Episode Book

EPISODE #6449

Why PHP2Shell Was a Sniper Rifle, Not a Nuke

PUBLISHED	4 November 2026 at 12:28 UTC
DURATION	27:36
LISTEN	myweirdprompts.com/episode/php2shell-cve-exploit-analysis-30490/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	5 November 2026 at 07:03 UTC

SYNOPSIS

Critical CVE scores rarely match actual threat reality. An analysis of the PHP2Shell unauthenticated remote code execution vulnerabilities highlights the gap between catastrophic headlines and real-world exploitation. Despite 46 million vulnerable web servers and an embargo breach, attacks required hyper-specific target configurations. The discussion covers decade-old memory bugs found by modern AI models, rapid patch-diff exploitation by adversaries, rolling deployment failures, and comparisons to historic dependencies like Log4j.

TRANSCRIPT



Corn

Last month, the language running most of the web had a very bad couple of weeks, and the details were loose in the wild before the patches landed. That's the hook Daniel sent us: PHP2Shell, two unauthenticated remote code execution flaws, and then the wider question he actually wants answered. What have been the biggest CVEs of the last few years, and what made them big?



Herman

And I love that framing, because it sounds like a list request, but it's really a definition request. Big how?



Corn

Big according to whom. The score? The headlines? The body count?



Herman

Herman Poppleberry here, and yes, I've been waiting for a vulnerability episode like a kid before a field trip. Because PHP2Shell is the perfect test case for the gap between "critical" and "catastrophic," and that gap is where the whole story lives.



Corn

So let's put the two flaws on the table. What actually broke?



Herman

Two memory bugs, both in PHP core, both reachable without credentials. The first one, twenty-six-109618, lives in the out-of-memory path. When PHP runs out of memory it uses longjmp, which is basically a controlled panic jump out of the allocation code. Problem is, some contexts don't account for that jump, so you end up with request memory sitting there uninitialized while the cleanup code reads it and branches on it. If you can control what lands in that memory, you can steer the cleanup.



Corn

The janitor reading notes off the floor before the lights go out.



Herman

That's... yes, actually, that's a decent picture of it. The janitor doesn't write the notes, but the janitor acts on them. And in a shared hosting environment, the floor is full of other people's scraps, so an attacker's goal is to litter it with exactly the notes they want read.



Corn

How would an attacker even get their content into that memory in the first place?



Herman

That's the craft part. You spray the request with data you control, upload fields, headers, whatever the application passes through, and then you trigger the out-of-memory condition at the right moment so your data happens to be occupying the space the cleanup routine reads. It's timing plus garbage placement. Which is exactly why it's not one-size-fits-all, and that comes back to bite the whole story later.



Corn

And the second one?



Herman

The second one, twenty-six-109619, is about global state pointing into memory that belongs to an individual request. If that pointer isn't reset properly between requests, a later request reads memory that's already been freed. And that second one reaches back to PHP five point four onward, which is a shocking range.



Corn

Five point four. When did five point four come out?



Herman

Twenty-twelve. So on that second flaw, every PHP deployment for over a decade, including versions that are themselves years out of support, is in scope. And the use-after-free shape matters too, because freed memory on a busy server gets reused constantly. On a shared host processing thousands of requests, the odds that something interesting lands in that slot before your follow-up request reads it go way up. It's a race the attacker gets to run over and over until they win it.



Corn

Both rated nine point two under CVSS four.



Herman

Critical, both of them, and the PHP team's own rating. Under the older three point one scale they come out nine point eight and eight point one. CISA put both on the Known Exploited Vulnerabilities catalog five days after disclosure, which tells federal agencies to patch on a clock.



Corn

And the reach number everybody quoted was seventy percent of websites run PHP.



Herman

Roughly, yes. Wiz put it at sixty-five percent of organizations in their dataset. And the part that made this a scare rather than a bug report: it's the runtime. Not one app, not one vendor product. Every host running PHP, every version back to five point four for that second flaw, is standing in the open.



Corn

There was a detail I liked, in a grim way. The bug had been sitting in core for over a decade. Doug Jacobson at Iowa State pointed that out, and the reason it surfaced at all is that a researcher at a firm called Calif, in August, used an advanced OpenAI model to find it.



Herman

Found a decade-old memory bug in one of the most audited codebases on earth. That part of the story is new. We've had AI-assisted fuzzing before, but a longjmp cleanup bug that survived years of eyeballs, found by a model, that's the line being crossed.



Corn

Nine point two, seventy percent of the web, and here we are two weeks later. Which is the question the whole thing keeps circling. Herman, before we get into the comparison set, take me through the week, because the disclosure itself became half the story.



Herman

The timeline is the story. Patches and public disclosure landed together on October twenty-first, ten a.m. Eastern. Governments took sites offline preemptively. Wikipedia suspended its bug tracking and project management systems rather than risk it.



Corn

And to be clear for listeners who don't live in this, that's Wikipedia choosing to partially blind itself, taking its own coordination tools down, because the alternative was leaving doors open into its infrastructure.



Herman

The calculation was that a few days of a worse workflow beat the risk of an unauthenticated RCE in the runtime underneath it. But AWS's research, published the next day, showed a honeypot exploitation attempt on October seventeenth. Four days before anyone was supposed to know the details. From an IP on a commercial anonymization network.



Corn

So somebody had it before the embargo broke.



Herman

Somebody had something. CJ Moses at AWS was careful about it. "Remains unclear how the actor acquired exploit information prior to disclosure." And then proofs of concept appeared hours before the ten a.m. announcement, which means the leak had already circulated by the time the patch train arrived.



Corn

The whole premise of coordinated disclosure is that the bad guys find out when the patch does. A four-day head start breaks the model, not the code.



Herman

Moses went further and said it's structural. "The time from patch to working exploit is shrinking for structural reasons. AI-assisted analysis makes it cheaper to turn a patch diff or advisory into a working exploit. As a result, more actors can act within hours or even minutes once technical details of a vulnerability is disclosed." I'd underline that, because people keep treating this as a PHP problem. It isn't. The patch diff itself has become the exploit blueprint, and reading diffs is something models are now good at.



Corn

There's the symmetry nobody wants to sit with. The model that found the decade-old bug in August is the same class of tool that turns October's diff into November's weapon.



Herman

Offense and defense got the same upgrade on the same day, and the defense only gets to use it if it's looking. That's the asymmetry that actually matters. Calif had to be pointed at PHP core on purpose. An attacker just has to be pointed at a patch.



Corn

And the patch, in this case, wasn't even the end of the incident.



Herman

Not remotely. October twenty-third, PHP warns of a third flaw, twenty-six-115560, and declines to ship an emergency patch for it, because the configuration required to trigger it is exotic enough that they judged the fix riskier than the exposure. They told people to disable session upload progress instead.



Corn

A vendor telling you the safest patch is a setting.



Herman

Which is honestly the right call sometimes, but it reads terribly in a crisis. Then October twenty-fifth, four days after the main event, a second round of builds, because the first patches missed a flaw in the Brotli compression library, a separate CVE. First attack against that one recorded the next day.



Corn

So the story everyone reported as "PHP patched it, whew" was actually a rolling week of "PHP patched part of it."



Herman

Patching a language runtime is a multi-week process, not an event. You've got shared hosts, you've got containers nobody rebuilds, you've got that PHP five point four long tail. Censys scanned it: forty-six million vulnerable servers reachable, and two weeks later, twenty-nine million still unpatched.



Corn

Twenty-nine million.



Herman

Against that, Wiz did see a sixty-seven percent drop in internet-exposed vulnerable machines after disclosure, so the people paying attention moved fast. The exposure didn't vanish, it stratified. The ones who patch in a week were never the ones the attackers want anyway.



