
































-  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... 

	passed in the correct order
	Ternary operators should not be nested
	Reflection should not be used to increase accessibility of classes, methods, or fields
	Multiline blocks should be enclosed in curly braces
	Parameters should be passed in the correct order
	Classes named like "Exception" should extend "Exception" or a subclass
	Two branches in a conditional structure should not have exactly the same implementation
	Unused assignments should be removed
	Method arguments with default values should be last
	A reason should be provided when skipping a test
	"__construct" functions should not make PHP 4-style calls to parent constructors

Replacement strings should reference existing regular expression groups

Analyze your code

 Bug

 Major 

 regex

The PHP function `preg_replace` can be used to perform a search and replace based on regular expression matches. The `$replacement` parameter can contain references to capturing groups used in the `$pattern` parameter. This can be achieved with `\n` or `$n` to reference the `n`'th group.

When referencing a nonexistent group, it will be substituted with an empty string. This is in general not the intended behavior, and might stay unnoticed since no warning is raised.

Noncompliant Code Example

```
preg_replace("/(a)(b)(c)/", "\\1, \\9, \\3", "abc"); //
```

Compliant Solution

```
preg_replace("/(a)(b)(c)/", "\\1, \\2, \\3", "abc");
```

See

- [preg_replace](#) - PHP Documentation

Available In:


sonarlint  | **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](#)

PHP 4 constructor declarations
should not be used

 Code Smell


Deprecated predefined variables
should not be used

 Code Smell

"switch" statements should not have
too many "case" clauses

 Code Smell

Classes should not have too many
methods

 Code Smell

Functions should not have too many