

PHP security alert: What to do now

For website owners, agencies and hosting providers
General edition | Prepared: 22 October 2026

Author: **Dominik Jedro** · justdigital.world

Priority: confirm the PHP patch status now. Updates released on October 21 address two critical flaws that can allow remote code execution without signing in. [1]

1. Assign one person to coordinate

Assign a website owner to open an urgent hosting support ticket, follow up by phone if needed, and record the contact, status and next update time.

2. Confirm which systems use PHP

Your own website or server: Ask the host or administrator to check every PHP application, domain, subdomain, staging site and embedded PHP component. Include Joomla, Drupal, shops and custom applications.

Website builder or managed service: Ask the provider to confirm PHP use and remediation. **Just visiting websites?** PHP is updated on the server; there is no PHP patch to install in your browser.

Scope: This general guide does not assess any specific website. The operator must establish whether each installation is affected or already patched.

WordPress and plugin updates do not replace the PHP fix: attacks can occur before application code runs. The flaws are CVE-2026-109618 and CVE-2026-109619. [1]

3. Apply the appropriate security update [1]

PHP branch	Fixed release*
8.2	8.2.35
8.3	8.3.36
8.4	8.4.27
8.5	8.5.12

* Or a later fixed release in the same branch. A documented vendor backport covering both CVEs is also acceptable. For PHP 8.1 or older, require evidence of a vendor fix or upgrade to a supported branch. [1]

How to fix the vulnerability

Choose the route that matches your hosting setup.

A. WordPress on shared or managed hosting

- 1. Read the PHP version:** In WordPress admin, open Tools > Site Health > Info > Server. Record the full PHP version. This screen reports information; it does not install PHP patches. [3]
- 2. Check with the host:** Open the PHP settings or version manager in the hosting account. The provider must supply the fix for the affected website. A selector showing “PHP 8.3” alone does not confirm patch status; require the full version or vendor backport evidence.
- 3. Activate the patch:** Check backups and restoration, then have the fixed release applied. If the panel does not let you select a patched release, send the support ticket on page 4. Avoid an unnecessary PHP branch change.
- 4. Verify the result:** Recheck the PHP version, obtain confirmation that the fix is active and run the checks on page 3. Keep WordPress, themes and plugins current too; that does not replace the PHP fix. [1]

B. Self-managed PHP server, VPS or container

Linux packages: Identify the distribution, package source and actual web PHP service. Install the corrected PHP packages from a maintained source, with compatible extensions. Check vendor coverage of both CVEs. Validate configuration and restart the affected FPM/web-server processes. Updating only the CLI package is insufficient.

Containers: Obtain a verified fixed base image, rebuild the application image if needed and replace every running instance. Changing an image tag alone does not update running containers.

Windows or source builds: Deploy the matching fixed vendor build or patched source with compatible extensions and restart affected services. Exact commands must match the operating system, installation method and service.

If an update fails: block access to the vulnerable system and coordinate recovery with the operator. Do not restore an unpatched version to public service.

Hosting provider checklist

Goal: the fix is active in every affected web environment.

Before and during the update

- ☐ Record the PHP version, vendor package and responsible provider for each domain or environment. Explicitly document any environment that does not use PHP.
- ☐ Check current file and database backups and the ability to restore. If compromise is suspected, preserve existing logs and evidence.
- ☐ Prefer the security release in the existing supported PHP branch. Avoid an unnecessary major upgrade during the emergency; check compatibility when migrating an obsolete branch.
- ☐ Update every affected instance, including multiple servers, containers and separate PHP pools. Restart or redeploy PHP services in a controlled way.
- ☐ Verify the runtime actually serving web requests. Command-line `php -v` alone does not prove the website is patched. Do not leave a public `phpinfo` page exposed.

If immediate patching is not possible

Arrange temporary isolation of the affected application from the public internet until the fix is applied and verified. CERT-EU recommends considering this measure. [2]

If a static maintenance page is used, access to the unpatched PHP origin must remain blocked. A WordPress maintenance plugin or a firewall rule alone is not evidence of complete protection. [1,2]

After the update: owner and provider together

- ☐ Test the homepage, key pages, forms and email delivery, login and checkout where applicable. Check each application separately.
- ☐ Review logs, unknown admin accounts, scheduled jobs and changed files for suspicious activity. Patching does not remove an existing intrusion.
- ☐ If there are concrete signs of compromise: isolate the application, preserve evidence and involve incident response specialists.

