

- 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... 🔍

Boolean checks should not be inverted
Local variables should not be declared and then immediately returned or thrown
Unused local variables should be removed
"switch" statements should have at least 3 "case" clauses
A "while" loop should be used instead of a "for" loop
Overriding methods should do more than simply call the same method in the super class
"empty()" should be used to test for emptiness
Interface names should comply with a naming convention
Return of boolean expressions should not be wrapped into an "if-then-else" statement
Boolean literals should not be redundant

All "catch" blocks should be able to catch exceptions

Analyze your code

Bug

Major ?

unused

Exceptions handlers (catch) are evaluated in the order they are written. Once a match is found, the evaluation stops.

In some contexts a catch block is dead code as it will never catch any exception:

- If there is a handler for a base class followed by a handler for class derived from that base class, the second handler will never trigger: the handler for the base class will match the derived class, and will be the only executed handler.
- When multiple catch blocks try to catch the same exception class, only the first one will be executed.

This rule raises an issue when a catch block catches every exception before a later catch block could catch it.

### Noncompliant Code Example

```
class MyException extends Exception {}
class MySubException extends MyException {}

try {
    doSomething();
} catch (MyException $e) {
    echo $e;
} catch (MySubException $e) { // Noncompliant: MySubExc
    echo "Never executed";
}
```

### Compliant Solution

```
class MyException extends Exception {}
class MySubException extends MyException {}

try {
    doSomething();
} catch (MySubException $e) {
    echo "Executed";
} catch (MyException $e) {
    echo $e;
}
```


Available In:

sonarlint


sonarcloud

sonarqube


Empty statements should be removed

 Code Smell


A close curly brace should be located at the beginning of a line

 Code Smell


URIs should not be hardcoded

 Code Smell

Class names should comply with a naming convention

 Code Smell

Track uses of "TODO" tags

 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.  
[Privacy Policy](#)