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...

Authorizations should be based on strong decisions
Vulnerability

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

Back references in regular expressions should only refer to capturing groups that are matched before the reference

Analyze your code

Bug | Critical | regex

When a back reference in a regex refers to a capturing group that hasn't been defined yet (or at all), it can never be matched. Named back references throw a warning in that case; numeric back references fail silently when they can't match, simply making the match fail.

When the group is defined before the back reference but on a different control path (like in `(.)|\1` for example), this also leads to a situation where the back reference can never match.

Noncompliant Code Example

```
preg_match("/\\1(.)/", $str); // Noncompliant, group 1 is de
preg_match("/(.)\\2/", $str); // Noncompliant, group 2 isn't
preg_match("/(.)|\\1/", $str); // Noncompliant, group 1 and
preg_match("/(?<x>.)|\k<x>/", $str); // Noncompliant, group
```

Compliant Solution

```
preg_match("/(.)\\1/", $str);
preg_match("/(?<x>.)\\k<x>/", $str);
```

Available In:

sonarlint | sonarcloud | sonarqube

Array values should not be replaced unconditionally

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