

PHP2Shell: Interim Protection Where Patching Is Delayed

CVE-2026-109618 and CVE-2026-109619

22 October 2026 | Reference TA-2026-1022-001 | Supplements TA-2026-1021-001

1. Purpose of This Supplement

Our advisory TA-2026-1021-001 recommended upgrading all PHP runtimes to a fixed release. Where that cannot happen straight away, for example because of change freezes or vendor dependencies, this supplement sets out the temporary measures that reduce the risk, why each one helps and how to confirm it has worked. They reduce exposure to the attack methods demonstrated so far but do not repair the flaws in PHP, so they should stay in place for days, not weeks.

Permanent fix	Upgrade to PHP 8.5.12, 8.4.27, 8.3.36 or 8.2.35, or a vendor package listing both CVEs
This supplement	Temporary risk reduction only; not a substitute for patching
Restart required	Yes. Restart every PHP-FPM, Apache and long-running PHP worker after each change

Coordinate before applying. Apply these changes in consultation with the application development and support teams. Memory, request size and feature settings can cause availability issues or break dependencies such as uploads, imports, scheduled jobs and integrations. Test outside production and agree a rollback plan first.

2. Before You Start: Where Settings Must Go

Both flaws are triggered before the website's own code runs, so settings must already be active when a request arrives. Set them in `php.ini`, the PHP-FPM pool or the Apache configuration, not with `ini_set()` or `.user.ini`, and use the `php_admin_*` forms so applications such as WordPress cannot change them back:

```
; PHP-FPM: add to every pool file, e.g. /etc/php/8.x/fpm/pool.d/www.conf
php_admin_value[memory_limit]           = -1
php_admin_value[post_max_size]          = 8M      ; 32M if uploads are needed
php_admin_value[upload_max_filesize]    = 6M      ; keep below post_max_size
php_admin_flag[session.upload_progress.enabled] = off
php_admin_value[fatal_error_backtraces] = 0        ; PHP 8.5 only
# Apache mod_php (httpd.conf or <VirtualHost>, not .htaccess): php_admin_value memory_limit -1
```

How to confirm: after a full restart, check live values with a temporary page printing `ini_get()`, then delete it; `php -i` shows the command-line configuration, not the web server's.

3. Measures for CVE-2026-109618 (Memory Exhaustion)

This flaw is triggered when a large or complex request makes PHP run out of memory part-way through preparing it. The aim is that this never happens inside PHP: oversized requests are rejected first, and if memory does run out, the operating system stops the process cleanly.

Ref	Measure	What to Do	Why It Helps
1	Remove PHP's own memory cap Locked at server level	Set <code>memory_limit=-1</code> with <code>php_admin_value</code> in every pool or virtual host, and remove any finite limit set elsewhere (e.g. <code>WP_MEMORY_LIMIT</code>).	Removes the per-request memory limit used in confirmed exploits. Other allocation failures can still reach the vulnerable code, so Refs 2–4 remain essential.
2	Reduce the maximum request size PHP setting	Set <code>post_max_size</code> to 8M where no file uploads are needed, or 32M where they are. Keep <code>upload_max_filesize</code> below <code>post_max_size</code> .	Limits how much data one request can force PHP to process.
3	Reject oversized requests earlier Web server, proxy or WAF	Nginx <code>client_max_body_size</code> ; Apache <code>LimitRequestBody</code> , <code>LimitRequestFields</code> (headers only). Field, part and nesting limits need a WAF. Where an endpoint takes no uploads, block multipart/form-data at the edge.	Stops hostile requests before PHP sees them. Limits must also cover chunked requests with no declared length and the decoded size of compressed bodies.
4	Add an outer memory boundary Container or cgroup	Set a memory ceiling on the PHP service, e.g. <code>systemd MemoryMax=</code> with <code>OOMPolicy=continue</code> , or a container memory limit. Where possible, give each worker its own cgroup.	Replaces the cap removed in Ref 1. Normally only the affected worker is ended and PHP respawns it; test that the master process survives, especially with container limits.

