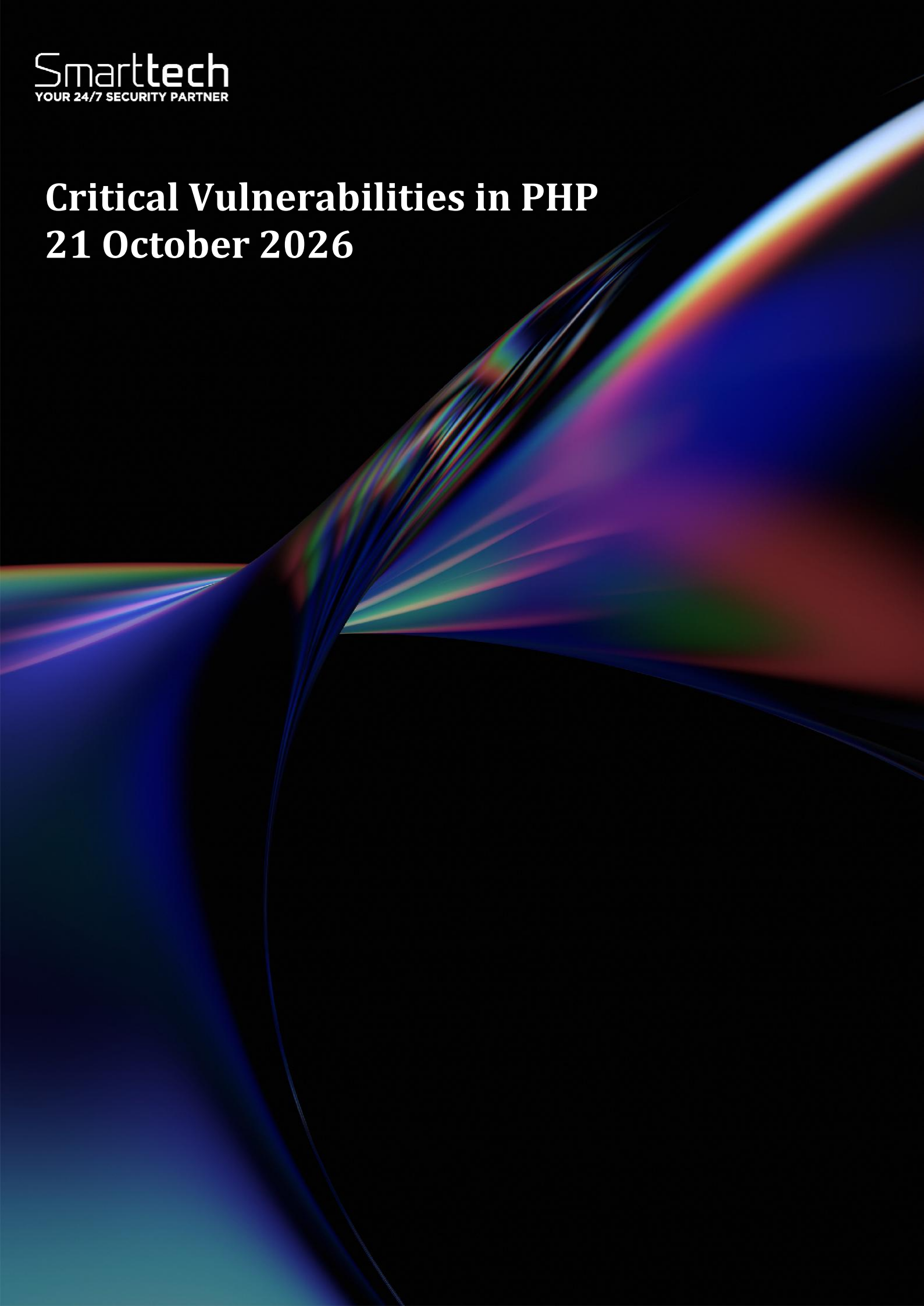
Smarttech
YOUR 24/7 SECURITY PARTNER

Critical Vulnerabilities in PHP
21 October 2026



Document ID	SMA-Threat Report
Document status	ISSUED
Issue Number	181
Authors	Gabriel Alexandru< gabriel.alexandru@smarttech247.com >
Verified by	Alin Curcan < alin.curcan@smarttech247.com >
Last modified	2026-10-21
Issue Date	2026-10-21

This document and any information therein are confidential property of Smarttech247 and without infringement neither the whole nor any extract may be disclosed, loaned, copied or used for manufacturing, provision of services or other purposes whatsoever without prior written consent of Smarttech247, and no liability is accepted for loss or damage from any cause whatsoever from the use of the document. Smarttech247 retain the right to alter the document at any time unless a written statement to the contrary has been appended.

Threat Reports are reports created by Smarttech247 based on high and critical severity vulnerabilities that may have a high potential to be exploited in the wild i.e. vulnerabilities that are present in most used products by companies and do not have an auto-update option or they are usually not automatically updated in case that could lead to some service disruption. This report is usually created as soon as the vulnerability is released, therefore we strongly recommend that the information is reviewed, tests are performed and patches are applied before the first proof-of-concept is released.
Even though certain vulnerabilities may not have an active exploit in the wild at the time that we report on them, we take into consideration the wider risk and the impact it could have on systems, should an exploit like that be available after a while. Our duty is to report them on time and we recommend enterprises that, in order to keep critical business systems protected, they should consider, on average, ten working days to check whether or not the new vulnerability affects them, and if so, to implement actions in order to remove the risk.

Overview:

Two related PHP vulnerabilities (CVE-2026-109618 and CVE-2026-109619) allow unauthenticated remote code execution by corrupting or reusing memory that belongs to a single request. The PHP project released fixes on October 21, 2026, and urges all users to update immediately.

