

- 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

Controlling permissions is security-sensitive

Analyze your code

Security Hotspot

Minor

The access control of an application must be properly implemented in order to restrict access to resources to authorized entities otherwise this could lead to vulnerabilities:

- [CVE-2018-12999](#)
- [CVE-2018-10285](#)
- [CVE-2017-7455](#)

Granting correct permissions to users, applications, groups or roles and defining required permissions that allow access to a resource is sensitive, must therefore be done with care. For instance, it is obvious that only users with administrator privilege should be authorized to add/remove the administrator permission of another user.

Ask Yourself Whether

- Granted permission to an entity (user, application) allow access to information or functionalities not needed by this entity.
- Privileges are easily acquired (eg: based on the location of the user, type of device used, defined by third parties, does not require approval ...).
- Inherited permission, default permission, no privileges (eg: anonymous user) is authorized to access to a protected resource.

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

Recommended Secure Coding Practices

At minimum, an access control system should:

- Use a well-defined access control model like [RBAC](#) or [ACL](#).
- Entities' permissions should be reviewed regularly to remove permissions that are no longer needed.
- Respect [the principle of least privilege](#) ("an entity has access only the information and resources that are necessary for its legitimate purpose").

Sensitive Code Example

CakePHP

```
use Cake\Auth\BaseAuthorize;
use Cake\Controller\Controller;


abstract class MyAuthorize extends BaseAuthorize { // Sensitive
    // ...
}

// Note that "isAuthorized" methods will only be detected in
abstract class MyController extends Controller {
    public function isAuthorized($user) { // Sensitive. Method
        return false;
    }
}
```

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

See

- [OWASP Top 10 2017 Category A5](#) - Broken Access Control
- [SANS Top 25](#) - Porous Defenses
- [MITRE, CWE-276](#) - Incorrect Default Permissions
- [MITRE, CWE-732](#) - Incorrect Permission Assignment for Critical Resource
- [MITRE, CWE-668](#) - Exposure of Resource to Wrong Sphere
- [MITRE, CWE-277](#) - Insecure Inherited Permissions

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud  **sonarqube** 