

CRITICALSTART® Security Advisory

TLP CLEAR // [CS-SA-26-1002] PHP2SHELL RCE Vulnerabilities

Executive Summary

The **CRITICALSTART®** Cyber Research Unit (CRU) is tracking PHP2Shell, the name for two critical flaws in the PHP: Hypertext Preprocessor (PHP) runtime, tracked as Common Vulnerabilities and Exposures (CVE) identifiers CVE-2026-109618 and CVE-2026-109619. Both can allow unauthenticated remote code execution (RCE) with the privileges of the web worker account, and PHP rates each 9.2 on Common Vulnerability Scoring System (CVSS) version 4.0. PHP and its research partners disclosed the flaws on October 21, 2026, after a leak on October 20 forced a release one day ahead of the planned October 22 disclosure. The flaws stem from how PHP handles fatal errors, particularly out-of-memory (OOM) errors, during request processing. A request that ends abnormally can leave corrupted state behind, and a later request on the same worker can reuse it. CRU has not identified any known threat actor and the sources reviewed describe it without attribution. PHP released fixes on October 21, 2026, in versions 8.5.12, 8.4.27, 8.3.36, and 8.2.35. Every PHP deployment that serves web requests on an unpatched release is exposed, and the flaws sit in the runtime rather than in any application.^[1, 2] PHP runs on 69.3% of websites with a known server-side language, according to W3Techs. That figure measures PHP adoption not confirmed vulnerable sites.^[4, 6] Estimates put the number of internet-exposed PHP hosts between 1.3 million and 5.1 million. Researchers report exploitation attempts prior to patch release. CISA, NSA, and other authorities have put out advisories and mitigation guidance. Organizations should inventory every PHP runtime that serves web requests, patch internet-facing systems first, and restart PHP processes after updating.

Vulnerability Context

CVE	CVSS v4.0	Description
CVE-2026-109618	9.2	RCE via out-of-memory (OOM) corruption. Request memory is left in an invalid state when an allocation fails during early request handling. ^[1]
CVE-2026-109619	9.2	RCE via stale global state. Request-owned state survives after its memory is freed and is reused by a later request on the same worker. ^[2]

Both flaws corrupt PHP's memory when a request ends abnormally. This section summarizes the mechanisms in PHP's advisories and omits exploitation detail. Both CVEs share the vector

CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N. Attack complexity is rated High and attack requirements are present. PHP states that exploit reliability depends on the PHP build, extensions, frontend behavior, and memory allocator layout.^[1, 2]

Affected and Fixed Versions

Branch	Affected	Fixed	Notes
PHP 8.5	Before 8.5.12	8.5.12	Supported branch. ^[1, 2]
PHP 8.4	Before 8.4.27	8.4.27	Supported branch. ^[1, 2]
PHP 8.3	Before 8.3.36	8.3.36	Supported branch. ^[1, 2]
PHP 8.2	Before 8.2.35	8.2.35	Reaches end of life on December 31, 2026. ^[8]
PHP 8.1 and older	Reported affected back to PHP 5.4 ^[3, 4]	-	PHP provides minimal source backports for 7.4, 8.0, and 8.1. Distributions and long-term support (LTS) vendors may provide their own backports. ^[5, 7]

CVE-2026-109618: Memory Corruption After Out-of-Memory Errors

PHP uses a jump function, `longjmp()`, to abandon a request when memory runs out. In some contexts, request memory is reachable before PHP fully initializes it. If an allocation fails at that point, request cleanup reads invalid state, which can lead to code execution.^[1] PHP's advisory identifies several multipart request parsing problems that can push a request to the memory limit while it stays within its apparent size limits.^[1] On Apache `mod_php`, this happens entirely before application code begins to run.^[1]

CVE-2026-109619: Stale State Carried Between Requests

A PHP worker handles many requests in sequence. Some internal state, including error, session, object, and extension state, can still point to memory that a finished request owned. A later operation on the same worker can use that stale state before PHP re-initializes it, which creates a use-after-free condition that can lead to code execution.^[2] Exploitation requires a producer, which leaves stale state behind after a fatal error or memory exhaustion, and a consumer, which acts on that state in a later request on the same worker.^[2] PHP confirms two consumers: diagnostics raised while applying configuration before request initialization, and session upload progress handling.^[2] Two cases are confirmed. A PHP 8.5 fatal error backtrace path has demonstrated `mod_php` exploitation, and a session upload progress path was validated against both `mod_php` and PHP-FPM.^[2] Session upload progress is enabled by default, and the application does not need to call `session_start()` for it to be reachable.^[2] Running multiple workers lowers the success rate per request but does not remove the condition.^[2]

