SECURITY GUIDANCE · TLP:CLEAR · VERSION 1.0

PHP2Shell

Find, fix, hunt, prevent.

CVE-2026-109618 / CVE-2026-109619

Practical guidance for UK organisations: find every PHP runtime, including the ones hidden inside firewalls, appliances and supplier-run services; patch it and prove the fix is running; hunt for compromise; and govern the response.

9.2
CVSS v4.0 for both CVEs: unauthenticated remote code execution

21 Oct
Fixes released: 8.2.35 · 8.3.36 · 8.4.27 · 8.5.12

17 Oct
First exploitation attempt recorded, four days before disclosure

5.4+
Vulnerable code present in every PHP release since 2012

How to use this guide

Executives: sections A and J, and the executive checklist in K. **IT managers:** D, H, I, J and K. **Sysadmins and security teams:** B, C, E, F, G and H.

Facts carry a source and date. Text marked JUDGEMENT is IT Revolution's professional view rather than a sourced fact. The situation is described as of 24 October 2026; vendor advisories change daily, so check them for the latest status.

Contents

A	Executive summary	Executives	3
B	Technical overview	Security teams	4
C	Asset discovery and inventory	IT managers · Sysadmins	6
D	Prioritisation model	IT managers	9
E	Remediation	Sysadmins	10
F	Interim mitigations and compensating controls	Sysadmins	14
G	Compromise assessment	Security teams	16
H	Validation and testing	IT managers · Sysadmins	19
I	Prevention	IT managers	21
J	Executive governance	Executives	23
K	Checklists	Everyone	26
L	References	Reference	28
i	Document control and disclaimer		30

EXECUTIVES

Executive summary

> **Any system running PHP that serves web requests can be taken over by an attacker without a password until it is patched, and that includes firewalls, VPN gateways, storage and monitoring appliances, not just websites.** On 21 October 2026 the PHP project fixed two critical flaws, CVE-2026-109618 and CVE-2026-109619, known as PHP2Shell. Both are rated 9.2 out of 10. The UK's NCSC has told organisations to find every PHP installation, patch internet-facing systems first, and consider taking systems offline if they cannot be fixed or mitigated quickly.

What is at risk. A successful attacker runs their own code on the server with the web service's privileges. The researchers also report that the flaws can leak memory, which may contain passwords, API keys, session tokens and certificates. From there an attacker can steal data, plant backdoors, deface sites or move into the internal network. Updating your WordPress, Drupal or Laravel application does not help: the flaw is in PHP underneath it.

Why it is not just a website problem. PHP is built into many products' web management interfaces. Palo Alto Networks has confirmed all PAN-OS and Panorama versions are affected, with fixed releases still pending as of 24 October. Netgate (pfSense) and OPNsense have issued patches. Cisco is investigating 31 products and cloud services, and Fortinet 12. These fixes come from the vendor, not from a PHP upgrade you can run yourself.

Current threat (as of 24 October 2026). Exploitation attempts are widely reported, including one Amazon recorded on 17 October, four days before disclosure. Public proof-of-concept code is mostly unreliable, but at least one vendor says it holds a private, reliable exploit. Neither CVE was in CISA's Known Exploited Vulnerabilities catalogue as of 23 October. JUDGEMENT Treat the current difficulty of exploitation as a short window to patch, not a reason to wait.

Decisions the board or leadership must make this week

Decision	Recommended position JUDGEMENT	Owner
Authorise emergency change outside normal windows	Yes, for all internet-facing PHP and appliance management interfaces, within 24 to 72 hours	CIO / CTO
Systems that cannot be patched or mitigated in 72 hours	Take offline, or restrict to VPN or internal-only access, until fixed. Accept the downtime	Business owner, with CISO advice
Compromise assessment for systems exposed since 17 October	Commission now, internally or via an NCSC-assured incident response provider	CISO
Secrets rotation for exposed systems	Approve rotation of credentials, keys and sessions that passed through exposed systems	CISO / service owners
End-of-life PHP (8.1 and older, and 8.2 after 31 Dec 2026)	Fund migration to PHP 8.4 or 8.5 with dated plans; backports buy time only	CIO, budget holder
Suppliers, hosts and SaaS	Require written confirmation of patch status and date from every supplier that runs PHP for you	Procurement / supplier owner
Regulatory notification	Legal and DPO on standby: a confirmed breach of personal data may need ICO notification within 72 hours	DPO / General Counsel

What good looks like by day 7. Every PHP-bearing asset is on a single register with an owner; every internet-facing one is patched (verified, not assumed), mitigated or offline; appliance vendors have been chased in writing; and the compromise assessment has a result for every system exposed since 17 October.

B

SECURITY TEAMS

Technical overview

Two memory-safety bugs in PHP's request handling let an unauthenticated HTTP request corrupt memory before or around application code, and both are fixed only by PHP 8.2.35, 8.3.36, 8.4.27, 8.5.12 or a vendor package carrying the same patch. This section stays at defensive level: enough to scope, prioritise and detect, with no exploitation detail.

Facts at a glance

Item	Fact	Source, date
CVE-2026-109618 (GHSA-w8hw-w9fx-j79w)	"RCE via OOM corruption". When a memory allocation fails mid-request, PHP can abandon processing while internal request data is half-initialised; cleanup then treats it as valid. Crafted multipart (form and upload) requests can exhaust memory even within configured limits.	Rapid7, 21–22 Oct; NCSC, 21 Oct
CVE-2026-109619 (GHSA-6wmf-4rv5-p3rp)	"RCE via stale global state". State from one request can survive after its memory is freed; a later request on the same worker uses it. Confirmed paths: session upload progress (on by default) and, on PHP 8.5, `fatal_error_backtraces` (on by default).	Rapid7; PHP Foundation, 21 Oct
Severity	CVSS v4.0 9.2 for both (AV:N/AC:H/AT:P/PR:N/UI:N, high C/I/A). The 10.0 that circulated on 20 Oct was wrong. Vendors rescore for their own products (Fortinet 9.8 v3; Palo Alto 8.7 CVSS-B for PAN-OS).	PHP Foundation; Help Net Security, 22 Oct
Affected	Every PHP release before the fixed versions. The vulnerable code dates to PHP 5.4.0 (2012).	Sansec; Debian tracker
Fixed releases	8.5.12, 8.4.27, 8.3.36, 8.2.35, published 21 Oct 2026 around 13:30–14:10 UTC (sources differ by minutes)	php-src 8.4.27; CERT-EU 2026-018
End-of-life branches	No php.net release for 8.1 and older. PHP published "minimal mitigation" source patches for 7.4, 8.0 and 8.1. Many vendors backported (section e).	Rapid7; ma.ttias.be, 23 Oct
PHP's own position	"There are no complete mitigations, updating is the only safe choice."	php-src release notes
Reporters	Calif (reported 11 Aug), Wiz (24 Aug) and OpenAI. The underlying out-of-memory problem was first spotted in 2024 during PHP's external audit.	PHP Foundation

Exposure by SAPI

PHP says every installation that serves web requests is at risk. The SAPI (the way PHP is attached to the web server) changes how easily.

SAPI	Exposure	Notes
Apache mod_php	Highest	PHP rates this the highest risk: full unauthenticated RCE was demonstrated over HTTP with default settings, before any application code runs.

SAPI	Exposure	Notes
PHP-FPM (nginx, Apache proxy_fcgi, Caddy)	High	RCE confirmed via the session upload-progress path when the application also hits a fatal error. For the OOM bug, PHP's code review says FPM is affected but no complete FPM exploit is published. Sources differ slightly on how far FPM RCE is proven.
php-cgi, FrankenPHP, LiteSpeed (lsphp)	In scope	Named by PHP as at risk. FrankenPHP and some products bundle their own PHP binary.
CLI only (cron, CI, build tools)	Lower	Not reachable by web requests, but still update; the same binary is often also serving web traffic.