Corn

That's the part I keep coming back to. The people who move in week one are fine, and the twenty-nine million are the ones who find out from the ransom note.



Herman

Now the contested part, because this is where the official record fell apart. October twenty-second, Elizabeth Barron at the PHP Foundation tells WIRED, "to our knowledge, there has been no successful attempt at remote code execution." Same day AWS is saying broadly exploited in the wild. Two days after that, October twenty-fourth, Adel Karimi at Perplexity confirms actual RCE on honeypots running Apache mod_php on eight point four point twenty-six. His phrasing was good, "confirmed RCE with usable memory disclosure, not just a crash followed by a suspicious process."



Corn

The suspicious process defense. Every incident report has one.



Herman

Then Palo Alto's unit reported an attack on a European government agency and retracted it. And Dream reported twenty-five compromised firewalls and management servers across the US, Japan, Taiwan, South Korea, three of them in a US-government-linked environment. Meanwhile Wiz on the twenty-seventh says no successful end-to-end exploitation in its telemetry, and Kaspersky's two-week retrospective, out yesterday, says "no wave of breaches has materialized."
Ivan: Nightjar, nightjar. Sixteen, forty two. Nine, three. Sixteen, forty two. The goose has left the porch light on. Count the eggs under the hedge, not the ones in the kitchen. The nest is warm but the gate is open. Cuckoo.



Corn

So you have confirmed real RCE in honeypots, twenty-five real boxes compromised, and no mass breach. All simultaneously true.



Herman

All simultaneously true, and it's not a contradiction, it's the shape of the thing. This exploit is not one-size-fits-all. You have to match the exact PHP build, the memory settings, how the application handles requests. Kaspersky's number I find quietly staggering: of about nineteen thousand Palo Alto devices found running the vulnerable code, attackers picked two hundred and forty. Two hundred forty. They scanned nineteen thousand and cherry-picked the ones their exploit actually fit.



Corn

A sniper rifle that the headlines called a nuke.



Herman

And that's the lesson I want to carry into the list, because every big CVE is big on a different axis, and the score captures almost none of them. So, the comparison set. Log4Shell.



Corn

2021, the ten point zero, the one that got executives to learn what a logging library was.



Herman

December of twenty-one, Chen Zhaojun at Alibaba Cloud Security, and the reason it detonated is almost poetic: it first got noticed in Minecraft servers. People were putting strings in chat messages and getting servers to fetch code from remote LDAP servers. The bug is JNDI lookup injection. Log4j2, from two point zero beta nine through two point fifteen, would parse attacker-controlled log text, follow a lookup, and go fetch from wherever the string told it to.



Corn

The logger logged the attack and executed it.



Herman

The logging library was the attack surface. And the reason it was everywhere: a logging library is the single most transitively included dependency in Java. Nobody chooses it, it arrives. The NVD lists over three hundred and twenty-seven affected products, and that's the ones somebody bothered to enumerate. An attacker could send a string in a user agent, a user agent, Corn, and own the box.



Corn

Compare that with PHP2Shell's two hundred forty out of nineteen thousand. Log4Shell was a string in anything.



Herman

Any field that ends up in a log. That's the exploitability axis maxed out. Reach plus trivial exploitability equals a global fire drill that took, realistically, years to fully burn out, because the transitive dependency meant people didn't know they had it. The invisible layers are the load-bearing ones.



Corn

And there's a detail from that December I still think about. For the first forty-eight hours or so, the advice people were passing around included mitigations that turned out to be incomplete, and the version guidance kept shifting as the maintainers found bypasses for their own patches. It was the same rolling-patch shape we just watched with PHP, just compressed and spread across more vendors.



Herman

Same shape, different scale. Log4Shell had dozens of vendors each shipping their own sequence of fixes, so the "did you patch it" question had a different answer depending on whose product you were standing in front of. At least with PHP there was one upstream issuing the builds.



Corn

Then the opposite shape. One product, no reach, ninety-five million people.



