



ABAP

Apex

С

C++

CloudFormation

COBOL

C#

CSS

Flex

Go **GO**

HTML 5

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 315 6 Vulnerability 10 rules

R Bug (75)

o Security Hotspot

⊗ Code (212)

O Quick 13 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

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 nonowned memory should not be freed

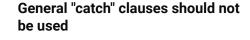
📆 Bug

Closed resources should not be accessed

📆 Bug

Dynamically allocated memory should be released

📆 Bug



Analyze your code

☼ Code Smell ♥ Minor ②

error-handling

A general catch block seems like an efficient way to handle multiple possible exceptions. Unfortunately, it traps all exception types, casting too broad a net, and perhaps mishandling extraordinary cases. Instead, specific exception sub-types should be caught.

Noncompliant Code Example

```
try {
  file.open("test.txt");
} catch (...) { // Noncompliant
  // ...
```

Compliant Solution

```
file.open("test.txt");
} catch (std::ifstream::failure e) {
```

Exceptions

There are cases though where you want to catch all exceptions, because no exceptions should be allowed to escape the function, and generic catch handlers are excluded from the rule:

- · In the main function
- In a class destructor
- In a noexcept function
- In an extern "C" function

Additionally, if the catch handler is throwing an exception (either the same as before, with throw; or a new one that may make more sense to the callers of the function), or is never exiting (because it calls a noreturn function, for instance exit), then the accurate type of the exception usually does not matter any longer: this case is excluded too.

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**

Freed memory should not be used Recursion should not be infinite Bug Recursion should not be infinite Bug Resources should be closed Bug Resources should be closed Code Smell Switch labels should not be nested inside non-switch blocks Code Smell Memory access should be explicitly bounded to prevent buffer overflows Replication should not lead to unexpected behavior at runtime Bug Recursion should not be infinite Security Bug Resources should be closed Code Smell Switch labels should not be nested inside non-switch blocks Code Smell	
Memory locations should not be released more than once	Freed memory should not be used
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	₩ 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	
Printf-style format strings should not lead to unexpected behavior at runtime	∰ 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	
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	Rug
Recursion should not be infinite 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	lead to unexpected behavior at
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	Rug
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	Recursion should not be infinite
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	∰ 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	Resources should be closed
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	n Bug
"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	
Iater 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	Security Hotspot
Only standard forms of the "defined" directive should be used Code Smell Switch labels should not be nested inside non-switch blocks	
directive should be used Code Smell Switch labels should not be nested inside non-switch blocks	Code Smell
Switch labels should not be nested inside non-switch blocks	
inside non-switch blocks	
Code Smell	