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...

Code Smell

Duplicate values should not be passed as arguments

Code Smell

Configuration should not be changed dynamically

Code Smell

Class constructors should not create other objects

Code Smell

PHP parser failure

Code Smell

The names of methods with boolean return values should start with "is" or "has"

Code Smell

"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

Functions should use "return" consistently

Analyze your code

Code Smell   Major   api-design confusing

Because it is dynamically typed, PHP does not enforce a return type on a function. This means that different paths through a function can return different types of values, which can be very confusing to the user and significantly harder to maintain.

In particular, it is consequently also possible to mix empty `return` statements (implicitly returning `null`) with some returning an expression. This rule verifies that all the `return` statements from a function are consistent.

Noncompliant Code Example

```
function foo($a) { // Noncompliant, function will retur
  if ($a == 1) {
    return true;
  }
  return;
}
```

Compliant Solution

```
function foo($a) {
  if ($a == 1) {
    return true;
  }
  return false;
}
```

or

```
function foo($a) {
  if ($a == 1) {
    return true;
  }
  return null;
}
```

Available In:

sonarlint sonarcloud sonarqube

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