

MERCHANTS, PROVIDERS & ISOS

PHP2Shell: what the new PHP flaws mean for anyone taking card payments

Published 22 October 2026

Two critical PHP flaws can let an unauthenticated attacker run code on a web server. For a merchant that is a risk to checkout integrity, payment credentials and uptime. For a provider it is a portfolio risk. What happened, what to check and what to ask your host.

On 21 October 2026 the PHP project released emergency fixes for two critical flaws in the language itself, tracked as CVE-2026-109618 and CVE-2026-109619 and nicknamed **PHP2Shell**. Both are rated 9.2 on CVSS 4.0. Either can allow an unauthenticated attacker to run code on an affected web server, and the flaws are in PHP, not in any one application built on it.

For merchants that creates a risk to checkout integrity, payment credentials and availability. For payment providers and ISOs it creates potential exposure across their merchant portfolios. Here is what happened, what to check and what to ask your hosting provider.

PHP runs a very large share of the web, including most of the shop software small and mid-size merchants use. That makes this a payments story as much as a security one.

What was found

- **CVE-2026-109618** is a memory corruption bug. If PHP runs out of memory while it is part-way through setting up a request, the clean-up afterwards can treat half-built data as valid.
- **CVE-2026-109619** is a stale-state, use-after-free bug. PHP can keep internal state pointing at request memory that has already been freed, and later processing may use it. PHP lists remote code execution and disclosure of sensitive memory as the effects.
- **Reach**. The earliest affected code goes back to PHP 5.4. The PHP project says every installation serving web requests is at risk, whether it runs PHP-FPM, Apache with mod_php, CGI, FrankenPHP or LiteSpeed. PHP assesses the risk as highest on mod_php, where the demonstrated attack can land before any of the site's own code runs. That describes the attack that has been shown, not a claim that other setups are safe.

- **Fixed versions.** PHP 8.5.12, 8.4.27, 8.3.36 and 8.2.35. Versions 7.4 to 8.1 have no official release, only minimal source backports. Versions 5.4 to 7.3 are end of life and will not be fixed by the PHP project at all. Some Linux distributions and long-term-support vendors do back-port fixes to older branches, so check your own vendor's security advisory before assuming a given installation is unpatched.

Where things stand

The details leaked on 20 October, a day before the planned release, which was itself brought forward from 22 October. Amazon's threat intelligence team reported an attempt against its honeypots on 17 October, four days before the public release, and more attempts since, apparently from several actors. Cloudflare and watchTowr both report attack traffic, and watchTowr says it has reproduced both flaws. Public exploit code exists, though its reliability is unverified, and the PHP project said at release that it had not identified a functional exploit. Whether the attempts seen so far would succeed against real production systems is not yet established.

This is a credible and urgent threat. Exploitation attempts and proof-of-concept code do not, by themselves, show that payment data has been stolen from merchants, and as of 22 October we are not aware of a confirmed payment-data compromise attributable to PHP2Shell. Patch affected systems now, investigate anything suspicious and keep the relevant logs. The absence of a confirmed breach is no reason to delay.

Why it matters for card payments

A merchant does not need to be running PHP code of its own to be affected. The software underneath the shop is enough, and the consequences reach the payment side in four ways.

1. **The server that builds the checkout page.** Code execution on that server lets an attacker change what the checkout page does. Injecting a script that copies card details as customers type them (often called skimming or Magecart-style attacks) is the long-standing way stolen card data leaves PHP shops.
2. **Credentials and keys.** Payment gateway API keys, webhook secrets and admin logins usually sit in the shop's configuration. Anyone who can run code on the server can read them.
3. **Fraud and abuse.** Stolen keys can be used to issue refunds or run test transactions. A compromised site can also host a second, hidden checkout, which looks to the provider very like [transaction laundering](#).
4. **Downtime.** Even an unsuccessful attack can crash workers. A shop that is down is not taking orders, and a recovery that takes days can break [subscription billing](#) cycles and drive refunds and disputes.

Merchants that redirect to a provider's hosted payment page, or embed the provider's own fields, are better placed: the card number is never on their server. They still have the admin, refund and API-key exposure above.

What to check this week

- **Find out which PHP you run.** On your own server, run `php -v`. On managed or shared hosting, look in the control panel or ask the host. Plugins and shop software do not report it, and a shop can sit on an old PHP version years after its software was updated.
- **Apply the vendor's fix.** For most Linux distributions and hosts that means the vendor's package update, followed by a restart of PHP so the change takes effect. Do not rely on the PHP version number alone, because distributions often back-port the fix without changing it.
- **If you are on PHP 7.3 or older, plan to move.** The PHP project will not fix these versions. Ask your host or developer whether their vendor back-ports a fix, and for the date, not the intention.
- **Mitigations are temporary.** The PHP project documents settings that narrow each flaw, and a web application firewall or network restrictions may reduce exposure while you prepare an update. None is a complete fix. Upgrade to a fixed release or apply your distribution's verified security patch as soon as you can, and if an internet-facing system cannot be patched promptly, consider restricting its exposure until it is.
- **Rotate secrets if you cannot rule out a compromise.** Payment gateway keys, webhook secrets, admin passwords and database credentials. Check for new admin users, changed checkout files and unexpected scripts on the payment page.
- **Look at the checkout.** Compare the scripts the payment page loads against what you expect. Unknown third-party domains are the warning sign.

PCI DSS and your provider

If you are in scope for PCI DSS, this falls within your existing vulnerability-management responsibilities. Requirement 6.3.1 requires you to identify new vulnerabilities and assign risk rankings. Requirement 6.3.3 requires critical or high-security patches, according to that ranking, to be installed within one month of release.

That is a maximum, not a recommendation to wait. Given the severity of PHP2Shell and the reports of exploitation activity, affected internet-facing systems should be remediated immediately. Where the checkout page is in scope, the requirements on payment-page scripts may also be relevant.

A merchant that cannot show it patched promptly, and has no record of why, is in a weak position if anything goes wrong. A short written note of when the flaw was identified, what was done and when is enough to begin with.

Tell your provider early if you have reason to think the shop was compromised. Payment providers may have reporting and escalation obligations under the applicable card scheme rules, and a merchant that raises it on the first day is dealt with very differently from one the provider finds through a scheme alert or a run of fraud reports.

For providers and ISOs

The exposure is across your portfolio and it is not evenly spread. Merchants on end-of-life PHP, on cheap shared hosting and on self-managed shop software are the ones to look at first. Reasonable steps are:

- Contact merchants on self-hosted shop software with a short, plain notice and a request for the date they patched.
- Watch for the signs of a compromised checkout: a sudden rise in card-testing, unexpected declines, refunds that do not match orders, or disputes reporting fraud on cards that were used only at that merchant.
- Re-run website monitoring on the highest-volume merchants and look for new scripts or unfamiliar payment destinations.
- Remember that the same flaw affects your own applications and the sites of your partners and affiliates.

At onboarding, one useful question is which platform and hosting the merchant runs, and who is responsible for keeping them current. The answer says a good deal about how an account will behave in a week like this one.

Sources

- [Tenable: PHP2Shell \(CVE-2026-109618, CVE-2026-109619\) frequently asked questions](#)
- [Qualys: PHP announces critical security release for two vulnerabilities](#)
- [Cisco Talos: Two critical RCEs in PHP core](#)
- [CERT-EU: Critical vulnerabilities in PHP \(2026-018\)](#)
- [watchTowr: PHP remote code execution exploited](#)

Read online: <https://payments-marketplace.com/insights/php2shell-payments/>

Payments Marketplace Ltd

Registered in England & Wales No. 17424288
Garden Ctg, Dereham Road, Dereham, Norfolk, NR19 2PW, UK
connect@payments-marketplace.com · +44 1366 727140