

- 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

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

Reading the Standard Input is security-sensitive

Analyze your code

Security Hotspot Critical

Reading Standard Input is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2005-2337
- CVE-2017-11449

It is common for attackers to craft inputs enabling them to exploit software vulnerabilities. Thus any data read from the standard input (stdin) can be dangerous and should be validated.

This rule flags code that reads from the standard input.

Ask Yourself Whether

- data read from the standard input is not sanitized before being used.

You are at risk if you answered yes to this question.

Recommended Secure Coding Practices

Sanitize all data read from the standard input before using it.

Sensitive Code Example

```
// Any reference to STDIN is Sensitive
$varstdin = STDIN; // Sensitive
stream_get_line(STDIN, 40); // Sensitive
stream_copy_to_stream(STDIN, STDOUT); // Sensitive
// ...


// Except those references as they can't create an injection
ftruncate(STDIN, 5); // OK
ftell(STDIN); // OK
feof(STDIN); // OK
fseek(STDIN, 5); // OK
fclose(STDIN); // OK

// STDIN can also be referenced like this
$mystdin = 'php://stdin'; // Sensitive


file_get_contents('php://stdin'); // Sensitive
readfile('php://stdin'); // Sensitive

$input = fopen('php://stdin', 'r'); // Sensitive
fclose($input); // OK
```


See

 Code Smell


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

- [MITRE, CWE-20](#) - Improper Input Validation

#### Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

**sonarcloud**  | **sonarqube** 

© 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](#)