

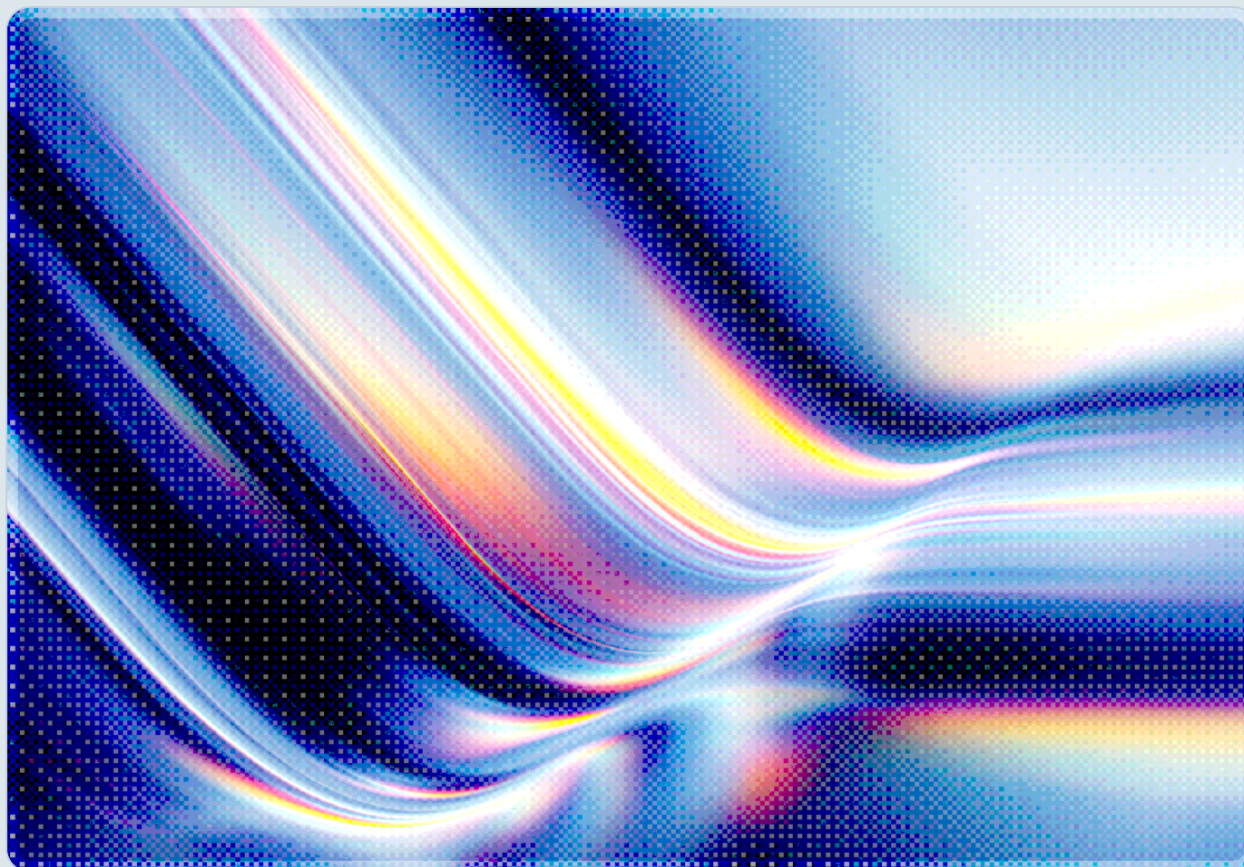
PHP2Shell: Exploitation Before Disclosure for an AI-Found Flaw

Security Implications and Guidance

2026-10-23



AI-assisted Rapid Research



CSAI

cloud
CSA security
alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

PHP2Shell is the collective name for two critical vulnerabilities in the PHP runtime, CVE-2026-109618 and CVE-2026-109619. Each is rated 9.2 under CVSS v4.0 and allows unauthenticated remote code execution on servers that handle web requests [2][5][7]. Because the flaws sit in the PHP engine and not in application code, an otherwise secure application can still be exposed. Wiz reports that even a minimal "Hello World" application on an affected PHP version may be vulnerable [1]. Fixed releases are PHP 8.5.12, 8.4.27, 8.3.36, and 8.2.35. The PHP project no longer ships updates for PHP 8.1 and older, so those branches depend on distribution backports or a migration [2][5].

The flaws were found with AI assistance. Wiz Research attributes its discovery to its Wiz Atlas platform working alongside frontier models from Google and OpenAI [1]. The PHP Foundation reports that three parties, Calif, Wiz, and OpenAI, each independently reported the same out-of-memory root cause between August 11 and September 18, 2026 [2].

The disclosure process broke down at several points. The CVE identifiers leaked on October 20, which led the PHP project to move the coordinated release forward by a day [2]. Within hours, one of the reporting firms accidentally published a proof-of-concept video, and a downstream distributor later shipped patched binaries more than six hours before the rescheduled release [2][4].

The most important finding for defenders comes from Amazon Threat Intelligence. Its MadPot honeypot recorded a PHP2Shell exploitation attempt on October 17, 2026. That was four days before public disclosure and three days before the CVE identifiers leaked. Amazon states it is unclear how the actor obtained exploit information [3]. Investigations should therefore start their review window no later than October 17, rather than at the October 21 patch date, and should extend further back where log retention allows, since the earliest exploitation is unknown [3].

Patching the PHP runtime is the only complete fix. The PHP team describes the configuration mitigations as partial, and Amazon reports that no configuration-only workaround addresses both CVEs [3][5]. One of the PHP2Shell bug paths also enables remote disclosure of process memory, which Wiz reports can include passwords, tokens, API keys, and certificates [1][5]. CSA therefore recommends that organizations that cannot rule out exposure rotate any secrets those PHP workers held.

Background

PHP is the server-side language behind a large share of the public web. Cloudflare, citing W3Techs, describes it as used by almost 70% of websites [6]. Estimates of directly reachable PHP infrastructure vary with methodology. VulnCheck counted 2,608,039 internet-facing hosts that expose PHP, and Modat counted 4.38 million internet-facing PHP services as of October 20, 2026. Modat estimated that about 79% of those services run branches that no longer receive upstream support [4]. That last figure matters as much as the vulnerability itself, because a large part of the exposed population has no official upstream release to apply; for those branches, the PHP security team has provided only minimal source backports for 7.4, 8.0, and 8.1 [5].

Both vulnerabilities come from how PHP handles fatal errors during request processing, especially out-of-memory conditions. An aborted request can leave internal state corrupted or dangling, and a later request on the same reused worker process then touches that state [1]. According to the PHP Foundation, CVE-2026-109618 is an out-of-memory corruption flaw that leads to code execution, while CVE-2026-109619 is a stale global-state flaw that produces a use-after-free [2]. Wiz describes four related bug paths behind the two CVEs: an upload-progress unserializer use-after-free, a dangling session-ID pointer that leaks heap contents through the response cookie, a fatal-error backtrace use-after-free, and an OPcache type confusion across workers [1]. Sources differ on which CVE covers the memory-disclosure path. Wiz places it under CVE-2026-109618 [1], whereas BleepingComputer associates memory disclosure with CVE-2026-109619 [5]. Wiz reports demonstrating unauthenticated code execution against current container images of WordPress, Drupal, and Joomla, and adds that the memory-disclosure path can expose passwords, tokens, API keys, and certificates belonging to other requests [1].

The flaws affect every PHP deployment model that serves web requests, including PHP-FPM, Apache mod_php, php-cgi, FrankenPHP, and LiteSpeed [5]. The PHP security advisory considers the risk highest for Apache mod_php [5][7]. In that configuration, according to the PHP Foundation, an unauthenticated request can trigger the out-of-memory bug before any application code runs, and full code execution was demonstrated with default settings. On PHP-FPM, code execution was confirmed through application-level code paths [2]. Wiz states that every PHP version released in the last decade is affected by at least one of the bugs [1].

PHP2Shell is also a documented case of AI-assisted vulnerability research. Wiz reports that it found its first issue on August 19, 2026, using Wiz Atlas with Gemini 4 Argon and GPT-5.6 Sol. Wiz also claims that AI helped adapt exploits across PHP versions and memory layouts, so that one researcher did in about a week what previously would have taken a team months, a vendor self-assessment rather than an independent measurement [1]. The PHP Foundation describes the receiving side of the same trend. AI-

assisted research has sharply increased its report volume: the project has published 15 CVEs since July 1, 2026, and has handled more than 100 reports that were invalid, duplicates, or not reproducible. It now uses AI tooling to validate and group reports, although a person still makes every decision [2].

Security Analysis

The Disclosure Timeline

The disclosure timeline shows an embargo of roughly ten weeks, counted from the first report, that came apart within its last 24 hours. Separately, exploitation activity appeared before any of the known leaks. The table below combines the reporter, maintainer, and threat-intelligence accounts, with all times given in UTC. Read in sequence, it shows that the decision to move the release forward preceded both the proof-of-concept video and the early distribution of patched binaries.

Date / Time (UTC)	Event	Source
Jul 30 – Aug 11, 2026	Calif audits PHP and reports an unconditional RCE with the out-of-memory root cause	[2]
Aug 19, 2026	Wiz Research independently discovers its first PHP vulnerability with AI-assisted tooling	[1]
Aug 20–21, 2026	Unauthenticated RCE confirmed against Drupal, then WordPress	[1]
Aug 24, 2026	Wiz reports to the PHP security team	[1][2]
Sep 18, 2026	OpenAI reports further attack vectors	[2]
Sep 21, 2026	PHP acknowledges the reports and begins a coordinated fix	[1]
Oct 17, 2026	Amazon MadPot records an exploitation attempt from a commercial anonymization network	[3]
Oct 20, ~15:00	CVE IDs circulate online with a wrong release time and a wrong CVSS of 10.0; no technical details	[2]

Date / Time (UTC)	Event	Source
Oct 20, 19:30	PHP publishes its pre-announcement early and moves the release to Oct 21, 14:00	[2][7]
Oct 20, ~20:00	Calif mistakenly publishes a PoC video; copies spread after deletion	[2]
Oct 20, ~23:00	Mitigation advice shared with some critical infrastructure providers may also have leaked	[2]
Oct 21, ~02:00–08:00	Amazon observes PoC exploitation attempts from multiple anonymized sources	[3]
Oct 21, 03:02	Cloudflare enables default WAF protections for all plans	[6]
Oct 21, 07:41	Chainguard publishes built images containing the fix, ahead of the embargo	[2][4]
Oct 21, 14:00	Patched PHP releases and public disclosure	[2][5]

Exploitation Before Disclosure

The October 17 event changes how this incident should be read. Amazon reports that the attempt came from 103.216.220[.]92, an address tied to a commercial anonymization network. Amazon judges that an anonymized probe against a honeypot during the embargo suggests a threat actor, and states that how the actor got exploit information remains unknown [3]. An Amazon spokesperson described the activity as largely a one-off probe, and Amazon notes that it is unclear whether the attempts it observed would have succeeded [3][4]. The event predates every leak the PHP Foundation has documented, so the October 20 leak, the Calif video, and the Chainguard binaries cannot explain it.

The available evidence fits at least three explanations, and CSA cannot currently distinguish among them. First, an attacker may have found the same bug independently. The PHP Foundation's account makes this plausible: three separate organizations reached the same root cause within roughly five weeks, and at least two used frontier AI models [1][2]. When several defenders converge this quickly, an attacker with similar tools may have converged too. Second, information may have leaked from inside the embargo. That group was fairly large, since it included the reporters, distributors with early patch access, and critical infrastructure providers given mitigation guidance [2]. Third, a participant's systems may have

been compromised. None of these has been confirmed, and treating any of them as established would be speculation. Amazon draws the operational lesson in its own analysis: an embargo protects defenders only until someone else finds the same bug, and the public disclosure date should not be treated as the earliest possible exploitation date [3].

AI Compresses Both Ends of the Window

PHP2Shell shows AI shortening the timeline at both ends. On the discovery side, AI-assisted tooling helped several teams find decade-old memory-safety bugs in a mature, heavily scrutinized runtime at about the same time [1][2]. On the exploitation side, Amazon observes that "AI-assisted analysis makes it cheaper to turn a patch diff or advisory into a working exploit." It argues that most actors no longer need to wait for a few skilled exploit developers to publish working code, and that more actors can therefore act within hours of disclosure [3]. The Chainguard episode illustrates the risk. The PHP Foundation notes that once fixed binaries were public, "any attacker could now study the fix in binary form while every other PHP user was still waiting for the release" [2]. ELLIO reported its first observed exploitation request less than an hour later [4]. However, PoC-based attempts had already been seen since about 02:00 UTC [3], so that request cannot be tied to the binaries specifically, and no source attributes it to binary diffing or to AI.

Post-disclosure activity followed the broader pattern Amazon describes. Amazon reports broad exploitation attempts in the wild, with payloads and infrastructure varied enough to suggest multiple actors [3]. Pantheon reported blocking more than 60,000 suspicious requests by the evening of October 21, VulnCheck identified scanners using the user-agent string `php2shell-scan/109618`, and Cloudflare observed attackers changing payloads to get past its rules [4][6]. JFrog also found fake proof-of-concept repositories on GitHub carrying prank malware [4]. As a result, defenders who download unvetted exploit code to test their own exposure take on a supply-chain risk of their own.

Why Detection Is Difficult

Exploitation happens below the application layer, so application logs may show little. CrowdStrike, as reported by The Hacker News, describes the only server-side trace as a single memory-limit fatal error in the web server error log [4]. Wiz recommends looking for unusual multipart requests followed by memory-limit errors, repeated HTTP 500 responses, unexpectedly large or malformed `PHPSESSID` response cookies, and unexplained churn in PHP workers [1]. Amazon advises searching form-upload POST bodies for the string `PHP_SESSION_UPLOAD_PROGRESS` and checking web servers for unexpected process execution or file changes [3]. The FBI, as reported by The Hacker News, has advised watching for PHP processes that launch command shells or download files [4].

These signals are not consistent across sources. CrowdStrike reports that successful exploitation can return HTTP 200 with stable workers [4], so the absence of 500 errors or worker churn should not be read as an absence of compromise. Because no CVE-specific signatures existed before disclosure, behavioral signals such as these were the only practical option for that period, consistent with Amazon's recommendation to invest in detections that do not depend on CVE signatures [3].

Interim Mitigations and Their Limits

For organizations that cannot patch at once, the PHP team and Amazon have described partial mitigations. Each has costs, and none replaces the runtime update [3][5]. The table below maps the reported mitigations to each vulnerability, along with the caveats that limit them.

Vulnerability	Reported interim mitigation	Caveat
CVE-2026-109618	Set <code>memory_limit=-1</code> via <code>php_admin_value</code> so applications cannot override it; lower <code>post_max_size</code> (8M, or 32M if uploads are needed); enforce a container or cgroup memory limit that kills only the affected worker [5]	Removing the PHP memory limit moves the risk of resource exhaustion to the host or container layer
CVE-2026-109619	Set <code>session.upload_progress.enabled=0</code> and restart all workers; on PHP 8.5, set <code>fatal_error_backtraces=0</code> ; remove invalid per-request settings and fix attacker-reachable fatal errors [5]	The backtrace setting blocks only one confirmed path; hiding errors with <code>display_errors</code> or <code>@</code> does not help [5]
Both	Block multipart file uploads where the business allows it [3]; keep WAF managed rules in block mode [6]	Can break application features; WAF rules are temporary and attackers are already mutating payloads [5] [6]

Recommendations

Immediate Actions

Organizations should first build a complete inventory of PHP runtimes. It should cover web servers, containers and base images, managed hosting, appliances, and third-party products that embed PHP, and it should not rely only on application inventories. Supported branches should be updated to PHP 8.5.12, 8.4.27, 8.3.36, or 8.2.35 right away, starting with internet-facing Apache mod_php deployments, where the PHP security advisory considers the risk highest [5][7]. For PHP 8.1 and older, teams should get backported fixes from their distribution or LTS vendor. The PHP security team has provided minimal source backports for 7.4, 8.0, and 8.1, but the durable answer is migration to a supported branch [5]. The UK NCSC, as reported by BleepingComputer, has advised taking unpatched systems offline if they cannot be updated [5]. Where patching takes longer than hours, the interim mitigations above should be applied and WAF protections confirmed in blocking mode.

Incident response teams should start their review window no later than October 17, 2026, rather than October 21, and should look further back where logs allow [3]. That means searching web and error logs for memory-limit fatals, upload-progress strings, and abnormal session cookies. It also means checking hosts for unexpected child processes, web shells, and modified files, and treating any PHP worker suspected of exposure as having leaked the secrets in its memory. Exposed database credentials, API keys, session-signing keys, and TLS private keys should be rotated. The IP addresses Amazon published belong to shared anonymization infrastructure, so they should drive investigation rather than blanket blocking [3].

Short-Term Mitigations

Over the next 30 days, organizations should change remediation targets for critical, internet-facing, pre-authentication flaws in core runtimes from days to hours, as Amazon recommends [3]. They should also confirm that container image pipelines rebuild and redeploy promptly when a base image is patched, since many PHP workloads inherit the runtime from base images whose update ownership is unclear. Detection engineering should favor behavioral analytics that do not depend on CVE-specific signatures, such as web-server processes spawning shells, worker crash-and-restart patterns, and outbound connections from PHP workers. Finally, security teams that test exposure should use scanners from established vendors rather than unvetted GitHub proof-of-concept code, given the malicious repositories JFrog identified [4].

Strategic Considerations

PHP2Shell suggests that coordinated vulnerability disclosure is under strain from AI-accelerated discovery in ways that existing norms do not address. When independent teams find the same flaw within weeks, an embargo lasting more than two months assumes secrecy that may not hold. The October 17 probe indicates that, in at least this case, someone not authorized to hold the embargoed details had enough exploit information to attempt exploitation before any documented leak. Maintainers and reporters may therefore need shorter embargoes for high-impact bugs that many parties can find. They also need clear rules for what early-access distributors may publish, and they should plan on the assumption that a public patch, in source or binary form, may be turned into exploit attempts within hours.

For enterprises, the more durable lesson concerns architecture. Runtime-level flaws in widely used language platforms can bypass application security entirely. Defense in depth therefore needs to include worker isolation, least-privilege service accounts, egress restrictions on application tiers, and secrets that are short-lived enough to limit what memory disclosure can expose. The PHP Foundation's investment in dedicated security staff and AI-assisted triage [2] is a model other critical open-source projects may need to follow, and organizations that depend on those projects should consider supporting it.

CSA Resource Alignment

PHP2Shell follows the trends set out in CSA's 2026 research on AI-accelerated vulnerability discovery. *AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis* [8] argues that AI-driven discovery and automated patch analysis are collapsing the exploitation window, often to before disclosure, and recommends a 72-hour fast lane for internet-facing, high-likelihood vulnerabilities. PHP2Shell is consistent with both points. Exploitation attempts began within hours of the PoC leak and continued within hours of the patch, and a probe was observed days earlier [3]. For pre-authentication runtime flaws, this single case suggests that even 72 hours may be too slow, a hypothesis that further incidents should test.

Reforming Coordinated Vulnerability Disclosure for the Autonomous Bug Hunter Era [9] argues that CVD norms built around human researchers working at human speed do not fit AI-scale findings. It calls for clear human accountability for AI-generated reports and for triage driven by exploitability. The PHP2Shell timeline adds a failure mode that paper anticipated only in part. Several AI-assisted teams arrived at the same bug, and the embargo then had to protect a secret held by a widening circle of

reporters, distributors, and infrastructure providers. *AI Vulnerability Discovery Is Outpacing Patch Capacity* [10] describes the remediation side of the same imbalance, and the large exposed population running unsupported PHP branches [4] is an example of the capacity gap that note examines.

At the control level, the CSA AI Controls Matrix (AICM) v1.1 [11] offers a framework for operationalizing these lessons. Its Threat and Vulnerability Management (TVM) domain covers remediation timelines, vulnerability prioritization, and detection updates. Hardening and timely updating of the runtimes beneath applications draws on its Infrastructure Security (I&S) and Change Control and Configuration Management (CCC) domains, while Application and Interface Security (AIS) addresses the application layer that these runtime flaws bypass. Organizations using AICM can map the immediate actions in this note to TVM controls and use the incident to test whether their remediation targets for critical internet-facing flaws reflect the hour-scale exploitation now being observed.

References

- [1] Wiz Research (S. Tzadik, S. Tamari, N. Ohfeld, T. Schwartz). "[PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70% of the Internet.](#)" Wiz Blog, October 21, 2026.
- [2] The PHP Foundation. "[Today's PHP Security Release, and How We're Strengthening PHP Security.](#)" The PHP Foundation Blog, October 21, 2026.
- [3] C. Moses, Amazon Threat Intelligence. "[Amazon Threat Intelligence identifies PHP2Shell exploitation prior to public disclosure in MadPot.](#)" AWS Security Blog, October 21, 2026.
- [4] R. Lakshmanan. "[Someone Tried to Exploit PHP2Shell 4 Days Before the Vulnerabilities Were Publicly Disclosed.](#)" The Hacker News, October 22, 2026.
- [5] S. Gatlan. "[PHP ships emergency patches for critical PHP2Shell flaws.](#)" BleepingComputer, October 21, 2026.
- [6] D. Molteni, D. Jones, K. Nandwani. "[Cloudflare deploys protections against php2shell vulnerability.](#)" Cloudflare Blog, October 21, 2026.
- [7] The PHP Group. "[PHP News Archive – 2026.](#)" php.net, October 20, 2026.
- [8] Cloud Security Alliance. "[AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis.](#)" CSA AI Safety Initiative, June 17, 2026.
- [9] Cloud Security Alliance. "[Reforming Coordinated Vulnerability Disclosure for the Autonomous Bug Hunter Era.](#)" CSA AI Safety Initiative, June 7, 2026.
- [10] Cloud Security Alliance. "[AI Vulnerability Discovery Is Outpacing Patch Capacity.](#)" CSA AI Safety Initiative, August 2, 2026.
- [11] Cloud Security Alliance. "[AI Controls Matrix \(AICM\) v1.1.](#)" Cloud Security Alliance, 2026.