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

Go static code analysis

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

All rules 38 | Bug 7 | Security Hotspot 2 | Code Smell 29

Tags

Search by name...

Hard-coded credentials are security-sensitive
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

"if ... else if" constructs should end with "else" clauses

Analyze your code

Code Smell | Critical

This rule applies whenever an `if` statement is followed by one or more `else if` statements; the final `else if` should be followed by an `else` statement.

The requirement for a final `else` statement is defensive programming.

The `else` statement should either take appropriate action or contain a suitable comment as to why no action is taken. This is consistent with the requirement to have a final `default` clause in a `switch` statement.

Noncompliant Code Example

```
if x == 0 {
        doSomething()
} else if x == 1 {
        doSomethingElse()
}
```

Compliant Solution

```
if x == 0 {
        doSomething()
} else if x == 1 {
        doSomethingElse()
} else {
        return errors.New("unsupported int")
}
```

Exceptions

When all branches of an `if-else if` end with `return` or `break`, the code that comes after the `if` implicitly behaves as if it was in an `else` clause. This rule will therefore ignore that case.

Available In:

sonarcloud | sonarqube

"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