



SECURITY ADVISORY

# PHP2Shell

**CVE-2026-109618 & CVE-2026-109619**

## Assessment & Action Plan

Two critical remote code execution flaws in the PHP engine: what they are, who is exposed, and what to do this week.

|             |                                       |         |                   |
|-------------|---------------------------------------|---------|-------------------|
| PREPARED BY | Hung Le, NamiQ                        | DATE    | 22 October 2026   |
| FACTS AS OF | 22 October 2026 (CISA KEV 2026.10.21) | VERSION | 1.1               |
| SEVERITY    | <b>Critical: CVSS 4.0 9.2</b>         | STATUS  | Patches available |

COMPANION ARTICLE [namiq.io/insights/php2shell-cve-2026-109618-109619](https://namiq.io/insights/php2shell-cve-2026-109618-109619)

## Executive summary

### In brief

On **21 October 2026** the PHP project released fixes for two critical flaws in the PHP engine itself. They sit underneath WordPress, Drupal, Laravel, Magento and every other PHP application, so **a clean application on an unpatched PHP is still exposed**.

- **Fixed in PHP 8.5.12, 8.4.27, 8.3.36 and 8.2.35.** Everything from PHP 5.4 up to those releases is affected.
- **PHP 8.1 and older receive no new upstream release.** Fixes come from Linux distributions or LTS vendors, or from upgrading.
- **Severity: 9.2 Critical (CVSS 4.0)**, with no login and no user interaction. The official vector rates attack complexity as **high**.
- **Exploitation attempts are real** (from 17 October, before disclosure). As of 22 October **neither CVE is in CISA's Known Exploited Vulnerabilities catalog**, and no reliable exploit against a default install has been shown publicly.
- **Patching and restarting every PHP worker is the only complete fix.** Configuration workarounds reduce exposure but do not close it.

**Recommendation:** patch internet-facing PHP within 24 hours, `mod_php` first. Where that is not possible, follow CISA and take the system offline, or apply the compensating controls in section 6.3 as a deliberate, time-boxed risk decision. Review logs from **17 October** onwards, not from the disclosure date.

## At a glance

|                   | CVE-2026-109618                                                                                                | CVE-2026-109619                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| PHP advisory      | GHSA-w8hw-w9fx-j79w                                                                                            | GHSA-6wmf-4rv5-p3rp                                                                                        |
| Title             | RCE via OOM corruption                                                                                         | RCE via stale global state                                                                                 |
| Root cause        | Out-of-memory abort ( <code>longjmp()</code> ) leaves request data partly initialised; shutdown then trusts it | Global state outlives the request that owns it; the next request on the same worker reads freed memory     |
| Severity          | 9.2 Critical, CVSS 4.0                                                                                         | 9.2 Critical, CVSS 4.0                                                                                     |
| Authentication    | None required                                                                                                  | None required                                                                                              |
| Fixed in          | 8.2.35 · 8.3.36 · 8.4.27 · 8.5.12                                                                              | 8.2.35 · 8.3.36 · 8.4.27 · 8.5.12                                                                          |
| Config workaround | Partial: <code>memory_limit=-1</code> plus input limits                                                        | Partial:<br><code>session.upload_progress.enabled=0</code> ; on 8.5, <code>fatal_error_backtraces=0</code> |
| CISA KEV          | Not listed (catalog 2026.10.21)                                                                                | Not listed (catalog 2026.10.21)                                                                            |

# 1. What the vulnerabilities are

Both flaws are in **php-src**, the PHP interpreter, not in a framework, extension or application. They were reported by researchers credited in the PHP advisories (including Wiz Research) and fixed by the PHP security team. Public write-ups call the pair **PHP2Shell**.

## 1.1 CVE-2026-109618: RCE via out-of-memory corruption

PHP handles an out-of-memory (OOM) error by calling `longjmp()`, which jumps straight out of the code that was running to abort the request. Code that was half-way through building internal request data never gets to finish. If the allocation fails at the wrong moment, that data is left **partly initialised**, and the request-shutdown procedure that runs next can **read or branch on invalid state**. The PHP advisory states this can lead to remote code execution.

**How an attacker gets there.** The advisory names malformed multipart (file-upload) requests as a way to push PHP into OOM at a chosen point:

- multipart parts with an **undeclared content length**;
- an **unlimited number of headers per part**;
- **empty file controls**.

**What the fix does.** The patch removes `longjmp()` from OOM handling entirely. It replaces it with a delayed "soft" OOM, and with an immediate "hard" OOM that ends the process through `_exit()`, so no cleanup ever runs on half-built state.

## 1.2 CVE-2026-109619: RCE via stale global state

PHP keeps some process-wide state (error, session, object and extension state) that can end up **pointing into memory owned by a request that has already finished**. When the same worker process picks up the next request, some operations run **before normal initialisation resets that state**. They read memory that has been freed: a classic **use-after-free**. With a crafted heap layout, the advisory says, this becomes remote code execution.

**Triggers named in the advisory:**

- fatal errors that skip cleanup, **including out-of-memory errors and execution timeouts**;
- warnings raised during shutdown, or during early INI processing.

**Confirmed paths:** PHP 8.5's fatal-error backtraces (with `fatal_error_backtraces` enabled), and session upload-progress handling. **Precondition:** the request that plants the state and the request that consumes it must reach **the same worker**. This is why the advisory suggests `pm.max_requests=1` as a hardening option.

## 1.3 How they combine

The first bug corrupts state inside one request. The second carries corrupted state from one request into the next. Researchers chained the two into unauthenticated code execution, and Calif reports confirming pre-authentication RCE against specific versions of WordPress, Drupal, Joomla and Magento. Public descriptions of the internals vary. Sansec, for example, describes 109618 in terms of an upload-progress unserializer use-after-free. **This document follows the PHP project's own advisories wherever sources differ.**

## 1.4 Weakness and technique mapping

| ID               | Name                                               | How it applies here                                                                                                 |
|------------------|----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| CWE-416          | Use After Free                                     | Stale global state points at freed request memory and is used by the next request (109619).                         |
| CWE-824          | Access of Uninitialized Pointer                    | An OOM abort leaves pointers in request data that were never written (109618).                                      |
| CWE-755          | Improper Handling of Exceptional Conditions        | The OOM and fatal-error paths are the exceptional conditions that are mishandled.                                   |
| CWE-1325         | Improperly Controlled Sequential Memory Allocation | Many small allocations (multipart parts and headers) are not bounded in total, which makes OOM reachable on demand. |
| CWE-1419         | Incorrect Initialization of Resource               | State is used before the next request's initialisation has reset it.                                                |
| ATT&CK T1190     | Exploit Public-Facing Application                  | The initial-access route: an unauthenticated request to an internet-facing PHP server.                              |
| ATT&CK T1499.003 | Application Exhaustion Flood                       | Resource-heavy requests that drive PHP into OOM. Even a failed exploit can crash workers.                           |

CWE identifiers are those listed in the PHP advisory for CVE-2026-109619. The ATT&CK mapping is NamiQ's own.

## 2. Severity: what the scores mean

The PHP security team scores both CVEs **9.2 Critical under CVSS 4.0**. Other numbers are in circulation, and they are not contradictions so much as different scoring systems and assumptions:

| Source                        | Score                  | Note                                                                                      |
|-------------------------------|------------------------|-------------------------------------------------------------------------------------------|
| PHP security team (GHSA)      | 9.2 Critical, CVSS 4.0 | The authoritative score. Vector below.                                                    |
| Fortinet PSIRT, some trackers | 9.8 Critical, CVSS 3.1 | The older scale has no "attack requirements" metric, which inflates scores like this one. |

| Source                  | Score         | Note                                                       |
|-------------------------|---------------|------------------------------------------------------------|
| Amazon Linux            | 8.1, CVSS 3.1 | Vendor re-assessment for its own packages.                 |
| Pre-release CVE records | 10.0          | The PHP Foundation says this figure was <b>incorrect</b> . |

The vector, decoded: `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`

| Metric                        | Value              | What it means for you                                                                                      |
|-------------------------------|--------------------|------------------------------------------------------------------------------------------------------------|
| Attack vector                 | Network            | Reachable over the internet if the PHP service is.                                                         |
| Attack complexity             | High               | The attacker must defeat conditions such as heap layout. This is <b>not</b> a trivial one-request exploit. |
| Attack requirements           | Present            | Deployment conditions must hold, e.g. the same-worker requirement for 109619.                              |
| Privileges / user interaction | None / None        | No account and no victim click are needed.                                                                 |
| Impact on the PHP system      | High / High / High | Full loss of confidentiality, integrity and availability on the host.                                      |
| Impact on other systems       | None               | Scored for the PHP host only. Lateral movement afterwards is a separate question.                          |

## 3. Real-world exploitability

### 3.1 Timeline

| When (UTC)     | What happened                                                                                                                                                            | Source             |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| 17 Oct 2026    | Amazon Threat Intelligence sees an exploitation attempt against its honeypots, <b>four days before disclosure</b> . How the attacker got exploit information is unclear. | Amazon via Tenable |
| 21 Oct, ~07:43 | Cloudflare's CTO reports "early attacks emerging", hours before the release.                                                                                             | Tenable; Sansec    |
| 21 Oct, 14:00  | PHP releases 8.5.12, 8.4.27, 8.3.36 and 8.2.35 and publishes both advisories. The PHP security team "did not identify any functional exploits at the time of release".   | PHP; Tenable       |
| 21 Oct         | CISA issues "Critical Vulnerabilities in PHP Require Immediate Patching": it <b>has not observed exploitation</b> at publication.                                        | CISA               |

| When (UTC) | What happened                                                                                                                                                                                                                        | Source             |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| 21 Oct     | Amazon sees further attempts whose payloads and infrastructure suggest <b>multiple actors</b> . It is unclear whether they would have succeeded. watchTowr reports attempts against both CVEs on its honeypot network.               | Tenable; watchTowr |
| 21 Oct     | CISA KEV catalog version 2026.10.21: <b>neither CVE listed</b> . NamiQ re-checked the live feed on 22 Oct with the same result.                                                                                                      | CISA KEV feed      |
| 22 Oct     | Sansec revises its view: mass exploitation is " <b>less likely</b> ". The public proof-of-concept code it examined fails against a default PHP install, and most of it needs a separate local-file-inclusion bug in the application. | Sansec             |

## 3.2 Exposure by deployment

| Deployment                                               | Exposure                      | Why                                                                                                             |
|----------------------------------------------------------|-------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Apache + mod_php, internet-facing                        | Highest                       | PHP says a demonstrated exploit fires from an unauthenticated request <b>before any application code runs</b> . |
| PHP-FPM, php-cgi, FrankenPHP, LiteSpeed, internet-facing | High                          | PHP's position: every installation serving web requests is at risk.                                             |
| Same, internal only                                      | Medium                        | Needs a foothold on the network first, but intranet apps are often the least-patched.                           |
| Vendor appliances and panels with embedded PHP           | Unknown until the vendor says | Fortinet (FG-IR-26-186) and F5 (K000163821) have published advisories. Check every vendor.                      |
| PHP CLI only (cron, queue workers)                       | Not addressed by sources      | No public analysis covers the CLI. Patch it in the normal cycle.                                                |

## 3.3 NamiQ assessment

### Our reading, 22 October 2026

Attempts are real, and they began before disclosure, so **some exposure window has already passed** for anyone who was unpatched on 17 October. Reliable mass exploitation has **not** been shown, and the official vector rates complexity as high.

That combination argues for **patching on a plan, starting today with whatever faces the internet**. It does not justify waiting for a confirmed in-the-wild exploit or a KEV listing. The window between a credible proof of concept and broad exploitation of a runtime this widespread is typically short, and a KEV listing tends to follow exploitation rather than precede it.

## 4. Should you patch now?

**Yes.** Treat internet-facing PHP as a P0 change. The decision for each system:

| Your situation                                      | Recommended action                                                                                                                                         | Target                     |
|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| Internet-facing, supported branch (8.2–8.5)         | Upgrade to the fixed release, restart all workers, verify (section 5.2).                                                                                   | Within 24 h; mod_php first |
| Internet-facing, 7.4–8.1 on a distro or LTS package | Install the vendor's backported package, restart, and verify through the changelog, not <code>php -v</code> .                                              | Within 24 h                |
| Internet-facing, 8.1 or older with no vendor fix    | CISA: <b>take it offline</b> until updated. Otherwise isolate it behind strict controls (6.3) and start the upgrade.                                       | Today                      |
| Cannot patch yet (change freeze, testing)           | Make a deliberate, owned risk decision: offline, per CISA, or compensating controls, per Sansec's view for stores. Record who decided and when it expires. | Today; days, not weeks     |
| Internal-only systems                               | Patch in the next change window.                                                                                                                           | Within 72 h                |
| Vendor appliances                                   | Follow each vendor's advisory; do not patch PHP inside an appliance yourself.                                                                              | As vendors publish         |

Targets are NamiQ's recommendation for a typical small or mid-size estate, not a regulatory deadline. A KEV listing would set formal deadlines for US federal agencies.

## 5. Exposure assessment

### 5.1 Inventory: where PHP hides

- **Web servers:** Apache (mod\_php), Nginx or Caddy with PHP-FPM, IIS with FastCGI, LiteSpeed, FrankenPHP.
- **Applications:** WordPress, Drupal, Joomla, Magento, Nextcloud, Laravel and Symfony apps, plus admin tools such as phpMyAdmin.
- **Containers:** images you built yourself. Rebuilding on the same base-image tag may not pull a patched PHP; check the image that is actually running.
- **Managed platforms:** on Google Cloud, App Engine standard (8.2+) is patched automatically, App Engine flexible must be redeployed, and Cloud Run is patched only if automatic base-image updates are enabled. Check your provider's equivalent.
- **Embedded PHP:** hosting control panels, network and security appliances, monitoring tools, and marketplace LAMP or CMS images.
- **Orphans:** old internal tools on 7.x with no owner, which are exactly the systems with no upstream fix.

## 5.2 Verify the version you actually run

This is where most assessments go wrong. **Debian, Ubuntu, Red Hat and others backport security fixes without changing the version number**, so `php -v` can report 8.1.2 on a fully patched Ubuntu server and 8.3.20 on an unpatched one. Use the table for upstream builds and the changelog check for distribution packages.

| Branch    | Fixed upstream release | Upstream status                                                                                                      |
|-----------|------------------------|----------------------------------------------------------------------------------------------------------------------|
| 8.5       | 8.5.12                 | Supported                                                                                                            |
| 8.4       | 8.4.27                 | Supported                                                                                                            |
| 8.3       | 8.3.36                 | Supported                                                                                                            |
| 8.2       | 8.2.35                 | Security fixes only; end of life 31 Dec 2026                                                                         |
| 7.4 – 8.1 | None                   | End of life. The PHP security team published minimal source backports; fixes come from distributions and LTS vendors |
| 5.4 – 7.3 | None                   | End of life. No fix from the PHP project                                                                             |

```
# 1. Version string (useful, but not proof on distribution packages)
php -v

# 2. Debian / Ubuntu: is the fix in the installed package?
zgrep -l CVE-2026-109618 /usr/share/doc/php*-common/changelog.Debian.gz

# 3. RHEL / Alma / Rocky / Fedora
rpm -q --changelog php-common | grep -E 'CVE-2026-10961(8|9)'

# 4. Containers: the running image, not the Dockerfile
docker exec <container> php -v

# 5. What the WEB server runs (FPM can differ from the CLI): serve a
#   phpinfo() page bound to localhost, read it, then delete it.
curl -s http://127.0.0.1/__ver.php | grep -E 'PHP Version|Server API'
```

## 5.3 Attack-surface questions for each system

| Question                                                          | Why it matters                                                                                                  |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Does it accept <b>multipart/form-data</b> (file uploads)?         | Every known trigger uses multipart. If the app never needs uploads, blocking multipart removes the known paths. |
| Is <b>session.upload_progress.enabled</b> on? (It is by default.) | This is a confirmed path for CVE-2026-109619.                                                                   |
| PHP 8.5 with <b>fatal_error_backtraces</b> on?                    | This is the other confirmed path for CVE-2026-109619.                                                           |
| Can a request cause a fatal error or hit the execution timeout?   | Both are listed triggers that skip cleanup.                                                                     |

| Question                                                     | Why it matters                                                |
|--------------------------------------------------------------|---------------------------------------------------------------|
| Are workers long-lived (high <code>pm.max_requests</code> )? | Stale state needs the next request on the same worker.        |
| Is the origin reachable directly, bypassing the WAF or CDN?  | Edge rules protect only the traffic that passes through them. |

## 6. Action plan

### 6.1 By urgency

#### Immediate (0–4 hours)

- List every **internet-facing** PHP service, with its SAPI and its version as verified in 5.2.
- Patch them, **mod\_php first**, and **restart every PHP worker** (PHP-FPM, Apache, long-running workers). A patched binary on disk does nothing for a process that loaded the old one.
- For any you cannot patch, make the offline-or-compensate decision from section 4 explicitly, and give it a named owner.
- Switch on edge protection where you have it: Cloudflare's managed rules (available on the free plan; Cloudflare calls them a stopgap) or Google Cloud Armor's recommended rules.
- On any system you suspect, **preserve logs and, if possible, a disk image before patching**. Rebuilding destroys evidence.

#### Short term (4–24 hours)

- Patch internal PHP: intranet apps, CI/CD, monitoring and admin tools.
- Rebuild container images from a patched base, redeploy, and **verify the running image**.
- Redeploy managed-platform services that are not patched automatically.
- Collect vendor advisories for every appliance or product with embedded PHP.
- Keep `session.upload_progress.enabled=0` wherever the feature is unused; PHP recommends this even after patching.

#### Medium term (24–72 hours)

- Hunt for compromise from **17 October** onwards (6.4).
- Scan to confirm. Tenable Nessus plugin **369483**, Web App Scanning plugins **115719–115722**, and equivalent checks in other scanners. A clean scan is one input, not proof.
- Schedule upgrades off 8.1 and older; 8.2 itself reaches end of life on **31 December 2026**.
- Record both CVEs in your vulnerability register, with an owner and evidence of closure for every system.

#### Ongoing

- Watch the CISA KEV catalog and the PHP security advisories. If either CVE is listed as exploited, re-run the hunt.

- Add a pipeline check that fails builds whose PHP is below the fixed baseline.
- Set a patch SLA for critical, unauthenticated RCE in shared runtimes. Under 24 hours for internet-facing systems is a reasonable target.

## 6.2 Patch procedure (per host)

1. Take a backup or snapshot; on suspect hosts, preserve evidence first.
2. Install the fixed release or the vendor's backported package.
3. Restart PHP-FPM, the web server and any PHP workers; confirm that no old worker processes remain.
4. Verify through the web SAPI (5.2, step 5) and the package changelog.
5. Remove temporary workarounds that cost functionality (e.g. a multipart block), but keep upload progress off if unused.
6. Record the verification output as evidence of closure.

## 6.3 Compensating controls if you cannot patch today

Both PHP advisories are explicit: these **reduce exposure but do not make an installation safe**. Restart all workers after any change, and treat them as days of cover, not weeks.

| Control                                                                                                                 | Addresses                      | Cost and caveats                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Block <code>multipart/form-data</code> at the proxy or web server</b>                                                | All known triggers (both CVEs) | File uploads break. Test checkout, cart and admin uploads. Remove after patching.                                                               |
| <code>session.upload_progress.enabled = 0</code>                                                                        | 109619 session path            | Low cost. Disabling it <b>alone is not enough</b> (Sansec).                                                                                     |
| <code>fatal_error_backtraces = 0</code> (PHP 8.5)                                                                       | 109619 backtrace path          | Low cost. Leaves other stale-state paths open.                                                                                                  |
| <code>memory_limit = -1</code> , enforced so apps cannot override it                                                    | 109618 OOM trigger             | <b>Removes PHP's guard against runaway scripts.</b> One request can exhaust the server, so monitor memory and restore the limit after patching. |
| <b>Conservative <code>post_max_size</code>; cap decoded body size at the front end</b>                                  | OOM reachability               | Can reject legitimate large uploads.                                                                                                            |
| <b>Lower <code>max_input_nesting_level</code>; limit input variables, multipart parts, headers per part and uploads</b> | OOM reachability               | Needs testing against real forms.                                                                                                               |
| <code>pm.max_requests = 1</code> (PHP-FPM)                                                                              | 109619 same-worker requirement | Costs throughput; measure before rolling out.                                                                                                   |
| <b>Container or cgroup <code>memory.max</code></b>                                                                      | Availability boundary          | Not a complete mitigation (PHP).                                                                                                                |
| <b>Do not use <code>RLIMIT_AS</code> / <code>RLIMIT_DATA</code></b>                                                     | —                              | The PHP advisory warns these <b>trigger the vulnerable path</b> .                                                                               |

| Control                                     | Addresses             | Cost and caveats                                                                            |
|---------------------------------------------|-----------------------|---------------------------------------------------------------------------------------------|
| WAF managed rules (Cloudflare, Cloud Armor) | Known exploit vectors | A stopgap that covers only traffic routed through the WAF. Lock the origin down to the WAF. |
| Take the system offline                     | Everything            | CISA's advice if you cannot patch or run 8.1 or older. Business cost is yours to weigh.     |

## 6.4 Detection and hunting (look back to 17 October)

| Where              | What to look for                                                                                                                                                                                                                                                                                  |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Web and proxy logs | Multipart requests with unusual structure: huge or missing content lengths, very many parts, many headers per part, empty file fields.<br>Bursts of such requests from one source, especially against upload endpoints.                                                                           |
| PHP and FPM logs   | Bursts of "Allowed memory size exhausted" fatal errors.<br>Worker crashes, segmentation faults, "child exited on signal" messages, and maximum-execution-time errors on endpoints that normally never hit them.                                                                                   |
| Host / EDR         | PHP worker processes (php-fpm, apache2, httpd) <b>starting shells or downloading tools</b> , which CISA calls out specifically.<br>Outbound connections from web workers; new or modified files in web-writable directories.                                                                      |
| Security tooling   | Make sure vendor content is enabled: Google Security Operations (Mandiant Frontline Threats rule pack, Emerging Threat Center GLOBAL.26.015), Wiz Threat Intel Center, Tenable plugins.<br>Do not run unverified public exploit code to test yourself; some circulating PoCs may contain malware. |

**If you find signs of compromise:** isolate the host, preserve evidence, rotate every credential the PHP application could read (database, API keys, cloud tokens in environment variables), and rebuild from a known-good image rather than cleaning in place.

