



ABAP

Apex

C С

CloudFormation

COBOL

C#

3 CSS

 \bowtie Flex

-GO Go

5 HTML

Java

JS JavaScript

Kotlin

Objective C

PHP

PL/I

PL/SQL

Python

RPG

Ruby

Scala

D Swift

Terraform

Text

тѕ 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)

sensitive

sensitive

Security Hotspot

Security Hotspot

Security Hotspot

flag is security-sensitive

Security Hotspot

security-sensitive

concisely

Code Smell

expressions

Code Smell

Code Smell

A Code Smell

Code Smell

redundant

rethrow

Security Hotspot

Creating cookies without the

"HttpOnly" flag is security-sensitive

Creating cookies without the "secure"

Using hardcoded IP addresses is

Regular expression quantifiers and

Character classes should be preferred over reluctant quantifiers in regular

A subclass should not be in the same "catch" clause as a parent class

Jump statements should not be

"catch" clauses should do more than

character classes should be used

6 Vulnerability (40)

Resource Sharing policy is security-

Delivering code in production with

debug features activated is security-

♣ Bug (51)

Security Hotspot 33

Code Smell (144)

Identical expressions should not be used on both sides of

Analyze your code

Search by name...

📆 Bug 🔷 Major 🕝

Tags

a binary operator

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified. In the case of bitwise operators and most binary mathematical operators, having the same value on both sides of an operator yields predictable results, and should be simplified.

Noncompliant Code Example

```
if ( $a == $a ) { // always true
                       doZ();
  if ( $a != $a ) { // always false
                       doY();
if ( a == b \& a == b ) { // if the first one is tr
                       doX();
if ( a == b \mid \ a ==
                       doW();
  j = 5 / 5; //always 1
  k = 5 - 5; //always 0
```

Exceptions

Left-shifting 1 onto 1 is common in the construction of bit masks, and is ianored.

```
$i = 1 \ll 1; // Compliant
$j = $a << $a; // Noncompliant</pre>
```

• {rule:php:S1656} - Implements a check on =.

Available In:



"&&" and "||" should be used

A Code Smell

© 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. $\underline{\text{Privacy Policy}}$

Boolean checks should not be inverted

Code Smell

Local variables should not be declared and then immediately returned or thrown

Code Smell

Unused local variables should be removed

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell