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...

naming convention
Code Smell

Return of boolean expressions should not be wrapped into an "if-then-else" statement
Code Smell

Boolean literals should not be redundant
Code Smell

Empty statements should be removed
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

URIs should not be hardcoded
Code Smell

Class names should comply with a naming convention
Code Smell

Track uses of "TODO" tags
Code Smell

"file_uploads" should be disabled
Vulnerability

"enable_dl" should be disabled
Vulnerability

"session.use_trans_sid" should not be enabled
Vulnerability

"allow_url_fopen" and "allow_url_include" should be disabled
Vulnerability

Allowing all external requests from a WordPress server is security-sensitive

Analyze your code

Security Hotspot | Major | cwe owasp

External requests initiated by a WordPress server should be considered as security-sensitive. They may contain sensitive data which is stored in the files or in the database of the server. It’s important for the administrator of a WordPress server to understand what they contain and to which server they are sent.

WordPress makes it possible to block external requests by setting the `WP_HTTP_BLOCK_EXTERNAL` option to `true`. It’s then possible to authorize requests to only a few servers using another option named `WP_ACCESSIBLE_HOSTS`.

Ask Yourself Whether

- Your WordPress website contains code which may call external requests to servers you don’t know.
- Your WordPress website may send sensitive data to other servers.
- Your WordPress website uses a lot of plugins or themes.

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

Recommended Secure Coding Practices

- Uninstall WordPress plugins which send requests to servers you don’t know.
- Make sure that `WP_HTTP_BLOCK_EXTERNAL` is defined in `wp-config.php`.
- Make sure that `WP_HTTP_BLOCK_EXTERNAL` is set to `true`.
- Make sure that `WP_ACCESSIBLE_HOSTS` is configured to authorize requests to the servers you trust.

Sensitive Code Example

```
define( 'WP_HTTP_BLOCK_EXTERNAL', false ); // Sensitive
```

Compliant Solution

```
define( 'WP_HTTP_BLOCK_EXTERNAL', true );
define( 'WP_ACCESSIBLE_HOSTS', 'api.wordpress.org' );
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2021 Category A10 - Server-Side Request Forgery (SSRF)
- wordpress.org - Block External URL Requests
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- OWASP Attack Category - Server Side Request Forgery

"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

- MITRE, CWE-918 - Server-Side Request Forgery (SSRF)

Available In:

sonarcloud sonarqube