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...*

Bug

All code should be reachable
Bug

Loops with at most one iteration should be refactored
Bug

Short-circuit logic should be used to prevent null pointer dereferences in conditionals
Bug

Variables should not be self-assigned
Bug

Useless "if(true) {...}" and "if(false){...}" blocks should be removed
Bug

All "catch" blocks should be able to catch exceptions
Bug

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

Allowing unfiltered HTML content in WordPress is security-sensitive
Security Hotspot

Allowing unauthenticated database repair in WordPress is security-sensitive
Security Hotspot

Allowing all external requests from a WordPress server is security-sensitive
Security Hotspot

Disabling automatic updates is

\`str_replace\` should be preferred to \`preg_replace\`

Analyze your code

Code Smell Critical regex performance

An `preg_replace` call always performs an evaluation of the first argument as a regular expression, even if no regular expression features were used. This has a significant performance cost and therefore should be used with care.

When `preg_replace` is used, the first argument should be a real regular expression. If it's not the case, `str_replace` does exactly the same thing as `preg_replace` without the performance drawback of the regex.

This rule raises an issue for each `preg_replace` used with a simple string as first argument which doesn't contains special regex character or pattern.

Noncompliant Code Example

```
$str = "Bob is a Bird... Bob is a Plane... Bob is Super
$changed = preg_replace("/Bob is/", "It's", $str); // N
$changed = preg_replace("/\.\.\./", ";", $changed); //
```

Compliant Solution

```
$str = "Bob is a Bird... Bob is a Plane... Bob is Super
$changed = str_replace("Bob is", "It's", $str);
$changed = str_replace("...", ";", $changed);
```

Or, with a regex:

```
$str = "Bob is a Bird... Bob is a Plane... Bob is Super
$changed = preg_replace("/\w*\sis/", "It's", $str);
$changed = preg_replace("/\.{3}/", ";", $changed);
```

Available In:
sonarlint | sonarcloud | sonarqube

security-sensitive

Security Hotspot

WordPress theme and plugin editors are security-sensitive

Security Hotspot

Allowing requests with excessive content length is security-sensitive

Security Hotspot

Manual generation of session ID is security-sensitive

Security Hotspot

Setting loose POSIX file permissions is security-sensitive