

- 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...

|   |                  |
|---|------------------|
| Signalling processes is security-sensitive                            | Security Hotspot |
| Configuring loggers is security-sensitive                             | Security Hotspot |
| Using weak hashing algorithms is security-sensitive                   | Security Hotspot |
| Disabling CSRF protections is security-sensitive                      | Security Hotspot |
| Using pseudorandom number generators (PRNGs) is security-sensitive    | Security Hotspot |
| Dynamically executing code is security-sensitive                      | Security Hotspot |
| `str_replace` should be preferred to `preg_replace`                   | Code Smell       |
| "default" clauses should be first or last                             | Code Smell       |
| A conditionally executed single line should be denoted by indentation | Code Smell       |
| Conditionals should start on new lines                                | Code Smell       |
| Cognitive Complexity of functions should not be too high              | Code Smell       |
| Parentheses should not be used for calls to "echo"                    |                  |

Cipher algorithms should be robust

Analyze your code

Vulnerability

Critical

cwe

privacy

owasp

sans-top25

**Strong cipher algorithms** are cryptographic systems resistant to cryptanalysis, they are not vulnerable to well-known attacks like brute force attacks for example.

A general recommendation is to only use cipher algorithms intensively tested and promoted by the cryptographic community.

More specifically for block cipher, it's not recommended to use algorithm with a block size inferior than 128 bits.

Noncompliant Code Example

```
<?php
// mcrypt_encrypt is deprecated since PHP 7.1
$c1 = mcrypt_encrypt(MCRYPT_DES, $key, $plaintext, $mode);
$c2 = mcrypt_encrypt(MCRYPT_DES_COMPAT, $key, $plaintext, $mode);
$c3 = mcrypt_encrypt(MCRYPT_TRIPLEDES, $key, $plaintext, $mode);
$c4 = mcrypt_encrypt(MCRYPT_3DES, $key, $plaintext, $mode);
$c5 = mcrypt_encrypt(MCRYPT_BLOWFISH, $key, $plaintext, $mode);
$c6 = mcrypt_encrypt(MCRYPT_RC2, $key, $plaintext, $mode);
$c7 = mcrypt_encrypt(MCRYPT_RC4, $key, $plaintext, $mode);

$c8 = openssl_encrypt($plaintext, "bf-ecb", $key, $options=0);
$c9 = openssl_encrypt($plaintext, "des-ede3", $key, $options);
$c10 = openssl_encrypt($plaintext, "des-efb", $key, $options);
$c11 = openssl_encrypt($plaintext, "rc2-cbc", $key, $options);
$c12 = openssl_encrypt($plaintext, "rc4", $key, $options=OPE);
```

Compliant Solution

```
<?php
$c1= openssl_encrypt($plaintext, "aes-256-gcm", $key, $options=OPE);
```

See


- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-327](#) - Use of a Broken or Risky Cryptographic Algorithm
- [SANS Top 25](#) - Porous Defenses

Available In:


sonarlint

sonarcloud

sonarqube

 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