



Apex

**ABAP** 

С

C++

CloudFormation

COBOL

C#

**CSS** 

Flex

Go **GO** 

5 HTML

JavaScript

Kotlin

Java

Kubernetes

Objective C

PHP

PL/I

PL/SQL

Python

Ruby

Scala

**RPG** 

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

ΑII 578 6 Vulnerability (13) rules

**R** Bug (111)

o Security Hotspot

⊗ Code (436)

Quick 68 Fix

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

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:

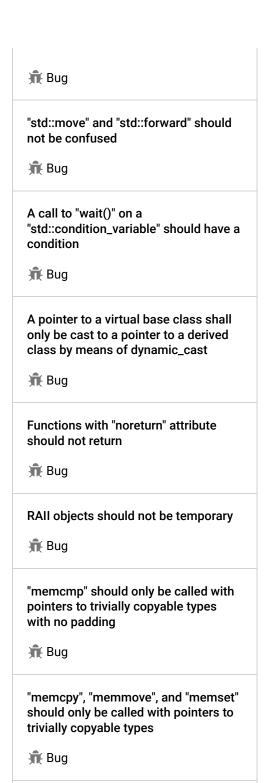
```
const char* root_dir = "/jail/";
if (chdir(root_dir) == -1) {
  exit(-1);
} // Compliant
int fd = open(any_dir, O_RDONLY | O_DIRECTORY);
if(fchdir(fd) == -1) {
  exit(-1);
} // Compliant
```

#### 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



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

Destructors should be "noexcept"

📆 Bug

📆 Bug

© 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