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...

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

Configuration should not be changed dynamically
Code Smell

Class constructors should not create other objects
Code Smell

PHP parser failure
Code Smell

Character classes in regular expressions should not contain the same character twice

Analyze your code

Code Smell | Major | regex

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If the same character is listed twice in the same character class or if the character class contains overlapping ranges, this has no effect.

Thus duplicate characters in a character class are either a simple oversight or a sign that a range in the character class matches more than is intended or that the author misunderstood how character classes work and wanted to match more than one character. A common example of the latter mistake is trying to use a range like `[0-99]` to match numbers of up to two digits, when in fact it is equivalent to `[0-9]`. Another common cause is forgetting to escape the `-` character, creating an unintended range that overlaps with other characters in the character class.

Noncompliant Code Example

```
"/[0-99]/" // Noncompliant, this won't actually match string
"/[0-9.-_]/" // Noncompliant, .-_ is a range that already co
```

Compliant Solution

```
"/[0-9]{1,2}/"
"/[0-9.\\-_]/"
```

Available In:

sonarlint | sonarcloud | sonarqube

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

Code Smell

"php_sapi_name()" should not be used

Code Smell

Classes should not have too many lines of code

Code Smell

Deprecated features should not be used

Code Smell