Herman

MOVEit Transfer. CVE twenty-three-34362, May 2023, CLOP ransomware went after it. A managed file transfer product, which is not a common thing to run, thousands of installations, not millions. But every installation is a pipeline into other companies' data, because that's what the product does, it moves files between organizations. One victim's MOVEit instance handed over the data of hundreds of downstream partners. As of October 2024, Tenable counted two thousand seven hundred seventy-three organizations and over ninety-five million individuals.



Corn

Ninety-five million people affected by a product most of them had never heard of.



Herman

Blast radius, not reach. Log4Shell was everywhere and exploitable. MOVEit was rare and surgical, and the cascade did the damage. That's the axis CVSS has no column for.



Corn

And the victims in that cascade weren't even the ones who failed to patch. A pension administrator runs MOVEit, patches late, and the retirees whose data flowed through it had no say in any of it. The people hurt were three links away from the vulnerable software.



Herman

Which is why the ninety-five million number exists at all. If you counted only the operators, it would be a modest incident. The multiplication happens because the product's entire job is to be the multiplication.



Corn

EternalBlue is the third shape.



Herman

The wormable axis. An NSA-discovered flaw in Windows SMB that leaked when the Shadow Brokers dumped the agency's tools. On its own, a nasty remote bug. Weaponized in WannaCry and NotPetya, it became self-propagating lateral movement with zero user interaction. NotPetya alone ranks among the costliest cyberattacks in history, and the original bug wasn't even targeting those victims, it just spread through anything reachable.



Corn

No click, no lure, no phishing email. The network cable was the lure.



Herman

Heartbleed is the fourth, and it's the stealth axis. OpenSSL, the Heartbeat extension. You send a length field that lies about how long your heartbeat message is, and the server, being polite, hands back that much memory from beyond the actual message. Server private keys, session tokens, credentials, whatever was sitting adjacent in process memory. Somewhere between a quarter and half of popular HTTPS servers were affected.



Corn

Nothing crashed. Nothing was down. You just bled.



Herman

No downtime, no logs of consequence, secrets walking out quietly for who knows how long before anyone knew to look. A wormable bug announces itself. Heartbleed's whole danger was that it didn't.



Corn

And the aftermath problem there was uniquely bad. You can't un-leak a private key. With a code execution bug you patch and move on. With Heartbleed, every affected site had to assume its keys were already gone and reissue everything, because there was no way to look back and check.



Herman

Right, the absence of evidence was the evidence. The clean logs that would normally reassure you were exactly what made it worse.



Corn

So that's four axes so far, and then the recent benchmark, the one from last December. React2Shell.



Herman

React2Shell, CVE twenty-five-55182, unauthenticated RCE in React Server Components, Next.js. Disclosed early December 2025. What makes it the reference point for this whole discussion is the speed of the ecosystem response. VulnCheck counted two hundred thirty-six valid public exploits in the last four weeks of the year. Two hundred thirty-six, Corn. That surpassed PwnKit as the most publicly researched CVE they'd ever tracked.



Corn

Research is a generous word for what some of those were.



Herman

Fair, but follow the consequences. By December thirty-first, eleven distinct threat actors, two ransomware families, four botnets using it. By late January, VulnCheck's canaries had recorded over twenty-six thousand exploit attempts. Talos said it "rocketed to the highest percentage of attacks for the entire year within the last three weeks of 2025."



Corn

A whole year of attack share claimed in three weeks.



Herman

And the runner-up was SharePoint, thirty-six public exploits, six ransomware families on it. The pattern is the tempo. React2Shell's four-week ecosystem is the world Moses was describing, and PHP2Shell is the first case where we got to watch the compressed window happen to the runtime under the entire web.



Corn

And yet, the punchline of the whole comparison. A nine point two with seventy percent of the web exposed, and no mass breach materialized.



