bug

Immediately dangling references

"pthread\_mutex\_t" should be unlocked in the reverse order they were locked

"pthread\_mutex\_t" should be properly

"pthread\_mutex\_t" should not be

consecutively locked or unlocked

initialized and destroyed

should not be created

Using "strncpy" or "wcsncpy" is security-sensitive

Security Hotspot Major

Analyze your code

Search by name...

cwe owasp cert

In C, a string is just a buffer of characters, normally using the null character as a

low-level details such as buffer sizes or having an extra character to store the final null character. Doing that correctly and consistently is notoriously difficult and any error can lead to a security vulnerability, for instance, giving access to sensitive data or allowing arbitrary code execution.

sentinel for the end of the string. This means that the developer has to be aware of

The function char \*strncpy(char \* restrict dest, const char \* restrict src, size\_t count); copies the first count characters from src to dest, stopping at the first null character, and filling extra space with 0. The wcsncpy does the same for wide characters and should be used with the same quidelines.

Both of those functions are designed to work with fixed-length strings and might result in a non-null-terminated string.

## **Ask Yourself Whether**

- There is a possibility that either the source or the destination pointer is
- The security of your system can be compromised if the destination is a truncated version of the source
- The source buffer can be both non-null-terminated and smaller than the
- The destination buffer can be smaller than the count
- You expect dest to be a null-terminated string
- There is an overlap between the source and the destination

There is a risk if you answered yes to any of those questions.

## **Recommended Secure Coding Practices**

- C11 provides, in its annex K, the strncpy\_s and the wcsncpy\_s that were designed as safer alternatives to stropy and woscpy. It's not recommended to use them in all circumstances, because they introduce a runtime overhead and require to write more code for error handling, but they perform checks that will limit the consequences of calling the function with bad arguments.
- Even if your compiler does not exactly support annex K, you probably have access to similar functions
- If you are using strncpy and wsncpy as a safer version of strcpy and wcscpy, you should instead consider strcpy s and wcscpy s, because these functions have several shortcomings:
  - o It's not easy to detect truncation
  - o Too much work is done to fill the buffer with 0, leading to suboptimal
  - Unless manually corrected, the dest string might not be null-terminated
- If you want to use strcpy and wcscpy functions and detect if the string was truncated, the pattern is the following:
  - Set the last character of the buffer to null
  - Call the function
  - o Check if the last character of the buffer is still null
- If you are writing C++ code, using std::string to manipulate strings is much simpler and less error-prone

# Sensitive Code Example

int f(char \*src) { char dest[256];



"std::move" and "std::forward" should not be confused



A call to "wait()" on a "std::condition\_variable" should have a condition



A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic\_cast



Functions with "noreturn" attribute should not return



RAII objects should not be temporary



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



"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types

Rug Bug

"std::auto\_ptr" should not be used

📆 Bug

Destructors should be "noexcept"

```
📆 Bug
```

```
strncpy(dest, src, sizeof(dest)); // Sensitive: might silen
return doSomethingWith(dest);
```

#### **Compliant Solution**

```
int f(char *src) {
  char dest[256];
  dest[sizeof dest - 1] = 0;
  strncpy(dest, src, sizeof(dest)); // Compliant
  if (dest[sizeof dest - 1] != 0) {
    // Handle error
  return doSomethingWith(dest);
}
```

### See

- OWASP Top 10 2021 Category A6 Vulnerable and Outdated Components
- OWASP Top 10 2017 Category A9 Using Components with Known Vulnerabilities
- MITRE, CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer
- CERT, STR07-C. Use the bounds-checking interfaces for string manipulation

Available In:

sonarcloud 🚳 | 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