Contradictions and unconfirmed points

- **Which CVE covers upload progress.** PHP's advisory and php2shell.com put the upload-progress path under CVE-2026-109619; Wiz's and Sansec's tables put it under CVE-2026-109618. The same update fixes both, so remediation is unaffected.
- **Memory disclosure.** The researchers say the bugs can leak worker memory without running code. PHP's advisories describe RCE. JUDGEMENT Plan as if secrets in memory were exposed.
- **Exploitation in the wild.** watchTowr reports in-the-wild exploitation since 21 Oct; Amazon saw an attempt on 17 Oct; CISA and Rapid7 had not observed exploitation as of 21–22 Oct; VulnCheck reports a private ASLR-bypassing exploit for x86-64. Attempts are certain; confirmed successful compromises were not public as of 23 Oct.
- **CISA KEV.** Not listed as of 23 Oct per several trackers. One blog claims it was added; this is unconfirmed. Check the KEV catalogue directly.
- **Possible follow-on bug.** A researcher reported a separate, older memory bug still present in 8.5.12. No CVE or fix as of 23 Oct. It does not reduce the need to install the 21 Oct releases.

How the disclosure unfolded

17 Oct	Amazon's honeypots record a PHP2Shell exploitation attempt.
20 Oct ~15:00	CVE IDs and a wrong CVSS 10.0 circulate online; no technical detail.
20 Oct 19:30	PHP publishes advance notice; release moved forward a day to 21 Oct 14:00.
20 Oct ~20:00	A reporter's PoC video is published by mistake, then deleted; copies circulate.
21 Oct 07:41	Chainguard publishes fixed images early, letting attackers study the fix.
21 Oct 07:43	Cloudflare reports early attacks.
21 Oct ~14:00	**PHP releases 8.2.35, 8.3.36, 8.4.27, 8.5.12 and the advisories. NCSC issues its alert.**
22 Oct	Palo Alto confirms PAN-OS and Panorama affected; Cisco and Fortinet investigating; Debian, Ubuntu, RHEL fixes shipping.

Times UTC. Sources: PHP Foundation; Sansec; Help Net Security, 21 Oct; watchTowr.

C

IT MANAGERS · SYSADMINS

Asset discovery and inventory

> **Build one register of every PHP runtime you own or depend on, from four directions at once: external, internal, cloud and supplier, because each one finds assets the others miss.** NCSC's first priority action is to identify PHP installations generally, not only outdated versions, and to include appliances and services that bundle PHP (NCSC, 21 Oct).

Discovery methods

Method	Finds	Misses / caveats
External attack-surface discovery: DNS zones, certificate transparency logs, your IP ranges, cloud public IPs, ASM tools	Internet-facing hosts, forgotten subdomains, exposed admin portals on odd ports	Origins behind CDNs; hosts that hide `X-Powered-By`; anything internal
Unauthenticated network scanning (e.g. Rapid7 remote checks, Qualys QID 520310)	Responding web services, some PHP fingerprints	Cannot see backported package versions; false negatives behind CDNs and WAFs; false positives on hosts carrying PHP's EOL minimal patch
Authenticated scanning and agents (Qualys QID 45110 for PHP inventory; Nessus, Rapid7, Greenbone credentialed scans)	Exact package versions, every SAPI on the host, internal-only systems	Only hosts you have credentials for; appliances usually excluded
EDR / endpoint inventory, osquery, Velociraptor `Generic.Applications.PHPInventory`	Running PHP processes, binaries in non-standard paths, developer machines	Hosts without an agent
Configuration management and package data (Ansible facts, Puppet, SCCM, Intune)	What was deployed where	Drift and hand-built servers
Cloud inventories (AWS Config, Azure Resource Graph, GCP Asset Inventory) and container registries	VMs, PaaS runtimes, images with PHP layers	Images not yet scanned; running containers on old images
SBOMs and vendor VEX statements	PHP inside commercial products and your own builds	Only as good as the supplier's SBOM; many appliances publish none
Supplier and MSP questionnaire	Hosted, SaaS and managed services that run PHP for you	Relies on supplier honesty; get it in writing
NCSC-NL product list (github.com/NCSC-NL/php2shell)	Known products with a PHP dependency and their status	Community-maintained; check each vendor's own advisory

Scanner references: Qualys, Rapid7. Check with your scanner vendor that its plugin understands distribution backports before trusting a clean result.

Where PHP lives: asset categories

Status reflects what we could verify on 24 October 2026. `VERIFIED` means a vendor or authoritative source confirmed PHP use or impact for this event. `ARCHITECTURE` means the product is known to be built on PHP but we found no PHP2Shell statement. `UNKNOWN` means do not assume either way.

Category	Examples and status	How to detect PHP and version	Where the fix comes from
Web servers, CMS, frameworks	WordPress, Drupal, Joomla, Magento, MediaWiki, Laravel, Symfony, Nextcloud. VERIFIED Calif reported pre-auth RCE on Magento, WordPress, Drupal, Joomla, MediaWiki, Nextcloud and Laravel (Sansec)	On host: SAPI checks in section e. Externally: headers are hints only	PHP runtime update. Application updates do not fix it
Hosting panels and admin tools	cPanel (EasyApache 4), Plesk, DirectAdmin, CloudLinux alt-php, phpMyAdmin, Adminer. VERIFIED Plesk patched 7.4–8.1 but not 5.6–7.3; DirectAdmin requires a CustomBuild rebuild (ma.ttias.be; NCSC-NL)	Panels run several PHP versions per server under `/opt/cpanel/ea-php*`, `/opt/plesk/php/*`, `/opt/alt/php*`, `/usr/local/php*`	Panel vendor's packages; rebuild where required. Each PHP version on the box separately
Firewalls, VPN gateways, routers	VERIFIED Palo Alto PAN-OS and Panorama, all versions affected, fix still pending at 24 Oct. pfSense (PHP GUI and Captive Portal) patched in CE 2.9.0, CE 2.8.1, Plus 26.07 and 26.03.1; 2.7.2 unsupported. OPNsense hotfixes 26.7.7 and 26.4.2p3. Fortinet: FortiOS confirmed not affected (Help Net Security; Netgate; Notebookcheck)	You usually cannot see the PHP version. Rely on the vendor's affected-version table and your firmware version	Vendor firmware only. pfSense updates must be applied from console or SSH, not the GUI
Load balancers, NAC, wireless controllers, network management	UNDER INVESTIGATION Cisco (22 Oct): ISE, Catalyst Center, Catalyst SD-WAN Manager, Nexus Dashboard Insights, WAE, Expressway (bug CSCwx37062, fix TBD), URWB, Unified Industrial Wireless. Fortinet investigating FortiADC, FortiClient EMS, FortiSIEM, FortiSOAR and others	Track the vendor PSIRT advisory and bug IDs	Vendor fix; Cisco says no workarounds
NAS, storage, backup	VERIFIED PHP RELATIONSHIP QNAP QTS and QuTS hero built-in PHP, impact under investigation (NCSC-NL). Synology published advisory Synology-SA-26:14 for Web Station PHP packages (cited by Notebookcheck; we did not open it)	Package Centre / App Center lists PHP packages and versions	Vendor package or firmware update
Monitoring, ITSM, helpdesk	VERIFIED affected when on a vulnerable runtime: Icinga Web, GLPI, Faveo, Hesk (NCSC-NL). ARCHITECTURE Zabbix frontend, Cacti and Nagios XI are PHP applications; check vendor	Self-hosted: check the host's PHP. Appliance or vendor ISO: vendor statement	Self-hosted: PHP update. Vendor appliance: vendor image
Webmail, collaboration, VoIP / PBX	VERIFIED Roundcube (Calif confirmed pre-auth RCE), RainLoop, grommunio Web. ARCHITECTURE FreePBX web UI is PHP; check vendor	As above	As above. Distro-based PBXs often pin old PHP
Printers, MFDs, IoT, building management, OT/ICS HMIs	UNKNOWN No vendor statements found. Some embedded web UIs use PHP; many use other stacks. RaspAP (a Raspberry Pi network GUI) is verified PHP-CGI	Ask the vendor in writing; check SBOMs; look for `.php` paths in the web UI	Vendor only. JUDGEMENT For OT, prefer network isolation over rushed firmware changes
Hypervisor and infrastructure consoles	UNKNOWN FOR MOST JUDGEMENT Major hypervisor consoles are not generally PHP-based, but adjacent dashboards and HCI tools may be. Verify via PSIRTs	Vendor statement; SBOM	Vendor
Containers and images	Official Docker `php` images fixed for all four branches on amd64 and arm64 by 21 Oct evening; FrankenPHP 1.13.2 fixed; Alpine 3.24 fixed, older Alpine releases will not be	`trivy image`, `grype`, registry scanning; `docker exec <c> php -v` on running containers	Rebuild with `docker build --pull` and **recreate** containers. Running containers keep the old image
CI/CD runners	GitHub Actions `setup-php` does not update an already-present PHP by default; add `update: true`	Pipeline config review	Pipeline change

Category	Examples and status	How to detect PHP and version	Where the fix comes from
Developer workstations, test, staging	XAMPP's last release ships PHP 8.2.12; MAMP and static-php-cli builds were unfixed at 23 Oct; Herd, Homebrew, Laragon (8.3) fixed	EDR inventory; look for `php.exe` / `php` outside managed paths	Tool vendor or reinstall. Staging is often internet-exposed and forgotten
Windows / IIS	windows.php.net builds fixed. IIS runs PHP via FastCGI (php-cgi)	Search for `php.exe` and `php-cgi.exe`; review IIS handler mappings	Replace PHP binaries, recycle app pools
PaaS, managed hosting, SaaS, MSPs	AWS Elastic Beanstalk issued fixed platform versions; **Lightsail running instances are not updated**, only new blueprints; Heroku buildpacks fixed. Cisco is investigating 13 cloud services	Supplier statement; your own runtime check where you have a shell	Supplier, or you on IaaS

Why version strings mislead

- **Backports keep the old number.** Ubuntu 24.04's fixed PHP still reports `8.3.6`; the fix is in package `8.3.6-0ubuntu0.24.04.13`. Check the package version, not `php -v`.
- **EOL minimal patch hides itself.** PHP's 7.4/8.0/8.1 backport forces upload progress off but `php -i` may still say `On`, so configuration-reading scanners flag patched hosts.
- **Appliances report firmware, not PHP.** A PAN-OS or Cisco version number is the only reliable signal; PHP inside is a vendor build.
- **Banners are suppressed or belong to the CDN.** `expose_php=Off` and CDN termination hide the origin entirely.
- **One host, several PHPs.** CLI, mod_php and each FPM pool can be different versions; panels add more under `/opt`.

Inventory record: minimum fields

Asset ID and hostname or workload; business service and owner; supplier (if any); exposure (internet, partner, internal, management network); every PHP runtime and SAPI with exact package version and source repository; support status of that branch; fix status and evidence; mitigation status; exposure window start and end; compromise-assessment result.

D

IT MANAGERS

Prioritisation model

> **Work internet-facing systems first, then anything that holds credentials or reaches many networks, then the rest, and never let any in-scope system pass 14 days.** NCSC says to prioritise internet-facing systems and then internal ones; the tiers and targets below are JUDGEMENT built on that, on PHP's SAPI risk ranking and on Cyber Essentials' 14-day ceiling.

P0	**24 hours**	Internet-facing Apache mod_php. Appliance management interfaces or captive portals reachable from the internet (firewalls, VPN gateways, Expressway, NAS admin pages). Internet-facing EOL PHP. Anything with signs of probing or crashes since 17 Oct. *If not achievable, take offline or restrict to VPN or allow-listed IPs now.*
P1	**72 hours**	All other internet-facing PHP (FPM, php-cgi, FrankenPHP, LiteSpeed), partner-facing portals, PaaS and SaaS where you hold the runtime, shared hosting.
P2	**7 days**	Internal PHP reachable from user networks. High-value internal tools that hold secrets: monitoring, ITSM, password or config tools, webmail. Appliance admin interfaces on internal management networks.
P3	**14 days**	CLI-only PHP, CI runners, isolated dev and test, workstations. 14 days is the Cyber Essentials ceiling for in-scope devices.

Raise a system one tier if it runs mod_php, runs an EOL branch, stores or processes personal or payment data, holds credentials for other systems, is reachable from several network zones, or sits on shared infrastructure where one tenant's request can affect another.

Lower a tier only if isolation is verified (a firewall rule you have tested, not a diagram) or the runtime is CLI-only with no web listener.

Appliances awaiting a vendor fix stay at their tier. If the fix does not exist yet, the target applies to the compensating control: restrict the management interface to a jump host or management VLAN, as Palo Alto recommends for PAN-OS (Help Net Security).

E

SYSADMINS

Remediation

> **Install the fixed PHP from the same source you installed PHP from, restart every process that loads it (a full restart for Apache mod_php), then prove nothing is still running the old code.** The fixed upstream versions are 8.5.12, 8.4.27, 8.3.36 and 8.2.35; distributions often ship the fix under their own version numbers. Commands below are for common platforms; paths and service names vary, so test before fleet-wide use.

Decision path per runtime

1. **Appliance or vendor product?**
   Apply the vendor's fixed firmware or image. If none exists, go to section f and isolate the management interface. You cannot fix it by upgrading PHP yourself.

2. **Supported branch (8.2 to 8.5)?**
   Update to the fixed release or your distribution's fixed package. Restart. Verify.

3. **EOL branch (8.1 and older) with a vendor backport?**
   Install it, restart, verify, and open a dated migration ticket to 8.4 or 8.5.

4. **EOL branch with no backport?**
   Apply section f mitigations today, restrict access, and migrate. If none of that is possible within your tier's target, take it offline.

Step 1: find every runtime on the host

```
# Running PHP processes (FPM, mod_php children, LiteSpeed, FrankenPHP)
ps -eo pid,user,cmd | grep -E '[p]hp|[l]sphp|[f]rankenphp|[h]ttpd|[a]pache2'

# PHP binaries in common locations, including panel and SCL builds
ls -1 /usr/bin/php* /usr/sbin/php-fpm* /usr/local/bin/php /usr/local/sbin/php-fpm \
  /opt/remi/php*/root/usr/{bin,sbin}/php* /opt/cpanel/ea-php*/root/usr/{bin,sbin}/php* \
  /opt/plesk/php/*/{bin,sbin}/php* /opt/alt/php*/usr/{bin,sbin}/php* 2>/dev/null

# Is mod_php loaded? (apache2ctl on Debian/Ubuntu, httpd -M on RHEL)
apache2ctl -M 2>/dev/null | grep -i php || httpd -M 2>/dev/null | grep -i php

# FPM services and the version of each FPM binary
systemctl list-units --type=service --all | grep -Ei 'php.*fpm'
for b in /usr/sbin/php-fpm*; do echo "$b: $($b -v | head -1)"; done
```

> `php -v` **tells you about the CLI only.** Check every FPM binary, the Apache module and any panel-managed version separately.

Step 2: fixed packages by source

Checked 21–23 October 2026. Always confirm against your vendor's own tracker on the day.

Source	Fixed package or version	Notes
php.net, Docker official `php` images, Homebrew, windows.php.net, Gentoo	8.2.35, 8.3.36, 8.4.27, 8.5.12	Docker: rebuild with `--pull` and recreate containers
Debian 13 (php8.4)	`8.4.27-1~deb13u1` (DSA-6574-1)	Debian ships the upstream release
Debian 12 (php8.2)	`8.2.35-1~deb12u1` (DLA-4848-1)	
Debian 11 (php7.4)	`7.4.33-1+deb11u14` via Freexian ELTS only (ELA-1853-1)	Debian LTS for 11 ended 31 Aug 2026
Ubuntu 26.04 / 24.04 / 22.04	`8.5.4-0ubuntu1.5` / `8.3.6-0ubuntu0.24.04.13` / `8.1.2-1ubuntu2.28` (USN-8948-1)	Backported: `php -v` still shows the old version
Ubuntu 20.04 and older	Ubuntu Pro ESM (USN-8948-2)	Needs ESM enabled; `pro fix CVE-2026-109618`
RHEL 8, 9, 10 and clones	Fixed in all supported streams **except RHEL 9** `php:8.4` (pending at 23 Oct)	RHEL 8 default `php:7.2` stream and other retired streams get no fix
Remi (RHEL 8–10)	8.2–8.5 upstream releases; 7.4, 8.0, 8.1 patched; 7.2, 7.3 not	Some mirrors lag: `dnf clean metadata`
Ondřej Surý (ppa:ondrej/php, packages.sury.org)	All branches 5.6–8.5 on amd64; some arm64 builds lagged	Changelog may only say "new upstream version"
Alpine	3.24 and edge fixed for 8.3–8.5	Older Alpine releases will not get a fix
Amazon Linux	AL2023 php8.2–8.5; AL2 php8.2 extra	No fix for AL2023 php8.1 or AL2 core php
SUSE / openSUSE	Not confirmed in our research	Check the SUSE CVE page
CloudLinux alt-php, Plesk, DirectAdmin, cPanel EA4	Fixed packages published (Plesk not for 5.6–7.3; DirectAdmin needs a rebuild)	Update every PHP version installed by the panel

Sources: Debian tracker, Freexian, ma.ttias.be patch guide, updated 23 Oct, SANS ISC.

Step 3: apply the update

```
# Debian / Ubuntu (repeat per installed PHP version)
dpkg -l 'php*' | awk '/^ii/{print $2, $3}'
sudo apt-get update
sudo apt-get install --only-upgrade 'php8.3*' libapache2-mod-php8.3
# Ubuntu changelogs name the CVEs; Debian and sury may not, so compare versions with the table
zgrep -E 'CVE-2026-10961[89]' /usr/share/doc/php8.3-fpm/changelog.Debian.gz

# RHEL / Alma / Rocky (Red Hat and Remi modular repos)
rpm -qa 'php*' | sort
dnf module list --enabled php          # RHEL 8 and 9 only
sudo dnf upgrade 'php*'                # Remi SCL builds: sudo dnf upgrade 'php84-*'
dnf updateinfo list --security | grep -i php

# Alpine 3.24 (package names follow the branch, e.g. php83)
sudo apk update && sudo apk upgrade php83 php83-fpm

# SUSE family
sudo zypper refresh && zypper list-patches | grep -i php

# Containers
docker build --pull --no-cache -t myapp:patched .
```

```
docker compose up -d --force-recreate
kubectl rollout restart deployment/<name>
```

Windows/IIS: replace the PHP directory with the fixed build from windows.php.net, keep your `php.ini`, then recycle the IIS application pools.

Step 4: restart the right thing

How PHP runs	What to do	Trap
PHP-FPM behind nginx, Caddy or Apache proxy_fcgi	`systemctl restart php8.x-fpm` (Debian/Ubuntu) or `php-fpm` (RHEL)	Each version has its own service
Apache mod_php	Full `systemctl restart apache2` or `httpd`	**A reload or `apachectl graceful` keeps the old `libphp` in memory**
LiteSpeed / OpenLiteSpeed	Upgrade `lsphpXY`, then `pkill lsphp`	Restarting LiteSpeed alone does not restart PHP
FrankenPHP, Laravel Octane	Install FrankenPHP 1.13.2 or later; delete Octane's bundled `frankenphp` binary and restart	apt/dnf do not update it
Queue workers, Supervisor, Swoole, RoadRunner	Restart them (`php artisan queue:restart` for Laravel)	Long-running workers hold old code for days
Containers	Recreate from the rebuilt image	Pulling a tag does not change running containers

Source for the Apache behaviour, tested: ma.ttias.be.

Step 5: prove the fix is loaded

```
# Any process still mapping a deleted (pre-upgrade) library? Empty output = good
sudo grep -lE ' /(usr|lib|opt).*\(deleted\)$' /proc/[0-9]*/maps

# Debian / Ubuntu alternative
sudo needrestart -b -r l
```

Record the package version, restart time and this output as evidence (section h).

End-of-life PHP (8.1 and older)

A backport closes these two CVEs; it does not make an EOL branch supported. PHP's own 7.4/8.0/8.1 patches are a "minimal mitigation" that switches upload progress off entirely and adds no hard memory limit.

Option	Use when	Limits
Vendor backport: Ubuntu (22.04 free; Pro for older), Freexian ELTS, Red Hat (`php:7.4` on RHEL 8, 8.0 on RHEL 9), Remi, Ondřej Surý, CloudLinux, Plesk (7.4–8.1), Zend LTS, TuxCare	You already use that source, or can switch quickly	Fixes these CVEs only. Upload progress bars stop working
Build from PHP's source backport branches	You compile PHP yourself; 7.4, 8.0 or 8.1 only	You own the build, testing and future fixes
Third-party extension (e.g. hp_uploadprogress_guard)	Last resort only JUDGEMENT	Not written or reviewed by PHP

Option	Use when	Limits
Migrate to PHP 8.4 or 8.5	Always, with a dated plan	Application and framework compatibility work; often an OS rebuild
Isolate or take offline	No fix and no viable mitigation within target	Business impact; agree with the owner

Known no-fix cases at 23 Oct: RHEL 8 `php:7.2`, `7.3`, `8.0` and RHEL 9 `php:8.1` streams; Debian 11 without ELTS; Ubuntu 25.04 and 25.10; Alpine before 3.24; Amazon Linux 2 core php and AL2023 php8.1; Plesk 5.6–7.3; Remi 7.2 and 7.3; pfSense 2.7.2.

> **Do not migrate to 8.2.** It reaches end of life on 31 December 2026 (php-src 8.2.35 notes). Target 8.4 (security support to end of 2028) or 8.5 (to end of 2029).

Vendor appliances awaiting a fix

1. Record the vendor advisory ID, your firmware version and any bug ID (e.g. Cisco CSCwx37062 for Expressway).
2. Restrict management interfaces to a jump host or management VLAN now, and confirm with a test from an untrusted network.
3. Enable vendor threat-prevention signatures where offered (Palo Alto threat IDs 543004 and 97622–97625; Cisco Snort rules 67359 and 67360; FortiGuard IPS 37.316).
4. Chase the vendor in writing every 48 hours until a fixed version is published; escalate through your account manager.
5. Apply the fix as a P0/P1 change the day it ships, then verify the firmware version.

After patching: expect these

- **New hard memory limit.** The fix adds `memory_hard_limit_percent` (by default twice `memory_limit` up to 1G). Workers that reach it exit with code 137 and skip shutdown functions. Watch FPM logs.
- **Regressions in 8.4.27 and 8.5.12.** A SimpleXML garbage-collection crash (`zend_mm_heap corrupted`) and some out-of-memory cases now ending in exit 137. Fixed in source on 22 Oct; 8.3.36 and 8.2.35 contain only the security fix. JUDGEMENT None of these justifies staying unpatched.
- **Roll back temporary mitigations.** Restore a real `memory_limit`; remove blanket multipart blocks. Keep `session.upload_progress.enabled=0` unless you use it, and keep `session.use_strict_mode=1`.

Source: ma.ttias.be, updated 23 Oct.

SYSADMINS

F Interim mitigations and compensating controls

> **Every measure here reduces risk on known paths; none of them fixes the flaws, and PHP says no configuration fully protects against CVE-2026-109619.** Use them only to bridge the gap to a fix, record each one as a time-limited risk acceptance, and remove the ones that hurt once you have patched.

PHP's own configuration mitigations

Setting	Addresses	Where it must be set	Limits and side effects
`memory_limit = -1`, enforced so applications cannot override it	CVE-2026-109618: removes the predictable memory-limit failure the confirmed exploit uses	`php_admin_value` in the FPM pool or Apache config	Real allocation failures can still hit the bug. One request can now consume all memory, so add the limits below. Revert after patching
`session.upload_progress.enabled = 0`	Confirmed session paths of CVE-2026-109619	`php.ini`, FPM pool (`php_admin_flag`) or server config that applies before request parsing	`.user.ini` and `ini_set()` are too late. mod_php ignores `.user.ini`. Upload progress bars stop
`fatal_error_backtraces = 0` (PHP 8.5 only, on by default)	The 8.5 backtrace path of CVE-2026-109619	Administratively, then restart workers	Other stale-state paths remain
`post_max_size = 8M` (32M if uploads needed), `upload_max_filesize` below it	Makes memory exhaustion harder	`php.ini` / pool	Large uploads fail
Move per-request settings out of `.htaccess` and `PHP_VALUE`; fix deprecated or invalid INI values	Per-request INI warnings can feed CVE-2026-109619	Server config	Needs an application review
Hiding errors (`display_errors`, `@`, `error_reporting`)	Nothing	—	Does not prevent stale-state access. Absence of log entries is not proof of safety

Sources: PHP's advisories as summarised by Rapid7 and SANS ISC.

```
; FPM pool, e.g. /etc/php/8.3/fpm/pool.d/www.conf (repeat for every pool)
php_admin_value[memory_limit] = -1
php_admin_flag[session.upload_progress.enabled] = off
php_admin_value[post_max_size] = 8M
; PHP 8.5 only
php_admin_flag[fatal_error_backtraces] = off

; Apache mod_php (vhost or server config), then a full restart
php_admin_value memory_limit -1
php_admin_flag session.upload_progress.enabled off
```

Confirm the pool settings took effect with `php-fpm8.3 -tt 2>&1 | grep -E 'memory_limit|upload_progress|post_max_size'`, which dumps the parsed pool configuration, then restart every worker.

Network and edge controls

Control	What it does	Honest limits
Take offline, or restrict to VPN or allow-listed IPs	Removes unauthenticated reachability	The only interim measure that is close to complete. Business impact
Restrict appliance management interfaces to a jump host or management VLAN	Removes the main attack surface on firewalls, NAS and controllers	Insiders and compromised admin workstations can still reach it
Front-end request limits: body size, number of multipart parts, headers per part, nesting depth, requests without a usable `Content-Length`	Starves the memory-exhaustion technique	Must be tuned per application; PHP recommends it alongside, not instead of, patching
Block `multipart/form-data` where the application never accepts uploads or multipart forms	Removes the confirmed request type	Breaks uploads. An attacker who reaches the origin directly bypasses an edge-only rule
WAF and IPS signatures (Cloudflare managed rules, Palo Alto threat IDs, Cisco Snort 67359/67360, FortiGuard 37.316)	Blocks known exploit patterns	Signature-based; will lag new variants. Useless if the origin is reachable without going through the WAF
Egress filtering from web and appliance hosts	Limits command-and-control and data theft after compromise	Does not prevent exploitation
Own FPM pool and user per application; read-only webroot; AppArmor/SELinux confinement	Limits blast radius	Does not prevent exploitation
`pm.max_requests = 1` for FPM JUDGEMENT	Reduces cross-request state reuse	Heavy performance cost; not an official mitigation
Container or cgroup memory limits	—	PHP warns these are incomplete and can kill the whole pool. Do not use `RLIMIT_AS` or `RLIMIT_DATA`, which can trigger the vulnerable path

Example nginx edge limits JUDGEMENT (adapt to your application):

```
client_max_body_size 8m;
# Only where the site never takes uploads or multipart forms:
if ($http_content_type ~* "multipart/form-data") { return 403; }
```

> **Lock the origin.** If you rely on a CDN or cloud WAF, make sure the origin only accepts traffic from that provider (firewall allow-list or authenticated origin pulls). JUDGEMENT Attackers routinely find origin IPs through historic DNS and certificate data, and an edge rule does nothing if they can go round it.

SECURITY TEAMS

Compromise assessment

> **Patched is not clean: every PHP system that was reachable while unpatched since at least 17 October needs a compromise assessment, and any internet-facing one should have its secrets rotated whatever the assessment finds.** NCSC says to assess for evidence of compromise and backdoors before or alongside remediation, and to use an NCSC-assured Cyber Incident Response provider if you lack the capability (NCSC).

1. Fix the exposure window per system

- **Start:** 17 October 2026 (first recorded attempt, Amazon) or the system's go-live, whichever is later. JUDGEMENT If logs allow, look back further; the bugs date to 2012 and were known to researchers from August, though no earlier exploitation has been reported.
- **End:** the verified time the fixed code was running (section e, step 5) or the system was isolated, not the time the package was installed.
- **Highest concern:** 20 October evening onwards (leak and PoC video), and especially 07:41 UTC on 21 October onwards, when the fix could be studied.

2. Preserve before you look

Snapshot or image the system, export logs for the whole window with SHA-256 hashes, and capture memory on high-value systems where your tooling allows. Do this before cleaning anything. Work on copies.

3. What to hunt for

Layer	Indicators	Where
Exploit attempts	Multipart POSTs to any PHP path including `/` ; a `PHP_SESSION_UPLOAD_PROGRESS` form field; very large part headers or filenames; many empty file parts; requests with no usable `Content-Length` ; bursts of 5xx from one source	CDN/WAF logs, reverse-proxy logs with content type, IDS. Default access logs do not record content type or body
PHP failures	Apache `child pid ... exit signal Segmentation fault` ; FPM `child ... exited on signal 11 (SIGSEGV)` ; `Allowed memory size ... exhausted ... in Unknown on line 0` (memory exhausted before application code); coredumps; after patching, exit code 137	Web server and FPM error logs, journal, `coredumpctl`
Web shells and droppers	New or changed `.php` , `.phtml` , `.phar` , `.user.ini` , `.htaccess` in webroots, upload, cache and temp directories; `auto_prepend_file` or `auto_append_file` added	File system, deployment records, integrity tools
Application tampering	New administrator accounts; changed core or plugin files; injected JavaScript (payment skimmers on e-commerce)	App admin UI, database, checksum tools
Execution	Web server or FPM user spawning `sh` , `bash` , `curl` , `wget` , `python` , `perl` , `base64` , `whoami` , `id` , `uname`	EDR, auditd `execve` records
Persistence	New cron entries or systemd timers; changed `authorized_keys` ; new local users; new services	Host
Objectives	Sustained high CPU (cryptominers); outbound connections from web hosts to unfamiliar IPs or mining-pool ports; large outbound transfers; database dumps in web-accessible paths	NetFlow, firewall, proxy and DNS logs, EDR

Layer	Indicators	Where
Appliances	Configuration changes, new admin accounts, unexpected logins, unexpected reboots	Appliance audit logs and syslog; vendor IoC guidance when published

VulnCheck expects floods of attempts from broken public PoCs, so attempts in logs alone say little about whether a server was compromised (Help Net Security). Weight host evidence above request evidence.

4. Read-only hunting commands (Linux)

```
SINCE='2026-10-17'

# PHP worker crashes and pre-application memory exhaustion
journalctl --since "$SINCE" -u 'php*-fpm*' | grep -Ei 'SIGSEGV|signal 11|exited on signal|code 137'
grep -h 'Segmentation fault' /var/log/apache2/error*.log /var/log/httpd/error_log* 2>/dev/null
grep -h 'Unknown on line 0' /var/log/php*fpm*.log /var/log/apache2/error*.log /var/log/httpd/error_log*
2>/dev/null
journalctl -k --since "$SINCE" | grep -i segfault
coredumpctl list --since "$SINCE" 2>/dev/null

# Files created or changed in the window (adjust roots; compare with deploy records)
find /var/www /srv /home/*/public_html /tmp /dev/shm -xdev -type f \
  \( -name '*.ph*' -o -name '.user.ini' -o -name '.htaccess' \) -newermt "$SINCE" -ls 2>/dev/null

# Higher-signal web shell patterns (still review each hit)
grep -rlE --include='*.ph*' \
  'eval\s*\(\s*(base64_decode|gzinflate|str_rot13)|\$_(POST|GET|REQUEST|COOKIE)\[[^]]*\]\s*\(' /var/www
2>/dev/null
grep -rnE 'auto_(prepend|append)_file' /var/www /etc/php* 2>/dev/null

# Persistence
ls -la /etc/cron.* /var/spool/cron/ 2>/dev/null; systemctl list-timers --all
find / -xdev -name authorized_keys -newermt "$SINCE" 2>/dev/null

# What is the web user running now? (www-data on Debian/Ubuntu; apache or nginx on RHEL)
ps -u www-data -o pid,ppid,etime,cmd
ss -tnp state established | grep -E 'php|www-data|apache|nginx'

# 5xx bursts by source IP (combined log format; adjust field numbers)
awk '$9 ~ /^5/ {print $1}' /var/log/nginx/access.log* | sort | uniq -c | sort -rn | head -20
```

WordPress example: `wp user list --role=administrator --fields=ID,user_login,user_registered`, `wp core verify-checksums` and `wp plugin verify-checksums --all`. Use the equivalent integrity checks for Drupal, Magento and others, or compare against your Git repository.