Herman

Because the exploit needs per-target tuning. Two hundred forty out of nineteen thousand. The reach was total and the practical exploitation was artisanal. Log4Shell was a ten point zero with a trivial string. MOVEit was one product and ninety-five million people. EternalBlue spread itself. Heartbleed whispered.



Corn

Score one of those, at best.



Herman

The score grades the bug. It doesn't grade the situation, and the situation is the thing that hurts you. Reach, exploitability, blast radius, and I'd add propagation and stealth under those. CVSS is a measurement of the fall.
And that's the thing about a CVE score, it tells you how bad the fall is, not how many people are standing under the ladder.



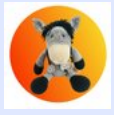
Herman

Understood. Same place as last time.



Corn

What on earth was that?



Herman

The patch-rate numbers are the other half, though. Censys via Kaspersky, twenty-nine million of forty-six million still unpatched at two weeks. Even with CISA on the clock and a sixty-seven percent drop in exposure among the attentive, that long tail is where the real infections will come from, months from now, and nobody's headline will connect them back.



Corn

The breach that eventually matters won't say PHP2Shell on it. It'll say some hospital trust or a mid-size retailer, and the forensics will quietly note the box hadn't been rebuilt since October.



Herman

And that's the quiet privilege of the exploitability question. The attackers have to do the tuning work, so they triage. Two hundred forty targets. They're not wasting a good exploit on an uncertain target when the string-based ones are free.



Corn

Which inverts the usual advice, slightly. The patch-your-stuff crowd says do it before you're a headline. Here it's more like, do it because you're precisely the kind of target that survives on being annoying to attack.



Herman

Effort as armor. Not a strategy I'd sign off on, but it is measurably what happened. Dream's twenty-five boxes, and honestly, government-adjacent management servers are exactly the target worth tuning an exploit for.



Corn

Worth the craftsman's hours.



Herman

The bug broker's hourly rate. Corn, one more comparison worth making, because you asked what the biggest of the last few years were, and there's one from the old era that belongs in the set. Shellshock. Bash, twenty fourteen, but the flaw dated back to the late eighties, to the day bash learned to export functions through environment variables. The parser would execute whatever came appended after the function definition.



Corn

A bug older than most of the people exploiting it.



Herman

Sat there for two and a half decades, reachable through CGI on half the Unix web servers of the era. That's the deep-time version of the PHP2Shell story, and PHP's second flaw reaching back to five point four is the same shape at smaller scale.



Hilbert

The bash one is older than twenty fourteen by more than you said.



Herman

Did I get the version range wrong?



Hilbert

The bug. Bash got the exported functions in nineteen eighty-nine. People say twenty fourteen like the bug arrived then. It was in the shell I was running when it was two years old. Mid-eighties I had a Tandy six thousand with a Unix bolted on, can't have been cheap, I forget. I typed a function into an environment variable to carry settings between two programs and noticed the shell would finish the line for me. Anything after the function. Wrote a letter to the man who maintained it. Never got a reply. Kept the carbon for a while.



Corn

You kept the carbon copy of a bug report from nineteen eighty-five.



Hilbert

The stamp was the interesting part. Corn, you say nobody counts the tail, but the two hundred forty were counted because someone had to read nineteen thousand banners. That's the whole job. Reading banners.



Herman

He's not wrong, the targeting data comes from someone doing the boring enumeration at scale.



Hilbert

Twenty-five is what it costs when the exploit is fussy. In eighty-five it cost a stamp.



Corn

So the disclosure model. That's the open wound in all this. If the pre-patch window is now structural, and Moses says it is, what's left of coordinated disclosure? The entire practice, going back to the nineties, assumes the information moves with the patch.



Herman

And the assumption has quietly died. PHP2Shell's window was four days, from a source nobody can name. React2Shell showed the post-patch window collapsing to hours. Put together, the usable patch window, the time you have between "fix exists" and "fix is being bypassed," is now roughly the time it takes a competent admin to actually deploy, and nothing more.



Corn

Which turns every emergency release into a race where the defender runs uphill.



