PHP Security Concepts

Overview

Web applications written in PHP may contain security vulnerabilities that malicious users can exploit to gain sensitive information about your system or your users. These vulnerabilities can include the following:

- Unverified executable files.
- Development server environment functions that run in a production server environment.
- Error messages that include sensitive system information.

This document lists several methods that you can use to harden your system's PHP configuration.

Restrict file inclusion attacks

File inclusion attacks often occur when an attacker exploits a file-inclusion vulnerability in a web application that dynamically includes files and scripts. A user may create applications that do not properly validate `include` and `require` statements, or use filenames as parameters. An attacker may pass a malicious file to the application that contains the same name as a file that already exists on the server. The attacker can use this file to pull sensitive information about your system.

File inclusion attacks include Local File Inclusion (LFI) attacks and Remote File Inclusion (RFI) attacks.

For more information about file inclusion attacks, read Wikipedia's File inclusion vulnerability article.

Local File Inclusion attacks

LFI attacks occur when an attacker pulls local files into PHP scripts in order to view sensitive information on or about your system. For example, an attacker may use a local file inclusion vulnerability in a PHP script to view the `/etc/passwd` file. This would allow an attacker to discover basic information about your web server's accounts.

To limit the impact of local file inclusion vulnerabilities in PHP scripts, enable the *open_basedir* feature in WHM's *PHP open_basedir Tweak* interf ace (*WHM >> Home >> Security Center >> PHP open_basedir Tweak).* This feature limits an attacker's access to a single directory via local includes and makes local file inclusion attacks more difficult.

Remote File Inclusion attacks

RFI attacks occur when an attacker pulls files from a remote location on your server. For example, an attacker can write a PHP script and host it on a server, and then use a remote inclusion method to take advantage of inclusion vulnerabilities on your server. An insecure PHP configuration allows attackers to execute the malicious data from their servers, even without read or write permissions on your server.

To prevent remote file inclusion attacks, set the *allow_url_fopen* and *allow_url_include* directives to *Off* in the *Advanced Mode* section of WHM's *P HP Configuration Editor* interface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

> **Note:**
> On systems that run EasyApache 4, set these directives in the *Basic Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).

The disable_functions directive

Some PHP functions are **not** safe for a production environment. If your PHP developers do not require these functions, we **strongly** recommend

that you disable them so that an attacker cannot use them. Generally, when you disable these functions, you can stop an attacker who manages to load a malicious PHP script on to your system.

To disable a list of functions, enter them in a comma-delimited list to the *disable_functions* directive's text box in the *Advanced Mode* section of WHM's *PHP Configuration Editor* interface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

> **Note:**
> On systems that run EasyApache 4, enter these functions in the *Editor Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).

For an example of functions to disable, read nixCraft's post on disable_functions.

> **Important:**
> - Many functions in PHP perform the same tasks. Ask your developers to standardize on one or two of these functions for a task so that you can disable the rest.
> - You can **only** disable internal PHP functions.

Prevent information disclosure

Error messages that disclose important system information can help attackers plan an attack strategy. This information includes your directory structure, database names, and usernames. If PHP does not print errors to the web application's user interface, you can inhibit attackers' ability to gain information that they could use to compromise your system.

To limit the display of error messages, set the *display_errors* directive to *Off* in the *Advanced Mode* section of WHM's *PHP Configuration Editor* int erface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

> **Notes:**
> - On systems that run EasyApache 4, set this directive to *Off* in the *Basic Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).
> - When you disable the *display_errors* directive, your developers can still retrieve helpful information from debug codes in the appropriate PHP logs.

Restrict file uploads

Attackers often upload malicious programs to vulnerable systems in order to compromise them. If you restrict all file uploads, this can ensure that attackers **cannot** exploit your PHP configuration to inject their own PHP scripts.

To restrict file uploads, set the *file_uploads* directive in the *Advanced Mode* section of WHM's *PHP Configuration Editor* interface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

> **Notes:**
> - On systems that run EasyApache 4, set this directive in the *Basic Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).
> - Some developers prefer to include the ability to upload files to your server via PHP. If you **must** allow file uploads, set the *uplo ad_tmp_dir* directive to *On* in order to change the default temporary directory for file uploads.
> - Many administrators also set the *upload_max_filesize* directive to limit the maximum file size that users can upload. This parameter does **not** improve the security of your PHP configuration. Administrators set this parameter in order to help manage the server's load from PHP scripts.

Protect sessions

Some attackers attempt to hijack sessions. This occurs when an attacker steals a user's web application session and performs actions as that user. PHP uses long, randomly-generated session identifiers for its URLs. While this makes session URLs exceedingly difficult to guess, the filesystem stores this value. Attackers can inject JavaScript into pages to steal cookies that contain these session IDs, which would allow them to hijack sessions.

To protect these session IDs from session hijackers, you can set the *session.cookie_httponly* directive in the *Advanced Mode* section of WHM's *P HP Configuration Editor* interface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

Notes:

- On systems that run EasyApache 4, set this directive in the *Editor Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).
- This directive makes certain that JavaScript **cannot** access a PHP application's session cookies. If your developers require that JavaScript possesses access to session cookies, do **not** enable this option.
- You may also wish to allow PHP to check HTTP referrer values. This ensures that sensitive session information passes internally during a user's session, so that users cannot accidentally publish sensitive session information when they share URLs.

Disable register globals

Global variables allow a PHP script to receive and process variables without a specified source. This allows attackers to overwrite configuration variables in order to gain access to areas of your system that the system ordinarily restricts.

To remove this vulnerability, set the *register_globals* directive to *Off* in the *Advanced Mode* section of WHM's *PHP Configuration Editor* interface (*WHM >> Home >> Service Configuration >> PHP Configuration Editor*).

Note:
On systems that run EasyApache 4, set this directive to *Off* in the *Basic Mode* section of WHM's *MultiPHP INI Editor* interface (*WHM >> Home >> Software >> MultiPHP INI Editor*).

Important:
This option no longer exists for PHP version 5.4 and above.

Additional documentation

Suggested documentationFor cPanel usersFor WHM usersFor developers

- PHP Security Concepts
- How to Install KernelCare
- How to Configure Your Firewall for cPanel Services
- Basic Security Concepts
- How to Purchase a KernelCare License

Error rendering macro 'contentbylabel' : parameters should not be empty

- PHP Security Concepts
- How to Install KernelCare
- How to Configure Your Firewall for cPanel Services
- Basic Security Concepts
- How to Purchase a KernelCare License

- Guide to the LiveAPI System - LiveAPI Methods
- Tutorial - Call UAPI's SSL::install_ssl Function in Custom Code
- WHM API 1 Functions - php_set_vhost_versions
- UAPI Functions - LangPHP::php_get_impacted_domains
- WHM API 1 Functions - fetch_security_advice