



Flex

Go **GO** 

5 HTML

Java

JavaScript

Kotlin

**Objective C** 

Kubernetes

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

ΑII 315 6 Vulnerability (10) rules

**R** Bug (75)

• Security Hotspot

⊗ Code (212)

O Quick 13 Fix

Tags

Search by name...

"memset" should not be used to delete

Vulnerability

sensitive data

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

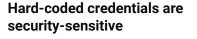
📆 Bug

Closed resources should not be accessed

📆 Bug

Dynamically allocated memory should be released

Bug



Analyze your code

cwe cert sans-top25 owasp

Because it is easy to extract strings from an application source code or binary, credentials should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- CVE-2019-13466
- CVE-2018-15389

Credentials should be stored outside of the code in a configuration file, a database, or a management service for secrets.

This rule looks for hard-coded credentials in variable names that match any of the patterns from the provided list.

#### **Ask Yourself Whether**

- Credentials allow access to a sensitive component like a database, a file storage, an API or a service.
- Credentials are used in production environments.
- · Application re-distribution is required before updating the credentials.

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

### **Recommended Secure Coding Practices**

- Store the credentials in a configuration file that is not pushed to the code repository.
- · Store the credentials in a database.
- Use your cloud provider's service for managing secrets.
- If a password has been disclosed through the source code: change it.

## **Sensitive Code Example**

```
dbi_conn conn = dbi_conn_new("mysql");
string password = "secret"; // Sensitive
dbi_conn_set_option(conn, "password", password.c_str());
```

### **Compliant Solution**

```
dbi_conn conn = dbi_conn_new("mysql");
string password = getDatabasePassword(); // Compliant
dbi conn set option(conn, "password", password.c str()); // C
```

### See

- OWASP Top 10 2021 Category A7 Identification and Authentication Failures
- OWASP Top 10 2017 Category A2 Broken Authentication
- MITRE, CWE-798 Use of Hard-coded Credentials
- MITRE, CWE-259 Use of Hard-coded Password
- CERT, MSC03-J. Never hard code sensitive information
- SANS Top 25 Porous Defenses
- Derived from FindSecBugs rule Hard Coded Password

Available In:

sonarcloud 🚳 sonarqube Developer

Freed memory should not be used
<b>∰</b> Bug
Memory locations should not be
released more than once
Rug
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
_
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
☼ Code Smell

© 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