

[Back](#)

There isn't an easy exploit for this PHP vulnerability

2026-10-21

I know, I know. It's all you're hearing about. That, and the number 4.

Don't get me wrong, there's clearly a bug here and it really does matter. But this is **not** easy to exploit. It is certainly nowhere near as easy as Log4Shell/Heartbleed/RoRYaml etc.

EDIT: Putting this up here so people read it. You should patch your servers anyway. There *is* a real vulnerability here, but exploiting it is significantly harder than people are making it out to be - it is certainly not as easy as e.g. log4shell.

Justify yourself, you lunatic!

I don't think this PHP vulnerability is that exploitable. I think that 5 years ago people would've understood this, but now people are being deluded by their LLMs telling them they have an exploit when they don't.

If you take a look at github right now, there are thousands, literally thousands, of "PoC" exploits for this vulnerability. One thing you'll notice about all of them though, is that they **do not work** against a vanilla PHP instance.

Go ahead, stand up a php server you think is vulnerable in Docker and then *run the PoC yourself*. Don't get your LLM to do it. No. Run the script, yourself. The exploits don't work.

Where's the exploit, Lebowski?

I think that a lot of people had assumed that LLMs will be able to build an exploit chain instantly from reading the diff, and that even if this was harder to exploit than other exploits, LLMs would've levelled the playing field. Well. There's no working exploit. Now what?

When I think about "we're very fucked" CVE10s on the Internet, I think of the following vulnerabilities:

- RoR YAML decoding (send a post body with bad yaml in it)
- Log4shell (set your user-agent to \${jndi:ldap:})
- Shellshock (copy `env x='() { :; }; echo vulnerable'` and edit to taste)
- Heartbleed (ask for Content-Length 10000000)

The second these vulnerabilities made the news, there were botnets doing the rounds causing damage. And this was before LLMs! People have to write the botnets. With their hands!

It's been 10 hours since this vulnerability has been posted, and half the software world is looking at it, and lord knows 100,000 people have got their "superintelligence" on the case. Nevertheless, there's no working exploit. The ones doing the rounds are fake, or only work in a lab, because...

Fucking LLMs, man.

Script kiddies (and script adults, who have also switched their brains off) now all think that, well, they don't understand UAF chaining, but this Claude guy knows everything and can totally make an exploit.

Hilariously, this results in the LLM giving their younger-brother an unplugged controller. It says exploit all across the screen... and has some smoking gun transcripts... but it's modified the server you're testing against to have a severe vulnerability. Cheating!!

- <https://github.com/david-alvarez-rosa/php-rce>
- <https://github.com/eness/php2shell-CVE-2026-109618-CVE-2026-109619-PoC>
- <https://github.com/ikow/php2shell-poc>

I just plucked these three because they were the ones with the most stars, but it goes on for miles and miles.

One really amusing thing in common with most of these "PoC"s is that they lean on "LFI", or local-file-inclusion, to make the vulnerability. In non-php terms, their exploit chain requires the server to already have a severe code-level vulnerability.

It's not rare for PHP apps to be fucked in some sense, but requiring the server to have an existing vulnerability where users could `cat /etc/password` is an *insane* pre-requisite. If you need this, this does not make this a severe bug.

The strangest thing is that so many people have independently made this extraordinarily weird defect in judgement. That is because they haven't independently made this decision. Claude/Etc. has done all of the judgement for them, hence the bizarre distribution in logical errors.

There's something important to be said here about how many people have Claude/GPT make design decisions for them, and how AI is the primary decision maker in the world right now. You've got this bizarre case of "Decision Schelling Points" where a method of solving a problem is extremely overrepresented, because it's what Claude does out of the box.

It's especially bizarre in this case, where AI is inserting a vulnerability on the server... to prove that the server is vulnerable to your "exploit any PHP instance" attack!

So why was everyone so scared?

Because:

1. A PHP core team member said that this was cataclysmic ("exploits will start appearing 15 minutes after the patch drops")
2. We all deep down know that PHP is insecure
3. The exploit video had an anime girl in it
4. We all know that LLMs are looming over the internet like a 100 ton chekov's anvil, and we know that code isn't secure

And we've had things like this before, but not in the LLM era yet. Luckily for us, this doesn't seem to be that day. This is too difficult to exploit easily. It isn't a Log4J moment.

Contact

EDIT: I have added my contact details to this post and fixed a broken link. I added the disclaimer at the top that you should still patch.

Signal: zk7.27

Email: zkldi@proton.me