## 6.5 Definition of done

- Every PHP instance in the inventory has a named owner and a verified fixed version (or vendor backport) recorded with evidence.
- Every PHP worker has been restarted since patching.
- No system older than 8.2 remains internet-facing without a dated exception and an upgrade plan.
- The 17-October-onwards hunt is complete for every internet-facing system, and its findings are recorded.

- Temporary workarounds that cost functionality have been removed; intentional hardening (upload progress off) is documented.

## 7. What we do not know yet

- **Whether CISA will add either CVE to the KEV catalog.** That would set formal deadlines, and it usually follows confirmed exploitation.
- **Whether a reliable exploit against a default configuration appears.** Sansec's 22 October view could change in either direction.
- **The full list of affected appliances and hosted products.** Vendor advisories are still arriving.
- **Conflicting exploitation claims.** HKCERT's bulletin describes the CVEs as exploited in the wild; CISA had not observed exploitation at publication. Treat "exploited" as unconfirmed until a primary source confirms it.
- **Items we could not verify.** Our threat feed references a Cisco Talos post on Snort coverage and a Cisco advisory on PHP in Cisco products. Neither could be confirmed from a public source at the time of writing, so neither is relied on here. Ask your Cisco account team directly.

## 8. References

- [1] PHP — GHSA-w8hw-w9fx-j79w, CVE-2026-109618: RCE via OOM corruption  
<https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>
- [2] PHP — GHSA-6wmf-4rv5-p3rp, CVE-2026-109619: RCE via stale global state  
<https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>
- [3] CISA — Critical Vulnerabilities in PHP Require Immediate Patching (21 Oct 2026)  
<https://www.cisa.gov/news-events/alerts/2026/10/21/critical-vulnerabilities-php-require-immediate-patching>
- [4] CISA — Known Exploited Vulnerabilities catalog  
<https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
- [5] Google Cloud — Responding to core PHP vulnerabilities CVE-2026-109618 and CVE-2026-109619  
<https://cloud.google.com/blog/products/identity-security/responding-to-core-php-vulnerabilities-cve-2026-109618-cve-2026-109619>
- [6] Tenable — php2shell: frequently asked questions  
<https://www.tenable.com/blog/php2shell-cve-2026-109618-cve-2026-109619-frequently-asked-questions-about-critical-php-remote>
- [7] Sansec — PHP2Shell research (updated 22 Oct 2026)  
<https://sansec.io/research/php2shell>
- [8] watchTowr — PHP remote code execution FAQ: CVE-2026-109618  
<https://watchtowr.com/intelligence/php-cve-2026-109618-faq/>
- [9] Fortinet PSIRT — FG-IR-26-186  
<https://fortiguard.fortinet.com/psirt/FG-IR-26-186>
- [10] F5 — K000163821: PHP vulnerabilities CVE-2026-109618 and CVE-2026-109619  
<https://my.f5.com/manage/s/article/K000163821>
- [11] HKCERT — PHP Multiple Vulnerabilities (22 Oct 2026)  
[https://www.hkcert.org/security-bulletin/php-multiple-vulnerabilities\\_20261022](https://www.hkcert.org/security-bulletin/php-multiple-vulnerabilities_20261022)

[12] Palo Alto Networks Unit 42 — Threat Brief: CVE-2026-109618, CVE-2026-109619

<https://unit42.paloaltonetworks.com/cve-2026-109618-cve-2026-109619/>

[13] NamiQ — PHP2Shell: what to do this week (companion article)

<https://namiq.io/insights/php2shell-cve-2026-109618-109619>

### About this document

Prepared by **Hung Le**, NamiQ. NamiQ provides managed IT and security for growing businesses. This assessment reflects public information as of **22 October 2026**, and the situation is changing quickly; check the references before acting on anything time-sensitive.

© 2026 NamiQ. This document is the property of NamiQ and may be shared unaltered, with attribution. It is not legal advice and does not guarantee the security of any system.

[namiq.io](https://namiq.io) · [namiq.io/contact](https://namiq.io/contact)