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...

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
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
Bug

Unary prefix operators should not be repeated
Bug

"=+" should not be used instead of "+="
Bug

A "for" loop update clause should move the counter in the right direction
Bug

Return values from functions without side effects should not be ignored

Assertion failure exceptions should not be ignored

Analyze your code

Bug | Critical | tests phpunit

PHPUnit assertions do throw a *PHPUnit\Framework\ExpectationFailedException* exception when they fail. This is how PHPUnit internally notices when assertions within testcases fail. However, if such an exception type or one of its parent types is captured within a try-catch block and not rethrown, PHPUnit does not notice the assertion failure.

This check raises an issue on assertions within the *try* body of a *try-catch* block that do catch exceptions of the type *PHPUnit\Framework\ExpectationFailedException*, *PHPUnit\Framework\AssertionFailedError*, or *Exception*, and do not handle the variable holding the exception.

Noncompliant Code Example

```
public function testA() {
    try {
        assertTrue(getValue()); // Noncompliant
    } catch (\PHPUnit\Framework\ExpectationFailedExcept

    }
}
```

Compliant Solution

```
public function testB() {
    try {
        assertTrue(getValue()); // Compliant
    } catch (\PHPUnit\Framework\ExpectationFailedExcept
        assertEquals("Some message", $e->getMessage());
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Bug

Values should not be uselessly incremented

Bug

Related "if/else if" statements and "cases" in a "switch" should not have the same condition

Bug

Objects should not be created to be dropped immediately without being used

Bug

Identical expressions should not be used on both sides of a binary