For future visibility, log process execution by the web user, for example an auditd rule `-a always,exit -F arch=b64 -S execve -F uid=33 -k web_exec` (uid 33 is `www-data` on Debian; 48 is `apache` on RHEL).

5. Log sources to collect

Web access and error logs; PHP-FPM logs; system journal and kernel log; auditd or EDR telemetry; CDN and WAF logs; firewall, proxy, DNS and NetFlow logs for egress; application audit logs (admin logins, user changes); database audit logs; appliance audit logs and syslog; cloud control-plane and flow logs.

6. Escalate to incident response when any of these is true

- A web shell, unexplained `.php` file or `auto_prepend_file` entry is found.
- The web or appliance service user ran shells, downloaders or reconnaissance tools.
- A new administrator account or unexplained configuration change appears.
- Web hosts made outbound connections you cannot explain.
- Crash clusters line up in time with suspicious multipart requests.
- JUDGEMENT A P0 system was exposed for more than 24 hours and you lack the logs to rule compromise out. Treat it as potentially compromised.

7. If compromised

1. Invoke your incident response plan; contain by isolating, not by powering off (memory evidence is lost).
2. Rebuild from known-good images on patched PHP rather than cleaning in place.
3. Rotate every secret the host could read: database passwords, API keys, framework keys (e.g. Laravel `APP_KEY`, WordPress salts), session secrets, SSH keys, cloud credentials and instance roles, TLS private keys stored on the host. Invalidate active sessions.
4. Hunt for lateral movement from the host into internal networks.
5. Report to the NCSC (report.ncsc.gov.uk); bring in an NCSC-assured CIR provider if needed.
6. Brief your DPO and legal adviser immediately on whether personal data was affected (section j sets out the 72-hour ICO duty and NIS reporting).

H

IT MANAGERS · SYSADMINS

Validation and testing

> **An asset is closed only when the fixed code is proven to be running, the proof is filed, and the compromise assessment is done; a clean scan on its own proves very little.** Validate with at least two independent methods per asset, because each method has blind spots.

Definition of done (per asset)

- [ ] Fixed package or vendor firmware installed from a trusted source; version recorded against the vendor's fixed-version table
- [ ] Every SAPI restarted; `/proc/*/maps` check or `needrestart` shows no process on old libraries
- [ ] Authenticated scan clean, or every finding explained (e.g. EOL minimal-patch false positive)
- [ ] External check from outside your network, including against the origin IP if behind a CDN
- [ ] Management interfaces confirmed unreachable from untrusted networks (tested, not assumed)
- [ ] Interim mitigations recorded, and rolled back where they harm service (section e)
- [ ] Compromise assessment completed with result recorded (section g)
- [ ] Secrets rotated where the risk assessment requires it
- [ ] Asset owner sign-off

Validation methods and their blind spots

Method	Proves	Blind spots
Package version check against vendor table	Fixed code is on disk	Not that it is running
`/proc` maps or `needrestart`	No process uses the old library	Containers: run inside each container or check the image ID of running containers
Authenticated scanner	Estate-wide coverage and trend	Plugin quality on backports; hosts without credentials; EOL minimal patch still shows `upload_progress => On`
Unauthenticated / external scanner	What an attacker sees	Behind CDNs it sees the edge; suppressed banners give false negatives
Container registry plus runtime check	Images are fixed **and** running containers use them	A fixed registry with old pods still running is common
Appliance firmware check	Vendor fix applied	Only as good as the vendor's advisory
Configuration dump (`php-fpm -tt`, `php -i` per SAPI)	Mitigations are in effect	`php -i` shows the CLI, not FPM pools

Safe testing

- **Do not run exploit PoCs against production.** The bugs are memory corruption: failed attempts crash workers and muddy the forensic picture. Version and configuration checks answer the same question safely.
- **Purpose-built checkers** (for example Searchlight Cyber's open-source php2shell-scanner, released 22 Oct per Help Net Security): read what each check sends, run it only against systems you own and are authorised to test, and treat results as one input. JUDGEMENT
- **If exploit validation is genuinely needed,** reproduce the exact build in an isolated lab.

Retest cadence JUDGEMENT

Daily authenticated and external scans until every asset is closed; weekly for the following four weeks; and again after every image rebuild, restore from backup, failover or new deployment, because those are how old PHP returns. Pin base images to fixed versions so rebuilds cannot regress.

Evidence to retain

Evidence	Why
Inventory record per asset, with owner and exposure	Shows scope was complete
Before and after package or firmware versions, with timestamps	Proves the fix and the exposure window end
Change tickets and restart records	Shows controlled, timely change
`/proc` / `needrestart` output, scan reports before and after	Proves the fix was running
Mitigation configuration diffs and risk acceptances with expiry dates	Shows interim risk was owned
Supplier and vendor written statements	Supply-chain due diligence
Compromise-assessment notes, log exports and hashes	Supports any ICO, NIS, insurer or customer inquiry
Sign-offs	Accountability

JUDGEMENT Keep the pack for at least 12 months, and longer if any incident, regulatory notification or insurance claim arose. It is also your evidence for Cyber Essentials and ISO 27001 audits.

IT MANAGERS

Prevention

> **The next PHP2Shell will be a two-hour exercise only for organisations that already know where every runtime is, patch critical exposures within days, keep admin interfaces off the internet, and hold suppliers to the same standard.** NCSC links its PHP alert to its guidance on preparing for a "vulnerability patch wave" (NCSC). The PHP Foundation reports 15 CVEs since 1 July 2026 alone.

Patch service levels JUDGEMENT, ALIGNED TO CYBER ESSENTIALS

Exposure	Critical (CVSS 9+) or known exploited	High (CVSS 7–8.9)	Medium and below
Internet-facing, including appliance management and captive portals	48 hours (mitigate or isolate within 24 hours)	7 days	30 days
Internal, holding credentials or reaching many zones	7 days	14 days	30 days
Other internal	14 days	14 days	60 days

Cyber Essentials v3.3 (Danzell question set, from 27 April 2026) makes failing to apply critical or high-risk updates within 14 days an automatic fail for in-scope devices, firmware included (summary; confirm against the NCSC/IASME requirements). Pre-approve an emergency change route so these SLAs are achievable at 2 a.m. on a Wednesday.

Asset and software management

- Keep a live register fed automatically by discovery, agents, cloud APIs and container registries. Reconcile monthly; audit quarterly.
- Record runtimes, not just hosts: every PHP (and other language runtime) per SAPI, with source repository and support end date.
- Require SBOMs (CycloneDX or SPDX) for software you build and buy, and VEX statements from vendors for critical CVEs.
- Maintain an end-of-life register with dated retirement plans. PHP support ends: 8.2 on 31 Dec 2026, 8.3 on 31 Dec 2027, 8.4 on 31 Dec 2028, 8.5 on 31 Dec 2029.

Scanning cadence JUDGEMENT

Scope	Cadence
External attack surface	Continuous or daily
Authenticated internal scanning	Weekly, plus on demand for critical advisories
Container images	Every build, plus daily registry rescans; block deploys with critical findings
Appliances	Subscribe to every vendor's PSIRT feed; check firmware against advisories monthly
UK organisations	Sign up to the free NCSC Early Warning service

PHP hardening baseline JUDGEMENT

- `expose_php = Off`; `session.use_strict_mode = 1`; `session.upload_progress.enabled = 0` unless the application needs it.
- Set PHP options in `php.ini` or the FPM pool, not in `.htaccess` or per-request `PHP_VALUE`.

- One FPM pool and system user per application; `open_basedir`; read-only code directories; uploads stored outside the webroot and never executable.
- AppArmor or SELinux profiles for web and PHP processes; egress allow-lists from web tiers.
- Prefer FPM over mod_php for new builds, given PHP's own risk ranking for this event.

