

































-  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

"switch case" clauses should not have too many lines

Analyze your code

Code Smell Major brain-overload

The `switch` statement should be used only to clearly define some new branches in the control flow. As soon as a `case` clause contains too many statements this highly decreases the readability of the overall control flow statement. In such case, the content of the `case` clause should be extracted into a dedicated method.

### Noncompliant Code Example

With the default threshold of 5:

```
func foo(tag int) {
    switch tag {
    case 0:
        methodCall1()
        methodCall2()
        methodCall3()
        methodCall4()
        methodCall5()
        methodCall6()

    case 1:
        bar()

    }
}
```

### Compliant Solution

```
func foo(tag int) {
    switch tag {
    case 0:
        executeAll()











    case 1:
        bar()

    }
}

func executeAll() {
    methodCall1()
    methodCall2()
    methodCall3()
    methodCall4()
    methodCall5()
    methodCall6()
}
```

Available In:

sonarcloud | sonarqube

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