

PHP2Shell (CVE-2026-109618/109619)

緊急対応策

認証なしでサーバーを乗っ取ることができるPHPの脆弱性、PHP2Shell。Xintが自ら再現・検証した緊急対応策と、侵害有無の確認方法をまとめました。

公開：2026年10月21日 | 最終更新：2026年10月21日

1. PHP2Shellの概要

PHPエンジン自体の脆弱性で、認証なしでサーバーに対するリモートコード実行や、プロセスメモリの読み取りが可能です。脆弱性の公表前に、PoCのデモ動画とパッチ適用済みのバイナリが流出し、それをもとにしたPoCが急速に広がっています。Xintチームが実際に攻撃コードを完成させてみたところ、他の攻撃者が同じコードを作るのも時間の問題だと判断しました。今すぐパッチを適用してください。

CVE	CVE-2026-109618、CVE-2026-109619
深刻度	CVSS v4.0 9.2 (Critical)
修正バージョン	8.2.35 / 8.3.36 / 8.4.27 / 8.5.12 (2026年10月21日 23:00 JST リリース)
影響を受けるバージョン	修正バージョン未満、およびPHP 8.1以下 (7.x、5.xを含む)
PHP 8.1以下	公式リリースはありません。ご利用のディストリビューション (Ubuntu、Debian、RHELなど) やLTSサポート事業者のセキュリティアップデートをご確認ください。
緩和策	PHP公式の見解は「完全な緩和策 (mitigation) はなく、アップデートが唯一の安全な選択肢」
すぐに実施できる対策	<code>mod_php</code> : <code>memory_limit = -1</code> 、 <code>session.auto_start = 0</code> PHP-FPM・CGI: 上記に加えて <code>session.upload_progress.enabled = 0</code> <code>auto_start</code> をオフにすると、 <code>session_start()</code> を呼び出さないページではセッション機能が動作しなくなる場合があります。変更前に3.2の確認手順を実施してください。
Xintの見解	独自の再現環境で実際にエクスプロイトが動作することを確認し、本書の対策を適用すると攻撃を緩和できることも確認しました。 PHP 8.1以下をご利用の場合は3.1をご参照ください。

2. 脆弱な環境の確認方法

パッチ未適用のPHPは、デフォルト設定のままだでも脆弱だとお考えください。設定の変更で防げるのは一部の攻撃経路だけで、同じ原因による新たな攻撃経路が今後見つかる可能性があります。

2.1 サーバー構成別の影響の有無

公式の分析とXintチームの再現結果をもとにまとめました。新たに確認された内容は随時更新します。(最終更新：2026年10月21日)

構成	判定	条件
Apache + mod_php (prefork)	脆弱	アプリケーションに関係なく、任意のPHPスクリプトでリモートコード実行が可能。セッション経由はPHP 5.4.0以降、リクエスト変数経由はPHP 7.3以降が対象
Apache + PHP-FPM (mod_proxy_fcgi)	脆弱	<code>session_start()</code> 、 <code>unserialize()</code> などのデコード処理が行われるアプリケーションでリモートコード実行が可能。WordPress、phpMyAdminなど主要なアプリケーションが該当
Nginx + PHP-FPM	脆弱	Apache + PHP-FPMと同じ。 <code>client_max_body_size</code> を大きくしている場合や、 <code>fastcgi_request_buffering off</code> を設定している場合は、攻撃がより容易になる可能性あり
Nginx + mod_php	該当なし	Nginxはmod_phpに対応していない

- 最初の報告者がリモートコード実行を確認したアプリケーション：WordPress、Drupal、Joomla、MediaWiki、Roundcube、Magento、phpMyAdmin、Nextcloud、Laravel、Symfony、WHMCS。これはデモの対象にすぎず、脆弱性はPHP自体にあります。
- リモートコード実行とは別に、プロセスメモリを読み取る脆弱性も公開されています。

2.2 設定確認スクリプト (php2shell_check_ja.php)

注意：コマンドライン (php -v、php -i) で確認しないでください

コマンドラインのPHPは、Webサービスとは別の設定ファイルを読み込みます。コマンドラインでは対策済みに見えても実際のWebには反映されていないことがあり、対策したつものまま脆弱な状態が続くことになります。

