

- 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...



vulnerable to argument injection attacks

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

Security Hotspot

Delivering code in production with debug features activated is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure" flag is security-sensitive

Security Hotspot

Using hardcoded IP addresses is security-sensitive

Security Hotspot

Regular expression quantifiers and character classes should be used concisely

Code Smell

Character classes should be preferred over reluctant quantifiers in regular

**Objects should not be created to be dropped immediately without being used**

Analyze your code

Bug Major

There is no good reason to create a new object to not do anything with it. Most of the time, this is due to a missing piece of code and so could lead to an unexpected behavior in production.

If it was done on purpose because the constructor has side-effects, then that side-effect code should be moved into a separate, static method and called directly.

### Noncompliant Code Example

```
if ($x < 0) {  
    new foo; // Noncompliant  
}
```

### Compliant Solution


```
$var = NULL;  
if ($x < 0) {  
    $var = new foo;  
}
```

Available In:


**sonarlint** | **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](#)


expressions

 Code Smell


A subclass should not be in the same  
"catch" clause as a parent class

 Code Smell


Jump statements should not be  
redundant

 Code Smell

"catch" clauses should do more than  
rethrow

 Code Smell

"&&" and "||" should be used

 Code Smell