

Stand: 22. Oktober 2026, 15:44 Uhr

PHP2SHELL: KRITISCHE RCE-SCHWACHSTELLEN IN PHP

CVE-2026-109618 & CVE-2026-109619, CVSS 9.2

Es wurden kritische Sicherheitslücken in PHP entdeckt, die Angreifenden die Ausführung beliebiger Befehle auf jedem Server (Remote-Code-Execution) ermöglichen, auf dem PHP läuft.¹

Betroffen sind alle Versionen ab einschließlich PHP 5 (also PHP 5, 7 und 8). Gefährdet sind alle PHP-Installationen, die Webanfragen bearbeiten, darunter PHP-FPM, Apache mit mod_php, php-cgi, FrankenPHP und LiteSpeed.

Die Schwachstellen werden als CVE-2026-109618² und CVE-2026-109619³ geführt und sind jeweils mit einem CVSS-v4.0-Score von 9.2 bewertet.

Offizielle Patches wurden für die Versionen PHP 8.2, 8.3, 8.4 und 8.5 am 21. Oktober 2026 gegen 16 (CEST) bereitgestellt. Für die älteren Versionen 7.4, 8.0 und 8.1 gibt es minimale Backports. Allerdings wird dringend geraten auf eine unterstützte Version zu aktualisieren, idealerweise PHP 8.3 oder neuer.

TECHNISCHE DETAILS

Die Schwachstelle CVE-2026-109618 betrifft die Speicherverwaltung von PHP. Wird das Speicherlimit erreicht, kann PHP unter bestimmten Vorbedingungen und beim Upload eines präparierten Formulars auf bereits freigegebenen Speicher zugreifen. Angreifende können so Speicher auslesen und manipulieren und Schadcode im Nutzerkontext des Webserver ausführen.

Die zweite Schwachstelle CVE-2026-109619 betrifft die Fehlerbehandlung in PHP. Hier können Angreifende unter ähnlichen Vorbedingungen wie bereits bei der ersten Schwachstelle mit mindestens zwei HTTP-Anfragen einen instabilen Zustand in der Fehlerbehandlung erzeugen und diesen anschließend ausnutzen.

Die Anfragen für die Ausnutzung beider Schwachstellen benötigen mitunter mehrere Tausend Anfragen für einen erfolgreichen Versuch.⁴

BETROFFENHEITSANALYSE

- Vollständige Inventarisierung aller Systeme und Anwendungen, die PHP verwenden. Das betrifft auch virtuelle Maschinen, Container-Images, Drittanbieter-Software, Managed Services, sowie vernetzte Geräte mit Weboberflächen⁴ wie bspw. Firewalls und NAS-Systeme.
- Ermittlung und Dokumentation der verwendeten exakten PHP-Versionen.
- Priorisierung derjenigen Instanzen, die aus dem Internet oder öffentlichen Netzen erreichbar sind.
- Triage der Instanzen hinsichtlich der Möglichkeit eines zeitnahen Patches.

¹ <https://www.php.net/archive/2026.php#2026-10-21-5>

² <https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>

³ <https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>

⁴ <https://www.heise.de/news/php2shell-Kritische-Sicherheitsluecken-erlauben-Codeeinschleusung-in-PHP-11492938.html>

MITIGATIONSMAßNAHMEN

Für aktualisierbare Systeme und Anwendungen:

Zur vollständigen Schließung der Schwachstellen ist ein zeitnahes Einspielen der Patches unumgänglich. Dabei sollte wie folgt vorgegangen werden:

- Durchführung von Update-Tests in einer Staging-Umgebung, sofern zeitnah umsetzbar.
- Zeitnahes Einspielen der offiziellen PHP-Sicherheitsupdates (ab 8.2.35, 8.3.36, 8.4.27 und 8.5.12).
- Neustart der betroffenen Dienste.
- Verifizierung der aktualisierten Versionsnummern auf allen Instanzen.
- Kontrolle der Systeme auf ihre Integrität und Anzeichen einer Kompromittierung.

Für nicht-aktualisierbare Systeme und Anwendungen (Workarounds):

- Organisatorische Vorbereitung auf Installation der Patches

Falls die betroffenen Dienste nicht betriebsnotwendig, aber aus öffentlichen Netzen erreichbar sind, sollten diese nach Möglichkeit über die Firewall isoliert oder sogar abgeschaltet werden.

Die folgenden Maßnahmen können alternativ als temporärer Workaround verwendet werden, falls ein Einspielen der Patches nicht zeitnah umsetzbar und eine Isolierung bzw. Abschaltung nicht möglich ist. Die Maßnahmen können negative Auswirkungen auf Geschwindigkeit oder Funktionalität von PHP haben und ersetzen keinesfalls eine zeitnahe Aktualisierung auf die Patch-Versionen, insbesondere stellen sie keinen umfassenden Schutz gegen die Schwachstellen dar.

