DFIR · CVE-2026-63030 / 60137 · WP2SHELL

Attacker request flow vs. the Apache access log

Every stage of the full `dfir` run reconstructed from a single access log — mapping what the attacker actually did against what the combined-format log line records, and what it silently omits.

source **/var/log/apache2/access_file.log** | host **localhost:8091** (WP 7.0.1)

attacker **172.20.0.1** | window **06:41:24–06:41:31 UTC** | UA **Chrome/126** (tool)

ANATOMY OF ONE COMBINED-FORMAT LINE

```
172.20.0.1 - - [19/Jul/2026:06:41:24 +0000] "POST /?rest_route=/batch/v1/ HTTP/1.1" 2
```

client IP | timestamp | request line — **method + URI only** | status | bytes sent | referer · user-agent

STAGE 0–1 **Recon & SQL-injection — shared by both chains** — the injection never appears in the log

0 Version fingerprint `GET recon`

Passive reads of generator strings to confirm the target is in the vulnerable range (6.9.x / 7.0.0–7.0.1).

```
… "GET /feed/ HTTP/1.1" 200 · "GET /readme.html" 200 · "GET /?rest_route=/" 200
```

IN LOG Benign-looking GETs to `/feed/`, `/readme.html`, `/wp-json/` — indistinguishable from a crawler.

1 Batch route-confusion + blind SQLi confirm `06:37:20 → 06:38:56`

Nested `batch/v1` requests desync the route handler and land `author__not_in` in **WP_Query**. A time-based `SLEEP()` differential confirms injection — visible only as evenly spaced requests.

```
06:37:20 "POST /?rest_route=/batch/v1/" 207 1542
06:37:44 "POST /?rest_route=/batch/v1/" 207 1543 ← +24s
06:38:08 "POST /?rest_route=/batch/v1/" 207 1543 ← +24s
06:38:32 "POST /?rest_route=/batch/v1/" 207 1542 ← +24s
```

IN LOG Repeated `POST /?rest_route=/batch/v1/ → 207 Multi-Status` ; small ~1.5 KB bodies; a regular **24-second cadence** = time-based blind extraction in progress.

HIDDEN The payload — `author_exclude=0) AND IF( ... ,SLEEP(3),0)-- -` , the nested `"///"` primer, the SQL — all live in the POST **body**, which the access log never captures.

PHASE 1 **Crack-based chain — dump the hash, log in, drop a shell** `plugin: asset-loader-crack`

2 Credential dump (in-band UNION) 06:41:24

A `UNION SELECT` forges a fake post row so `wp_users` (login + `$wp$2y$` bcrypt hash + email) reflects back in the response.

```
06:41:24 "POST /?rest_route=/batch/v1/" 207 19522
```

IN LOG Same batch URI/status as a probe — but the response **balloons to ~19 KB** (19522 vs ~1.5 KB). The size anomaly *is* the exfil signal.

HIDDEN The `UNION SELECT` and the dumped hash itself are in the request/response bodies — never logged.

3 Authenticate as admin 06:41:24

Fetch the login page (sets `wordpress_test_cookie`), then POST valid **admin** credentials — the `302` redirect to `/wp-admin/` means success.

```
06:41:24 "GET /wp-login.php" 200 5648
06:41:24 "POST /wp-login.php" 302 1232 ← auth OK
```

IN LOG GET then POST `/wp-login.php` with a `302` — a successful login. One request, first try = automated, not a human.

HIDDEN The username/password in the POST body; whether the creds were cracked or reused.

4 Deploy webshell plugin 06:41:25

Upload a ZIP plugin containing a token-gated PHP webshell via the admin plugin installer.

```
06:41:25 "POST /wp-admin/update.php?action=upload-plugin" 200 82635
```

IN LOG `action=upload-plugin` — a high-fidelity IOC. An unattended admin uploading a plugin seconds after logging in is not normal operations.

5 Webshell access → RCE 06:41:26

GET the freshly-created plugin path to verify, then run a command via `?cmd=`.

```
06:41:26 "GET /wp-content/plugins/asset-loader-crack/asset-loader-crack.php?token=8
06:41:26 "GET …asset-loader-crack.php?token=8f03…&cmd=whoami" 200 201
```

IN LOG GET to a **brand-new plugin path never requested before**, carrying a `token=` and `cmd=` query — the clearest RCE tell in the whole log, and here the payload *is* in the URI.

PHASE 2 — **Crack-free chain — forge an admin, no password needed** — `plugin: asset-loader-nocrack`

2 Forge admin via oEmbed + changeset bridge `06:41:27 → 06:41:28`

UNION-forges a fake `WP_Post`, primes the `oembed_cache`, and rides a `customize_changeset` to run `POST /wp/v2/users` as an existing admin — creating a **new administrator with no password crack**.

```
06:41:27 "POST /?rest_route=/batch/v1/" 207 19710
06:41:28 "POST /?rest_route=/batch/v1/" 207 19691
```

IN LOG Only more large `batch/v1` 207s — **identical shape to Phase 1**. The access log cannot tell the crack-free path from the crack-based one here.

HIDDEN The whole admin-creation — `POST /wp/v2/users`, the forged changeset, the borrowed admin ID — is tunnelled **inside the batch body**. No `/wp/v2/users` line ever appears in the access log.

3 Authenticate as the forged admin `06:41:29`

Log in as the just-created `pentest_<NNNN>` administrator.

```
06:41:29 "GET /wp-login.php" 200 5648
06:41:29 "POST /wp-login.php" 302 1253 ← auth OK
```

IN LOG A successful login — but for an account that was **never dumped and never existed before this session**. Correlated with user-audit data, "admin logs in that no one created" is the crack-free tell.

4 Deploy webshell plugin `06:41:30`

Same installer path, second plugin.

```
06:41:30 "POST /wp-admin/update.php?action=upload-plugin" 200 82662
```

IN LOG A **second** `upload-plugin` within seconds — same IOC, new artifact.

5 Webshell access → RCE `06:41:31`

Verify, then run `whoami`.

```
06:41:31 "GET /wp-content/plugins/asset-loader-nocrack/asset-loader-nocrack.php?tok
06:41:31 "GET …asset-loader-nocrack.php?token=3d3c…&cmd=whoami" 200 201
```

IN LOG A second new plugin path with `token=` + `cmd=` . Both shells return `www-data` .

What the access log can & can't see

- **No request bodies.** The combined format logs method + URI only. Every SQLi payload, the UNION dumps, and the entire crack-free `POST /wp/v2/users` admin creation are invisible — they ride inside `batch/v1` POST bodies.
- **Both chains look the same at the batch stage.** Crack-based vs crack-free is indistinguishable in the access log until the login/upload/exec tail — separate the two with the **DB query log** (`oembed_cache` / `customize_changeset` writes) and a **user-account audit**.
- **UA is not identity.** The tool ran as `Chrome/126` ; the analyst's browser heartbeat (`admin-ajax.php` , `Chrome/150`) is interleaved in the same log. Don't pivot on user-agent.

TELL · VOLUME

batch/v1 POST flood → 207s. A public site rarely sees a burst of `/batch/v1` at all.

TELL · SIZE

~19 KB responses to batch POSTs where probes are ~1.5 KB = in-band data exfiltration.

TELL · TIMING

Even N-second gaps between identical requests = time-based blind extraction stepping through characters.

TELL · SEQUENCE

login → upload-plugin → new `/plugins/…php?cmd=` within seconds is the RCE fingerprint.

Reconstructed from /var/log/apache2/access_file.log · wp2shell lab · CVE-2026-63030 (batch route-confusion → RCE) + CVE-2026-60137 (author__not_in SQLi). Query strings truncated for display; timestamps and status/size fields verbatim.