sonar RULES

Products

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

Functions should not contain too many return statements

Code Smell

Track uses of "FIXME" tags

Code Smell

Generic exceptions ErrorException, RuntimeException and Exception should not be thrown

Code Smell

Local variables should not have the same name as class fields

Code Smell

Redundant pairs of parentheses should be removed

Code Smell

Inheritance tree of classes should not be too deep

Code Smell

Nested blocks of code should not be left empty

Code Smell

Functions should not have too many parameters

Code Smell

Unused "private" fields should be removed

Code Smell

Collapsible "if" statements should be merged

Code Smell

OS commands should not be vulnerable to argument injection attacks

Unary prefix operators should not be repeated

Analyze your code

Bug Major

Calling the `!` or `~` prefix operator twice does nothing: the second invocation undoes the first. Such mistakes are typically caused by accidentally double-tapping the key in question without noticing. Either this is a bug, if the operator was actually meant to be called once, or misleading if done on purpose.

Noncompliant Code Example

```
$a = 0;
$b = false;

$c = !!$a; // Noncompliant
$d = ~~$b; // Noncompliant
```

Compliant Solution

```
$a = 0;
$b = false;

$c = !$a; // Compliant
$d = ~$b; // Compliant
```

Available In:

sonarlint | sonarcloud | sonarqube

Vulnerability

Logging should not be vulnerable to injection attacks

Vulnerability

Repeated patterns in regular expressions should not match the empty string

Bug

Function and method parameters' initial values should not be ignored

Bug

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive