

SPECIAL TOPIC · LIVE VULNERABILITY

PHP2Shell

CVE-2026-109618 & CVE-2026-109619

The bug you already know how to exploit — this time in the runtime
behind a fifth of the web

It started with a 6 a.m. phone call

October 21, 2026.

"Patch every PHP server. Now. Two critical RCEs, no authentication."

Then the real question:

How many PHP servers do we even have?

Inventory said 12.

Reality: nobody knew. One was running WordPress. Three were old internal tools. One was **under someone's desk.**



@IT_unhinged

Security called me at 6:12 AM today.

"Patch every PHP server immediately."

I asked what happened.

"Two critical vulnerabilities. Remote code execution. No authentication required."

Cool.

How many PHP servers do we have?

According to inventory: 12

According to reality:

Nobody knows.

We found one running WordPress.

Three running old internal tools.

One under someone's desk.

Post your reply

@IT_unhinged · the sysadmin reality

This is the tool that caused the panic

One HTTP request.

No login. A shell.

The researchers' tool auto-targets the big PHP apps:

WordPress ✓

Drupal ✓

Joomla ✓

+ remote memory disclosure of **any** PHP app.

Look at the terminal → "PRE-AUTH RCE CONFIRMED ·
uid=33 (www-data)"



sagitz ✓
@sagitz_

Follow

Today we disclose PHP2Shell - remotely exploitable memory corruptions in PHP

WordPress pre-auth RCE: ✓

Drupal pre-auth RCE: ✓

Joomla pre-auth RCE: ✓

Remote memory disclosure of any PHP application: ✓

Read more ↓

```
PHP2SHELL
exploit by WizResearch · v1.5.13 / automatic target discovery and command execution

00:00:00 00 DISCOVER fingerprint remote application
00:00:00 00 DETECTED WordPress at 10.254.252.3:80
00:00:00 00 PREFLIGHT validate target and execution policy
00:00:00 00 HTTP HTTP 301; server:Apache/2.4.68 (Debian); 0.05s
00:00:00 00 CHAIN worker capture -> ZendMM map -> PHP global overwrite
00:00:00 00 INFO CONTROL_FLOW resolved PG(auto_prepend_file) target -> in-memory payload -> /usr/bin/id
00:00:00 00 INFO GADGETS none; this chain uses a runtime-resolved PHP global-pointer override
00:00:00 00 EXECUTE run unauthenticated request sequence

RAW DISCLOSURE / 256 captured bytes; unredacted · bytes 00c8-00ff
0x7985403fdc28 00 00 00 00 00 00 00 58 dc 5f 43 85 79 00 00 [...]X.C.y..
0x7985403fdc40 00 00 00 00 00 00 00 00 dc 5f 43 85 79 00 00 [...]..C.y..
0x7985403fdc58 00 00 00 00 00 00 00 00 dc 5f 43 85 79 00 00 [...]..C.y..C.y..

00:00:00 00 OK CAPTURE remote worker memory acquired (256-byte live window)
00:00:00 00 INFO LEAK_BASE leak = 0x7985403f8000
00:00:00 00 INFO ALLOCATOR main chunk = 0x798550200000
00:00:00 00 INFO IMAGE_PTRS text=0x7985522a8c50 · rodata=0x798552ce9f88
00:00:00 00 INFO WRITE_TARGET PG(auto_prepend) = 0x798552e631a0
00:00:00 00 INFO HEAP_OBJECTS D=0x7985403f0000 · F=0x7985403f0000
00:00:00 00 INFO OVERRIDES D=0x7985435f1000 · F=0x798550200000 · victim=1259

00:00:00 00 OK RESULT PRE-AUTH RCE CONFIRMED
00:00:00 00 OK OUTPUT uid=33(www-data) gid=33(www-data) groups=33(www-data)
01:00:00 00 100% request sequence complete 65.2s
```

Post your reply

@sagitz_ · Wiz Research — the disclosure

Governments told people to pull the plug



高市早苗
@takaichi_sanae

Follow

Show translation

皆様に、今夜に対応して頂きたい大至急のお願いです。

多くのウェブサイトで使用されているPHPに関し、PHPのセキュリティチームが、重大な2件の脆弱性に関する注意喚起を行っています。

公開しているウェブサイトでPHPを使用しているか確認し、今夜、10月21日午後11時（日本時間）に公開される修正パッチについて、可及的速やかに適用できるよう準備をお願いします。

