

- 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...

"php_sapi_name()" should not be used

Code Smell

Classes should not have too many lines of code

Code Smell

Deprecated features should not be used

Code Smell

Files should not contain inline HTML

Code Smell

Files should contain only one top-level class or interface each

Code Smell

Classes should not have too many fields

Code Smell

Track uses of "NOSONAR" comments

Code Smell

Statements should be on separate lines

Code Smell

Classes should not be coupled to too many other classes (Single Responsibility Principle)

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Assignments should not be made from within sub-expressions

Code Smell

Files should not have too many lines of code

Code Smell

Multiline blocks should be enclosed in curly braces

Analyze your code

Code Smell Major

Curly braces can be omitted from a one-line block, such as with an `if` statement or for loop, but doing so can be misleading and induce bugs.

This rule raises an issue when the whitespacing of the lines after a one line block indicates an intent to include those lines in the block, but the omission of curly braces means the lines will be unconditionally executed once.

Note that this rule considers tab characters to be equivalent to 1 space. If you mix spaces and tabs you will sometimes see issues in code which look fine in your editor but are confusing when you change the size of tabs.

Noncompliant Code Example

```
if ($condition)
    firstActionInBlock();
    secondAction(); // Noncompliant; executed unconditionally
    thirdAction();

if($condition) firstActionInBlock(); secondAction(); // Non

if($condition) firstActionInBlock(); // Noncompliant
    secondAction(); // Executed unconditionally

$str = null;
for ($i = 0; $i < count($array); $i++)
    $str = $array[$i];
    doTheThing($str); // Noncompliant; executed only on last
```

Compliant Solution

```
if ($condition) {
    firstActionInBlock();
    secondAction();
}
thirdAction();

if($condition) { firstActionInBlock(); secondAction(); }


if($condition) {
    firstActionInBlock();
    secondAction();
}

$str = null;
for ($i = 0; $i < count($array); $i++) {
    $str = $array[$i];
    doTheThing($str);
}
```


Lines should not be too long

 Code Smell

HTTP response headers should not be vulnerable to injection attacks

 Vulnerability

"sleep" should not be called

 Vulnerability

Static members should be referenced with "static::"

 Bug

See

- [MITRE, CWE-483](#) - Incorrect Block Delimitation

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](#)