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...

be used to end lines
Code Smell

More than one property should not be declared per statement
Code Smell

The "var" keyword should not be used
Code Smell

"<?php" and "<?=" tags should be used
Code Smell

File names should comply with a naming convention
Code Smell

Comments should not be located at the end of lines of code
Code Smell

Local variable and function parameter names should comply with a naming convention
Code Smell

Field names should comply with a naming convention
Code Smell

Lines should not end with trailing whitespaces
Code Smell

Files should contain an empty newline at the end
Code Smell

Modifiers should be declared in the correct order
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

Delivering code in production with debug features activated is security-sensitive

Analyze your code

Security Hotspot | Minor | cwe error-handling debug user-experience owasp

Delivering code in production with debug features activated is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2018-1999007
- CVE-2015-5306
- CVE-2013-2006

An application's debug features enable developers to find bugs more easily and thus facilitate also the work of attackers. It often gives access to detailed information on both the system running the application and users.

Ask Yourself Whether

- the code or configuration enabling the application debug features is deployed on production servers or distributed to end users.
- the application runs by default with debug features activated.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Do not enable debug features on production servers or applications distributed to end users.

Sensitive Code Example

CakePHP 1.x, 2.x:

```
Configure::write('debug', 1); // Sensitive: development mode
or
Configure::write('debug', 2); // Sensitive: development mode
or
Configure::write('debug', 3); // Sensitive: development mode
```

CakePHP 3.0:

```
use Cake\Core\Configure;

Configure::config('debug', true); // Sensitive: development
```

WordPress:

```
define( 'WP_DEBUG', true ); // Sensitive: development mode
```

Compliant Solution

CakePHP 1.2:

An open curly brace should be located at the end of a line

Code Smell

Tabulation characters should not be used

Code Smell

Method and function names should comply with a naming convention

Code Smell

Creating cookies with broadly defined "domain" flags is security-sensitive

Security Hotspot

```
Configure::write('debug', 0); // Compliant; this is the prod
```

CakePHP 3.0:

```
use Cake\Core\Configure;

Configure::config('debug', false); // Compliant:  "0" or "fa
```

WordPress:

```
define( 'WP_DEBUG', false ); // Compliant
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-489 - Active Debug Code
- MITRE, CWE-215 - Information Exposure Through Debug Information

Available In:

sonarcloud | sonarqube