MAG212 CYBER

CYBER.MAG212.COM

INDEPENDENT CYBERSECURITY INTELLIGENCE



SPECIAL REPORT · CVE-2026-109618 / 109619

PHP2Shell. Patch the runtime, not just the site.

A technical briefing for IT administrators, hosting customers and security teams: the flaws, the client impact, firewall rules and the fix.

CRITICAL · CVSS 9.2

INSIDE THIS REPORT

- What the two bugs are
- Who is exposed
- Impact on clients
- What you need to do
- Firewall rules
- Detection & response

BY EL MOSTAFA OUCHEN

SPECIAL REPORT · WEB SERVERS · RUNTIME SECURITY

PHP2Shell: when a crashed request becomes a shell

Two critical flaws in the PHP engine expose almost every PHP website, whatever runs on top. What they are, how they reach your clients, and how to shut them out.

BY EL MOSTAFA OUCHEN · PUBLISHED BY MAG212 CYBER · 22 OCTOBER 2026

Two critical flaws in the PHP runtime itself, CVE-2026-109618 and CVE-2026-109619 (together called "PHP2Shell"), let an unauthenticated attacker run code as the web server user, or read live worker memory, on almost any PHP installation that serves web requests. The application on top does not matter. PHP shipped fixed releases **8.5.12, 8.4.27, 8.3.36 and 8.2.35** on 21 October 2026 at 14:00 UTC, a day earlier than planned after an embargo leak, and the project says **updating is the only complete fix** (GHSA-w8hw-w9fx-j79w). Exploitation attempts are confirmed by several independent sensor networks, some from before disclosure. As of 22 October no source had confirmed a successful compromise of a production server. Successful exploitation can return an ordinary HTTP 200 and write nothing to the application log, so "no confirmed compromises" is not the same as "no compromises". Businesses should treat this as an emergency runtime patch. Hosting and MSP customers should confirm who owns their PHP runtime today. Anyone who served traffic on vulnerable PHP since 17 October should plan to rotate secrets even if they find no webshell.

KEY FACTS

CVE
CVE-2026-109618 · CVE-2026-109619

SEVERITY
CRITICAL
CVSS v4 9.2 (PHP) · NVD v3.1 9.8

AFFECTED
All web-serving PHP: mod_php, PHP-FPM, php-cgi, FrankenPHP, LiteSpeed. Back to PHP 5.4

FIXED IN
8.5.12 · 8.4.27 · 8.3.36 · 8.2.35

STATUS (22 OCT)
Exploit attempts confirmed; no confirmed compromises; not in CISA KEV

ACT NOW

1 Update PHP to 8.5.12, 8.4.27, 8.3.36 or 8.2.35 and restart every PHP process.

2 Can't patch yet? Apply both sets of interim settings and put a WAF in front.

3 Hunt back to 17 Oct and rotate secrets on exposed hosts.

SECTION 01

Two runtime bugs share one root: requests that end abnormally

Both CVEs are memory-safety flaws in the PHP engine, not in WordPress, Drupal, Laravel or any other application. BleepingComputer puts it plainly: "Both stem from PHP leaving memory corrupted when a request ends abnormally" (BleepingComputer). That abnormal end is usually an out-of-memory (OOM) fatal error, or another fatal error, during request processing. Calif, one of the discovering teams, describes the pair as a chain

of "memory-exhaustion and interrupted-state bugs", with the earliest affected code path dating to **PHP 5.4.0** (Calif).

CVE-2026-109618, "RCE via OOM corruption" (GHSA-w8hw-w9fx-j79w). PHP handles memory exhaustion by jumping out of the code that is running. If that happens while internal request data is only partly initialized, request cleanup can then treat the half-built data as valid. In php.net's words, "request cleanup may then treat the invalid data as valid" (php.net). NVD adds that "multiple OOM strategies can consistently trigger this condition and lead to remote code execution" (NVD). Red Hat places the flaw in memory-limit handling during **request parsing** (Red Hat). PHP's interim mitigations all target request-body (POST and multipart) parsing under memory pressure, which supports that reading (php.net archive).

CVE-2026-109619, "RCE via stale global state" (GHSA-6wmf-4rv5-p3rp). Here, request-owned internal state survives after its memory has been freed. "A later request handled by the same worker may use that stale state before normal initialization resets it" (php.net). The advisory classifies it as an unauthenticated heap use-after-free that can disclose memory or give RCE as the web worker user. It requires a **persistent worker that serves multiple requests**, with the attacker's requests landing on the same worker. Confirmed paths include **session upload progress (enabled by default)**, **PHP 8.5 fatal-error backtraces**, Apache mod_php directory settings and PHP-FPM (GHSA-6wmf-4rv5-p3rp).

The two advisories cross-reference each other and tell users to remediate both. They share fixed versions, reporters and fix developers (GHSA-w8hw; GHSA-6wmf). The root causes are separate, so mitigating one leaves the other open. Beyond code execution, Wiz stresses an **unauthenticated remote memory-disclosure** capability that can expose "passwords, authentication tokens, API keys and certificates" from other users' requests, even when the application never calls `session_start()` (Wiz).

	CVE-2026-109618	CVE-2026-109619
Advisory	GHSA-w8hw-w9fx-j79w	GHSA-6wmf-4rv5-p3rp
PHP title	RCE via OOM corruption	RCE via stale global state
Class	Partly initialized request state trusted during cleanup after OOM	Cross-request use-after-free of stale global state
Key precondition	Memory exhaustion during request handling. Validated with default settings on mod_php	A persistent worker reused across requests
Confirmed paths (per PHP)	Request-body parsing under memory limit	Session upload progress, PHP 8.5 fatal-error backtraces, mod_php directory settings, FPM
CWE (PHP CNA)	CWE-416, 755, 824, 1325 (GHSA also lists 1419)	Same set

Sources: GHSA-w8hw, GHSA-6wmf, NVD.

A note on conflicting CVE mapping. Wiz's write-up assigns session upload-progress handling and memory disclosure to CVE-2026-109618, and fatal-error backtraces plus an OPcache type-confusion issue to CVE-2026-109619 (Wiz). PHP's official advisory places session upload progress and PHP 8.5 backtraces under **CVE-2026-109619**, and php.net's interim mitigations for that CVE are exactly `session.upload_progress.enabled=0` and `fatal_error_backtraces=0` (GHSA-6wmf; php.net archive). The Hacker News follows the same mapping (The Hacker News). This article follows PHP's mapping. The OPcache item appears only in Wiz's material, and its link to either CVE is unconfirmed. In practice the mapping changes nothing: both CVEs are fixed by the same releases, and both sets of interim settings should be applied.

SECTION 02

Scores run from 8.1 to 9.8 depending on the scorer

PHP, as the CVE Numbering Authority, scores both CVEs at **CVSS v4.0 9.2 (Critical)**, 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` (NVD 109618; NVD 109619). The "CVSS 10.0" that circulated with the leak never came from PHP or any official scorer (The Hacker News; Check Point).

Scorer	CVE-2026-109618	CVE-2026-109619
PHP CNA (v4.0)	9.2 Critical (AC:H, AT:P)	9.2 Critical (AC:H, AT:P)
NVD (v3.1)	**9.8 Critical** (AC:L)	Not yet scored (record status "Received")
Red Hat (v3.1)	**8.1 Important** (AC:H)	Not retrieved
Ubuntu priority	Not retrieved	Critical
CISA SSVC (21 Oct)	Exploitation: none; Automatable: no; Technical impact: total	Same
CISA KEV	Not listed at time of reporting	Not listed at time of reporting

Sources: NVD, Red Hat, Ubuntu, NVD 109619.

The scores differ mainly on attack complexity. PHP and Red Hat treat exploitation as dependent on configuration and harder to carry out. Red Hat notes that standard RHEL deployments use PHP-FPM, whose workers exit on early initialization failures, and that reliable exploitation needs precise heap manipulation and timing (Red Hat). NVD scores it as low-complexity network RCE. The CISA SSVC entry of "exploitation: none" was recorded at publication on 21 October, before watchTowr and Cloudflare reported attack traffic, and should be read as out of date. Our recommendation: do not let the lower Red Hat score set your priority. Working exploits exist for the most common configuration, and active probing has been confirmed.

SECTION 03

Every web-facing PHP is affected, and mod_php is the softest target

PHP's position is unambiguous: "All PHP installations serving web requests" are at risk, including **PHP-FPM, Apache mod_php, php-cgi, FrankenPHP and LiteSpeed** (php.net archive). The application is irrelevant. Wiz demonstrated RCE against the official `wordpress:latest`, `drupal:latest` and `joomla:latest` Docker images (Wiz), and Calif confirmed pre-authentication RCE against WordPress, Drupal, Joomla and Magento (Tenable). Risk does vary by server API (SAPI):

Deployment	CVE-2026-109618	CVE-2026-109619	Priority
Apache mod_php	**Full RCE validated with default settings.** A single unauthenticated request triggers it before application code runs	Confirmed path via mod_php directory settings (`.htaccess` `php_value` / `php_flag` apply before request startup)	Highest
PHP-FPM	Child exits on startup failure, which limits that case. Post-startup OOM may still trigger it. No complete FPM exploit shown	**RCE validated on a dynamic pool** via the session lifecycle. Also exposed when nginx or `mod_proxy_fcgi` passes `PHP_VALUE` / `PHP_ADMIN_VALUE` that cause early diagnostics	High

Deployment	CVE-2026-109618	CVE-2026-109619	Priority
FrankenPHP	Likely susceptible (unvalidated)	No early consumer in default config. Not immune	High
LiteSpeed / LSAPI	Likely, application-specific	Conditional candidate	High
php-cgi	Listed as affected by PHP	Listed as affected	Patch
CLI scripts	Not mentioned by any advisory	Not mentioned	Lower (inference: no HTTP body parsing, no request reuse)

Sources: GHSA-w8hw, GHSA-6wmf, php.net. The CLI row is MAG212 CYBER ' inference, not a vendor statement.

Versions: four fixed branches and a long unfixed tail

Affected releases reach back to PHP 5.4 (Wiz; Tenable). Upstream fixed only the supported branches. PHP 8.1 and older receive no project releases, but PHP published **minimal source backports for 7.4, 8.0 and 8.1**, "intended for source builds or users without an LTS vendor" (php.net). PHP 8.6.0 RC4 also carries the fixes, but it is a test release (php.net).

Branch	Fixed upstream release	Notes
8.5	**8.5.12**	Also disable `fatal_error_backtraces` if you cannot patch immediately
8.4	**8.4.27**	Debian trixie: `8.4.27-1~deb13u1` (DSA-6574-1)
8.3	**8.3.36**	Ubuntu 24.04 php8.3: "Needs evaluation" at time of research
8.2	**8.2.35**	Debian bookworm: `8.2.35-1~deb12u1` (DLA-4848-1). **Branch EOL 31 Dec 2026**
8.1 / 8.0 / 7.4	None (source backports only)	Depends on distro/LTS vendor (e.g. cPanel ea-php backports)
7.3 and older	None	No upstream fix. Commercial extended support or migration only

Sources: php.net, Debian tracker, Ubuntu, Tenable.

Distribution status was uneven on 22 October. Debian had fixed stable and oldstable, while Debian forky and sid still carried vulnerable php8.4 8.4.26-1 (Debian tracker). Ubuntu listed every PHP package from 14.04 to 26.10 as "Needs evaluation", with no USN (Ubuntu). Red Hat's CVE page showed no errata yet and pointed to a mitigation bulletin (Red Hat). Third-party repositories lagged as well. Remi's repository had no release-day update, and PHP advised its users to move to distribution packages or build from source (Help Net Security). Ondřej Surý's PPA had no fix on release day either (martincid.com). Laravel Forge told customers that its automatic updates do not patch PHP, and that Ubuntu unattended-upgrades does not pull from the third-party sources it installs PHP from (martincid.com). Admins on Ubuntu, RHEL or these repositories may therefore face a period with no patch available, and the interim settings below become essential during that time.

SECTION 04

From leak to internet-wide scanning in hours: attempts confirmed, compromises not

Calif and Wiz found the bugs independently in August 2026 and reported them to PHP between 11 and 24 August. OpenAI's Codex Security team is also credited (Calif; Wiz; The Hacker News). This is not entirely a new problem. A PHP core developer saw early signs of the OOM flaw during a 2024 external audit, but fix work

stalled. The PHP Foundation called the delay "a structural failure" (The Hacker News). Disclosure was planned for 22 October at 16:00 UTC (Threat Landscape).

Time (UTC)	Event
17 Oct	Amazon's MadPot honeypot records an exploitation attempt from an anonymization-network IP, four days before disclosure (Help Net Security)
20 Oct	A "Tell HN" post leaks two PHP RCEs and claims CVSS 10.0. Around 20:00, Calif mistakenly posts a video of its working PoC (Threat Landscape; The Hacker News)
20 Oct ~19:33	PHP brings the release forward to 21 Oct 14:00 UTC (Check Point)
21 Oct 03:02	Cloudflare enables WAF rules by default for all plans (The Hacker News)
21 Oct 07:41	Chainguard publishes patched images about six hours before the official release. The PHP Foundation criticizes the move (The Hacker News)
21 Oct 08:36	ELLIO's sensors log their first PHP2Shell request (ELLIO)
21 Oct 14:00	PHP releases 8.5.12, 8.4.27, 8.3.36, 8.2.35 (php.net)
21 Oct 14:19	watchTowr reports "active exploitation attempts" against both CVEs (The Hacker News)
21 Oct 20:40	A multi-vulnerability bot adds PHP2Shell; one address hits 123 ELLIO sensors in 18 minutes (ELLIO)

What is confirmed is a large volume of exploitation *attempts*. Cloudflare saw attack traffic against customers before the patch and watched attackers "mutating the attack payload" to evade its rules (Cloudflare). Pantheon blocked more than **60,000** probing requests. Amazon assesses that multiple threat actors are involved (The Hacker News). By 21:00 UTC on 21 October, ELLIO had counted 25 source addresses grouped into a handful of tool families (ELLIO). **What is not confirmed** is any successful break-in. No source had confirmed one, no threat actor had been named, and no ransomware or botnet had been attributed as of 22 October (The Hacker News). Accounts of working exploits also differ. The PHP Foundation said it found no functional exploits at release, while watchTowr says it reproduced both flaws (Tenable). Given how little trace exploitation leaves, the absence of confirmed compromises probably reflects a detection gap as much as attacker failure.

Exposure: between 1.3 million and 5.1 million hosts, depending on who counts

Exposure estimates vary by a factor of four because each source counts differently, using banner detection versus version detection and "PHP host" versus "vulnerable host". **Silas Cutler counts over 1.3 million potentially vulnerable hosts**, and a Shodan search by Henk Poley finds over 1.9 million internet-exposed PHP hosts (Tenable). **VulnCheck counts about 2.6 million** internet-facing PHP hosts. Over 700,000 of them run PHP 3.x–7.3, which will never get an upstream fix, about 400,000 run 7.4–8.1, and about 403,000 run supported 8.2–8.5 (The Hacker News). At the top end, **VirusTotal counts over 5.1 million PHP hosts**, about 3.8 million with at least one open port (BleepingComputer). Patch uptake started slowly. A secondary report citing mySites.guru says only **0.08%** of monitored live sites reported a fixed PHP version by 13:52 UTC on release day, and that 22.7% of monitored Joomla and WordPress sites still ran PHP 8.1 or older (martincid.com).

How it reaches clients: outages, shared hosting and appliances

The business impact confirmed so far is **precautionary downtime, not breaches**. BambooHR took its service offline in the US, Canada and Ireland, interrupting payroll. In the Netherlands, DigiD, MijnOverheid, the lower house of parliament's site and several education systems went offline (The Hacker News). The UK NCSC told organizations to update immediately or take affected systems offline (BleepingComputer). CERT-EU advised isolating internet-facing PHP applications until a fix is applied and verified (Tenable).

Shared-hosting and managed-WordPress customers face a particular problem: they cannot patch the runtime themselves. PHP's own guidance confirms that `ini_set()` and `.user.ini` cannot apply the upload-progress mitigation (php.net), so tenant-level self-help is limited to a WAF in front of the site and a support ticket. Updating WordPress or Laravel does not update PHP (The Hacker News). Vendors responded quickly. cPanel shipped EasyApache 4 fixes on 21 October: ea-php82–85 at upstream versions, backports for ea-php81, ea-php80 and ea-php74, and no fix for cPanel-built ea-php73 and older. CloudLinux alt-php fixes reached the regular repositories at 00:07 UTC on 22 October (computingforgeeks). Rollout still happens server by server, and each server may run several PHP branches that all need patching. The same source reports a pitfall: backported packages keep old version strings, so a patched CloudLinux ea-php84 can still report 8.4.26 (computingforgeeks). Shared memory pools also raise the stakes. Because memory disclosure can expose "data belonging to other requests" (Wiz), we advise MSPs to treat co-tenants on a shared worker pool as affected together. No source has yet confirmed cross-tenant exploitation.

Appliances that embed PHP form a further long tail. As of 22 October, Palo Alto Networks listed all PAN-OS and Panorama versions as affected with no fixed release. Fortinet was checking 12 products and Cisco 31, with none confirmed vulnerable (Help Net Security). The PAN-OS exploit path has not been described publicly. MSPs should nonetheless make sure firewall management interfaces are not reachable from the internet.

Action plan

What you need to do

Who has to act depends on who runs PHP on your server. Find the card that fits you, then use the firewall steps below for extra cover while the update rolls out.

MANAGED BY MAG212 CYBER

Our team handles it

If MAG212 CYBER manages your servers or hosting, we are applying the patched PHP builds and the firewall rules described here. There is nothing to install yourself. Contact us if you want written confirmation of the status of your systems.

SHARED OR MANAGED HOSTING ELSEWHERE

Check with your provider

On shared hosting you cannot update PHP or apply the main workaround yourself, because `.user.ini` and `ini_set()` do not work for it. Check your host's status page or open a ticket and ask whether the fixed build is installed. Some hosts have already posted updates: Hostinger, for example, says its managed plans other than VPS were patched automatically (status page). cPanel and CloudLinux shipped patched packages on 21–22 October.

YOU RUN PHP YOURSELF (VPS, DEDICATED, CLOUD)

Act now

Server-level protections from a hosting company do not reach a server you manage yourself.

1. **Update to PHP 8.5.12, 8.4.27, 8.3.36 or 8.2.35 as soon as you can.** It is the only complete fix. Restart Apache, PHP-FPM and every long-running PHP worker afterwards.
2. **Can't update straight away?** Set `session.upload_progress.enabled = Off` in `php.ini` or your PHP-FPM pool config, then restart all PHP workers. Setting it in code with `ini_set()` will not work. This blocks only one of the two bugs, so treat it as a stopgap and apply the memory settings in the "Interim settings" section too.
3. **Turn off uploads you don't need and keep request size limits low** (`post_max_size`, web server body limits). This lowers risk but is not a substitute for updating.
4. **Update your firewall** using the steps below, and keep it in front of the server until the update is verified.

For everyone. PHP 8.1 and older no longer get releases from the PHP project, only source backports for 7.4, 8.0 and 8.1, and PHP 7.3 and older get nothing. Even where a vendor or host has back-patched an old version, plan an upgrade: older branches miss many other security fixes, and PHP 8.2 stops receiving security fixes on 31 December 2026. Move to 8.3 or newer after checking that your theme, plugins and framework support it.

How to update your firewall

Firewall rules buy time; they do not fix the bug. The steps below are STANDARD PRACTICE configurations recommended by MAG212 CYBER, built on the indicators and limits published by PHP, Cloudflare, ELLIO and Abstract Security. Menu names and syntax vary by product version, so test in monitoring mode before you switch to blocking.

1. Cloudflare or another cloud WAF

- Cloudflare enabled PHP2Shell managed rules for all plans, including Free, at 03:02 UTC on 21 October. On Pro, Business and Enterprise plans, open *Security → WAF → Managed rules* and confirm the Cloudflare Managed Ruleset is enabled and the PHP2Shell rules are set to **Block**, not Log.
- **Lock your origin to the WAF.** Rules only protect traffic that passes through the proxy. Allow inbound HTTP/HTTPS on the origin only from Cloudflare's published IP ranges (cloudflare.com/ips), so attackers cannot hit the server's IP directly.
- On AWS WAF, Azure Front Door, Akamai, Imperva and similar services, check the vendor's changelog for a PHP2Shell rule, enable it, and add the size and indicator rules below as custom rules.

2. ModSecurity with the OWASP Core Rule Set

We found no CRS rule written specifically for PHP2Shell as of 22 October, but you can block the known scanner signatures and cap the request shapes the attacks rely on. Rule IDs and limits below are illustrative.

```
# [STANDARD PRACTICE] /etc/modsecurity/php2shell.conf
# Known PHP2Shell scanner user agents (ELLIO)
SecRule REQUEST_HEADERS:User-Agent "@rx (?i)php2shell-" \
  "id:1026101,phase:1,deny,status:403,log,msg:'PHP2Shell scanner UA'"
# Multipart boundaries used by public tooling (----p2s / ----p2shell)
SecRule REQUEST_HEADERS:Content-Type "@rx boundary=-*p2s" \
  "id:1026102,phase:1,deny,status:403,log,msg:'PHP2Shell multipart boundary'"
# Hundreds of empty file fields: cap the number of uploaded parts
SecRule &FILES "@gt 20" \
  "id:1026103,phase:2,deny,status:413,log,msg:'Too many multipart file parts'"
# Request body caps (bytes)
SecRequestBodyLimit 8388608
SecRequestBodyNoFilesLimit 131072

# crs-setup.conf: tighten CRS argument limits (rules 920360-920390)
SecAction "id:900300,phase:1,nolog,pass,t:none,\
  setvar:tx.max_num_args=255,setvar:tx.arg_name_length=100,\
  setvar:tx.combined_file_sizes=8388608"
```

Reload the web server afterwards and watch the audit log for false positives, especially on sites that take large uploads.

3. Imunify360 (cPanel and CloudLinux servers)

- Make sure ruleset **8.85 or later** is installed. Version 8.84 was the first to block on LiteSpeed, and ImunifyAV does not receive these rules.
- As of 22 October the rules covered only the CVE-2026-109619 path, so the PHP update is still required.

4. The web server as a first filter

Apply the request limits from the "Interim settings" section (`client_max_body_size` in nginx; `LimitRequestBody`, `LimitRequestFields` in Apache), and reject `multipart/form-data` on paths that never accept uploads.

5. Host firewall: block known sources and outbound downloads

Blocking the IPs seen so far is a short-term measure, because attackers rotate addresses. Blocking outbound connections from the web server user is more durable: it stops the loader seen in the wild from downloading its second stage (`http://92.118.112.100/nfs`).

```
# [STANDARD PRACTICE] Block sources reported on 17-21 Oct (short-term)
for ip in 103.216.220.92 31.223.184.166 37.19.200.5 134.209.159.64 \
          103.114.163.188 95.217.98.37 85.11.167.243 45.198.224.169 \
          109.71.252.97 92.118.112.100; do
  iptables -I INPUT -s "$ip" -j DROP
  iptables -I OUTPUT -d "$ip" -j DROP
done

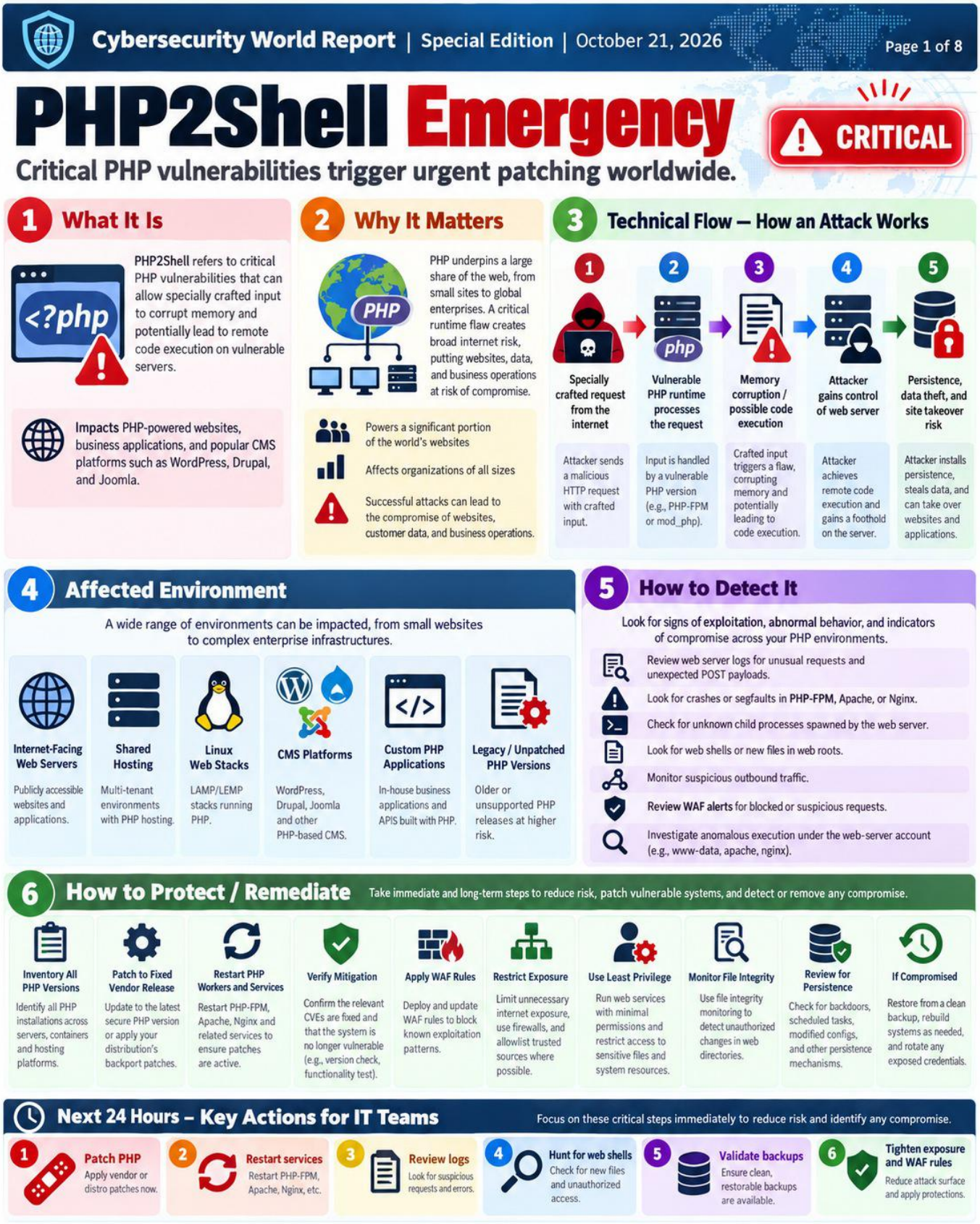
# [STANDARD PRACTICE] Stop the PHP worker user opening new outbound connections.
# First allow what the site needs (database, SMTP, payment/API hosts), then:
iptables -A OUTPUT -m owner --uid-owner www-data -d 127.0.0.1 -j ACCEPT
iptables -A OUTPUT -m owner --uid-owner www-data -m conntrack --ctstate NEW -j REJECT
# (Use 'apache' or 'nginx' instead of www-data on RHEL-family systems; persist with your distro's tooling.)
```

6. Network firewalls and appliances

- Install the latest IPS/threat signature updates on your perimeter firewall, and check your vendor's advisories for a PHP2Shell signature before relying on one.
- Some appliances run PHP themselves. Palo Alto Networks lists all PAN-OS and Panorama versions as affected with no fix yet, and Fortinet and Cisco are still assessing products. Make sure firewall management interfaces are reachable only from trusted internal addresses, never from the internet.

Why you still need the PHP update

Firewall rules are a safety net. They stop the attacks we already know about, but attackers were already changing their requests to slip past Cloudflare's rules before the patch shipped. The PHP update fixes the bug itself. Treat the two as a pair: the update for the fix, the firewall for extra cover.



Visual briefing: PHP2Shell at a glance. Cybersecurity World Report, Special Edition 01.

SECTION 05

Patching is the only fix, and verifying it is harder than it looks

"Updating to a PHP release containing the patch to this vulnerability is the *only* complete mitigation," says GHSA-w8hw. GHSA-6wmf adds that "no complete configuration-based mitigation exists" (GHSA-w8hw; GHSA-6wmf). Three rules follow from the sources. First, verify the fix by **CVE ID in the package changelog, not by version number**, because vendors ship backports under old-looking version strings. Second, **restart every PHP process**: Apache, PHP-FPM and any long-running workers (The Hacker News). Third, check the **web SAPI, not just the CLI**. `php -v` reports the CLI binary, which can differ from the FPM or mod_php build serving traffic.

The fix also changes runtime behavior. It removes the jump-out on OOM and replaces it with a delayed "soft OOM" at the next safe point and a hard OOM via `_exit()` at a higher threshold (GHSA-w8hw). A process that reaches the hard limit terminates without running shutdown handlers, so review `memory_limit` and `post_max_size` against application needs after updating (The Hacker News).

The commands below are STANDARD PRACTICE. They are general administration commands that MAG212 CYBER recommends, not commands published by PHP or any vendor for these CVEs. Adjust package names to your branch.

```
# [STANDARD PRACTICE] Debian / Ubuntu
apt update && apt install --only-upgrade 'php8.*'
dpkg -l 'php*' | grep ^ii
apt changelog php8.4 | grep -E 'CVE-2026-10961[89]'      # must list both CVEs
systemctl restart php8.4-fpm apache2

# [STANDARD PRACTICE] RHEL / Alma / Rocky / Fedora
dnf upgrade 'php*'
dnf updateinfo list --cve CVE-2026-109618
rpm -q --changelog php-common | grep -E 'CVE-2026-10961[89]'
systemctl restart php-fpm httpd

# [STANDARD PRACTICE] cPanel EasyApache 4
dnf update 'ea-php*'        # or yum; or via WHM
/scripts/restartsrv_apache_php_fpm && /scripts/restartsrv_httpd

# [STANDARD PRACTICE] Check the binary actually serving traffic
php-fpm8.4 -v               # FPM binary (CLI 'php -v' may differ)
# For mod_php: serve a temporary phpinfo() page via the web server, read it, delete it.
```

On cPanel, forum reports say `/scripts/upcp --security` updated cPanel's internal PHP but not the ea-php packages, so update the ea-php packages explicitly (computingforgeeks). For **containers**, rebuild from a base image that contains the fix and redeploy rather than patching running containers. We have not verified which official Docker Hub `php` tags carry the fixes. Confirm that a tag's PHP version or changelog includes the fix before relying on it, and check inside the running container (for example `docker exec <container> php -v`, plus the FPM binary). Chainguard published patched images on 21 October (The Hacker News). For **Windows/IIS and Plesk**, our sources contained no vendor-specific PHP2Shell build information. As standard practice, replace the PHP build with the matching patched release (same thread-safety mode and architecture) and recycle the IIS application pools or PHP-FPM handlers.

SECTION 06

Interim settings buy time, not safety

Where patching must wait, for example on Ubuntu or RHEL before errata, behind a slow hosting provider, or on legacy branches, PHP publishes interim settings for each CVE. **The advisory says none of them make an installation safe** (php.net archive). Apply both sets: memory settings alone leave CVE-2026-109619 open, and the reverse is also true.

For CVE-2026-109618 (OOM) (php.net; GHSA-w8hw): set `memory_limit=-1` administratively (`php_admin_value`) so applications cannot override it, and confirm that no pool, directory or application setting reintroduces a finite limit. Set `post_max_size=8M` (or `32M` if uploads are needed) with `upload_max_filesize` below it. Lower `max_input_nesting_level` and bound input variables, multipart parts and uploads. Enforce request-size, field-count, multipart-part, header-count and nesting limits at the web server or proxy, including for requests without a usable Content-Length. Block multipart entirely where it is unused. Contain the resulting memory risk with a cgroup or container limit such as `memory.max`, ideally one that kills only the affected worker. **Never use `RLIMIT_AS` or `RLIMIT_DATA`**, because those limits trigger the vulnerable allocation-failure path. Even with `memory_limit=-1`, a failed `mmap()` can still reach the vulnerable path.

For CVE-2026-109619 (stale state) (php.net; GHSA-6wmf): set `session.upload_progress.enabled=0` in php.ini, the FPM pool or server config, never through `ini_set()` or `.user.ini`, then **restart every worker**. On PHP 8.5, set `fatal_error_backtraces=0`. On FPM, if performance allows, set `pm.max_requests=1` to limit worker reuse. Remove INI settings that emit warnings before request initialization, such as deprecated `auto_detect_line_endings`, non-default `filter.default`, legacy `assert.*` and session settings, or an invalid `memory_limit` or `date.timezone`. Audit `.htaccess` `php_value` / `php_flag` lines on mod_php and `PHP_VALUE` / `PHP_ADMIN_VALUE` passed by nginx. Hiding errors with `display_errors`, `error_reporting` or `@` **does not help**. Fix fatal errors and timeouts that attackers can reach. PHP recommends keeping upload progress disabled even after patching if it is unused, and InMotion Hosting notes that WordPress, Joomla, Drupal, Magento and PrestaShop do not use it (martincid.com).

The configuration below expresses those sourced settings in STANDARD PRACTICE syntax. The specific `max_input_*` values and web-server limits areMAG212 CYBER' illustrative choices, not vendor numbers. Test upload-heavy applications before rollout.

```
; [STANDARD PRACTICE syntax for PHP-sourced settings] FPM pool, e.g. /etc/php/8.x/fpm/pool.d/www.conf
php_admin_value[memory_limit] = -1
php_admin_value[post_max_size] = 8M
php_admin_value[upload_max_filesize] = 4M
php_admin_flag[session.upload_progress.enabled] = off
php_admin_value[max_input_nesting_level] = 16   ; illustrative (PHP default 64)
php_admin_value[max_input_vars] = 1000          ; illustrative
php_admin_flag[fatal_error_backtraces] = off    ; PHP 8.5 only
pm.max_requests = 1                             ; only if performance allows
; Pair with a cgroup cap, e.g. systemd MemoryMax= on the FPM unit. Never RLIMIT_AS/RLIMIT_DATA.
; Then: systemctl restart php8.x-fpm   (restart, not reload)
```

```
# [STANDARD PRACTICE] Apache mod_php — vhost/server config, NOT .htaccess
php_admin_value memory_limit -1
php_admin_value post_max_size 8M
php_admin_flag  session.upload_progress.enabled off
LimitRequestBody 8388608
LimitRequestFields 50
LimitRequestFieldSize 8190
```

```
# [STANDARD PRACTICE] nginx front end
client_max_body_size 8m;
large_client_header_buffers 4 8k;
# Audit: grep -R 'PHP_\(ADMIN_\)\?VALUE' /etc/nginx
# On locations that never accept uploads:
# if ($content_type ~* "multipart/form-data") { return 403; }
```

```
# [STANDARD PRACTICE] find per-directory PHP overrides that act as early consumers on mod_php
grep -R --include=.htaccess -nE 'php_(admin_)?(value|flag)' /var/www
```

Edge WAFs are a useful stopgap, not a fix. Cloudflare deployed managed rules to all plans, including Free, and Pro, Business and Enterprise customers should confirm the rules are in block mode (Cloudflare). Attackers are already mutating payloads, and the rules protect only traffic that passes through the proxy. The origin must reject direct-to-IP requests (martincid.com). On cPanel/CloudLinux servers, Imunify360 rulesets 8.83–8.85 add virtual patches. Version 8.84 is the first to block on LiteSpeed, ImunifyAV gets nothing, and at 22 October the rules covered only the CVE-2026-109619 path (computingforgeeks; Imunify360). Finally, mod_php hosts that cannot be patched should be moved to PHP-FPM or taken offline, as the NCSC advises.

SECTION 07

Hunting a bug that can return HTTP 200

CrowdStrike notes that successful exploitation can produce an HTTP 200 with no application log entry, leaving only a memory-limit fatal in the web server error log (The Hacker News). The exploit runs commands as the web user (www-data, apache or nginx) without crashing the worker (Abstract Security). PHP warns that "a clean log does not prove safety" (GHSA-6wmf). Detection therefore rests on three layers: version inventory (the most reliable), request and log anomalies, and post-exploitation behavior on the host.

Behavioral indicators from researchers and the vendor include unusual multipart requests followed by memory-limit errors or repeated HTTP 500s, oversized or malformed `PHPSESSID` response cookies, and unexplained PHP worker churn (Wiz; The Hacker News). PHP adds repeated warnings or core errors before target scripts run, diagnostics during cleanup, "Maximum execution time exceeded", and failures confined to attacker-reachable routes (GHSA-6wmf). Abstract Security adds "in Unknown on line 0" fatals, HTTP 200 responses to large POSTs whose bodies contain "Allowed memory size", Apache `AH00052` child-exit entries, field names of 1,000 bytes or more, and an address-space brute-force phase of about **590 oversized requests from one IP over roughly 15 minutes** (Abstract Security).

Observed indicators. These are from sensor networks. IP addresses churn and should be used for retro-hunting, not as a blocklist that proves safety.

Type	Indicator	Source
User-Agent	`php2shell-scan/109618`, `php2shell-109618`, `php2shell-verify/1.0`	ELLIO
Multipart boundary	Begins `----p2s` or `----p2shell`	ELLIO
Form field	`PHP_SESSION_UPLOAD_PROGRESS` paired with `PHPSESSID` (a signal, not a verdict: legitimate uploads use it too)	ELLIO; Abstract Security
Body shape	Hundreds of empty file fields; deeply nested field names; PHP open tag or serialized-object syntax in a multipart filename	ELLIO; Abstract Security

Type	Indicator	Source
Request body SHA-256	Scanner `43898a0a53bbd25d31ad9f5018ed3182be98a66109b22c9cab545a9fc447992a`; OOM tool `7bc41ed79a9ae909927a9e40701736460a8cb70fca16ce1510e58677aca764b6`	ELLIO
Payload	`uname -a; id; echo JACKALPOC` (validation)	ELLIO
Loader	Fetches `http://92.118.112.100/nfs` to `/tmp`, runs it with argument `php`. Unclassified static x86-64 ELF, SHA-256 `0c5486f06620e26e316f4b33f318b0d646d5e1133789331aa2fed012360d1486`	ELLIO
Source IPs (21 Oct)	31.223.184.166, 37.19.200.5, 134.209.159.64, 103.114.163.188, 95.217.98.37, 85.11.167.243, 45.198.224.169, 109.71.252.97	The Hacker News
Pre-disclosure IP (17 Oct)	103.216.220.92	The Hacker News

The hunts below are STANDARD PRACTICE implementations of the sourced behaviors. They are not vendor-published rules. Run them back to **at least 17 October 2026**, the earliest documented attempt.

```
# [STANDARD PRACTICE] Memory/timeout fatals and "Unknown on line 0"
grep -E 'Allowed memory size of [0-9]+ bytes exhausted|Out of memory|Maximum execution time|in Unknown on line 0' \
  /var/log/php*-fpm.log /var/log/apache2/error.log /var/log/httpd/error_log

# [STANDARD PRACTICE] Worker crashes / churn
grep -E 'exited on signal (11|6)|SIGSEGV|core dumped|AH00052' /var/log/php*-fpm.log /var/log/apache2/error.log
journalctl -u 'php*-fpm' | grep -i signal
dmesg | grep 'segfault at'

# [STANDARD PRACTICE] POST→500 bursts per source IP (combined log format)
awk '$6 ~ /POST/ && $9 == 500 {print $1}' access.log | sort | uniq -c | sort -rn | head

# [STANDARD PRACTICE] Known scanner user agents
grep -E 'php2shell-(scan|verify|109618)' /var/log/nginx/access.log /var/log/apache2/access.log
```