Management interface segmentation

Appliance, hypervisor, NAS and panel admin interfaces should never be reachable from the internet. Place them on a dedicated management network reachable only from hardened jump hosts or privileged access workstations, with MFA. Test it from outside quarterly. NCSC guidance: preventing lateral movement and vulnerability management.

Supplier security requirements

Write these into contracts and renewals JUDGEMENT:

1. Notify you within 24 hours of a critical vulnerability affecting the service or product, and state exposure and remediation plan.
2. Patch critical vulnerabilities within an agreed SLA no longer than yours, and confirm in writing with dates.
3. Provide SBOMs and VEX statements on request; disclose end-of-life components.
4. Use only supported runtimes, with migration commitments before end of life.
5. Give you the right to audit or to receive independent assurance (ISO 27001 certificate scope, Cyber Essentials Plus, pen-test summaries).
6. Cooperate with incident investigations and share relevant logs.

EXECUTIVES

Executive governance

Leadership's job is to make the response measurable and owned: one accountable executive, named owners per asset and supplier, a small set of KPIs reported on a fixed rhythm, and documented risk acceptances with expiry dates. Regulatory points below are summarised from the statutes and standards as generally understood; they were not re-researched for this event, so confirm details with your DPO or legal adviser. JUDGEMENT THROUGHOUT THIS SECTION UNLESS SOURCED

Policies to enforce

Policy	Minimum requirement
Vulnerability management	Severity- and exposure-based SLAs (section i); emergency route; exceptions need a named executive and an expiry date
Asset management	Every system and runtime registered with an owner before go-live; discovery reconciled monthly
Supported software / end of life	No unsupported runtime in production without an approved, time-limited exception and segregation
Secure configuration	PHP and appliance baselines; management interfaces never internet-exposed
Change management	Pre-approved emergency changes for critical vulnerabilities
Supplier security	Contract clauses in section i; supplier criticality tiers
Logging and monitoring	Web, PHP, host and egress logs retained at least 90 days online to support look-backs like this one
Incident response and breach notification	Playbooks covering the 72-hour clocks, NCSC reporting and CIR provider engagement

RACI for the PHP2Shell response

R = Responsible, A = Accountable, C = Consulted, I = Informed.

Activity	Board / SIRO	CISO	CIO / Head of IT	IT ops / sysadmins	Service owners	Procurement / supplier mgr	DPO / Legal	SOC / IR
Declare emergency and approve change route	I	C	A	I	I	I	I	C
Build and maintain PHP inventory	I	C	A	R	R	C	I	C
Patch, mitigate or isolate	I	C	A	R	R	I	I	I
Accept downtime or residual risk	A (if material)	C	C	I	R	I	C	C
Chase appliance vendors and suppliers	I	C	A	C	C	R	I	I
Compromise assessment	I	A	C	C	C	I	I	R
Breach and regulatory notification decisions	I	C	I	I	C	I	A / R	C

Activity	Board / SIRO	CISO	CIO / Head of IT	IT ops / sysadmins	Service owners	Procurement / supplier mgr	DPO / Legal	SOC / IR
Board reporting	I	R	A	I	I	I	C	C
Lessons learned and prevention programme	I	A	R	C	C	C	C	C

KPIs

KPI	Target
PHP-bearing assets in the register with a named owner	100%
Authenticated scan coverage of the register	≥ 95%
Internet-facing PHP assets fixed (verified running), mitigated or offline	100% within 72 hours
All in-scope assets fixed	100% within 14 days
Median and 90th-percentile time to remediate, by tier	Within tier targets
EOL PHP runtimes in production	Falling to zero, with dated plans
Appliances awaiting vendor fix, and days open	Tracked daily; each with a compensating control
Suppliers that have confirmed status in writing	100% of critical suppliers within 7 days
Compromise assessments completed for exposed systems	100%
Risk acceptances past expiry	Zero

Board reporting

Daily during the emergency, then weekly until closure, then folded into monthly risk reporting. One page: exposure (assets by tier and status), progress against KPIs, assets awaiting vendors, compromise findings, residual risks and decisions needed. NCSC's cyber governance for boards material is a good frame for the board's role.

Auditing compliance and holding owners and suppliers to account

- Second-line or internal audit re-verifies a random sample of "closed" assets using the section h evidence pack.
- Any SLA miss is escalated to the owning executive within one working day; repeated misses go to the risk committee.
- Exceptions live in one register with owner, compensating control and expiry; expired exceptions are reported to the board.
- Supplier responses are scored (speed, completeness, evidence) and fed into renewals; non-responders are escalated under contract.

Regulatory and framework links

Framework	What PHP2Shell touches	What to do
Cyber Essentials (v3.3, Danzell, from 27 Apr 2026)	Critical and high-risk updates within 14 days, firmware included; unsupported software in scope is a fail	Patch or remove within 14 days; segregate any EOL PHP out of scope or retire it. Record evidence for your next assessment

Framework	What PHP2Shell touches	What to do
ISO/IEC 27001:2022 Annex A	5.9 inventory; 5.19–5.23 supplier and cloud security; 5.24–5.28 incident management and evidence; 8.8 technical vulnerabilities; 8.9 configuration; 8.15–8.16 logging and monitoring; 8.20–8.22 network security and segregation; 8.32 change management	Use this event as a nonconformity check for each control; record corrective actions
NIS Regulations 2018	Operators of essential services report NIS incidents to their competent authority, and relevant digital service providers to the ICO, without undue delay and within 72 hours where thresholds are met	Check thresholds with your competent authority. Watch the Cyber Security and Resilience (NIS) Bill, still in the House of Lords at September 2026 with Royal Assent expected late 2026 to spring 2027, which is expected to widen scope and tighten reporting
UK GDPR	Art. 33: notify the ICO within 72 hours of becoming aware of a personal data breach unless it is unlikely to result in risk. Art. 34: tell individuals without undue delay if high risk. Art. 33(5): document every breach, reported or not	Memory disclosure or RCE on a system handling personal data can be a confidentiality breach. Bring in the DPO as soon as compromise is suspected, not confirmed. ICO breach reporting
NCSC	Alert of 21 Oct sets UK expectations: inventory, patch, mitigate, consider offline, assess for compromise	Cite your alignment with the NCSC actions in board and regulator communications

K

EVERYONE

Checklists

Tick these off in order; each list assumes the one above it is running.

Sysadmins and engineers

- [ ] List every PHP runtime on each host: CLI, mod_php, every FPM pool, panel builds under `/opt`, LiteSpeed, FrankenPHP, containers (section e, step 1)
- [ ] Record exact package versions and the repository each came from
- [ ] Preserve logs from 17 October onwards before changing anything; hash the exports
- [ ] Install the fixed package from the same source (8.2.35 / 8.3.36 / 8.4.27 / 8.5.12 or your vendor's fixed build)
- [ ] Restart properly: FPM per version; **full** Apache restart for mod_php; `pkill lsphp` for LiteSpeed; recreate containers; restart queue workers
- [ ] Run the `/proc/*/maps` deleted-library check or `needrestart`; output must be empty
- [ ] Where you cannot patch today: apply section f settings in the pool or server config, restart, confirm with `php-fpm -tt`, and restrict access
- [ ] Lock the origin to your CDN or WAF; confirm admin interfaces are unreachable from outside
- [ ] Rebuild images with `--pull`; set `update: true` in `setup-php`; replace bundled binaries (FrankenPHP, Octane)
- [ ] Run the section g hunt; escalate any hit; do not delete suspicious files
- [ ] After patching: watch for exit code 137 and the SimpleXML crash; restore a real `memory_limit`
- [ ] File the evidence pack (section h) and update the register

IT managers

- [ ] Open a single incident or programme ticket and a single PHP register; name an owner for every row
- [ ] Run external discovery, authenticated scans, EDR inventory and cloud and registry queries; reconcile them
- [ ] Add appliances and vendor products: firewalls, VPNs, NAS, monitoring, VoIP, panels; record vendor advisory IDs
- [ ] Assign each asset a tier (section d) and a due date; track daily
- [ ] Approve emergency changes; agree downtime with business owners for anything that cannot meet its tier
- [ ] Send the supplier request (template below) to every host, MSP and SaaS provider that runs PHP for you; chase every 48 hours
- [ ] Commission compromise assessments for every system exposed since 17 October; book an NCSC-assured CIR provider if capacity is short
- [ ] Approve secrets rotation for exposed internet-facing systems
- [ ] Report KPIs daily to the CISO and CIO
- [ ] Open dated migration tickets for every EOL PHP, targeting 8.4 or 8.5

