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...

Cognitive Complexity of functions should not be too high
Code Smell

Parentheses should not be used for calls to "echo"
Code Smell

Functions should not be nested too deeply
Code Smell

References should not be passed to function calls
Code Smell

"switch" statements should have "default" clauses
Code Smell

Control structures should use curly braces
Code Smell

String literals should not be duplicated
Code Smell

Methods should not be empty
Code Smell

Constant names should comply with a naming convention
Code Smell

Secret keys and salt values should be robust
Vulnerability

Authorizations should be based on strong decisions
Vulnerability

Server-side requests should not be vulnerable to forging attacks

Cryptographic keys should be robust

Analyze your code

Vulnerability | Critical | cwe privacy owasp

Most of cryptographic systems require a sufficient key size to be robust against brute-force attacks.

NIST recommendations will be checked for these use-cases:

Digital Signature Generation and **Verification:**

- p ≥ 2048 AND q ≥ 224 for DSA (`p` is key length and `q` the modulus length)
- n ≥ 2048 for RSA (`n` is the key length)

Key Agreement:

- p ≥ 2048 AND q ≥ 224 for DH and MQV
- n ≥ 224 for ECDH and ECMQV (Examples: `secp192r1` is a non-compliant curve (`n` < 224) but `secp224k1` is compliant (`n` >= 224))

Symmetric keys:

- key length ≥ 128 bits

This rule will not raise issues for ciphers that are considered weak (no matter the key size) like `DES`, `Blowfish`.

Noncompliant Code Example

```
$config = array(
    "digest_alg" => "sha512",
    "private_key_bits" => 1024, // Noncompliant
    "private_key_type" => OPENSSL_KEYTYPE_RSA,
);
$res = openssl_pkey_new($config);
```

Compliant Solution

```
$config = array(
    "digest_alg" => "sha512",
    "private_key_bits" => 2048 // Compliant
    "private_key_type" => OPENSSL_KEYTYPE_RSA,
);
$res = openssl_pkey_new($config);
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography

Vulnerability

The number of arguments passed to a function should match the number of parameters

Bug

Non-empty statements should change control flow or have at least one side-effect

Bug

Variables should be initialized before use

Bug

Replacement strings should reference existing regular expression groups

- NIST 800-131A - Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
- MITRE, CWE-326 - Inadequate Encryption Strength

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