- ダウンロード：https://gist.github.com/xint-io/14630b00ed2ff90cdde3bb92162e316f#file-php2shell_check_ja-php
 - SHA-256：2fa23a2bfadb6beeeff815a4da872fcaca3f62caaaaff6912cc9715f543bc92e3
1. スクリプト内の `$TOKEN` の値を任意の文字列に変更します。変更しない場合は404のみを返します。
 2. 推測されにくいファイル名（例：chk-8f3a2c.php）で保存し、アプリのエントリーポイント（index.php）と同じディレクトリにアップロードします。
 3. ブラウザで `https://<ドメイン>/<ファイル名>.php?k=<TOKEN>` にアクセスします。
 4. 確認が終わったら、すぐに削除します。

サイト（vhost、PHP-FPMプール）が複数ある場合は、それぞれ確認する必要があります。結果は以下のように読み取ります。

出力項目	次の対応
PATCH = 未適用 / PATCH = 公式リリースなし	3.1 パッチ適用
MITIGATION = 未適用	パッチ適用までの間、3.2の暫定緩和策を適用
実行方式 (SAPI)	3.2の「適用場所」の表でご自身の環境を確認

<code>session.save_path</code> 、 <code>error_log</code>	3.4 侵害有無の確認で使うパス
---	------------------

注意：ディストリビューションのパッケージはバージョン番号が変わらないことがあります

Ubuntu、RHELなどのパッケージは、バージョン番号をそのままにしてセキュリティパッチだけをバックポートすることがあります。スクリプトが「未適用」と判定した場合も、ディストリビューションのセキュリティアドバイザーをあわせてご確認ください。

3. 対策

パッチの適用が唯一の根本対策です。公開されたバグのほかにも同じパターンの亜種があることが知られており、公式パッチはその原因自体を修正するため、亜種による攻撃もあわせて防ぐことができます。以下の暫定緩和策とWAFは、パッチを適用するまでの時間を稼ぐ手段とお考えください。

3.1 (P0) パッチ適用

- PHP 8.2～8.5は、修正バージョン（8.2.35 / 8.3.36 / 8.4.27 / 8.5.12）にアップデートします。OS・ディストリビューション別のアップデート方法は、PHP公式のセキュリティアドバイザーにまとめられています。
- アップデート後は、必ずApacheまたはPHP-FPMを再起動してください。再起動しないと、旧バージョンが使われ続けます。
- コンテナ環境では、ベースイメージを再ビルドして再デプロイします。
- 2.2のスクリプトで、バージョンを再確認します。

PHP 8.1以下には、PHP公式のリリースがありません。ディストリビューションやLTSサポート事業者がセキュリティアップデートを提供しているかを、まずご確認ください。提供されていない場合はサポート対象バージョンへのアップグレードが原則で、それまでは3.2の暫定緩和策を適用してください。8.2も2026年末にセキュリティサポートが終了するため、8.3以上へのアップグレードをおすすめします。

3.2 (P1) パッチ適用までの暫定緩和策（部分的な緩和）

Califが示した暫定緩和策に、Xintの再現結果を加えました。一部の攻撃経路だけを防ぐもので、亜種までは防げません。推奨設定は実行方式（2.2スクリプトのSAPIの値）によって異なります。

実行方式	Califの推奨	Xintの追加推奨
mod_php (apache2handler)	<code>memory_limit = -1</code>	<code>session.auto_start = 0</code>
PHP-FPM (fpm-fcgi)、 CGI/FastCGI (cgi-fcgi)	<code>session.upload_progress.enabled = 0</code> + <code>memory_limit = -1</code>	<code>session.auto_start = 0</code>

auto_startもオフにする理由

Xintチームの再現では、`session.auto_start = 0`のサーバーは、`session.upload_progress.enabled`をオフにしても脆弱であることが分かりました。セッションを自動で開始する設定のため、攻撃に必要なセッション処理がすべてのリクエストで行われるからです。

