
































-  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 lookahead assertions should not be contradictory

Analyze your code

 Bug

 Critical



 regex

Lookahead assertions are a regex feature that makes it possible to look ahead in the input without consuming it. It is often used at the end of regular expressions to make sure that substrings only match when they are followed by a specific pattern.

However, they can also be used in the middle (or at the beginning) of a regex. In that case there is the possibility that what comes after the lookahead does not match the pattern inside the lookahead. This makes the lookahead impossible to match and is a sign that there's a mistake in the regular expression that should be fixed.

Noncompliant Code Example

```
preg_match("/(?:=a)b/", $str); // Noncompliant, the same char
```

Compliant Solution





```
preg_match("(?<=a)b/", $str);  
preg_match("/a(?:=b)/", $str);
```

Available In:

 sonarlint

 sonarcloud

 sonarqube

 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