Do not use RLIMIT_AS or RLIMIT_DATA limits. Limits of this type (`ulimit -v`, `ulimit -d`, or `systemd LimitAS=` and `LimitDATA=`) cause memory requests inside PHP to fail, which is exactly the condition the attack needs. Remove any that already apply to PHP services.

4. Measures for CVE-2026-109619 (Leftover Request Data)

Data from one request can survive and be misused by the next request on the same worker. The demonstrated attacks rely on PHP raising an error or warning at a sensitive moment, so the aim is to remove those triggers.

Ref	Measure	What to Do	Why It Helps
5	Turn off upload progress tracking All PHP versions	Set <code>session.upload_progress.enabled=0</code> at server level and restart every worker.	Closes the confirmed session-based attack routes. It does not cover every variant and has no effect on CVE-2026-109618.
6	Turn off fatal error backtraces PHP 8.5 only	Set <code>fatal_error_backtraces=0</code> at server level and restart every worker.	Removes a PHP 8.5 error-handling path identified by the PHP project as part of the mitigation.
7	Remove faulty per-site settings Configuration clean-up	Remove invalid or deprecated PHP settings in pool files, <code>.htaccess</code> and <code>.user.ini</code> . In the error log these appear as warnings attributed to Unknown on line 0.	Such settings raise warnings before a request is ready; hiding them with <code>display_errors=0</code> or <code>@</code> does not help.
8	Fix errors attackers can trigger Application	Review logs for PHP Fatal error and Maximum execution time on pages that need no login, including upload, import and clean-up functions.	Any error a remote user can cause is a possible trigger; fix or block it.

5. Monitoring and Response

5.1 What to Watch For

- Memory errors:** repeated `Allowed memory size ... exhausted` messages; after Ref 1 these should stop.
- Worker crashes:** PHP-FPM log entries such as `child ... exited on signal 11 (SIGSEGV)`, Apache Segmentation fault messages or kernel OOM kills.
- Unexpected programs:** alert when the web server account (`www-data`, `apache` or `nginx`) starts a shell, `curl`, `wget` or `python` (auditd `execve` rule on hosts without EDR).

5.2 If You See Signs of Attack

Treat an unpatched, internet-facing PHP server showing these signs as compromised: start incident response, isolate it if possible, and preserve logs, processes and memory before updating, as restarting PHP erases worker memory. Then rotate database credentials, API keys and session keys, and on Apache `mod_php` hosts, TLS private keys.

5.3 After the Upgrade Is Applied

- Restore a memory limit:** revert `memory_limit=-1` to a finite value sized to each application. The fix adds a hard memory limit that ends the process without running shutdown handlers, so test applications that rely on them for logging or clean-up.
- Keep and confirm:** retain edge limits and monitoring, set `expose_php=0`, and confirm the fix in the package changelog for both CVEs, as backports often keep the old version string.

5.4 Implementation Checklist

#	Check	Owner	Done
1	Change approved with application owners and a rollback plan agreed	App owner	☐
2	Settings applied at server level in every PHP-FPM pool and virtual host	Platform	☐
3	Workers restarted, live values confirmed; edge and cgroup limits tested	Platform	☐
4	Log and process alerts active for the indicators in Section 5.1	SOC	☐
5	Upgrade to a fixed PHP release scheduled, with a target date	App owner	☐

How NetSentries can support. Our team can verify these measures on each server, validate remediation once patched, and assess systems exposed before the update for compromise.

Sources: [PHP security advisory](#) | [GHSA-w8hw-w9fx-j79w](#) | [GHSA-6wmf-4rv5-p3rp](#) | [php2shell.com](#)

Based on public information as at 22 October 2026, for the recipient's internal use. Test configuration changes before production deployment.