Executives

- [ ] Name one accountable executive for the response
- [ ] Authorise emergency change and accept planned downtime where systems cannot be fixed fast
- [ ] Ask: "Do we have a complete list of everything running PHP, including appliances and suppliers, and who owns each item?"
- [ ] Ask: "How do we know the fix is running, not just installed?"
- [ ] Ask: "Which systems were exposed since 17 October, and has each had a compromise assessment?"
- [ ] Confirm the DPO and legal adviser are briefed on the 72-hour ICO clock and any NIS duties
- [ ] Fund migration off end-of-life PHP, including 8.2 before 31 December 2026
- [ ] Require written supplier confirmations and review supplier performance at renewal
- [ ] Receive the one-page report daily until internet-facing risk is closed, then weekly
- [ ] Commission a lessons-learned review within 30 days of closure

Supplier request template

```
Subject: [ACTION REQUIRED by <DATE>] PHP2Shell (CVE-2026-109618 / CVE-2026-109619) status for <SERVICE>

Please confirm in writing by <DATE>:
1. Whether any component of <SERVICE/PRODUCT> uses PHP, and which branch(es).
2. Whether it is affected by CVE-2026-109618 and CVE-2026-109619.
3. The fixed version or package deployed, and the date and time it was running in production.
4. Interim mitigations in place until then.
5. Whether you have found any evidence of exploitation or compromise affecting our data or service.
6. Your advisory reference, and the contact for follow-up.

This request supports our obligations under <contract clause / UK GDPR / NIS>.
```

Internal notification template

```
Subject: [ACTION REQUIRED by <DATE>] Critical PHP vulnerabilities (PHP2Shell): response needed

What happened: On 21 October 2026 PHP fixed two critical flaws (CVE-2026-109618, CVE-2026-109619)
that can let an attacker take control of a server running PHP without logging in. The NCSC has
asked UK organisations to act immediately. Exploitation attempts have been reported by several
security firms.

What this means for us: any system that runs PHP may be affected, including websites, internal
tools, firewalls, NAS and other appliances, and services suppliers run for us.

What we need from you by <DATE>:
1. The PHP version serving web traffic on each system you own (not just the command line).
2. Whether the fix is installed AND the service restarted, with the package version.
3. For appliances: the firmware version and the vendor's advisory status.
4. Anything unusual since 17 October: crashes, unknown files, new admin accounts. Do not delete
   anything; contact <SECURITY CONTACT> immediately.
5. For supplier-run services: forward the supplier's written confirmation.

Guidance: <LINK TO THIS GUIDE>. Questions: <CONTACT / TRACKER>.
```

REFERENCE

References

All sources accessed 24 October 2026. "Opened" means the full page was read; "Summary" means the fact was taken from the publisher's summary. Vendor and tracker pages change daily; check them for the latest status.

Primary: PHP project, governments, CERTs

Source	Date	Status
PHP Foundation: Today's PHP Security Release, and How We're Strengthening PHP Security	21 Oct 2026	Opened
php-src release 8.5.12 · 8.4.27 · 8.2.35	21 Oct 2026	Summary
PHP advisories GHSA-w8hw-w9fx-j79w (CVE-2026-109618) and GHSA-6wmf-4rv5-p3rp (CVE-2026-109619); php.net announcement of 21 Oct	21 Oct 2026	Via Rapid7, NCSC and SANS ISC; check the advisories directly for the latest wording
NCSC: Critical security issue affecting PHP	21 Oct 2026, updated	Opened
CERT-EU Security Advisory 2026-018	21 Oct 2026	Summary
Centre for Cybersecurity Belgium warning	22 Oct 2026	Summary
NCSC-NL php2shell software list (letters G, I, J, Q)	22–23 Oct 2026	Summary
CISA Known Exploited Vulnerabilities catalogue	Live	Check on the day

Distributions and package sources

Source	Date	Status
Debian security tracker CVE-2026-109618; DSA-6574-1	21–23 Oct 2026	Summary
Freexian ELTS tracker CVE-2026-109618	21–23 Oct 2026	Summary
Ubuntu USN-8948-1	21 Oct 2026	Via SANS ISC and ma.ttias.be
Red Hat CVE-2026-109618	21–23 Oct 2026	Via ma.ttias.be; check directly
SUSE CVE database	—	Status not confirmed in our research
Mattias Geniar: how to apply the patches (independent, tested package tables)	Updated 23 Oct 2026	Opened
Mattias Geniar: early mitigation steps	21 Oct 2026	Summary

Vendors and PSIRTs

Source	Date	Status
Help Net Security: what PHP2Shell means for Fortinet, Cisco and Palo Alto customers (summarises Palo Alto CVE-2026-109618, Fortinet FG-IR-26-186, Cisco cisco-sa-php-october26-Jj384Daq)	22 Oct 2026	Opened
Cisco advisory cisco-sa-php-october26-Jj384Daq	v1.4, 22 Oct 2026	Summary

Source	Date	Status
Netgate forum · Netgate blog	21–23 Oct 2026	Summary
Notebookcheck: QNAP, Nextcloud, OPNsense, pfSense status (cites Synology-SA-26:14 and OPNsense hotfixes)	24 Oct 2026	Summary
Cisco Community: Expressway CSCwx37062	23 Oct 2026	Summary (user forum)
SecurityWeek: tech and security companies respond	22–23 Oct 2026	Summary

Research and threat intelligence

Source	Date	Status
Rapid7 ETR: CVE-2026-109618 and CVE-2026-109619	21–22 Oct 2026	Opened
SANS ISC diary 33440	21 Oct 2026	Opened
Sansec: PHP2Shell	21–22 Oct 2026	Summary
watchTowr: PHP2Shell · FAQ	21–23 Oct 2026	Summary
Help Net Security: PHP2Shell patched (Amazon MadPot attempt on 17 Oct)	21 Oct 2026	Summary
Qualys TRU: Find, Mitigate, and Patch	20–24 Oct 2026	Summary
Datadog Security Labs	21 Oct 2026	Summary
VirusTotal: sizing the exposure window	22 Oct 2026	Summary
Brandefense (CISA alert and KEV status, VulnCheck private exploit)	24 Oct 2026	Summary
Threat Frontier (KEV not listed at 23 Oct)	23 Oct 2026	Summary

UK frameworks

Source	Status
NCSC Cyber Essentials overview and v3.3 summary via Logiq	Summary; read the v3.3 Requirements for IT Infrastructure directly
NCSC vulnerability management · preventing lateral movement · patch wave blog · Early Warning · CIR providers · cyber governance for boards	Linked from the NCSC alert
UK Parliament: Cyber Security and Resilience (Network and Information Systems) Bill	Summary; check for Royal Assent
ICO: report a breach; UK GDPR Arts. 33–34; NIS Regulations 2018; ISO/IEC 27001:2022	General knowledge; not re-researched for this event

PUBLICATION

Document control and disclaimer

Title	PHP2Shell (CVE-2026-109618 / CVE-2026-109619): Find, Fix, Hunt, Prevent
Publisher	IT Revolution (itrev.co.uk)
Version	1.0, 24 October 2026 (first publication)
Classification	TLP:CLEAR, may be shared freely with attribution
Next review	31 October 2026, or sooner on a material change: PAN-OS or Cisco fixes, a CISA KEV listing, a new PHP security release

Disclaimer. This guidance is provided in good faith to help organisations respond to CVE-2026-109618 and CVE-2026-109619. It reflects public information available on 24 October 2026 and IT Revolution's professional judgement where marked. It is not legal advice. Vendor and PHP project advisories take precedence over this document. Test commands and configuration changes before using them in production. IT Revolution accepts no liability for actions taken or not taken on the basis of this guidance.

Product and company names are trademarks of their owners and are used for identification only; no affiliation or endorsement is implied.

© 2026 IT Revolution · itrev.co.uk