
































-  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... 

Server-side requests should not be vulnerable to forging attacks

 Vulnerability

The number of arguments passed to a function should match the number of parameters

 Bug

Non-empty statements should change control flow or have at least one side-effect

 Bug

Variables should be initialized before use

 Bug

Replacement strings should reference existing regular expression groups

 Bug

Alternation in regular expressions should not contain empty alternatives

 Bug

Unicode Grapheme Clusters should be avoided inside regex character classes

 Bug

Assertions should not compare an object to itself

 Bug

Regex alternatives should not be redundant

 Bug

Alternatives in regular expressions should be grouped when used with anchors

 Bug

Array values should not be replaced unconditionally

Regex boundaries should not be used in a way that can never be matched

Analyze your code

 Bug  Critical   regex

In regular expressions the boundaries `^` and `\A` can only match at the beginning of the input (or, in case of `^` in combination with the `MULTILINE` flag, the beginning of the line) and `$`, `\z` and `\Z` only at the end.

These patterns can be misused, by accidentally switching `^` and `$` for example, to create a pattern that can never match.

Noncompliant Code Example

```
// This can never match because $ and ^ have been switched
preg_match("/$[a-z]^/", $str); // Noncompliant
```

Compliant Solution

```
preg_match("/^[a-z]$/", $str);
```

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](#)

unconditional

 Bug

Exceptions should not be created without being thrown

 Bug

Array or Countable object count comparisons should make sense

 Bug

All branches in a conditional structure should not have exactly the same implementation

 Bug

The output of functions that don't return anything should not be used