Message to the hosting provider

Copy this into an urgent support ticket.

Subject: Urgent - confirm PHP security patches for our websites

Hi team,

Please urgently check our hosting environments for [insert domains / applications], including subdomains and staging environments, for CVE-2026-109618 and CVE-2026-109619.

For each environment, please confirm:

1. Does it use PHP, and what exact web PHP version or vendor package is active?
2. Are both flaws fixed by the October 21, 2026 security releases or a documented vendor backport?
3. When did the fix become active, and were all PHP workers or instances restarted or redeployed?
4. Have you found suspicious activity in logs or files, and what period did you review?

If patching is still pending, please provide a firm completion time and propose temporary isolation of the affected application. Please identify any environments outside your responsibility so we can follow up separately.

Please reply with technical evidence of the active patch status for each environment and a named contact. Thank you.

Close the task when ...

every environment is accounted for, the active fix or lack of PHP exposure is documented, functional checks pass, and any suspicious findings have been assessed. A reply saying “WordPress is up to date” is insufficient.

Owner: _____ Ticket: _____
Completed on / by: _____

Sources and scope

[1] PHP Project: Security advisory, 21 October 2026.

<https://www.php.net/>

[2] CERT-EU: Security Advisory 2026-018.

<https://cert.europa.eu/publications/security-advisories/2026-018/>

[3] WordPress.org: Site Health screen.

<https://wordpress.org/documentation/article/site-health-screen/>

Source date: October 22, 2026. General operational guidance, not a security assessment. The provider must verify actual remediation. Check current vendor guidance if using this document at a later date.

Prompt for Claude / GPT

Copy, fill in the placeholders and paste into Claude or ChatGPT.

Without connected hosting or server tools, the AI can guide you but cannot update the server itself. A WordPress MCP connection alone usually cannot manage PHP packages. Use secure connections for credentials; never paste secrets into the chat.

Help me secure my own WordPress/PHP installation against CVE-2026-109618 and CVE-2026-109619. Explain the steps clearly and guide me through verified remediation.

Website(s): [domains]

Application: [WordPress / other / unknown]

Hosting and operating system: [provider / system / unknown]

Access: [WordPress admin / hosting panel / SSH / containers / MCP / none]

Backup status: [date / unknown]

1. Check current primary sources first: <https://www.php.net/> and <https://cert.europa.eu/publications/security-advisories/2026-018/> plus my vendor's advisories. As of October 22, 2026, fixed releases are PHP 8.2.35, 8.3.36, 8.4.27 and 8.5.12, or later fixed releases in the same branch. Accept documented vendor backports covering both CVEs. If you cannot browse, say so and ask for the current vendor advisory.

2. Begin read-only. Ask for missing details and use only available, authorized tools. Identify the web PHP runtime, installation method and every affected environment. Distinguish CLI PHP from web PHP. For WordPress: Tools > Site Health > Info > Server. Without access, write a specific hosting support ticket. Never claim to have performed checks you could not run.

3. Present a short patch plan covering backups, recovery and functional checks. Show exact commands or panel steps matched to my installation and explain their effects. Ask for approval before production changes or downtime. Prefer a patch in the existing supported branch; for obsolete branches, plan a compatible migration or documented vendor fix.

4. After approval, make only agreed changes, restart affected services or replace containers, and verify every instance. Do not claim that plugin updates, WAF rules or a maintenance plugin fix PHP itself. Do not run exploit tests or apply blanket memory-limit workarounds. Never ask me to paste passwords, API keys or complete configuration files into chat.

5. Prove the fix is active in web PHP. Test pages, forms, email delivery, login and checkout where applicable; agree on a test before sending real messages or placing orders. Review logs and files for signs of compromise. If suspicious activity is found, preserve evidence and involve specialists.

6. If immediate patching is impossible, propose controlled isolation from the public internet. Finish with each environment's before/after state, evidence, test results, unresolved items and owner. Use only these statuses: "confirmed patched", "demonstrably not affected" or "unverified". A reachable website or green WordPress status is not proof of patching.