

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- 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... 🔍

Octal values should not be used
Code Smell
Switch cases should end with an unconditional "break" statement
Code Smell
Session-management cookies should not be persistent
Vulnerability
Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)
Vulnerability
SHA-1 and Message-Digest hash algorithms should not be used in secure contexts
Vulnerability
Assertions should not be made at the end of blocks expecting an exception
Bug
Regular expressions should be syntactically valid
Bug
Only one method invocation is expected when testing exceptions
Bug
Reading the Standard Input is security-sensitive
Security Hotspot
Using command line arguments is security-sensitive
Security Hotspot
Using Sockets is security-sensitive
Security Hotspot
Using system calls is security-sensitive
Security Hotspot

Superfluous curly brace quantifiers should be avoided

Analyze your code

Code Smell

Major ?

regex

Curly brace quantifiers in regular expressions can be used to have a more fine-grained control over how many times the character or the sub-expression preceeding them should occur. They can be used to match an expression exactly n times with $\{n\}$, between n and m times with $\{n,m\}$, or at least n times with $\{n, \}$. In some cases, using such a quantifier is superfluous for the semantic of the regular expression, and it can be removed to improve readability. This rule raises an issue when one of the following quantifiers is encountered:

- $\{1, 1\}$ or $\{1\}$: they match the expression exactly once. The same behavior can be achieved without the quantifier.
- $\{0, 0\}$ or $\{0\}$: they match the expression zero times. The same behavior can be achieved by removing the expression.

Noncompliant Code Example

```
"/ab{1,1}c/"
"/ab{1}c/"
"/ab{0,0}c/"
"/ab{0}c/"
```


Compliant Solution

```
"/abc/"
"/ac/"
```


Available In:

sonarlint | sonarcloud | sonarqube


Encrypting data is security-sensitive

 Security Hotspot


Using regular expressions is security-sensitive

 Security Hotspot

Deserializing objects from an untrusted source is security-sensitive

 Security Hotspot

Literal boolean values and nulls should not be used in equality assertions

 Code Smell

"global" should not be used

 Code Smell