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...

Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

LDAP connections should be authenticated
Vulnerability

Cryptographic keys should be robust
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks
Vulnerability

Hashes should include an unpredictable salt
Vulnerability

Regular expressions should have valid delimiters
Bug

Regex lookahead assertions should not be contradictory
Bug

Back references in regular expressions should only refer to capturing groups that are matched before the reference
Bug

Regex boundaries should not be used in a way that can never be matched
Bug

Regex patterns following a possessive quantifier should not always fail
Bug

Class of caught exception should be defined

Analyze your code

Bug | Blocker | pitfall

When specifying the class of objects in a catch clause it is important to make sure that the class exists.

Since no PHP error will be raised if the class does not exist, this can lead to difficult to debug problems as the catch clause will have no effect and the reason might not be obvious.

This mistake often occurs when being in a namespace and catching PHP built-in exception classes without escaping to the global namespace or importing the classes.

This rule raises an issue when, being in a namespace, an undefined class belonging to that namespace is caught.

Noncompliant Code Example

```
namespace Foo\Bar;

try {
    doSomething();
} catch (Exception $e) { // Noncompliant - Exception will ne
    echo $e->message;
}
```

Compliant Solution

```
namespace Foo\Bar;

try {
    doSomething();
} catch (\Exception $e) { // Compliant used by global namesp
    echo $e->message;
}

// or

namespace Foo\Bar;

use Exception;

try {
    doSomething();
} catch (Exception $e) { // Compliant imported by use statem
    echo $e->message;
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Assertion failure exceptions should not be ignored

Bug

References used in "foreach" loops should be "unset"

Bug

Using clear-text protocols is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive