

The Legacy Hype

How an old PHP problem became world news, and what the coverage left out

J. Konstapel, Leiden, 22-10-2026.

© 2026 J. Konstapel, Constable Research, Leiden. All rights reserved. Quotation with attribution is permitted; reproduction of the whole requires written permission.

Version 1.0 · facts as of 22 October 2026, 22:00 CEST

What was found

In October 2026 two flaws in PHP took government websites offline in a dozen countries. The coverage presented it as a new threat to “70% of the internet”. It was not new. The same class of flaw has hit PHP before. Those earlier cases stayed in the specialist press.

What changed was not PHP. Four things coincided. The flaw leaked before its fix. Its discovery was marketed as an AI achievement. Governments visibly switched off their own services. And one large number gave the story a scale.

The real problem sat underneath and was hardly reported. Almost half of all PHP sites run versions that no longer receive security fixes. On 1 January 2027 that share rises to two thirds. That is legacy. The flaw was the occasion. The legacy was the story.

The occasion

On Tuesday 20 October 2026, at 17:20 CEST, an anonymous post appeared on Hacker News. It announced two critical remote-code-execution flaws in PHP, “CVSS 10.0”. Both the score and the date in the post were wrong.

PHP’s own security team spoke four hours later, at 21:32. It gave the real score, 9.2, and moved the fix to Wednesday 16:00 CEST. That evening a researcher’s exploit video leaked by accident. On Wednesday morning a software supplier published patched images before the embargo ended.

From about 10:00 on Wednesday, Dutch organisations began to switch off websites. DigiD’s information site, MijnOverheid and the Tweede Kamer went dark. Finland, France, Germany, Belgium, Denmark and Czechia followed. The fixes appeared at 15:43. By Thursday the United Kingdom had 32 services offline and Japanese banks had limited theirs.

Anatomy of the hype

Where the frames came from

“70% of the internet.” W3Techs measures that PHP runs on 69.3% of websites whose server language is known. That is a usage figure. Wiz, the security firm owned by Google, wrote “With 70% of the Internet affected”. Use became exposure. Wiz also runs the campaign site php2shell.com, which ends with “Contact Wiz”.

“AI found it.” This frame came from the finders. Calif found the flaw with OpenAI’s GPT-5.6 Sol, in a programme paid for by OpenAI. Wiz found it with Gemini 4 Argon. PHP’s own advisories make no AI claim.

“Take it offline.” Every official source made this conditional. CERT-EU, NCSC-UK, the BSI, the FBI and Japan’s prime minister all said the same. Patch first. Isolate only what cannot be patched at once. In the coverage the condition disappeared.

“The internet is melting down.” This was an editorial headline, in TheBlaze on Wednesday evening. No researcher said it.

Who carried the story

Almost every amplifier had something to show. Wiz had its site and its products. OpenAI had proof for its programme. Cloudflare had its filtering rules and its attack counts. Amazon had a sighting of an attempt on 17 October. Palo Alto had a count of 22.6 million “potentially vulnerable” instances.

None of this is improper. Security firms publish what they see. But in this wave they supplied the frames, the numbers and the drama. The newsrooms passed them on.

What actually happened

Millions of attack attempts were recorded worldwide. As of Thursday evening, no breach through these flaws had been confirmed with a named victim. The Dutch NCSC said on Thursday at 16:00 that it was investigating “first signals of successful abuse”. These were not confirmed.

Several analysts found the flaw hard to exploit reliably. The BSI called it “more complicated than expected”. Rapid7, watchTowr and VulnCheck said public exploits worked only in narrow setups. The PHP Foundation had seen no working exploit.

The disruption was real. It came mainly from precaution. Citizens could not reach MijnOverheid. Employees could not report sick or recovered to the benefits agency online. Council documents of about a hundred municipalities were unavailable. Nobody had been hacked, and still part of daily life stood still for an afternoon.

What did not change

PHP has had this kind of flaw before.

Flaw	Year	What happened	Coverage
CVE-2012-1823 (php-cgi)	2012	Also disclosed too early by accident	Specialist press; no government action
CVE-2019-11043 (PHP-FPM)	2019	Remote code execution under nginx	Specialist press
CVE-2024-4577 (php-cgi on Windows)	2024	Actively exploited by ransomware; ~458,800 exposed instances (Censys, likely an overcount)	Specialist press; listed by CISA as exploited

None of these had an AI angle, mass takedowns or a 70% frame. The 2024 flaw was worse in one respect: it was actually used by criminals. It made no front pages.

The real story: legacy PHP

How large it is

W3Techs gives the version shares of all PHP sites on 22 October 2026. Within PHP 8, versions 8.0 and 8.1 are no longer supported. Versions 4, 5 and 7 are all unsupported. The calculation below was done three times, independently, in exact fractions.

Group	Share of PHP sites
Official fixed release (8.2–8.5)	51.6%
Source-code backport only (7.4, 8.0, 8.1)	33.5%
No official fix (7.3 and older)	15.0%
Outside support today	48.4%
Outside support from 1 January 2027	66.5%

Measured on servers, the picture is similar. VulnCheck counted 2.6 million PHP hosts, about 42% of them end-of-life. After 24 hours, 99.3% of PHP instances that show their version were still unpatched (OX Security). In the Netherlands, NOS counted about 145,000 sites that may still be exposed.

Where it hides

PHP is not only on websites. The Dutch NCSC keeps a list of products that contain it. On 22 October it held 724 products from 456 suppliers. 216 were marked vulnerable. These include network devices, storage boxes and management interfaces. Their owners cannot update the PHP inside. They depend on the supplier.

Why it persists

The official fix covers only the four supported versions. For 7.4, 8.0 and 8.1 the PHP team published source-code branches, not ready-made releases. Whoever runs those versions must compile the fix or rely on a distribution or hosting provider. For 7.3 and older there is nothing upstream.

Many organisations do not know what runs where. The Dutch central government said most of its sites do not use PHP. It still took sites offline as a precaution. ENISA finds that few organisations keep a complete inventory of their software. Without one, switching everything off is the only safe move.

The project itself is thin. The PHP Foundation received \$730,534 in donations in 2025 and spent \$784,376. It funds 11 developers. Warnings from a 2024 security audit were not acted on in time. The Foundation’s own words: “the delay was a structural failure, caused by both the security team’s decision process and its lack of resources.”

A language that runs on most of the web is maintained on the budget of a small company.

Why the media do not explain legacy

Alarm outweighs explanation

Buse and Meissner analysed 581 articles on cyber incidents in six quality news outlets. 94.1% framed the story around responsibility. 72% used conflict. 76.2% were negative in tone. Only 9.7% mentioned specific security measures.

Maschmeyer, Deibert and Lindsay studied 629 commercial threat reports. Vendors choose topics that serve their market. Newsrooms that rely on vendor reports inherit those choices.

Maintenance has no news value

Vinsel and Russell, in *The Innovation Delusion*, describe how societies reward the new and ignore upkeep. A flaw with a name, a logo and a website is news. Ten years of postponed upgrades is not. There is no single culprit to quote, and the cost does not fall on those who postponed.

Who did explain

Some outlets did. BleepingComputer, Help Net Security, LWN and The Record mentioned end-of-life versions and backports, usually in a sentence or two. The Record cited Detectify: about half of PHP sites run older versions. In the Netherlands, Security.nl explained end-of-life, backports and paid extended support in detail on Wednesday afternoon. NOS mentioned old versions at small firms and hobby sites but did not explain support periods. Computable and AG Connect did not explain it in the articles checked.

The pattern is familiar. The first day carries the alarm. Explanation comes later, in specialist outlets, long reads and audit reports.

What can be done

Now, for anyone running PHP

Route	What it does	Limit
Upgrade to 8.3, 8.4 or 8.5	The only lasting route	Applications must be tested; cost sits there
Official source backports (7.4, 8.0, 8.1)	Closes these flaws	You compile it yourself
Distribution packages (Ubuntu Pro, Debian, Sury, CloudLinux)	Fixes in old version numbers	Only for users of that distribution
Paid extended support (Freexian €1,200–18,000 a year; Zend; TuxCare)	Security fixes past end-of-life	Postpones the upgrade; does not replace it
Web application firewall (e.g. Cloudflare, all plans)	Blocks known attack patterns	Only traffic routed through it; Cloudflare itself says to patch
Isolate or switch off	Removes exposure	An emergency brake, not a way of working

Structurally

Fund the maintenance. The PHP Foundation asks companies, hosting providers and governments for steady funding. Germany’s Sovereign Tech Agency put €205,000 into PHP in 2023–2024, including the audit whose warnings were later missed. The amounts are small next to the share of the web that depends on PHP.

Know what you run. A complete software inventory (an SBOM) is what allows an organisation to switch off only what is exposed. The NCSC’s product list shows how much PHP hides in bought equipment.

Use the law that now applies. The EU Cyber Resilience Act requires manufacturers to report actively exploited flaws since 11 September 2026. From 11 December 2027 it requires security support for at least five years. That makes the supplier, not the customer, answerable for the PHP inside a product.

Keep the condition in the advice. “Patch, or isolate what you cannot patch” is a precise instruction. “Take it offline” is not. The difference decides whether a country loses an afternoon of public services.

Corrections and open points

Earlier work on this subject stated that CERT-EU warned on Tuesday evening. Its own page dates the advisory to Wednesday 21 October, 15:20. The Dutch takedowns had started earlier that morning.

A statement that no Dutch system was breached held for Wednesday. From Thursday 16:00 the NCSC was investigating signals of abuse. They were unconfirmed at the time of writing.

Open: the exact times of the FBI's statement and of the first Wall Street Journal report; when php2shell.com was registered; the cost of upgrading PHP applications, for which no primary source was found.

Annotated references

PHP Foundation (2026). "Today's PHP security release and how we're strengthening PHP security." 21 October 2026. <https://thephp.foundation/blog/2026/10/21/todays-php-security-release-and-how-were-strengthening-php-security/> *Why read?* The project's own post-mortem. It names the "structural failure", the missed 2024 warnings and the embargo break. *Reading advice:* read the section on resources next to the Foundation's 2025 transparency report.

PHP Foundation (2026). "Impact and Transparency Report 2025." <https://thephp.foundation/blog/2026/05/27/impact-and-transparency-report-2025/> *Why read?* The budget and staffing behind a language used by most of the web.

PHP Group. "Supported Versions" and "Unsupported Branches." <https://www.php.net/supported-versions.php> · <https://www.php.net/eol.php> *Why read?* The support dates that turn a version into legacy. *Reading advice:* compare with the W3Techs shares below.

W3Techs (2026). "Usage statistics of PHP" and version breakdowns. <https://w3techs.com/technologies/details/pl-php> · <https://w3techs.com/technologies/details/pl-php/8> *Why read?* The source of the 69.3% figure and of the version shares used to compute the legacy share.

NCSC-NL (2026). PHP2Shell advisory and product list (GitHub). <https://github.com/NCSC-NL/php2shell> *Why read?* The fixed versions, the temporary measures and the list of products with PHP built in. *Reading advice:* open the software list to see how far PHP reaches beyond websites.

CERT-EU (2026). Security Advisory 2026-018. <https://cert.europa.eu/publications/security-advisories/2026-018/> *Why read?* The conditional wording: patch, and isolate only what cannot be patched.

Hacker News (2026). "Tell HN: 2x PHP RCE CVEs with CVSS 10.0 Incoming." Item 50167754. <https://news.ycombinator.com/item?id=50167754> *Why read?* The first public signal, with its wrong score and date.

Wiz (2026). "PHP2Shell: PHP RCE" and php2shell.com. <https://www.wiz.io/blog/php2shell-php-rce> · <https://php2shell.com/> *Why read?* Where "70% of the internet affected" was coined, and how a vendor campaign site frames a flaw.

Cloudflare (2026). "PHP vulnerabilities: WAF protection." <https://blog.cloudflare.com/php-vulnerabilities-waf-protection/> *Why read?* Observations of attack traffic, and a vendor stating the limits of its own protection.

Amazon Threat Intelligence (2026). PHP2Shell exploitation prior to public disclosure. <https://aws.amazon.com/blogs/security/amazon-threat-intelligence-identifies-php2shell-exploitation-prior-to-public-disclosure-in-madpot/> *Why read?* An attempt seen on 17 October, before any public signal. It shows the secret was out before the leak.

VulnCheck (2026). "PHP2Shell vulnerabilities." <https://www.vulncheck.com/blog/php2shell-vulnerabilities> *Why read?* Server-level counts of exposed and end-of-life PHP, and a sober view on exploitability.

OX Security (2026). "PHP2Shell: 24 hours in, 99.3% of measurable PHP instances are still unpatched." <https://www.ox.security/blog/php2shell-24-hours-in-99-3-of-measurable-php-instances-are-still-unpatched/> *Why read?* How slowly patches reach the long tail.

Hadrian (2026). PHP2Shell vulnerability alert. <https://hadrian.io/vulnerability-alerts/php2shell-leaked-critical-php-rce-cve-2026-109618-cve-2026-109619> *Why read?* The origin of the "roughly a third of websites" figure. It is a calculation from W3Techs, not a scan.

Security.nl (2026). "PHP publiceert patches voor kritieke kwetsbaarheden: 'Patch nu meteen!'" <https://www.security.nl/posting/958403/PHP+publiceert+patches+voor+kritieke+kwetsbaarheden%3A+%27Patch+nu+met> *Why read?* The Dutch exception: an article that explained end-of-life, backports and extended support on the day itself.

NOS (2026). Liveblog and follow-up reports, 21-22 October 2026. <https://nos.nl/artikel/2636149-nog-altijd-websites-niet-bereikbaar-na-lek-ook-internationaal-nog-problemen> · <https://nos.nl/artikel/2636225-signalen-bij-nationaal-cyber-security-centrum-van-misbruik-php-lek> *Why read?* The Dutch timeline, the human impact and the NCSC statements. *Reading advice:* note where attempts and breaches are kept apart, and where not.

TheBlaze (2026). "Most of the internet is suddenly melting down." <https://www.theblaze.com/tech/php-internet-meltdown> *Why read?* An example of a headline that no source supported.

Buse, S. and Meissner, K. (2019). "Much Ado About Hacking." GigaNet Symposium. https://www.giga-net.org/2019symposiumPapers/26_Buse_Meissner_Much-Ado-About-Hacking.pdf *Why read?* The measured gap between

alarm and advice in cyber reporting. *Reading advice*: the tables on frames and on security measures.

Maschmeyer, L., Deibert, R. and Lindsay, J. (2021). "A tale of two cybers." *Journal of Information Technology & Politics*. *Why read?* How commercial threat reporting selects its subjects, and what that means for those who rely on it.

Vinsel, L. and Russell, A. (2020). *The Innovation Delusion*. Currency. <https://arussell.org/the-innovation-delusion/>
Why read? Why maintenance is invisible in public debate, and what it costs. *Reading advice*: the chapters on infrastructure and on maintainers.

European Union (2024). Regulation (EU) 2024/2847, Cyber Resilience Act. <https://eur-lex.europa.eu/eli/reg/2024/2847/oj/eng> *Why read?* The obligations that move responsibility for embedded software to the manufacturer. *Reading advice*: Articles 13, 14 and 24.

Freexian, Zend, TuxCare. Extended PHP support offers. <https://www.freexian.com/lts/php/> · <https://www.zend.com/services/php-long-term-support> · <https://tuxcare.com/blog/php2shell-cve/> *Why read?* What it costs to keep old PHP safe, and for how long.