Herman

The defenders still hold one card, which is that AI can also read the diff and tell you what to do about it, faster than a human can. The same shrinkage on both sides. Whether the card is worth much depends on whether the twenty-nine million servers ever get around to playing it.



Corn

And the mass-compromise question is still open. Kaspersky yesterday, no wave of breaches. Wiz, no end-to-end exploitation in telemetry. Against Karimi's honeypots and Dream's twenty-five boxes and a KEV listing that says otherwise.



Herman

Both will turn out to have been right about what they each measured. The honeypots proved the exploit works on real configurations. The telemetry firms proved it hasn't industrialized. The retraction from Palo Alto is the detail I'd hold onto, though. Even the professionals, with real data, published a claim about a European government agency and took it back inside a news cycle.



Corn

Early official statements in a fast incident are wrong in both directions, and the corrections never travel as far as the claims.



Herman

That's the habit I'd want listeners to take from the whole episode, more than any list. Next time a nine point eight drops and the alerts go red, the first three questions aren't "what's the score." It's who's actually exposed, how hard is it to really exploit, and what does one box give you. The score is where the analysis starts, not where it ends.



Corn

The disclosure question is the one I'd leave open. If the leak is now structural and the patch-to-exploit window keeps shrinking, coordinated disclosure stops being a safety mechanism and becomes just a schedule. The industry hasn't decided what replaces it. Nobody has.



Herman

Thank you, as ever, to Hilbert Flumingtop producing the show. If you enjoyed this one, try episode thirty-five, The Privacy Gap; episode sixteen, On Deepfakes, SynthID, And AI Watermarking; and episode twelve, The AI Breakthrough. This has been My Weird Prompts. If you've got a minute, the review that helps us most is the honest kind, five stars optional but not discouraged.



Corn

Find everything we do at my weird prompts dot com. See you tomorrow, brother.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **NVD, published 2026-10-21, modified 2026-11-02 [primary]**
https://nvd.nist.gov/vuln/detail/CVE-2026-109618

2. **NVD, published 2026-10-21, modified 2026-11-02 [primary]**
https://nvd.nist.gov/vuln/detail/CVE-2026-109619

3. **NVD, Log4Shell [primary]**
https://nvd.nist.gov/vuln/detail/CVE-2021-44228

4. **Kaspersky, 2026-11-03**
https://www.kaspersky.com/blog/php2shell-two-weeks-after-patches/56647/

5. **WIRED, 2026-10-22 (updated 10:10am ET)**
https://www.wired.com/story/hackers-exploited-the-php-vulnerabilities-days-before-they-were-patched/

6. **Wiz Research, 2026-10-27**
https://www.wiz.io/blog/php2shell-exploitation-so-far

7. **Fortune / The Conversation (Doug Jacobson), 2026-10-23**
https://fortune.com/2026/10/23/openai-cyber-php2shell-openai-flaw-websites-hackers/

8. **Tenable, 2024-10-22**
https://www.tenable.com/blog/from-bugs-to-breaches-25-significant-cves-as-mitre-cve-turns-25

9. **BleepingComputer, Oct 2026**
https://www.bleepingcomputer.com/news/security/cisa-confirms-php2shell-vulnerabilities-now-exploited-in-attacks/

10. **BleepingComputer, Oct 2026**
https://www.bleepingcomputer.com/news/security/php2shell-security-flaws-now-sucessfully-exploited-in-the-wild/

11. **The Hacker News, Oct 2026**
https://thehackernews.com/2026/10/cisa-confirms-active-exploitation-of.html

12. **VulnCheck, 2025 year-end**
https://www.vulncheck.com/2025-routinely-targeted-vulnerabilities

13. **Cisco Talos, year in review**
https://blog.talosintelligence.com/year-in-review-vulnerabilities-old-and-new-and-something-react2/

14. **crackthe.net, biggest vulnerabilities since 1995**
https://crackthe.net/vulnerabilities/