To catch oversized session cookies, log response cookies (Apache `%{Set-Cookie}o`, nginx `$sent_http_set_cookie`) and flag `PHPSESSID` values far longer than the default 26–32 characters. On the host side, the most durable detection is a PHP worker (`php-fpm*`, `apache2`, `httpd`, `lsphp`, `frankenphp`, `php-cgi.exe`, `w3wp.exe`) spawning `sh`, `bash`, `curl`, `wget`, `nc`, `base64`, `python` or `perl`, or making unexpected outbound connections. That matches the FBI's advice to watch for PHP processes that launch shells or download files (The Hacker News). A STANDARD PRACTICE Defender for Endpoint example:

```
DeviceProcessEvents
| where InitiatingProcessFileName in~ ("php-fpm","php-fpm8.4","httpd","apache2","php-cgi.exe","w3wp.exe")
| where FileName in~ ("sh","bash","dash","curl","wget","nc","base64","cmd.exe","powershell.exe")
```

For fleet exposure testing, Searchlight Cyber has released an open-source **php2shell-scanner** (Help Net Security). Download scanners only from the vendor's official repository: a fake "React2Shell" scanner on GitHub in 2025 turned out to be malware (Hackread). Remote banner checks fail when `expose_php=Off` and are misled by backported version strings, so authenticated package-changelog checks are the most reliable signal. As of 22 October we found no vendor-published Sigma, YARA, Suricata, Nessus, Qualys or Nuclei content specific to PHP2Shell.

SECTION 08

Assume secrets leaked: verification and incident response

No vendor or CERT has published a PHP2Shell-specific incident response playbook. Two sourced facts shape our guidance. Successful RCE runs as the PHP web worker (php2shell.com). Memory disclosure, which leaves no process artifacts, can expose credentials, tokens, keys and certificates from other requests (The Hacker News). MAG212 CYBER therefore recommends that any internet-facing PHP host that served traffic on a vulnerable build after 17 October **rotate high-value secrets even if no webshell is found**. On shared worker pools, treat all co-hosted sites as in scope.

The following sequence is STANDARD PRACTICE webshell and RCE response adapted to these facts.

Contain and preserve first. Put the host behind a WAF or take it offline. Snapshot the VM or container before changing anything. Capture memory if possible (for example with LiME or AVML) before restarting workers, since a restart destroys the evidence. Copy web, PHP-FPM, PHP error, proxy/WAF/CDN and auth logs, the journal and the session directory (`session.save_path`), and record `ps auxf`, `ss -tupan` and `lsof -u www-data`. **Then patch and restart** every PHP SAPI.

Hunt for webshells and persistence.

```
# [STANDARD PRACTICE]
find /var/www -type f \( -name '*.php*' -o -name '*.phtml' -o -name '.htaccess' -o -name '.user.ini' \) -newermt
'2026-10-15' -ls
grep -RlE '(eval|assert|system|passthru|shell_exec|proc_open|popen|pcntl_exec)\s*\(|base64_decode\s*\
(|gzinflate\s*\(' /var/www
ls -la /tmp /dev/shm /var/tmp
grep -RnE 'auto_(prepend|append)_file' /var/www /etc/php* 2>/dev/null
crontab -u www-data -l; ls -la /etc/cron* /var/spool/cron
cat /etc/ld.so.preload 2>/dev/null
debsums -c 2>/dev/null || rpm -Va
wp core verify-checksums && wp plugin verify-checksums --all   # WordPress
```

Also check systemd units and timers, `authorized_keys` files, new user accounts, unexpected `.so` files in `extension_dir`, and changes to php.ini or the pool config. Scan with webshell YARA rule sets, PHP Malware Finder, ClamAV or maldet. Rebuild containers from clean, patched images rather than cleaning them in place.

Rotate credentials. Rotate database passwords in `wp-config.php`, `.env` or `config.php`; API, payment-gateway and cloud access keys, checking cloud audit logs for use of instance-role tokens; SMTP credentials; TLS private keys readable by the worker; application secrets such as `APP_KEY` and WordPress salts; and admin passwords. Flush the session store so all user sessions are invalidated.

Look for lateral movement and meet your obligations. Review database logs for dumps or new admin users, cloud audit trails, and SSH sessions originating from the web host. No PHP2Shell breach has yet triggered a reported regulatory notification. However, e-commerce platforms such as Magento and WooCommerce are confirmed affected, and memory disclosure on a checkout server could expose cardholder data or gateway keys. Organizations processing payment, personal or health data should assess PCI DSS, GDPR (72-hour clock from awareness) and sector notification duties early. The near-absence of logs makes "no evidence of breach" hard to establish. This last point is MAG212 CYBER's assessment, not a regulator's statement.

SECTION 09

Conclusion

PHP2Shell shows that keeping applications up to date is not enough to keep a server secure. The vulnerable code sits below every CMS and framework, and a patched WordPress on an unpatched runtime is still fully exposed. The organizations hit hardest will not be those that patch slowly but those that cannot answer, in minutes, which PHP version and SAPI each host runs and who is responsible for updating it. That includes shared-hosting tenants, Forge and PPA users, container fleets built on stale base images, and appliances such as PAN-OS. The 700,000-plus hosts on PHP 7.3 or older that VulnCheck counts will never receive an upstream fix and will form a permanent, scannable tail of exposure. PHP 8.2 joins the unsupported branches on 31 December 2026.

The incident also changes what "remediated" should mean for runtime memory-disclosure bugs. Patching closes the hole but does not cancel credentials that may already have been read from worker memory. Detection is also asymmetric: attackers can confirm success with an HTTP 200, while defenders may see one memory-limit line in an error log. For MSPs and hosting providers, the durable lessons are structural: per-tenant FPM pools instead of mod_php, egress filtering for web workers, cgroup memory limits in place of rlimits, and a runtime inventory that includes base images and embedded PHP. These measures shrink the blast radius of the next PHP runtime flaw before its advisory is published.

Cybersecurity World Report is published by MAG212 CYBER . Information is current to 22 October 2026. Distribution packages, WAF rules, KEV status and exploitation reporting were changing rapidly at the time of writing, so verify vendor advisories before acting.

MAG212 CYBER

El Mostafa Ouchen for Cybersecurity World Report. Published by MAG212 CYBER , an IT and cybersecurity services company. cyber.mag212.com