
































-  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... 

Classes should not have too many lines of code	
Deprecated features should not be used	
Files should not contain inline HTML	
Files should contain only one top-level class or interface each	
Classes should not have too many fields	
Track uses of "NOSONAR" comments	
Statements should be on separate lines	
Classes should not be coupled to too many other classes (Single Responsibility Principle)	
"switch case" clauses should not have too many lines of code	
Assignments should not be made from within sub-expressions	
Files should not have too many lines of code	
Lines should not be too long	

Parameters should be passed in the correct order

Analyze your code

 Code Smell  Major 

When the names of parameters in a method call match the names of the method arguments, it contributes to clearer, more readable code. However, when the names match, but are passed in a different order than the method arguments, it indicates a mistake in the parameter order which will likely lead to unexpected results.

Noncompliant Code Example

```
public function divide($divisor, $dividend) {
    return $divisor/$dividend;
}

public function doTheThing() {
    $divisor = 15;
    $dividend = 5;

    $result = $this->divide($dividend, $divisor); // Noncompliant
}
```

Compliant Solution


```
public function divide($divisor, $dividend) {
    return $divisor/$dividend;
}

public function doTheThing() {
    $divisor = 15;
    $dividend = 5;


    $result = $this->divide($divisor, $dividend); // Compliant
}{code}
h4.
```

Available In:
  

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

"require_once" and "include_once" should be used instead of "require" and "include"