PHP static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PHP code

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

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)

Reflection should not be used to increase accessibility of classes, methods, or fields

Analyze your code

Code Smell Major

This rule raises an issue when reflection is used to change the visibility of a class, method or field, and when it is used to directly update a field value. Altering or bypassing the accessibility of classes, methods, or fields violates the encapsulation principle and could lead to run-time errors.

Noncompliant Code Example

```
class MyClass
{
    public static $publicstatic = 'Static';
    private static $privatestatic = 'private Static';
    private $private = 'Private';
    private const CONST_PRIVATE = 'Private CONST';
    public $myfield = 42;

    private function __construct() {}
    private function privateMethod() {}
    public function __set($property, $value)  {}
    public function __get($property) {}
}

$clazz = new ReflectionClass('MyClass');

$clazz->getstaticProperties(); // Noncompliant . This gives

$clazz->setStaticPropertyValue('publicstatic', '42'); // OK
$clazz->getStaticPropertyValue('publicstatic'); // OK as the

// The following calls can access private or protected const
$clazz->getConstant('CONST_PRIVATE'); // Noncompliant.
$clazz->getConstants(); // Noncompliant.
$clazz->getReflectionConstant('CONST_PRIVATE'); // Noncompli
$clazz->getReflectionConstants(); // Noncompliant.

$obj = $clazz->newInstanceWithoutConstructor(); // Noncompli

$constructor = $clazz->getConstructor();
$constructorClosure = $constructor->getClosure($obj); // Non
$constructor->setAccessible(true); // Noncompliant. Bypassin

$prop = new ReflectionProperty('MyClass', 'private');
$prop->setAccessible(true); // Noncompliant. Change accessib
$prop->setValue($obj, "newValue"); // Noncompliant. Bypass o
$prop->getValue($obj); // Noncompliant. Bypass of the __get

$prop2 = $clazz->getProperties()[2];
$prop2->setAccessible(true); // Noncompliant. Change accessi
$prop2->setValue($obj, "newValue"); // Noncompliant. Bypass
$prop2->getValue($obj); // Noncompliant. Bypass of the __get

$meth = new ReflectionMethod('MyClass', 'privateMethod');
```

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

Lines should not be too long

```
$clos = $meth->getClosure($obj); // Noncompliant. It is poss
$meth->setAccessible(true); // Noncompliant. Change accessib

$meth2 = $clazz->getMethods()[0];
$clos2 = $meth2->getClosure($obj); // Noncompliant. It is po
$meth2->setAccessible(true); // Noncompliant. Change accessi

// Using a ReflectionObject instead of the class

$objr = new ReflectionObject($obj);
$objr->newInstanceWithoutConstructor(); // Noncompliant. Byp

$objr->getStaticPropertyValue("publicstatic"); // OK as ther
$objr->setStaticPropertyValue("publicstatic", "newValue"); /

$objr->getStaticProperties(); // Noncompliant. This gives ac

// The following calls can access private or protected const
$objr->getConstant('CONST_PRIVATE'); // Noncompliant.
$objr->getConstants(); // Noncompliant.
$objr->getReflectionConstant('CONST_PRIVATE'); // Noncomplia
$objr->getReflectionConstants(); // Noncompliant.

$constructor = $objr->getConstructor();
$constructorClosure = $constructor->getClosure($obj); // Non
$constructor->setAccessible(true); // Noncompliant. Bypassin

$prop3 = $objr->getProperty('private');
$prop3->setAccessible(true); // Noncompliant. Change accessi
$prop3->setValue($obj, "newValue"); // Noncompliant. Bypass
$prop3->getValue($obj); // Noncompliant. Bypass of the __get

$prop4 = $objr->getProperties()[2];
$prop4->setAccessible(true); // Noncompliant. Change accessi
$prop4->setValue($obj, "newValue"); // Noncompliant. Bypass
$prop4->getValue($obj); // Noncompliant. Bypass of the __get

$meth3 = $objr->getMethod('privateMethod');
$clos3 = $meth3->getClosure($obj); // Noncompliant. It is po
$meth3->setAccessible(true); // Noncompliant. Change accessi

$meth4 = $objr->getMethods()[0];
$clos4 = $meth4->getClosure($obj); // Noncompliant. It is po
$meth4->setAccessible(true); // Noncompliant. Change accessi
```

Available In:

sonarlint | sonarcloud | sonarqube