sonar RULES

Products

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML

PHP static code analysis

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

All rules (268) Vulnerability (40) Bug (51) Security Hotspot (33) Code Smell (144)

Tags

Search by name...

Code Smell

Cyclomatic Complexity of functions should not be too high

Code Smell

Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply

Code Smell

Cyclomatic Complexity of classes should not be too high

Code Smell

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

Code Smell

Expressions should not be too complex

Code Smell

"cgi.force_redirect" should be enabled

Vulnerability

Files that define symbols should not cause side-effects

Bug

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

Code Smell

Test methods should be discoverable

Code Smell

Use of namespaces should be preferred to "include" or "require" functions

Code Smell

Duplicate values should not be passed as arguments

Code Smell

Regular expressions should not be too complicated

Analyze your code

Code Smell Major regex

Overly complicated regular expressions are hard to read and to maintain and can easily cause hard-to-find bugs. If a regex is too complicated, you should consider replacing it or parts of it with regular code or splitting it apart into multiple patterns at least.

The complexity of a regular expression is determined as follows:

Each of the following operators increases the complexity by an amount equal to the current nesting level and also increases the current nesting level by one for its arguments:

- `|` - when multiple `|` operators are used together, the subsequent ones only increase the complexity by 1
- Quantifiers (`*`, `+`, `?`, `{n,m}`, `{n,}` or `{n}`)
- Non-capturing groups that set flags (such as `(?i:some_pattern)` or `(?i)some_pattern`)
- Lookahead and lookbehind assertions

Additionally, each use of the following features increase the complexity by 1 regardless of nesting:

- character classes
- back references

Noncompliant Code Example

```
if (preg_match("/^(?:(?:31(\\/|-|\\.)(?:0?[13578]|1[02]))\\1
    handleDate($dateString);
}
```

Compliant Solution

```
    if (preg_match("/^\\d{1,2}([-/.])\\d{1,2}\\1\\d{1,4}$/")
        $dateParts = preg_split("/[-/.]/", $dateString);
        $day = intval($dateParts[0]);
        $month = intval($dateParts[1]);
        $year = intval($dateParts[2]);
        // Put logic to validate and process the date based
    }
```

Available In:

sonarlint sonarcloud sonarqube

Configuration should not be changed dynamically

Code Smell

Class constructors should not create other objects

Code Smell

PHP parser failure

Code Smell

The names of methods with boolean return values should start with "is" or "has"

Code Smell