0-DAY NO CVE HIGH PUBLIC

phpMyAdmin CustomServer skips WebAuthn assertion signature

phpMyAdmin 5.2.3 CustomServer never verifies the WebAuthn assertion signature. After the MySQL password, a forged webauthn_request_response that matches challenge, origin, rpIdHash, and user-present is enough. Default tarball. No CVE yet.

NAME phpMyAdmin CustomServer skips WebAuthn assertion signature	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR phpMyAdmin
AFFECTED phpMyAdmin 5.2.3 default tarball (web-auth/webauthn-lib absent); CustomServer path. WebauthnLibServer is not this leftover.	PUBLISHED 12 Oct 2026
UPDATED 12 Oct 2026	TAGS 0-day, phpmyadmin, webauthn, 2fa, cwe-347, cwe-287, privileged

TL;DR

Lab files and proof-of-concept **here**.

01 The signature is parsed. Then it is ignored.

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/phpmyadmin-webauthn** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, config.inc.php, init.sql. Authorized lab only. It talks to loopback.

This is **phpMyAdmin 5.2.3** tag `RELEASE_5_2_3`. Default tarball. No CVE yet. CWE-347 / CWE-287. 8.1 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N`). Remote 2FA bypass. PR:L because you still need the MySQL password. After that password, WebAuthn 2FA does not verify `response.signature`. A garbage signature that still matches challenge, origin host, `rpIdHash`, and user-present is a full phpMyAdmin session. That is the database. That is not a PHP shell on the host.

`Plugins\TwoFactor\WebAuthn::createServer` uses `CustomServer` when `Webauthn\Server` is absent. The shipped tree lists `web-auth/webauthn-lib` as suggest / require-dev. `composer install --no-dev` does not install it. The official tarball does not ship it. The product's own `CustomServerTest` accepts an assertion with no public key.

If you actually installed `web-auth/webauthn-lib`, you are on `WebauthnLibServer` and not this bug.

02 Why phpMyAdmin

I had just finished unpublished Crit/High hunts of HTTP servers. phpMyAdmin is the SQL box a lot of those servers sit in front of. Cookie auth. Optional 2FA. A configuration-storage database with a control user that is not the logged-in account. A setup directory that is supposed to die after `config.inc.php` exists. CSRF tokens

that are supposed to empty the body. That is a lot of controls that can be leftover independently.

I only wanted Critical or High. Remote and local. Medium stays on the floor unless it chains. A finding on this hunt needed a leftover of a named PMASA, unauth mutating admin, SQLi as the control user, a 2FA skip, or a CSRF control that still runs SQL.

Eight parallel source hunts: SQLi/pmadb, XSS, auth/CSRF/setup, file/LFI, SSRF/deser, privileges/transformations, routes/CSRF, changelog leftovers. Five leftovers survived at High. This post is the first: WebAuthn. The others are central columns, designer JSON, setup/validate, and CSRF /sql.

03 Pinning RELEASE_5_2_3

Latest 5.2 stable on the day I cloned. Tag `RELEASE_5_2_3`. Commit `962857e`. `Version::VERSION = 5.2.3`. Not 6.x. 6.x already `quoteString`s some of the pmadb blobs this wave also hits. 5.2.3 is the tarball people still drop on a host.

04 The skip list is half the hunt

PMASA-2025-1 / 2 / 3 are already patched in 5.2.2. I did not re-file stored XSS in Check-tables headings, or the siblings that landed in that round.

Older names that look like this leftover and are not:

- WebAuthn "attestation: none" is a registration choice. It is not "skip assertion crypto on login."
- `WebauthnLibServer` is the real library. That path is not this bug.
- CVE-2016-5703 was the database name in central columns. Not WebAuthn.

- PMASA-2016-44 / CVE-2016-6621 was some setup SSRF. Not this 2FA class.

I am not filing "phpMyAdmin can run SQL." That is the product. I am filing the second factor that does not verify a signature.

05 Surfaces that returned nothing

File/LFI, deser, transformations, changelog leftovers at this bar came back empty or Medium. `isSuperUser()` is "can SELECT mysql.user" (Medium policy). `/config/set` persists `2fa` without step-up (Medium, and a chain from XSS or CSRF). ImportLdi `TERMINATED BY` is FILE-gated (Medium). Designer VQB history concatenates table names into HTML (High stored XSS, different leftover, not this post). Unquoted `authentication_plugin` is a different SQLi.

Auth held on password. Cookie login is password. HTTP auth is password. The WebAuthn prompt is the second factor. Structural checks on the assertion are real. The crypto is not.

06 CustomServer

`createServer`:

PHP Copy

```
1 return class_exists(WebauthnServer::class) ? new WebauthnLibServer($this->twofact
```

Default tarball: no `Webauthn\Server`. `CustomServer`. `parseAndValidateAssertionResponse` checks: credential id listed in `allowCredentials`, `clientData.type == webauthn.get`, challenge matches session, origin host, `rpIdHash`, user-present flag. `response.signature` must exist

and decode. Then the function returns. No COSE verify. No stored `credentialPublicKey`. No `signCount`. No `userHandle` bind.

The unit test does not pass a public key. That is not an oversight in my reading. That is the product's own test saying the assertion is fine without one.

HTTP is `POST /index.php?route=/` with `webauthn_request_response` after the password POST. Challenge plus `allowCredentials` are in the page. The PoC reuses the session challenge, forges origin host and `rpIdHash` and user-present, and puts `not-a-real-signature-PMA-WEBAUTHN-BYPASS` in the signature field. `check()` returns true. `$_SESSION['two_factor_check'] = true`. Home UI.

07 The lab

The Dockerfile copies 5.2.3, `composer install --no-dev`, then deletes `vendor/web-auth` so `CustomServer` is the one in play. The last `php -r` exits 1 if `Webauthn\Server` exists. `init.sql` enrolls WebAuthn 2FA for `appuser` without a hardware key. Password login reached the 2FA page. POST `webauthn_request_response` landed in the normal UI. `still_2fa=False` `landed=True`.

Clone the pin next to the Dockerfile, then run:

PLAIN TEXT Copy

```
cd lab
git clone --depth 1 --branch RELEASE_5_2_3 https://github.com/phpmyadmin/phpmyadm
./run.sh
```

Loopback `http://127.0.0.1:18080`. Compose project `pma-webauthn-lab`. Do not ship `src/`. Last line:

PLAIN TEXT Copy

```
SUCCESS PMA-WEBAUTHN-BYPASS WebAuthn 2FA accepted with garbage signature
```

Negative: install `web-auth/webauthn-lib` so `WebauthnLibServer` is used. Assertion must fail. That is a different server class, not a failed lab of this leftover.

08 Wrong turns, in the order I made them

POSTing WebAuthn JSON without a prior password session. Stays on login. PR:L is real. This is not unauthenticated PMA.

A signature that failed the structural checks. Challenge not from this session, origin host mismatch, user-present flag clear, credential id not in `allowCredentials`. Those checks run. The leftover is the one that does not.

`Webauthn\Server` **still present.** Wrong server class. The Dockerfile deletes `vendor/web-auth` and fails the build if the class exists. Without that, you lab the library path and write a FAIL that is about composer, not `CustomServer`.

Calling it unauthenticated RCE. It is not. You need the MySQL password. You get the SQL box as that user. You do not get a PHP shell on the host.

Treating "attestation: none" as this bug. Registration attestation is a different flag. Login assertion crypto is the leftover.

A reverse shell. Theatre. The oracle is a witness string in a 2FA POST on loopback.

09 What an attacker can do

They still need the MySQL password (cookie login). After that password, they POST `webauthn_request_response` with matching challenge, origin host, `rpIdHash`, user-present, enrolled credential id, and a garbage signature. They land in the normal UI. 2FA is gone. That is the database (SQL, users, export), not a PHP shell on the host.

PRIVILEGE	METHOD / PA
Complete WebAuthn 2FA without the authenticator (integrity of second factor)	POST `webauthn_r` after passw signature
Full PMA as that MySQL user (integrity + recon of the SQL box)	same sessi
Skip the MySQL password	unauthentic
Same bypass with `web-auth/webauthn-lib` installed	`WebauthnLi`
OS RCE / reverse shell	this leftover

Closed on this leftover: unauthenticated PMA, a PHP shell on the host, and treating a tarball that actually installed `web-auth/webauthn-lib` as this bug.

I am not going to print a forged PublicKeyCredential you can paste at someone else's PMA. If you own the instance, run the proof of concept against loopback.

10 The fix

Verify `authenticatorData || SHA-256(clientDataJSON)` with the stored COSE public key. Bind `userHandle`. Enforce `signCount`. Make `web-auth/webauthn-lib` a real dependency and delete `CustomServer`, or fail closed if the library is missing. Until then, disable WebAuthn 2FA or do not expose this phpMyAdmin.

A suggest / require-dev second factor is a comment. The tarball still offers the prompt.

Same pin leftovers: phpmyadmin-pmadb-central-sqli, phpmyadmin-designer-json-sqli, phpmyadmin-setup-validate, phpmyadmin-csrf-sql.

11 References

- Proof of concept: abraxas/phpmyadmin-webauthn · phpmyadmin-webauthn-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-347 · CWE-287 · 8.1 High
- Tree: phpmyadmin/phpmyadmin RELEASE_5_2_3 · `createServer` · `CustomServer`

Abraxas Labs

Independent security research by abraxas.

Analysis of 0-days, n-days, malware, and the

leftovers that still matter.

GitHub **@abraxas_null** **abraxas.null@proton.me** **abraxaslabs.tech**

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs