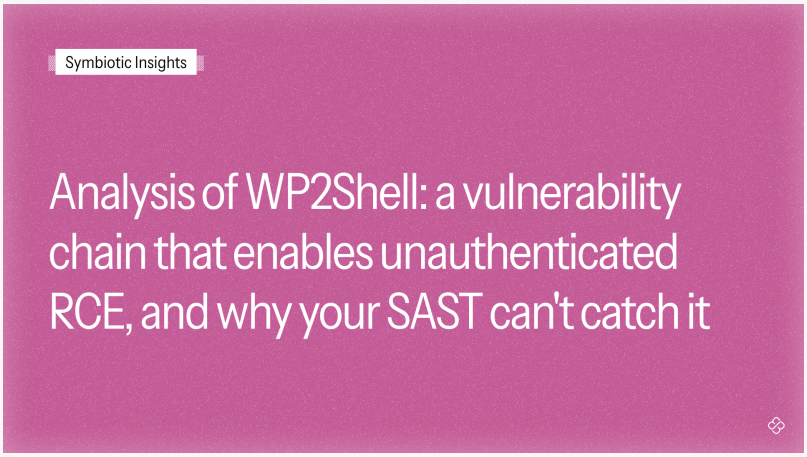
Symbiotic Security Breaks Down WP2Shell, a WordPress Core RCE Chain SAST Misses

New analysis of WP2Shell (CVE-2026-60137, CVE-2026-63030): two safe-looking WordPress core bugs compose into unauthenticated SQL injection and RCE.



Analysis of WP2Shell

BROOKLYN, NY, UNITED STATES, July 28, 2026 /EINPresswire.com/ -- Symbiotic Security today published a new technical analysis of WP2Shell, a two-vulnerability chain in WordPress core that turns an unauthenticated SQL injection into unauthenticated remote code execution (RCE). Beyond the technical severity, the research highlights a structural blind spot in modern application security: static analysis (SAST) reasons about one unit of code at a time, while WP2Shell only exists in the composition of multiple components across files and execution phases.

> SAST reasons about one unit at a time; WP2Shell is a property of the composition of three. The cheapest vulnerability to fix is the one the code was never able to contain."
>
> *Sébastien Rolland, security researcher at Symbiotic Security*

The chain combines two CVEs: CVE-2026-60137 and CVE-2026-63030. In isolation, each looks benign. The SQL injection sink in WP_Query appears unreachable because WordPress core never supplies a scalar author__not_in, and the batch REST endpoint bug looks like a routing oddity. Together, an attacker can manufacture the exact type shape, a scalar author__not_in string, that makes the "dead" sink live.

The enabler is a batch dispatcher desynchronization. The endpoint validates requests under one schema in one pass, then executes them under a different handler in a second pass. A malformed path inserts an error object into the request and validation lists but not the match list, shifting handler mapping by one and allowing a request to be sanitized under one schema and executed under another.

"What WP2Shell reveals isn't just a chain that can lead to RCE, it's an uncomfortable reality: there are countless security scanners on the market, and very few can detect this kind of vulnerability. It's a clear reminder that security after the fact, like scanning a merge request, is far from sufficient," said Edouard Viot, Co-founder and CTO at Symbiotic Security.

Why it matters: WordPress powers 59.1% of all sites whose CMS is known, or roughly 41.2% of the entire web, according to W3Techs. A flaw in WordPress core, not a plugin or theme, puts an enormous share of the internet within reach of unauthenticated attackers. Unauthenticated remote code execution is the most critical class of flaw a remote application can have, and it is rare in WordPress core, which makes WP2Shell notable in its own right.

The more durable lesson is for security teams everywhere. WP2Shell is a composition bug. Each half is defensible in isolation, and the danger only emerges when the two combine across files at runtime. That is precisely the blind spot in the static-analysis tools most AppSec programs rely on. Detection alone is not enough. Safety has to be enforced at the sink and, better still, built in when the code is written.

Why static analysis misses it: SAST excels at source-to-sink paths, missing sanitizers, and simple type flows. It struggles with relational invariants across loops, index-alignment assumptions, and runtime mapping tables that bridge identifiers across files, all of which WP2Shell relies on. Because each file looks safe on its own, a scanner that reasons one unit at a time has no way to see the chain.

Recommended defense-in-depth measures: normalize ID lists at the sink on both scalar and array paths, not only the array path; ban raw SQL concatenation into wpdb queries and enforce prepare or parameterized builders; treat schema-validated inputs as untrusted again at execution boundaries; use custom scanner rules as guardrails, not as a discovery mechanism for unknown chains; and shift enforcement earlier, so unsafe patterns are prevented as code is written rather than flagged after the fact.

The full technical write-up, including the dispatcher desynchronization walkthrough and prevention guidance, is available at https://www.symbioticsec.ai/blog/analysis-of-wp2shell-a-vulnerability-chain-that-enables-unauthenticated-rce-and-why-your-sast-cant-catch-it?ophqt=550a877f8c344b32d5d7202b46b070cd61447e6cbee27dab.

Symbiotic Security is reimagining code security for the AI era. As the creator of Symbiotic Code, the company has introduced the first generative agent that treats security as a fundamental constraint rather than an afterthought. Founded in 2024 by industry veterans Jérôme Robert and Edouard Viot, and backed by top-tier investors, Symbiotic bridges the gap between AI productivity and enterprise-grade trust. With a global team operating out of New York and Paris, Symbiotic's technology is already in active deployment across a variety of customers and industries in the US. www.symbioticsec.ai.

Darren McNelis
Symbiotic Security
+1 251-309-3711
email us here