

- 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...

Related "if/else if" statements and "cases" in a "switch" should not have the same condition	Bug
Objects should not be created to be dropped immediately without being used	Bug
Identical expressions should not be used on both sides of a binary operator	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

## Using pseudorandom number generators (PRNGs) is security-sensitive

Analyze your code

Security Hotspot

Critical

cwe owasp

Using pseudorandom number generators (PRNGs) is security-sensitive. For example, it has led in the past to the following vulnerabilities:

- CVE-2013-6386
- CVE-2006-3419
- CVE-2008-4102

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

As the `rand()` and `mt_rand()` functions rely on a pseudorandom number generator, it should not be used for security-critical applications or for protecting sensitive data.

### Ask Yourself Whether

- the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such as a password, is hashed.
- the function you use generates a value which can be predicted (pseudo-random).
- the generated value is used multiple times.
- an attacker can access the generated value.

There is a risk if you answered yes to any of those questions.

- ### Recommended Secure Coding Practices
- Use functions which rely on a cryptographically strong random number generator such as `random_int()` or `random_bytes()` or `openssl_random_pseudo_bytes()`
  - When using `openssl_random_pseudo_bytes()`, provide and check the `crypto_strong` parameter
  - Use the generated random values only once.
  - You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.

### Sensitive Code Example


```
$random = rand();
$random2 = mt_rand(0, 99);
```

### Compliant Solution

```
$randomInt = random_int(0,99); // Compliant; generates a cr
```

See

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 security-sensitive

 Security Hotspot

WordPress theme and plugin editors are security-sensitive

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [Mobile AppSec Verification Standard](#) - Cryptography Requirements
- [OWASP Mobile Top 10 2016 Category M5](#) - Insufficient Cryptography
- [MITRE, CWE-338](#) - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
- [MITRE, CWE-330](#) - Use of Insufficiently Random Values
- [MITRE, CWE-326](#) - Inadequate Encryption Strength
- [MITRE, CWE-1241](#) - Use of Predictable Algorithm in Random Number Generator
- Derived from FindSecBugs rule [Predictable Pseudo Random Number Generator](#)

Available In:

**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](#)