注意：memory_limit = -1 はPHPのメモリ上限をなくします

1つのリクエストがサーバーのメモリを使い切り、サービス停止（DoS）につながるおそれがあります。必ずOSレベルのメモリ上限をあわせて設定してください。

- コンテナ：メモリ制限（`--memory`、Kubernetesの `resources.limits.memory`）
- systemd：サービス単位の `MemoryMax`
- PHP-FPM： `pm.max_children` をサーバーのメモリに合わせて調整
- Apache： `MaxRequestWorkers` をサーバーのメモリに合わせて調整

注意：auto_startをオフにする前にご確認ください

コード内で `session_start()` を呼ばずに `$_SESSION` を使っている場合、`auto_start`をオフにした時点で、ログインやカートなどのセッション機能が停止します。

```
# セッションを使っているファイル
grep -rln '\$_SESSION' /var/www

# session_start() を直接呼び出しているファイル
grep -rln 'session_start' /var/www
```

共通のincludeファイル（config、headerなど）にすでに `session_start()` があれば、`auto_start`はすぐにオフにして構いません。ない場合は、共通のincludeファイルの先頭に `session_start();` を追加してから、`auto_start`を0にしてください。変更後は、ログインとセッション機能を必ずテストしてください。もともとオフ（デフォルトはOff）のサーバーでは、対応は不要です。`ob_start()` 系を使っている場合は、<https://www.php.net/manual/ja/function.session-start.php> を参考に修正してください。

`session.upload_progress.enabled = 0` を適用すると、サーバーに進捗を問い合わせるアップロードの進捗を表示する機能は、動作しなくなる場合があります。ブラウザ（JavaScript）側で進捗を計算する方式には影響しません。

適用場所

SAPI	適用場所	反映タイミング
apache2handler (mod_php)	Apacheの設定（vhostなど）に <code>php_admin_value memory_limit -1</code> 、 <code>php_admin_flag session.auto_start off</code>	Apache再起動
fpm-fcgi (PHP-FPM)	プール設定に <code>php_admin_value[memory_limit] = -1</code> 、 <code>php_admin_flag[session.upload_progress.enabled] = off</code> 、 <code>php_admin_flag[session.auto_start] = off</code>	PHP-FPM再起動
cgi-fcgi (CGI/FastCGI)	<code>php.ini</code> またはWebルートの <code>.user.ini</code>	<code>.user.ini</code> は最大5分で反映
コンテナ	イメージの <code>conf.d</code> にiniファイルを追加	再ビルド後に再デプロイ

- 可能であれば、`php_admin_value` / `php_admin_flag` で設定してください。アプリケーションがコード内の `ini_set()` で `memory_limit` を再び下げると緩和効果がなくなりますが、admin設定はコードから変更できま

せん。

- 共用サーバー（レンタルサーバー）：直接変更できない場合や、変更してもスクリプトの結果が変わらない場合は、事業者が設定の変更を制限している可能性があります。事業者に、PHPのアップデート予定と暫定緩和策の適用を依頼してください。
- 適用後は、2.2のスクリプトで再確認してください。

3.3 (P1) WAF・Webサーバー設定（暫定緩和策）

WAF

- ご利用中のWAFやCDN事業者のマネージドルールを、まずご確認ください。Cloudflare、Akamai、AWSなど主要な事業者が、今回の脆弱性向けのルールをすでに提供しています。ご自身のアカウントに実際に適用されているかは、事業者のお知らせや管理コンソールで確認する必要があります。
- 独自にルールを作成する場合は、以下のパターンをブロックします。
 - リクエストボディに `PHP_SESSION_UPLOAD_PROGRESS` という文字列が含まれる場合。サーバーで `session.upload_progress.name` を変更している場合は、変更後の名前でルールを作成してください。
 - 角括弧が深くネストしたフィールド名（`a[b][c][d]...`）
 - `filename=""` の空のアップロードパートを大量に含むmultipartリクエスト

注意：WAFの検知は容易に回避されます

charsetの変形、大容量データ、RFC 2231/5987エンコーディングなどの手法で、WAFの検知を回避するのは非常に容易です。ルールだけでは完全に防げないため、パッチ適用までの暫定措置とお考えください。

Apache

Apacheは、長さ情報のないchunked HTTP/1.1やHTTP/2のmultipartリクエストを、そのままPHPに渡します。PHPの `post_max_size` のチェックは宣言された長さしか見ないため、この経路でチェックを回避される可能性があります。`LimitRequestBody` で、リクエストボディのサイズを当該Webサービスに必要な分だけ許可してください。