修正パッチを速やかに適用することが困難な場合は、例えばウェブサイト等を一時的に停止させることも、リスクを低減させるための有効策の一つと考えられます。

国家サイバー統括室のXアカウントでも関係する情報を発信していますので、そちらも参考にして下さい。

国家サイバー統括室(注意・警戒情報)
(@cyber_forecast) / X

Post your reply

@takaichi_sanae · Japan (translated)

lans reposted



nyx
@nyxvoid92

今晚 10 点，全球 PHP 服务器准备迎来大逃

两个直通核心的 RCE 漏洞被提前外泄，传闻一个请求直接拿 Shell（PHP2Shell 工具出来了）。

最惨的是 PHP 8.1 以下完全不给补丁，而全达 59% 的对外主机还在跑旧版。今晚 22:00 一发 Patch 等于直接开战，15 分钟内全网

还在跑 PHP 7、PHP 5 祖传代码的公司，现拔网线跟锁 WAF 真的没别招了，好自为之

04:07 · 10/21/26 · 479K Views

150 345 2.1K 519

Relevant

View c



@nyxvoid92 · the panic (translated)

Japan, national advisory:

"Check if your public sites use PHP. Apply tonight's 11 PM patch ASAP. If you can't, taking the site offline is a valid way to cut risk."

The street view:

"PHP ≤ 8.1 gets NO patch. ~59% of public hosts still run old versions. When the patch drops, the whole net gets scanned in 15 minutes."

You have already exploited this exact bug

Everything in PHP2Shell is a bug class from THIS course — just in a huge target.

Lab 1–2

Overflow a buffer, corrupt memory you shouldn't reach

Class 12–14

Leak an address to beat ASLR; reuse libc / PLT-GOT for code exec

Week 11

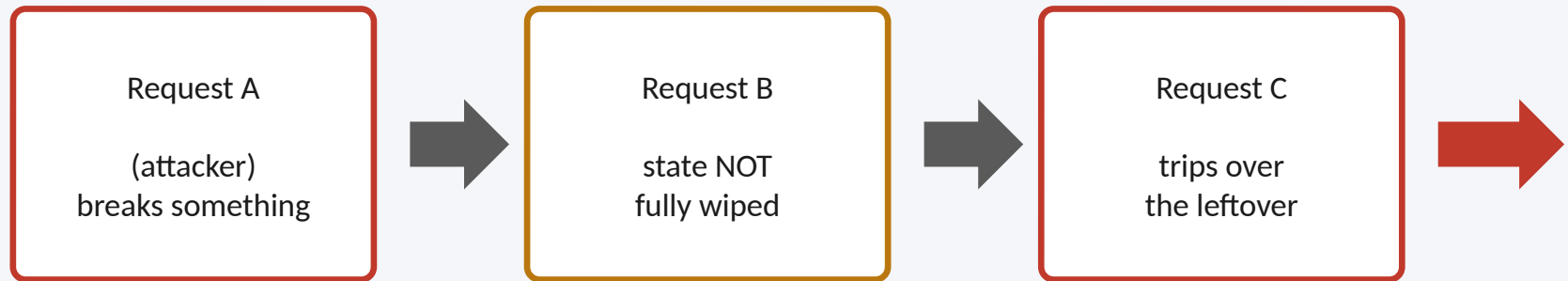
Use-after-free + heap grooming to land attacker bytes in a freed slot

PHP2Shell = all three, chained, against a server that never restarts.

The one new idea: the server never forgets

Our lab binaries start fresh and exit. A PHP worker is long-lived — it serves request after request, and is supposed to wipe its state between them.

ONE WORKER PROCESS — survives across requests



Remember two words for the rest of today:

Producer — the request that leaves broken state behind. **Consumer** — a later request that uses it.

Both bugs share one villain: how PHP gives up

When PHP runs out of memory, it doesn't clean up on the way out. It jumps.

What SHOULD happen

Finish the operation → tidy up each half-done thing → reset globals → end request cleanly.

What actually happens

```
zend_bailout() = longjmp()
```

a jump across functions, straight to shutdown — **skipping all the tidy-up.**

That “goto” leaves two kinds of wreckage — one bug each:

109618 a half-built object the cleanup still trusts

109619 a pointer to memory that's already freed

Bug #1 — the half-built object (CVE-2026-109618)

1

Start building

PHP marks “slot #5 exists” before it has finished writing slot #5.

