

- 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

"session.use_trans_sid" should not be enabled

Vulnerability

"allow_url_fopen" and "allow_url_include" should be disabled

Vulnerability

"open_basedir" should limit file access

Vulnerability

Neither DES (Data Encryption Standard) nor DESede (3DES) should be used

Vulnerability

"exit(...)" and "die(...)" statements should not be used

Bug

Functions and variables should not be defined outside of classes

Code Smell

Track lack of copyright and license headers

Code Smell

Octal values should not be used

Code Smell

Switch cases should end with an unconditional "break" statement

Code Smell

Session-management cookies should not be persistent

Vulnerability

Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)

Manual generation of session ID is security-sensitive

Analyze your code

Security Hotspot Major ? cwe owasp

If a session ID can be guessed (not generated with a secure pseudo random generator, or with insufficient length ...) an attacker may be able to hijack another user's session.

Ask Yourself Whether

- the session ID is not unique.
- the session ID is set from a user-controlled input.
- the session ID is generated with not secure pseudo random generator.
- the session ID length is too short.

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

Recommended Secure Coding Practices

Don't manually generate session IDs, use instead language based native functionality.

Sensitive Code Example

```
session_id(bin2hex(random_bytes(4))); // Sensitive: 4 b
session_id($_POST["session_id"]); // Sensitive: session
```

Compliant Solution


```
session_regenerate_id(); // Compliant
session_id(bin2hex(random_bytes(16))); // Compliant
```

See


- [OWASP Top 10 2021 Category A4](#) - Insecure Design
- [OWASP Top 10 2021 Category A7](#) - Identification and Authentication Failures
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [OWASP Session Fixation](#)
- [MITRE, CWE-330](#) - Use of Insufficiently Random Values
- [MITRE, CWE-340](#) - Generation of Predictable Numbers or Identifiers
- [PHP: random_bytes\(\)](#)
- [PHP: session_regenerate_id\(\)](#)

Available In:

sonardcloud | sonarqube

 Vulnerability

SHA-1 and Message-Digest hash algorithms should not be used in secure contexts

 Vulnerability

Assertions should not be made at the end of blocks expecting an exception

 Bug

Regular expressions should be syntactically valid

 Bug

Only one method invocation is expected when testing exceptions

 Bug

© 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](#)