Nginx + PHP-FPM

- `client_max_body_size` はWebサービスに必要な最小値にし、大きなアップロードが必要なパスだけ `location` 単位で引き上げます。
- `fastcgi_request_buffering` は、デフォルト値（on）のままにします。

3.4 侵害有無の確認

パッチの適用とあわせて、すぐに確認してください。この脆弱性はアプリケーションより下の層で動作するため、アプリケーションログにはほとんど痕跡が残りません。PHPとWebサーバーのログを確認する必要があります。攻撃が失敗しても、痕跡は残ります。

① 異常なメモリ超過ログ

正常なメモリ超過エラーには、PHPのファイル名と行番号が記録されます。`in Unknown on line 0` と記録されている場合は、リクエストを処理する前の段階で発生したもので、攻撃の可能性が高いといえます。主にmod_phpへの攻撃で見られます。

```
grep -cE 'Allowed memory size of [0-9]+ bytes exhausted.*in Unknown on line 0' \
/var/log/php_errors.log /var/log/apache2/error.log /var/log/httpd/error_log \
/var/log/php*-fpm.log 2>/dev/null
```

PHP-FPMへの攻撃では、アプリケーションの実行中にメモリ超過が発生するため、実際のファイル名が記録されます。セッションや `unserialize()` に関連する箇所でもメモリ超過が繰り返されている場合も、あわせて疑ってください。

② クラッシュとワーカーの異常終了ログ

```
grep -E "Segmentation fault|signal 11|SIGSEGV" \
/var/log/apache2/error.log /var/log/httpd/error_log /var/log/php*-fpm.log 2>/dev/null
dmesg | grep -i segfault | grep -E "php|httpd|apache2"
```

③ リクエスト・レスポンスの異常

- 異常なmultipartリクエストの直後に、メモリ超過エラーやHTTP 500が繰り返し発生している
- レスポンスの `PHPSESSID` クッキーが異常に長い、または壊れた値になっている（メモリ読み取り攻撃の痕跡。WAFやプロキシのログで確認）
- 原因不明のPHPワーカーの再起動が急増している

④ 異常に大きなセッションファイル

セッションの保存パスには、2.2スクリプトの `session.save_path` の値を使ってください。

```
find <セッション保存パス> -name 'sess_*' -size +1M -exec ls -lh {} \;
```

セッションファイルは、デフォルト設定では24分（`session.gc_maxlifetime = 1440` 秒）経過すると削除の対象になります。攻撃から時間が経つとこの方法では確認が難しくなるため、できるだけ早く確認してください。

⑤ 侵害後の痕跡

```
# Webサーバーのアカウントで実行中のプロセス（アカウント名は環境に合わせて：www-data、apache、nginx など）
ps -u www-data -o pid,ppid,lstart,cmd

# 直近3日以内に変更されたPHPファイル（Webシェル・改ざんの確認）
find /var/www -name '*.php' -mtime -3 -ls

# Webサーバー・PHPプロセスの外部接続
ss -tnp | grep -E "php|httpd|apache2|nginx"
```

痕跡が見つかった場合は、そのサーバーを外部インターネットから隔離し、ログを保全してください。メモリが読み取られた可能性があるため、DBパスワード、APIキー、セッション・暗号鍵などの機密情報も変更したうえで、Xintまでご連絡ください。

4. 参考

- PHP公式セキュリティアドバイザリ：<https://www.php.net/archive/2026.php#2026-10-21-5>

- PHP Foundationの解説 : <https://thephp.foundation/blog/2026/10/21/todays-php-security-release-and-how-we-re-strengthening-php-security/>
- GitHubセキュリティアドバイザリ CVE-2026-109618 : <https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>
- GitHubセキュリティアドバイザリ CVE-2026-109619 : <https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>
- Califの技術分析 : <https://calif.io/php2shell/>
- Wizの技術分析 : <https://www.wiz.io/blog/php2shell-php-rce>
- OpenAIの協調開示の発表 : <https://openai.com/index/coordinated-disclosure-of-critical-vulnerabilities/>

お問い合わせ : contact@xint.io | Webサイト : xint.io