Memory Disclosure and Public Demonstrations

Wiz reports that an attacker could read sensitive process memory, including passwords, tokens, application programming interface (API) keys, and certificates, without achieving code execution.^[3] PHP's advisory for CVE-2026-109619 also lists memory disclosure as a confirmed impact.^[2] Wiz demonstrated unauthenticated RCE against default container images of WordPress, Drupal, and Joomla, and states that exploitation does not depend on a flaw in the application.^[3]

Exposure by Server Interface

Risk differs by how PHP connects to the web server. PHP rates Apache `mod_php` highest because an unauthenticated request can trigger the attack before any application code runs.^[7]

Server Application Programming Interface (SAPI)	CVE-2026-109618	CVE-2026-109619
Apache <code>mod_php</code>	RCE validated over Hypertext Transfer Protocol (HTTP) with default input and memory limits. ^[1]	RCE validated through the PHP 8.5 fatal backtrace path and the session upload progress path. ^[2]
PHP-FPM	Susceptible after normal startup by source analysis. No complete exploit demonstrated. ^[1]	Session lifecycle RCE validated. It depends on a suitable operation inside the application. ^[2]
FrankenPHP	Likely susceptible to application-independent attacks. Source analysis only. ^[1]	No early consumer established in the default configuration. Application or extension behavior may still expose it. ^[2]
LiteSpeed Server Application Programming Interface (LSAPI)	Application-specific exploits likely. Unvalidated. ^[1]	Conditional candidate through early environment settings. ^[2]

MITRE ATT&CK Mapping

The mapping below, drawn from the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework, describes behaviors that are in scope.^[5]

Technique	Name	Context
T1190	Exploit Public-Facing Application	Crafted HTTP requests trigger memory corruption in PHP on internet-facing web servers. ^[1, 2, 4]
T1059	Command and Scripting Interpreter	PHP worker processes spawn shells or interpreters after code execution. ^[5]
T1059.004	Unix Shell	PHP worker processes spawn shells or interpreters after code execution. ^[5]
T1505.003	Server Software Component: Web Shell	Attackers may place web shells in the web root for persistence. ^[5]
T1105	Ingress Tool Transfer	Download utilities such as <code>curl</code> and <code>wget</code> retrieve additional tools. ^[5]
T1033	System Owner/User Discovery	Commands such as <code>whoami</code> and <code>id</code> run from the web server process. ^[5]
T1082	System Information Discovery	Commands such as <code>uname -a</code> run from the web server process. ^[5]
T1082	System Network Configuration Discovery	Network discovery commands run can be indicators

Implications for Organizations

The central risk is that these flaws sit below the application layer. Updating WordPress, its plugins, or any application code does not patch the PHP runtime that serves it, so each organization needs to confirm the runtime version on every host.^[4, 13] PHP often runs where teams do not track it. Control panels, vendor appliances, containers, and Windows software that bundles PHP can all carry an affected runtime, and managed hosting customers depend on their provider to patch.^[5, 13] Impact depends on the account PHP runs as. Successful exploitation yields code execution with the web worker's privileges, which does not automatically grant administrator control of the server.^[1, 2, 13]

The worker can still reach application files, credentials, and secrets available to that process, and Wiz reports that memory disclosure alone can expose passwords, tokens, API keys, and certificates.^[3, 13] Organizations that suspect exploitation should treat those secrets as exposed.

Configuration changes reduce risk but do not remove it. PHP states that updating is the only complete mitigation for both flaws, and that no generic configuration change covers every state-state path.^[1, 2] Cloudflare describes its web application firewall (WAF) rules as an additional layer of defense in depth, and those rules protect only traffic that passes through Cloudflare.^[4, 6]

Unsupported PHP is the main structural exposure. PHP 8.1 and older receive no official update, and PHP 8.2 reaches end of life on December 31, 2026.^[7, 8] Organizations that run these branches need a vendor backport, a source backport, or an upgrade.^[5, 7] The fix also changes memory behavior. The patch adds a hard memory limit that ends the process when reached, so teams should review `memory_limit` and `post_max_size` after updating.^[1, 8]

Note: Patching does not finish remediation until PHP restarts. Installing the fixed package does not load the patched runtime into running processes. Restart Apache, PHP-FPM, and every long-running PHP worker.^[8, 12] If a host may have been exposed before patching, rotate the secrets that the PHP process could read.

