



**ABAP** 

Apex

С

C++

CloudFormation

COBOL

C#

**CSS** 

Flex

Go

5 **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



# Go static code analysis

Unique rules to find Bugs, Security Hotspots, and Code Smells in your GO code

All rules (38)

**R** Bug (7)

Security Hotspot (2)

Code Smell (29)

Tags

Search by name...

Hard-coded credentials are securitysensitive

Security Hotspot

**Cognitive Complexity of functions** should not be too high

Code Smell

String literals should not be duplicated

Code Smell

Functions should not be empty

Code Smell

All branches in a conditional structure should not have exactly the same implementation

📆 Bug

"=+" should not be used instead of "+="

📆 Bug

Related "if/else if" statements should not have the same condition

📆 Bug

Identical expressions should not be used on both sides of a binary operator

📆 Bug

All code should be reachable

📆 Bug

Variables should not be self-assigned

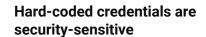
📆 Bug

Functions should not have identical implementations

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell



Analyze your code

🖣 cwe 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 flags instances of hard-coded credentials used in database and LDAP connections. It looks for hard-coded credentials in connection strings, and for variable names that match any of the patterns from the provided list.

It's recommended to customize the configuration of this rule with additional credential words such as "oauthToken", "secret", ...

## **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
- 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**

```
func connect() {
user := "root"
password:= "supersecret" // Sensitive
url := "login=" + user + "&passwd=" + password
```

### **Compliant Solution**

```
func connect() {
user := getEncryptedUser()
password:= getEncryptedPass() // Compliant
url := "login=" + user + "&passwd=" + password
```

See

"switch" statements should not have too many "case" clauses Code Smell Track uses of "FIXME" tags Code Smell Redundant pairs of parentheses should be removed Code Smell Nested blocks of code should not be left empty Code Smell Functions should not have too many parameters Code Smell Using hardcoded IP addresses is security-sensitive Security Hotspot Multi-line comments should not be empty Code Smell Boolean checks should not be inverted Code Smell Local variable and function parameter names should comply with a naming convention Code Smell

Boolean literals should not be

redundant

Code Smell

- 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
- SANS Top 25 Porous Defenses
- Derived from FindSecBugs rule Hard Coded Password

Available In:

sonarcloud 💩 | sonarqube

© 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