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

Objective C static code analysis

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

| All rules **315** | Vulnerability **10** | Bug **75** | Security Hotspot **18** | Code Smell **212** | Quick Fix **13** |

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

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

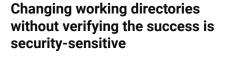
Stack allocated memory and non-owned memory should not be freed

🐞 Bug

Closed resources should not be accessed

🐞 Bug

Dynamically allocated memory should be released

🐞 Bug

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

Freed memory should not be used

🐞 Bug

Memory locations should not be released more than once

🐞 Bug

Memory access should be explicitly bounded to prevent buffer overflows

🐞 Bug

Printf-style format strings should not lead to unexpected behavior at runtime

🐞 Bug

Recursion should not be infinite

🐞 Bug

Resources should be closed

🐞 Bug

Hard-coded credentials are security-sensitive

🛡 Security Hotspot

"goto" should jump to labels declared later in the same function

☣ Code Smell

Only standard forms of the "defined" directive should be used

☣ Code Smell

Switch labels should not be nested inside non-switch blocks

☣ Code Smell