2

Run out of memory

The allocation for slot #5 fails — longjmp aborts. But the count already says it's there.

3

Cleanup trusts the count

Shutdown reads slot #5 as a real object and follows its (garbage) pointer.

With heap grooming (Week 11!), the attacker owns those garbage bytes → **code execution**.

Reaching “out of memory”: a crafted multipart upload inflates internal structures without tripping upload limits — under `post_max_size`, still over `memory_limit`. Path dates to PHP 5.4 (2012).

Bug #2 — the ghost pointer (CVE-2026-109619)

A global still points at memory that was already freed. The next request follows it.

PRODUCER

Request hits a fatal error (OOM, timeout, failed require...). `longjmp` aborts before the global is reset.



heap
FREED

*but the global
still points here*



CONSUMER

A later request touches that global before `init` resets it — attacker now controls freed memory. Use-after-free.

The sneaky part: the “consumer” can be a mere warning message. Hiding it with `@` or `display_errors=Off` doesn't help — **generating the warning is what touches the dead pointer.**

Default trigger: `session upload-progress` runs this path before your app code does — no `session_start()` needed. (Confirmed RCE case on PHP 8.5 via the fatal-error backtrace global.)

The payoff: read server memory from a cookie

You don't need full RCE to see the danger — just a read primitive, exactly like our GOT leak to beat ASLR.

- 1 Prime** A multipart upload makes a session pointer point into the request heap.
- 2 Free** Force OOM — longjmp aborts, the heap is freed, the pointer dangles.
- 3 Reuse** Send a second upload to the same worker; cleanup runs before reset.
- 4 Leak** PHP trusts the dead pointer and writes those bytes into your cookie.

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=%00%7f%45%4c%46%02%01%01...
                ↑ raw bytes of freed server memory, handed to the attacker
```

Same move as Class 14: leak first, then aim. Chain this read into the Bug #1 write and you have pre-auth RCE.

How they fixed it — and what you'd tell a client

The real fix

Stop jumping. OOM no longer calls longjmp mid-operation — it waits for a safe point, or exits hard.

Wipe the room. Shutdown now resets the full set of request-owned globals before freeing the heap.

If you can't patch tonight

- `memory_limit = -1` (admin-enforced)
- `session.upload_progress.enabled = 0`
- `fatal_error_backtraces = 0` (8.5)
- bound request size at the proxy; restart workers

Stopgaps only — other variants may exist.

Update is the only complete fix:

8.2.35

8.3.36

8.4.27

8.5.12

PHP 8.1 and older are end-of-life — no official patch. If you're on them, you're on your own.

What to take away

Memory bugs scale. The classes you practice on tiny binaries are what put 22M servers at risk.

The abort path is attack surface. “Safe” error handling that skips cleanup is where the bug lived.

Shared state is dangerous. If a process serves many requests, one can poison the next. Reset, never assume.

Talk it through

1. Why is Apache mod_php rated worse than PHP-FPM for the same bug?
2. Why don't @, display_errors=Off or error_reporting stop Bug #2?
3. We leaked bytes into a cookie — trace the path from that read to full RCE using Class 12–14.
4. Which Week-11 heap skill makes landing attacker bytes in the freed slot easiest?

Sources & further reading

- PHP official advisory (21 Oct 2026) — php.net/archive/2026.php#2026-10-21-5
- GHSA-w8hw-w9fx-j79w — RCE via OOM corruption (CVE-2026-109618)
- GHSA-6wmf-4rv5-p3rp — RCE via stale global state (CVE-2026-109619)
- Calif.io (LCFR) — original research write-up, calif.io/php2shell
- Wiz Research — PHP2Shell technical breakdown (symbols & confirmed cases)
- Unit 42 (Palo Alto) — threat brief + 22.6M exposure figure
- Tenable / watchTowr / Help Net Security — FAQ, exploitation status, timeline

Reaction posts: @sagitz_, @takaichi_sanae, @nyxvoid92, @IT_unhinged (translated). Credits: Calif.io, Wiz, OpenAI; fixes by arnaud-lb, iluuu1994, bukka, ndossche.