Für CVE-2026-109618:

- `memory_limit=-1` so setzen, dass der Wert nicht von Anwendungen überschrieben werden kann (z. B. über `php_admin_value`). Außerdem sicherstellen, dass Pool-, Verzeichnis- oder Anwendungseinstellungen den Wert nicht wieder zurücksetzen. Diese Einstellung behebt den Fehler am Speicherlimit, andere Fehler bei der Speicherreservierung können aber weiterhin ausgenutzt werden.
- `post_max_size` auf einen niedrigen Wert setzen, idealerweise auf ein Viertel des Speicherlimits. Für Anwendungen, die keinen Dateiupload benötigen, sollte der Wert auf **8M** gesetzt werden. Für Anwendungen mit Dateiuploads auf **32M**. Der Wert von `upload_max_filesize` sollte kleiner als `post_max_size` sein. Auf Webserver- oder Proxy-Ebene sollen die folgenden Eigenschaften von Anfragen über ein Limit beschränkt werden: dekodierte Anfragegröße (*decoded request size*), Zahl der Eingabefelder (*input field count*), Multipart- und Header-Teile (*multipart part count* und *header count*), sowie die Verschachtelungstiefe (*nesting depth*). Diese Limits sollten auch für Anfragen ohne `Content-Length` Header gelten.
- `max_input_nesting_level=8` setzen, falls die Anwendung es erlaubt. Diese Einstellung gehört in `php.ini`, FPM-Pool oder eine Serverkonfiguration, die vor Verarbeitung einer Anfrage greift. `ini_set()` oder FPM `.user.ini` sind nicht ausreichend.
- Ein Container- oder cgroup-Limit als Verfügbarkeitsgrenze nutzen, idealerweise mit einer Richtlinie, die nur den betroffenen Worker beendet. Nicht durch `RLIMIT_AS` oder `RLIMIT_DATA` ersetzen, da diese selbst den Fehler bei der Speicherreservierung auslösen können.

Für CVE-2026-109619:

- `session.upload-progress.enabled=0` setzen und jeden Worker neustarten. Diese Einstellung gehört in `php.ini`, FPM-Pool oder eine Serverkonfiguration, die vor Verarbeitung einer Anfrage greift. `ini_set()` oder FPM `.user.ini` sind nicht ausreichend. Das blockiert die bekannten Angriffswege über Sessions, aber keine anderen Varianten.
- Für PHP 8.5, `fatal_error_backtraces=0` setzen und jeden Worker neustarten. Das blockiert ausschließlich den bekannten Angriffsweg über Backtraces.
- Ungültige oder veraltete Einstellungen entfernen, die pro Anfrage gesetzt werden und schon vor der Initialisierung Meldungen ausgeben. Die Meldungen per `error_reporting`, `display_errors` oder `@` auszublenden, hilft nicht.
- Fatale Fehler und Execution Timeouts beheben, die Angreifende auslösen können, auch in Callbacks, bei Uploads, Importen und beim Cleanup.
- Für PHP-FPM, `pm.max_requests` auf einen niedrigen Wert setzen, sodass Worker regelmäßig ersetzt werden. Das verkürzt die Zeit, in der ein veralteter Zustand in einem Worker bestehen bleiben kann. Ein Angriff, der innerhalb der Lebensdauer eines einzelnen Workers abläuft, wird dadurch nicht verhindert. Zudem erhöhen sehr niedrige Werte den Aufwand durch häufige Prozessneustarts.

Allgemein:

- Verwendung von SELinux oder AppArmor zur Limitierung von Aktionen bei einer erfolgreichen Ausnutzung der Schwachstellen.
- `expose_php=0` setzen, um das Vorhandensein von PHP zu verschleiern. Dies verhindert allerdings keine Ausnutzung der Schwachstellen.

WEITERE MAßNAHMEN

Trotz Bereitstellung von Patches zur Schließung der Schwachstellen, sollten weiterhin die Nachrichten zur aktuellen Entwicklung beobachtet werden, da es zu Anschlussfunden kommen könnte (derzeit in Evaluation).

COMPROMISE ASSESSMENT

Es gibt Anzeichen dafür, dass vor Veröffentlichung der Schwachstellen und der Patches, bereits aktiv versucht wurde die Sicherheitslücken auszunutzen. Erste Ausnutzungsversuche lassen sich bis zum 17. Oktober 2026 zurückverfolgen.⁵

Daher kann auch bei einem zeitnahen Einspielen der Patches sowie bereits zuvor getroffenen Sofortmaßnahmen zur temporären Behebung nicht ausgeschlossen werden, dass die Schwachstellen bereits ausgenutzt wurden.

Aufgrund der Art der Schwachstellen werden bei einem erfolgreichen Angriff nur wenig Hinweise auf eine Kompromittierung zurückgelassen. Dennoch sollten zur Feststellung von Ausnutzungsversuchen die folgenden Indikatoren herangezogen werden:

⁵ <https://aws.amazon.com/de/blogs/security/amazon-threat-intelligence-identifies-php2shell-exploitation-prior-to-public-disclosure-in-madpot/>

- Angreifende erhalten bei erfolgreicher Ausnutzung die Rechte des Nutzers des PHP-Prozessors, also des Webserver oder des PHP-FPM. Meistens sind das die Nutzer **http**, **www-data**, **www**, o. ä. – manchmal aber auch **root**.
- Suche nach ungewöhnlichen oder vor kurzem geänderten Dateien im Home-Verzeichnis des Nutzers.
- Prüfung der ausgeführten Befehle in **.history** und **.bash_history** im Home-Verzeichnis des Nutzers.
- Den Webserver-Bereich auf PHP-Dateien prüfen, die seit dem 17. Oktober 2026 angelegt oder modifiziert wurden.
- Die **journald**-Ausgaben sowie **error.log** der Webserver auf ungewöhnliche Fehlermeldungen prüfen. Insbesondere solche, die Speicherfehler (z. B. „Memory exhausted“) melden:
 - **Allowed memory size of ... exhausted**
 - **Unknown on line 0**
 - **Failed to decode session object**
- Prüfung der Prozessliste auf ungewöhnliche Child-Prozesse vom Webserver oder von PHP-Prozessen.
- Auf ungewöhnliche Listener-Ports prüfen (**netstat -tlnp** oder **ss -tlnp**).