



PHP Update Guide

Step by Step

CRITICAL · 9.2 / 10

Two vulnerabilities in PHP itself may allow remote command execution on your server without logging in. The PHP team considers every installation that serves web requests at risk, with the highest risk on **mod_php**.

CVE-2026-109618 · CVE-2026-109619

What to do: update PHP to the latest security release for your branch, and make sure the PHP processes serving your site are actually running the patched version. Take a backup and have a rollback plan before changing production.

1 · Patched versions

Your PHP branch	First patched release
PHP 8.2	8.2.35
PHP 8.3	8.3.36
PHP 8.4	8.4.27
PHP 8.5	8.5.12

Important: later releases in the **same branch** include the fix. Security support for PHP 8.2 ends on December 31, 2026.

PHP 8.1 and older: no longer officially supported. Ask your OS vendor about extended security updates (LTS), or plan a move to a supported branch. Given the severity, the PHP team also published minimal source backports for 7.4, 8.0 and 8.1 for people who build PHP from source and have no LTS vendor.

2 · Who needs to check their server?

Any web application that runs on PHP, including **WordPress** and **Laravel**, whether through `PHP-FPM`, `mod_php`, `php-cgi`, `FrankenPHP` or `LiteSpeed`. Updating WordPress or Laravel alone is not enough, because the flaw is in PHP itself. The PHP team rates the risk highest for `mod_php`, with a demonstrated exploit that works through an unauthenticated request before any application code runs.

Diagnose before you update



Find the version you actually run

The terminal version can differ from the PHP that serves your website's requests.

3 • Check the version

First: command-line version (CLI)

```
php -v
```

This shows the CLI version only. It doesn't prove your website uses it.

Second: the PHP version serving the web

```
php-fpm8.3 -v    # example: installed binary  
# Inspect active FPM pools / web server config
```

Check how the site is wired to its FPM pool or Apache handler, or the PHP version in your hosting panel. The first command is an example of checking a binary, not proof of the version actually in use.

Don't leave a public `phpinfo()` file. It can expose paths and server configuration. Use your control panel or a trusted local check; if you must use it, restrict access and delete the file right away.

4 • Before touching production

1. Back up the site files and database, or take a `Snapshot`, and confirm you can restore it.
2. Identify the OS, the source of your PHP packages, the branch in use and the services tied to it.
3. Check plugin and dependency compatibility, especially when moving between PHP branches.
4. Plan a maintenance window and a rollback plan in case errors appear.

Linux distributions: Ubuntu / Debian / RHEL sometimes add the security fix to the original PHP version number (`backport`). In that case check your distribution's security notice and the package update, not the `php -v` number alone.

Examples, not commands that fit every server



How to update, by environment

Run these only after reviewing your server setup and the repositories you actually use.

5 • Common scenarios

Shared hosting • cPanel / Plesk

Open `MultiPHP Manager` or `Select PHP Version`, or the PHP settings in Plesk. Confirm with support that the installed runtime includes the fixes for `CVE-2026-109618` and `CVE-2026-109619`, because choosing a branch doesn't necessarily install the latest update.

Ubuntu / Debian • example for branch 8.3 from APT packages

```
sudo apt update
sudo apt install --only-upgrade 'php8.3*'
# If the active service is PHP 8.3 FPM:
sudo systemctl restart php8.3-fpm
```

First check the package names you have, where they come from and which services are active. With Apache and `mod_php` you may need to restart Apache. Check your distribution's notice if the fix is backported.

RHEL / AlmaLinux / Rocky • general example

```
sudo dnf upgrade 'php*'
# Restart only the services actually in use
```

Review AppStream, Remi or any third-party repository, the installed version, and the `php-fpm` / `httpd` services before restarting.

Docker • example for PHP 8.3 FPM

```
FROM php:8.3.36-fpm
# Then rebuild, pull and redeploy:
docker compose build --pull
docker compose up -d
```

These commands assume Compose builds the image from a Dockerfile. If you use `image:` directly, you need to pull and recreate the container, then check the version inside the new container.

Windows / IIS / Apache

Download the right security release from windows.php.net, check that your extensions and settings are compatible, then restart the related services and verify the version actually running.

Verify after updating + sources



Test... and confirm you're protected

Updating package files doesn't necessarily mean the running PHP workers picked up the new version.

6 • Final checklist

1. Restart PHP-FPM / Apache or redeploy your containers, depending on your setup.
2. Check both the CLI version and your site's runtime version, plus security notices if the fix is backported.
3. Test login, forms, file uploads, payments and other key features.
4. Review error logs and health checks after deploying.
5. If something breaks, use your rollback plan, fix the conflict quickly and update again. Don't stay on a vulnerable version.

If you can't update right now: these measures from the PHP team's announcement **reduce exposure to specific attack paths, but do not make the server safe**. Have your server admin review them; updating remains the only complete fix.

- **CVE-2026-109619:** set `session.upload_progress.enabled=0` in `php.ini`, an FPM pool or server config (not `ini_set()` or `.user.ini`), then restart every worker. This blocks the confirmed session paths.
- **PHP 8.5 only:** set `fatal_error_backtraces=0` and restart the workers.
- **CVE-2026-109618:** set `post_max_size = 8M` (or `32M` if you need uploads), keep `upload_max_filesize` below it, and ideally enforce request-size limits at the web server or proxy.
- `memory_limit=-1` (listed in the announcement) removes PHP's memory limit and can cause excessive usage or downtime. Don't apply it without a container or cgroup limit.
- Don't use `RLIMIT_AS` or `RLIMIT_DATA` as a substitute; they can trigger the same vulnerable path.

7 • Official sources

Official PHP announcement: php.net

CVE-2026-109618 · github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w

CVE-2026-109619 · github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp

Need help with the update?

Send your app type, hosting and PHP version only, with no passwords or sensitive files.

Call Abnaby on WhatsApp: 📞 [Tap here](#) • +971 56 308 1058