

PHP2Shell: что случилось 21 октября 2026 года и как проверить свой сервер

Интернет · 22.10.2026 · PDF-версия статьи с сайта cms.dolinais.ru

Две уязвимости в интерпретаторе PHP с оценкой 9.2 по CVSS 4.0 позволяют выполнить код на сервере без авторизации. Разбираем механику, зону риска и порядок действий — без паники и с проверяемыми шагами.

21 октября 2026 года команда PHP выпустила внеплановое обновление, которое касается практически любого сайта на этом языке. Причина — две уязвимости в самом интерпретаторе, объединённые общим названием **PHP2Shell** и оценённые в **9.2 из 10 по CVSS 4.0**. Обе позволяют выполнить произвольный код на сервере, не имея учётной записи и не эксплуатируя ошибку в конкретном приложении.

Это редкий случай, когда проблема находится не в плагине, не в теме и не в библиотеке, а в фундаменте, на котором работает примерно три четверти сайтов. Именно поэтому новость вызвала не обычный цикл «обновитесь при возможности», а экстренные меры: в Нидерландах временно отключили портал парламента и государственные сервисы, в Финляндии приостановили часть правительственных платформ, во Франции закрыли административные сайты местных органов власти, а Фонд Wikimedia вывел из работы вспомогательные сервисы для разработчиков. Во всех случаях речь шла о предосторожности, а не о подтверждённых взломах — но масштаб реакции хорошо показывает, насколько серьёзно отнеслись к ситуации.

Коротко

- **Что:** две критические уязвимости в PHP — CVE-2026-109618 и CVE-2026-109619.
- **Чем опасны:** удалённое выполнение кода без авторизации, чтение памяти процесса (пароли, токены), потенциальное развитие атаки внутри инфраструктуры.
- **Кого касается:** все, у кого PHP обслуживает веб-запросы. Наибольший риск — Apache с mod_php.
- **Что делать:** обновить PHP до 8.5.12, 8.4.27, 8.3.36 или 8.2.35. Обновление CMS, плагинов и шаблонов проблему не решает.
- **Есть ли обход настройками:** полноценного нет. Отдельные параметры снижают риск, но не закрывают уязвимости.

Что именно нашли

Обе уязвимости относятся к классу «сбой в управлении памятью», но проявляются по-разному.

CVE-2026-109618 — ошибка при нехватке памяти. PHP использует механизм аварийного выхода из текущего контекста, когда не может выделить память. В некоторых ситуациях запрос на этом этапе ещё не полностью инициализирован. Если выделение памяти срывается в такой момент, интерпретатор остаётся с некорректным внутренним состоянием, а процедура завершения запроса затем читает или интерпретирует эти данные как валидные. Исследователи показали, что несколькими способами можно довести сервер до нужного состояния через специально сформированный HTTP-запрос.

CVE-2026-109619 — обращение к уже освобождённой памяти между запросами. PHP-процесс живёт дольше одного запроса и хранит часть состояния между ними. Обычно это состояние сбрасывается перед обработкой следующего запроса. Но при определённых сценариях — например, после аварийного завершения — в глобальных структурах остаются ссылки на память, которая уже освобождена. Следующий запрос, обработанный тем же процессом, может использовать эти ссылки до того, как произойдёт нормальная инициализация. Здесь есть важная деталь: успешная атака требует пары «запись состояния» и «чтение состояния», причём оба запроса должны попасть в один и тот же рабочий процесс.

Почему это не история про WordPress

Самое неприятное в этой ситуации — уровень, на котором находится ошибка. WordPress, Drupal, Joomla, Magento, Bitrix и любая самописная система работают поверх PHP. Если уязвим сам интерпретатор, то качество кода приложения не имеет значения: атака может сработать до того, как выполнится первая строка вашего скрипта.

Именно поэтому совет «обновите CMS» здесь бесполезен. Проверять нужно **версию PHP на сервере**, а не версию движка сайта.

Кто в зоне риска

Разработчики PHP перечислили несколько вариантов интеграции, и риск для них различается.

- **Apache с mod_php** — самый опасный сценарий. Модуль вызывает процедуру завершения запроса даже при неудачном старте, поэтому атака может сработать полностью до выполнения кода приложения. Для этого варианта существует подтверждённый рабочий эксплойт.
- **PHP-FPM** — уязвим, но эксплуатация сложнее: при неудачном старте дочерний процесс завершается и заменяется новым. Тем не менее сценарий через обработку прогресса загрузки файлов был воспроизведён и на FPM.
- **php-cgi, FrankenPHP, LiteSpeed** — потенциально уязвимы; полнота подтверждённых сценариев зависит от конфигурации.

Отдельно стоит упомянуть особенность, которая делает такие ошибки особенно неприятными для администраторов: атака не оставляет очевидных следов в логах приложения. Если выполнение кода происходит на уровне интерпретатора, запись в журнале доступа может выглядеть как обычный запрос.

Как это проверяли

Уязвимости обнаружила команда Wiz Research, причём часть работы выполнялась с помощью инструментов анализа кода на базе ИИ. Первые проблемы нашли в середине августа 2026 года, через несколько дней получили выполнение кода на тестовых установках Drupal и WordPress, а 24 августа передали сведения разработчикам PHP. По оценке исследователей, некоторые уязвимые участки кода существовали больше десяти лет — но для построения надёжной атаки требовалась сложная последовательность действий.

Первоначально раскрытие планировали на 22 октября, но 20 октября информация появилась в открытом доступе, и команда PHP перенесла выпуск исправлений на 21 октября. Фактически обновления стали доступны немного раньше назначенного времени.

