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 578 | Vulnerability 13 | Bug 111 | Security Hotspot 18 | Code Smell 436 | Quick Fix 68

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

Assigning to an optional should directly target the optional
Bug

Result of the standard remove algorithms should not be ignored
Bug

"std::scoped_lock" should be created with constructor arguments
Bug

Objects should not be sliced
Bug

Immediately dangling references should not be created
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

Changing working directories without verifying the success is security-sensitive

Analyze your code

Security Hotspot | Critical | cwe owasp

The purpose of changing the current working directory is to modify the base path when the process performs relative path resolutions. When the working directory cannot be changed, the process keeps the directory previously defined as the active working directory. Thus, verifying the success of chdir() type of functions is important to prevent unintended relative paths and unauthorized access.

Ask Yourself Whether

- The success of changing the working directory is relevant for the application.
- Changing the working directory is required by chroot to make the new root effective.
- Subsequent disk operations are using relative paths.

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

Recommended Secure Coding Practices

After changing the current working directory verify the success of the operation and handle errors.

Sensitive Code Example

The `chdir` operation could fail and the process still has access to unauthorized resources. The return code should be verified:

```
const char* any_dir = "/any/";
chdir(any_dir); // Sensitive: missing check of the return val

int fd = open(any_dir, O_RDONLY | O_DIRECTORY);
fchdir(fd); // Sensitive: missing check of the return value
```

Compliant Solution

Verify the return code of `chdir` and handle errors:

```
const char* root_dir = "/jail/";
if (chdir(root_dir) == -1) {
  exit(-1);
} // Compliant

int fd = open(any_dir, O_RDONLY | O_DIRECTORY);
if(fchdir(fd) == -1) {
  exit(-1);
} // Compliant
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-252 - Unchecked Return Value
- man7.org - chdir

Available In:

sonarcloud | sonarqube Developer Edition

Bug

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

Bug

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

Bug

A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast

Bug

Functions with "noreturn" attribute should not return

Bug

RAII objects should not be temporary

Bug

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

Bug

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

Bug

"std::auto_ptr" should not be used

Bug

Destructors should be "noexcept"

Bug