

PHP releases emergency patches for two critical PHP2Shell remote code execution flaws



PHP has issued emergency updates for two critical runtime flaws, together called PHP2Shell, that let an unauthenticated attacker run code on a web server. Exploit attempts have emerged, and only updating fully protects a server.

- Two critical flaws, jointly named PHP2Shell, allow an attacker without an account to run code on a web server.
- The flaws are in the PHP runtime itself, not in any application, so a site is exposed even if its own code is sound.
- Fixed versions are PHP 8.5.12, 8.4.27, 8.3.36 and 8.2.35.
- The PHP team states that only updating fully protects a server.
- Exploit attempts have been reported.

PHP has released emergency security updates that address two critical flaws in the language itself. Together the flaws are called PHP2Shell, and they allow an attacker who has no account to execute code on a web server.

What is affected

The weaknesses sit in the PHP runtime rather than in any particular application. This means a website can be exposed even when its own code is well written and maintained. The fixes are included in PHP 8.5.12, 8.4.27, 8.3.36 and 8.2.35. The PHP team says that updating is the only measure that fully protects a server.

Exploit attempts have already emerged, so the window for unpatched systems is a practical concern and not only a theoretical one.

What to do next

Institutions running PHP-based websites, portals or internal tools should first identify where PHP is deployed, including on third-party hosting and in vendor-managed applications. They should then move each system to the fixed release that matches its current branch and confirm the change has taken effect.

The source does not describe interim mitigations or the technical conditions of exploitation, so teams should consult the official PHP release notes for further detail. Where patching must wait for a change window, treat the system as at elevated risk and increase monitoring of the web servers concerned.

What to do now

1. Inventory all systems running PHP, including third-party hosted and vendor-managed applications.
2. Upgrade to PHP 8.5.12, 8.4.27, 8.3.36 or 8.2.35, matching the branch in use, and verify the version afterwards.
3. Prioritise internet-facing web servers for emergency change approval.
4. Increase monitoring of web server logs and behaviour for signs of exploit attempts until patching is complete.
5. Ask vendors and hosting providers to confirm in writing that their PHP environments are patched.