

- 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

Unused local variables should be removed

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

A "while" loop should be used instead of a "for" loop

Code Smell

Overriding methods should do more than simply call the same method in the super class

Code Smell

"empty()" should be used to test for emptiness

Code Smell

Interface names should comply with a naming convention

Code Smell

Return of boolean expressions should not be wrapped into an "if-then-else" statement

Code Smell

Boolean literals should not be redundant

Code Smell

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

Constructing arguments of system commands from user input is security-sensitive

Analyze your code

Security Hotspot

Major

injection cwe owasp sans-top25

Constructing arguments of system commands from user input is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2016-9920
- CVE-2021-29472

Arguments of system commands are processed by the executed program. The arguments are usually used to configure and influence the behavior of the programs. Control over a single argument might be enough for an attacker to trigger dangerous features like executing arbitrary commands or writing files into specific directories.

Ask Yourself Whether

- Malicious arguments can result in undesired behavior in the executed command.
- Passing user input to a system command is not necessary.

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

Recommended Secure Coding Practices

- Avoid constructing system commands from user input when possible.
- Ensure that no risky arguments can be injected for the given program, e.g., type-cast the argument to an integer.
- Use a more secure interface to communicate with other programs, e.g., the standard input stream (stdin).

Sensitive Code Example

Arguments like `-delete` or `-exec` for the `find` command can alter the expected behavior and result in vulnerabilities:






```
$input = $_GET['input'];
system('/usr/bin/find ' . escapeshellarg($input)); // S
```

Compliant Solution

Use an allow-list to restrict the arguments to trusted values:

```
$input = $_GET['input'];
if (in_array($input, $allowed, true)) {
    system('/usr/bin/find ' . escapeshellarg($input));
}
```

See

 Code Smell
Class names should comply with a naming convention  Code Smell
Track uses of "TODO" tags  Code Smell
"file_uploads" should be disabled  Vulnerability
"enable_dl" should be disabled  Vulnerability
"session.use_trans_sid" should not be

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection
- [MITRE, CWE-88](#) - Argument Injection or Modification
- [SANS Top 25](#) - Insecure Interaction Between Components
- [CVE-2021-29472](#) - PHP Supply Chain Attack on Composer

Available In:



© 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](#)