


































- 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 rules38

 Bug7

 Security Hotspot2

 Code Smell29

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

Expressions should not be too complex

Analyze your code

 Code Smell  Critical   brain-overload

The complexity of an expression is defined by the number of `&&`, `||` and `condition ? ifTrue : ifFalse` operators it contains.

A single expression's complexity should not become too high to keep the code readable.

### Noncompliant Code Example

With the default threshold value of 3:

```
if (((condition1 && condition2) || (condition3 && condit
```











### Compliant Solution

```
if ( (myFirstCondition() || mySecondCondition()) && myLa
```

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](#)

<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>