Detection and Hunting Guidance

These flaws execute below the application layer, so application logs may contain little context.^[3] Treat the signals below as leads for investigation, not proof of compromise. Applying the PHP security update remains the only complete remediation.^[3]

Request and server signals

- Unusual multipart requests followed by memory-limit errors or repeated HTTP 500 responses.^[3]
- Unexpectedly large or malformed `PHPSESSID` response cookies.^[3]
- Unexplained PHP worker restarts or churn.^[3]
- Repeated warnings, deprecations, or core errors logged before target scripts run, and `Maximum execution time exceeded` messages.^[2]

Host and process signals

- PHP worker processes such as `php-fpm`, `php-cgi`, `httpd`, or `apache2` launching shells (`sh`, `bash`, `cmd.exe`, `powershell.exe`), interpreters (`python`, `perl`), or download and network tools (`curl`, `wget`, `nc`, `certutil.exe`).^[5]
- Reverse shell patterns in command lines, such as `/dev/tcp/`, `fsockopen`, and `mkfifo`.^[5]
- Reconnaissance commands such as `whoami`, `id`, `uname -a`, and `ipconfig` run from the web server.^[5]
- New files in the web root, new cron jobs or scheduled tasks, and new accounts.^[5]

No indicators of compromise (IOCs) have been confirmed in the sources reviewed as of October 21, 2026.^[5] If a host shows suspicious activity, isolate it and preserve its web server and PHP logs.^[5]

Prioritized Mitigation Strategies

To improve your organization's security posture considering the PHP2SHELL Vulnerability exploitation, CRU recommends the following:

Immediate (Within 24 Hours)

- **Inventory every PHP runtime that serves web requests.** Include containers, control panels such as cPanel and Plesk, vendor appliances, and Windows software that bundles `php.exe` or `php-cgi.exe`. Start with internet-facing systems, and treat Apache `mod_php` and FrankenPHP hosts as the highest priority.^[1, 5]
- **Install a fixed PHP release.** Update to 8.5.12, 8.4.27, 8.3.36, or 8.2.35, or install your distribution vendor's package. Confirm the fix by checking the package changelog for the CVE, because distributions can backport fixes under older version numbers.^[1, 2, 8] Then restart Apache, PHP-FPM, and every long-running worker.^[8, 12]
- **Apply interim mitigations on hosts that cannot be patched today.** Set `session.upload_progress.enabled=0` in `php.ini`, the PHP-FPM pool, or server configuration, not through `ini_set()` or `.user.ini`, and restart every worker.^[2, 7] On PHP 8.5, also set `fatal_error_backtraces=0`.^[2] Set and enforce `memory_limit=-1` administratively, keep `post_max_size` low, and enforce request size limits at the web server or proxy.^[1] If the application does not use multipart forms, block multipart requests before they reach PHP.^[1] Do not use `RLIMIT_AS` or `RLIMIT_DATA`, which trigger the vulnerable allocation failure path.^[1] Unlimited memory raises denial-of-service risk on shared servers, so apply it deliberately. These steps reduce exposure to specific paths only.^[1, 2]
- **Keep WAF protections enabled while patching.** On Cloudflare paid plans, confirm the Managed Ruleset is deployed in block mode.^[4] Treat WAF rules as a stopgap and not a substitute for the update.^[4, 6]
- **Isolate systems that cannot be updated.** Take internet-facing hosts that run unsupported PHP with no available backport offline, or restrict them behind a virtual private network (VPN) or WAF until they can be upgraded.^[5, 7]

Short-Term (Within Two Weeks)

- **Get written confirmation from hosting providers and managed vendors** that the PHP runtime running on your behalf is patched and restarted. Updating an application does not replace the server's PHP installation.^[13]
- **Hunt for compromise across the exposure window.** Details leaked on October 20, and some vendors report attack attempts on October 21.^[4, 5] Apply the detection guidance in this advisory to web server logs, PHP logs, and endpoint telemetry.^[3, 5]
- **If you find suspicious activity, contain first.** Isolate the server, preserve web server and PHP logs, and search for web shells, new web root files, cron jobs, scheduled tasks, and new accounts.^[5] Rotate database passwords, API keys, tokens, and certificates that the PHP process could read.^[3]
- **Audit PHP configuration.** Remove invalid or deprecated settings applied per request through `.htaccess`, Apache `php_value`, or nginx `PHP_VALUE`. Fix fatal errors and execution timeouts that an attacker can trigger, including in upload and import handling.^[2] Consider leaving `session.upload_progress.enabled=0` in place after patching if the application does not use the feature.^[2]
- **Review memory settings after updating.** The fix adds a hard memory limit, so check `memory_limit` and `post_max_size` against application needs.^[1, 8]