Как проверить свой сервер

Начните с версии интерпретатора. В командной строке это делается так:

```
php -v
```

Если PHP работает в контейнере, проверять нужно именно контейнер, а не хост:

```
docker exec php-fpm php -v  
docker exec php-fpm php -m
```

Веб-сервер удобно проверять по заголовку ответа — он показывает версию, которую реально использует сайт:

```
curl -sI https://example.com | grep -i x-powered-by
```

Важно проверить **все** окружения: веб-пул, фоновые обработчики, cron и CLI-скрипты. Частая ошибка — обновить контейнер, который обслуживает сайт, и забыть про контейнер с очередями или про отдельный CLI-интерпретатор на хосте.

Что обновить

Исправленные версии вышли одновременно для всех поддерживаемых веток. Достаточно перейти на ближайшую из них:

- PHP 8.5 → **8.5.12**
- PHP 8.4 → **8.4.27**
- PHP 8.3 → **8.3.36**
- PHP 8.2 → **8.2.35** (ветка доходит до конца поддержки 31 декабря 2026 года)

Уязвимыми признаны выпуски начиная с PHP 5.4. Ветки 8.1 и старше больше не получают обновлений от проекта PHP, но дистрибутивы с long-term support выпускают собственные патчи — например, обновления для Ubuntu вышли в рамках действующих программ поддержки. Если вы собираете PHP из исходников, для 7.4, 8.0 и 8.1 опубликованы минимальные backport-патчи.

Если PHP установлен из пакетов дистрибутива, обновляйте штатным пакетным менеджером и проверяйте, что перезапущены все рабочие процессы:

```
apt-get update && apt-get install --only-upgrade php8.3-fpm  
systemctl restart php8.3-fpm
```

Для контейнеров этого недостаточно: нужно пересобрать образ на новой базовой версии, потому что расширения, установленные вручную, при простой замене контейнера на чистый образ пропадут.

Если обновиться прямо сейчас нельзя

Разработчики PHP прямо говорят: полной конфигурационной защиты не существует, обновление — единственное надёжное решение. Но снизить риск на время можно.

- **Отключите обработку прогресса загрузки файлов** — параметр `session.upload_progress.enabled=0` в конфигурации PHP или пула FPM, с обязательным перезапуском всех процессов.
- **Уменьшите лимиты запросов:** `post_max_size`, `upload_max_filesize`, `max_file_uploads`, `max_input_vars`, `max_input_nesting_level`. Дополнительно ограничьте размер тела запроса на уровне веб-сервера или прокси.
- **Заблокируйте multipart-запросы** на публичных адресах, где загрузка файлов не предусмотрена.
- **Ограничьте число запросов на процесс** (`pm.max_requests=1`) — это разрывает сценарий «один процесс обрабатывает несколько запросов подряд», хотя и снижает производительность.
- **Уберите диагностику, которая возникает до инициализации запроса:** устаревшие и конфликтующие директивы в конфигурации, ошибки в путях, недопустимые значения параметров.

Отдельно подчеркну: `memory_limit=-1` действительно упоминается в рекомендациях как способ убрать один из сценариев, но применять его на общем сервере без внешнего ограничения памяти опасно — вместо аккуратного отказа вы получите падение всего контейнера.

Что сделать после обновления

Обновление закрывает уязвимость, но не отменяет гигиену. Разумный набор шагов на первые сутки:

- проверить, что все окружения действительно показывают новую версию;
- просмотреть журналы веб-сервера за 20–22 октября на подозрительные multipart-запросы, повторяющиеся ошибки 500 и запросы без корректного Content-Length;
- проверить каталоги загрузок на появление новых исполняемых файлов;
- при малейших признаках компрометации — сменить пароли и ключи, которые могли находиться в памяти процесса;
- убедиться, что база данных и служебные порты недоступны из интернета.

Отдельно стоит проверить конфигурацию: если у вас Apache с `mod_php` только потому, что «так исторически сложилось», это хороший повод перейти на PHP-FPM. Помимо эксплуатационных плюсов, для этой уязвимости FPM оказался заметно менее удобной мишенью.

Чек-лист

- Узнать фактические версии PHP во всех окружениях: веб, очереди, cron, CLI.
- Обновить до 8.5.12 / 8.4.27 / 8.3.36 / 8.2.35.
- Пересобрать образы контейнеров, сохранив набор расширений.
- Перезапустить все рабочие процессы, а не только веб-пул.
- Проверить версию через заголовок ответа сайта.
- Отключить `session.upload_progress.enabled`, если функция не используется.

- Ужесточить лимиты размера и структуры запросов.
- Просмотреть логи за период с 21 октября.
- Запланировать вывод из эксплуатации PHP 5.6 и 7.4, если они ещё где-то работают.

Вывод

PHP2Shell — напоминание о том, что безопасность приложения начинается ниже уровня приложения. Можно аккуратно писать код, использовать современный фреймворк и следовать практикам безопасной разработки, но ошибка в интерпретаторе всё равно поставит под удар весь сервер.

Хорошая новость в том, что исправления уже вышли, а сама процедура обновления предсказуема. Плохая — в том, что обновление PHP часто откладывают: оно затрагивает инфраструктуру, требует пересборки образов и проверки расширений. Именно поэтому такие задачи стоит держать в постоянном графике, а не вспоминать о них в день публикации очередного CVE.

Источники: официальный [анонс безопасности PHP](#), рекомендации [GHSA-w8hw-w9fx-j79w](#) и [GHSA-6wmf-4rv5-p3rp](#), исследование [Wiz Research](#), обзор [SecurityLab](#).

Источник: <https://cms.dolnais.ru/account/articles/1/98>