LESSONS LEARNED | VULNERABILITY RESPONSE

PHP2Shell: When the Patch Clock Ran Out Before It Started

Two pre-authentication remote code execution flaws in the PHP engine leaked before their fix, drew attacks before disclosure, and showed how few organizations know where PHP runs.

October 2026

Target audience: Executive and Senior Management

CVE-2026-109618 · CVE-2026-109619
Fixed in PHP 8.5.12 / 8.4.27 / 8.3.36 / 8.2.35

9.2
CVSS v4.0 score for each flaw; no login needed to run code
PHP / UNIT 42

22.6M
Internet-exposed instances potentially vulnerable
PALO ALTO CORTEX XPANSE

−4 days
First exploit attempt seen on honeypots before disclosure
AWS, 17 OCT

45 min
From disclosure to tens of thousands of attempts; millions by next day
CLOUDFLARE

5.4+
Every PHP release from 5.4 up to 8.5.11 affected
GOOGLE CLOUD / PHP

TEN DAYS IN OCTOBER

17 Oct — Exploit attempts hit AWS honeypots, ahead of any public advisory

20 Oct — Technical details leak online before coordinated disclosure

21 Oct — Emergency PHP releases ship; CISA urges immediate patching

22 Oct — Attack attempts climb into the millions across edge networks

23 Oct — Follow-on advisory CVE-2026-115560 for unusual session setups

What broke: two memory-safety bugs in PHP's own error handling. One leaves half-built request data behind after an out-of-memory event; the other lets a worker reuse stale state from an earlier request.

Why it matters: the flaw sits below WordPress, Drupal, plugins and custom code. A fully updated application is still exposed if the runtime underneath is not.

EIGHT LESSONS FOR BOARDS AND SENIOR MANAGEMENT

01 You cannot patch what you cannot see

PHP hides inside appliances, hosting panels, vendor products and bundled binaries. Most teams found their real exposure only after the advisory.

Keep a live runtime and component inventory (SBOM), not just an app list.

02 The disclosure window is gone

Attacks started before the advisory and scaled within the hour. A 30-day patch cycle for internet-facing criticals is now a breach window.

Set a pre-approved emergency change path measured in hours.

03 Patching the app is not patching the stack

Application owners updated WordPress and assumed safety. Ownership of the interpreter, container base image and OS package was unclear.

Name an owner for every layer: code, framework, runtime, host.

04 "Not exploitable" has an expiry date

The weakness was flagged in a 2024 audit but parked as unproven. AI-assisted research turned it into working code execution.

Revisit accepted risks and deferred findings at least yearly.

05 End-of-life software is debt with interest

PHP had to issue exceptional source fixes for unsupported 7.4, 8.0 and 8.1. Many estates still run them with no vendor package coming.

Fund retirement of EOL runtimes as a risk item, not an IT wish list.

06 Compensating controls buy time, not safety

WAF rules, edge filtering and config workarounds narrowed the attack paths but did not close them. Only the fixed build does.

Pair virtual patching with a hard deadline for the real fix.

07 Patch, then hunt

With attempts predating disclosure, a clean patch status says nothing about what happened before it. Watch for shells spawned by PHP workers.

Run a compromise assessment on every exposed PHP host.

08 Third parties carry your risk

For many firms the fix sat with a hosting or SaaS provider. Contract terms rarely set emergency patch timelines or evidence.

Request written patch attestation from critical suppliers.

FIVE QUESTIONS FOR THE BOARD

1. Do we know every system that runs PHP, including vendor and appliance builds?
2. How many hours did it take us to patch internet-facing critical systems?
3. Did we confirm no compromise before the patch, or only that the patch landed?
4. Which suppliers have confirmed they are fixed, in writing?
5. What end-of-life software are we still running, and what is the exit date?

72-HOUR RESPONSE CHECKLIST

- ☐ **Inventory:** scan for PHP across servers, containers, images and appliances
- ☐ **Patch:** move to 8.5.12 / 8.4.27 / 8.3.36 / 8.2.35 or the vendor's fixed package
- ☐ **Shield:** WAF rules and vendor-advised config changes where patching lags
- ☐ **Hunt:** review php-fpm and php-cgi child processes and web shells
- ☐ **Confirm:** supplier attestations and a report to the risk committee

CONTROL AND REGULATORY MAPPING

ISO/IEC 27001:2022	A.8.8 technical vulnerabilities; A.8.9 configuration; A.5.21 ICT supply chain
NIST CSF 2.0	ID.AM-02 software inventory; ID.RA-01 vulnerabilities; PR.PS-02 software maintenance
PCI DSS v4.0	6.3.2 component inventory; 6.3.3 critical patches; 6.4.2 automated web protection
NCA ECC / SAMA CSF	ECC 2-10 vulnerability management; SAMA CSF 3.3.17
CBB Rulebook	OM Module cyber security risk management and incident reporting



Kim | CISO

Kamran Israr Mirza · CISM, CCSP, CDPO, CBCMP, ISC² CC

Cybersecurity and GRC leader with 28+ years of experience, 20+ of them in Central Bank of Bahrain regulated financial institutions. Former Interim CISO at American Express Middle East, leading vulnerability management, incident response and PCI DSS compliance across 11 countries. Advises boards and FinTechs across the GCC and Pakistan on cyber, AI governance and digital asset security.

in linkedin.com/in/kimabdalian

REFERENCES

1. PHP Group, security release notes for 8.5.12, 8.4.27, 8.3.36, 8.2.35 (21 Oct 2026)
2. Google Cloud, Responding to core PHP vulnerabilities (Oct 2026)
3. Unit 42, Threat Brief: Critical Vulnerabilities in PHP (23 Oct 2026)
4. Datadog Security Labs, PHP2Shell detection and remediation (Oct 2026)
5. Huntress; watchTowr; SentinelOne; SecurityWeek; The Hacker News; SC World (Oct 2026)

Disclaimer: This poster reflects the author's independent professional view, drawn from public advisories available as of 25 October 2026. Figures belong to the cited sources and may change as investigations continue. It is not legal, regulatory or vendor-specific advice, and does not represent any employer or client. Validate actions against your own environment, vendor guidance and regulator expectations.