RISK:

Government:

- Large and medium government entities: **Critical**
- Small government entities: **Critical**

Businesses:

- Large and medium business entities: **Critical**
- Small business entities: **Critical**

SYSTEMS AFFECTED:

The PHP project states that all PHP installations serving web requests are at risk, including PHP-FPM, Apache mod_php, php-cgi, FrankenPHP and LiteSpeed, with the highest risk for mod_php. The fixed releases are PHP 8.5.12, 8.4.27, 8.3.36 and 8.2.35 (October 21, 2026), while PHP 8.1 and older no longer receive project updates and need a vendor backport or an upgrade.

Branch	Status	Fixed version	Notes
PHP 8.5	Supported	8.5.12	If patching is delayed, fatal_error_backtraces=0 is a temporary mitigation for a specific attack path.
PHP 8.4	Supported	8.4.27	Upgrade to the fixed release.
PHP 8.3	Supported	8.3.36	Upgrade to the fixed release.
PHP 8.2	Security-only	8.2.35	Security support ends 31 December 2026; plan migration.
PHP 8.1	EOL	No release	Use the PHP project's PHP-8.1-CVE-2026-109618-109619 source backport or a trusted vendor backport.
PHP	EOL	No release	Use the PHP project's PHP-8.1-CVE-2026-109618-109619 source

Branch	Status	Fixed version	Notes
8.0			backport or a trusted vendor backport.
PHP 7.4	EOL	No release	Use the PHP project's PHP-8.1-CVE-2026-109618-109619 source backport or a trusted vendor backport.

TECHNICAL SUMMARY:

Vulnerability 1 (CVE-2026-109618, GHSA-w8hw-w9fx-j79w): Out-of-memory (OOM) handling via longjmp()

PHP uses longjmp() to abort a request when an allocation fails. In some contexts, request memory is reachable before it is fully initialized. If an allocation fails at that point, the early exit leaves request memory in an invalid state, and the request shutdown procedure then reads or branches on it. The failed allocation does not need to be related to the invalid memory.

Vulnerability 2 (CVE-2026-109619, GHSA-6wmf-4rv5-p3rp): Stale request state causing use-after-free

PHP workers retain global state (error, session, object and extension state) that points into memory owned by one request. If that state is not reset before the request heap is freed, an operation in a later request that runs before normal request initialization can consume the dangling pointer. A crafted heap layout can turn this use-after-free into full RCE.

If you cannot update immediately: while only an update is a complete solution, the following measures reduce exposure to specific attack paths but do not make an installation safe.

RCE via OOM corruption (CVE-2026-109618):

- Set memory_limit=-1, preferably in a way applications can't override, such as php_admin_value in the FPM pool or Apache configuration, and check that no pool, directory or application setting puts a finite limit back. This removes the controllable PHP memory limit failure, but an underlying allocation failure can still reach the vulnerable path.
- Set a low post_max_size. For applications that don't require file uploads, set post_max_size = 8M. If file uploads are necessary, set post_max_size = 32M. Keep upload_max_filesize below post_max_size. Consider enforcing limits at the web server or proxy level for decoded request size, input field count, multipart part count, header count, and nesting depth. These limits must also cover requests without a usable Content-Length.
- Use a container or cgroup limit as an availability boundary, preferably with a policy that kills only the affected worker. Do not substitute RLIMIT_AS or RLIMIT_DATA, as these can cause the vulnerable allocation failure.

RCE via stale global state (CVE-2026-109619):

- Set session.upload_progress.enabled=0 and restart every worker. Use php.ini, an FPM pool or server configuration that applies before request parsing. Do not rely

on ini_set() or FPM .user.ini. This blocks the confirmed session paths but not other variants.
- On PHP 8.5, set fatal_error_backtraces=0 and restart every worker. This blocks the confirmed fatal backtrace path only.
- Remove invalid or deprecated settings applied per request that emit diagnostics before initialization. Hiding diagnostics with error_reporting, display_errors or @ does not help.
- Fix attacker-reachable fatal errors and execution timeouts, including failures in callbacks, upload or import processing and cleanup.

Recommendations:

We recommend the following actions be taken:

- Inventory all PHP runtimes (FPM, mod_php, php-cgi, FrankenPHP, LiteSpeed). Flag internet-facing hosts and EOL versions.
- Upgrade to PHP 8.5.12, 8.4.27, 8.3.36 or 8.2.35, and restart all workers afterwards.
- Use vendor or distro packages and check their advisories. Backports may not change the version string, so verify by package changelog.
- Prefer migration to a supported PHP branch. Where migration is not immediately possible, use a trusted vendor-supported backport or carefully validated source backport, and establish a migration plan.
- Where patching is delayed, apply interim mitigations as a stopgap only:
    - Set session.upload_progress.enabled=0.
    - Set memory_limit=-1 non-overridably.
    - Lower post_max_size.
    - Enforce proxy request-size limits.
    - Apply a cgroup memory limit.
    - On PHP 8.5, set fatal_error_backtraces=0.
- Do not run circulating PoC code outside an isolated lab. Some may be malware aimed at defenders.

References:

- https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w
- https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp
- https://www.php.net/archive/2026.php#2026-10-21-5
- https://www.tenable.com/blog/php2shell-cve-2026-109618-cve-2026-109619-frequently-asked-questions-about-critical-php-remote
- https://www.rescana.com/post/php-rce-flaws-patch-october-2026-prepare-now

CVE: CVE-2026-109618, CVE-2026-109619

Smarttech
YOUR 24/7 SECURITY PARTNER

www.smarttech247.com