Structural

- **Retire unsupported PHP.** Move every host off PHP 8.1 and older, and plan the upgrade of PHP 8.2 hosts before its end of life on December 31, 2026.^[7, 8]
- **Maintain an inventory of embedded runtimes,** including vendor products and container base images, and rebuild images from fixed tags once the image publisher releases them.^[5]

- **Restrict direct access to origin servers** so web servers accept traffic only from your WAF or content delivery network, because those protections cover only traffic routed through them.^[6]
- **Add process-level detection for PHP child processes** on Linux web servers that lack endpoint detection and response (EDR) coverage. Free tools such as process-watcher can provide visibility in observe mode.^[5]

Conclusion

PHP2Shell affects the PHP runtime that serves a large share of the web, and fixed versions are available now for supported branches. Updating the runtime and restarting PHP processes is the only complete remediation, and interim configuration changes reduce exposure only along specific paths. Organizations should patch internet-facing systems first, isolate unsupported PHP, and hunt for signs of compromise using the guidance above. For more threat reports, including H2 2025 detailing trending cybersecurity concerns visit [Critical Start's Intel Hub](#). Should anything new surface, this advisory will be updated. This advisory was written using the best intelligence available at the time and is subject to change as additional information becomes available.

What Critical Start is Doing

Critical Start is actively threat hunting for vulnerable versions and will be contacting customers through established channels for remediation. Likewise, the Security Engineering team is building applicable activity-based detections for global deployment. CRU continues to actively monitor the situation and work closely with the SOC for heightened vigilance. For future updates, CRU will post updates via Cyber Operations Risk & Response™ Bulletins and on the [CRITICALSTART® Intelligence Hub](#).

Further Reading

- [1] PHP Group (October 21, 2026). GHSA-w8hw-w9fx-j79w: RCE via OOM corruption (CVE-2026-109618). <https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>
- [2] PHP Group (October 21, 2026). GHSA-6wmf-4rv5-p3rp: RCE via stale global state (CVE-2026-109619). <https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>
- [3] Wiz Research (October 21, 2026). PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70% of the Internet. <https://www.wiz.io/blog/php2shell-php-rce>
- [4] Cloudflare (October 21, 2026). Cloudflare deploys protections against php2shell vulnerability. <https://blog.cloudflare.com/php-vulnerabilities-waf-protection/>
- [5] Huntress (October 21, 2026). Rapid Response: php2shell, Leaked Critical PHP Vulnerabilities (CVE-2026-109618 & CVE-2026-109619). <https://www.huntress.com/blog/php2shell-leaked-critical-php-vulnerabilities>
- [6] Tenable (October 21, 2026). php2shell (CVE-2026-109618, CVE-2026-109619): Frequently asked questions about critical PHP remote code execution vulnerabilities. <https://www.tenable.com/blog/php2shell-cve-2026-109618-cve-2026-109619-frequently-asked-questions-about-critical-php-remote>
- [7] BleepingComputer (October 21, 2026). PHP ships emergency patches for critical PHP2Shell flaws. <https://www.bleepingcomputer.com/news/security/php-ships-emergency-patches-for-critical-php2shell-flaws/>
- [8] The Hacker News (October 21, 2026). PHP Issues Emergency Patches for Two Critical PHP2Shell RCE Flaws as Exploit Attempts Emerge. <https://thehackernews.com/2026/10/php-issues-emergency-patches-for-two.html>
- [9] watchTowr (updated October 21, 2026). Frequently Asked Questions About PHP Remote Code Execution (PHP2Shell CVE-2026-109618 & CVE-2026-109619). <https://watchtowr.com/intelligence/php-cve-2026-109618-faq/>
- [10] Check Point (October 2026). php2shell: What We Know About the Leaked Critical PHP Vulnerabilities (CVE-2026-109618 & CVE-2026-109619). <https://blog.checkpoint.com/security/php2shell-what-we-know-about-the-leaked-critical-php-vulnerabilities-cve-2026-109618-cve-2026-109619>
- [11] SOCRadar (October 21, 2026). PHP2Shell: CVE-2026-109618 & CVE-2026-109619. <https://socradar.io/blog/php2shell-cve-2026-109618-cve-2026-109619/>
- [12] ma.ttias.be (October 21, 2026). PHP CVE-2026-109618 and CVE-2026-109619: how to apply the patches. <https://ma.ttias.be/php-cve-2026-109618-patch/>
- [13] Hackread (October 21, 2026). PHP Releases Emergency Fixes for Critical PHP2Shell Vulnerabilities. <https://hackread.com/php-releases-emergency-fixe-php2shell-vulnerabilities/>