0-DAY NO CVE HIGH PUBLIC

phpMyAdmin 5.2.3 WebAuthn 2FA, privileged RCE

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 5.2.3 WebAuthn 2FA, privileged RCE	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; vendor patch not yet assigned	PUBLISHED 20 Sept 2026
UPDATED 20 Sept 2026	TAGS 0-day, phpmyadmin, webauthn, 2fa, cwe-347, cwe-287, privileged, rce

TL;DR

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

01 The signature is never checked

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/PMA-5.2.3-WebAuthn-2FA** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, config.inc.php, init.sql, run.sh. Authorized lab only. It talks to loopback.

This is **phpMyAdmin 5.2.3**, default tarball, no CVE number yet. Filed with phpMyAdmin security. CWE-347 and CWE-287. 8.1 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N`). PR:L because you need the MySQL password (cookie login). It is not unauthenticated. 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 RCE on the data.

`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.

HTTP is `POST /index.php?route=/` with `webauthn_request_response` after the password POST. Function names in the write-up are PHP methods. I am not printing a forged PublicKeyCredential you can paste at someone else's PMA.

This is the map I used to get from a WebAuthn prompt to a session that no longer asked for the key. Isolated lab, loopback only. The stack is in the GitHub repo so you can run it at home. The client is the repo above.

02 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 RCE on the data, not a PHP shell on the host.

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

03 The lab (run this at home)

Source of truth is `lab/` on GitHub. Dockerfile is `php:8.2-apache`. It copies phpMyAdmin 5.2.3, `composer install --no-dev`, then **deletes** `vendor/web-auth` so `CustomServer` is the one in play. Compose adds `mysql:8.0`. Published port in that YAML is **18080**. Bind it to loopback.

DOCKERFILE Copy

```
# phpMyAdmin 5.2.3 lab: CustomServer WebAuthn (no web-auth/webauthn-lib).
FROM php:8.2-apache

ENV COMPOSER_ALLOW_SUPERUSER=1

RUN apt-get update && apt-get install -y --no-install-recommends \
      git unzip libzip-dev libonig-dev \
  && docker-php-ext-install mysqli mbstring \
  && rm -rf /var/lib/apt/lists/* \
  && curl -fsSL https://getcomposer.org/installer | php -- --install-dir=/usr/lo

WORKDIR /var/www/html
COPY src/ /var/www/html/
COPY config.inc.php /var/www/html/config.inc.php

# Git clone lockfile still lists webauthn-lib. Official tarball does not ship it
# CustomServer is used iff Webauthn\\Server is absent.
RUN git config --global --add safe.directory /var/www/html \
  && composer install --no-dev --no-interaction --prefer-dist --no-scripts \
  && rm -rf vendor/web-auth \
  && composer dump-autoload -o --no-interaction \
  && mkdir -p /tmp/pma /var/www/html/tmp \
```

```
23    && chown -R www-data:www-data /tmp/pma /var/www/html/tmp \
24    && php -r 'require "vendor/autoload.php"; exit(class_exists("Webauthn\\Server"
```

The last `php -r` **exits 1** if `Webauthn\Server` exists. The image must fail the build if the lib is still there.

`docker-compose.yml`:

YAML Copy

```yaml
services:
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: appdb
      MYSQL_USER: appuser
      MYSQL_PASSWORD: apppass
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/01-lab.sql:ro
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "127.0.0.1", "-uroot", "-prootpa
      interval: 3s
      timeout: 5s
      retries: 30

  pma:
    build: .
    ports:
      - "127.0.0.1:18080:80"
    depends_on:
      db:
        condition: service_healthy
```

The GitHub YAML as uploaded uses `"18080:80"`. Bind `127.0.0.1:18080:80`. Cookie auth, `pmadb`, `pma__userconfig`. `init.sql` enrolls WebAuthn 2FA for `appuser` without a hardware key. `config.inc.php` points at that control user.

Bring-up from the CVE repo `lab/`. You need the 5.2.3 tree as `src/` next to the Dockerfile (see lab/README.md).

PLAIN TEXT Copy

```
git clone https://github.com/abraxas/PMA-5.2.3-WebAuthn-2FA
cd PMA-5.2.3-WebAuthn-2FA/lab
# place phpMyAdmin 5.2.3 as src/ (RELEASE_5_2_3 tarball)
./run.sh
# or:
docker compose up -d --build
python3 ../PMA-5.2.3-WebAuthn-2FA-Abraxas-Labs.py http://127.0.0.1:18080
```

Do not publish the port off loopback. The proof of concept is written for `127.0.0.1:18080`.

04 What the tree actually registers

`createServer`:

PHP Copy

```
private function createServer(): Server
{
    return class_exists(WebauthnServer::class) ? new WebauthnLibServer($this→two
}
```

Password login, then `WebAuthn::check` reads `webauthn_request_response` and the session challenge. It calls `parseAndValidateAssertionResponse`. On `CustomServer` that method:

PHP Copy

```
public function parseAndValidateAssertionResponse(
    string $assertionResponseJson,
    array $allowedCredentials,
    string $challenge,
    ServerRequestInterface $request
): void {
    $assertionCredential = $this→getAssertionCredential($assertionResponseJson)
```

```
        // credential id in allowCredentials
        // clientData.type == webauthn.get
        // challenge matches session
        Assert::same($host, parse_url($clientData['origin'], PHP_URL_HOST));
        Assert::true(hash_equals($rpIdHash, $authenticatorData['rpIdHash']));
        $isUserPresent = (ord($authenticatorData['flags']) & 1) === 0;
        Assert::true($isUserPresent);
    }
```

It parses `response.signature` (the field must exist and decode). It never COSE-verifies it. It never uses the stored `credentialPublicKey`. `CustomServerTest::testParseAndValidateAssertionResponse` accepts an assertion with **no public key**. That is the product's own test.

WebauthnLibServer is the other branch. If you actually installed `web-auth/webauthn-lib`, you are not on this CVE. The default tarball is.

There is no assigned vendor patch yet. Until there is: install `web-auth/webauthn-lib` so `createServer` uses the lib, or disable WebAuthn 2FA, or do not expose this phpMyAdmin.

05 The 200 that was still the 2FA form

The first client I pointed at this was polite. It POSTed a WebAuthn JSON without a prior password session. You get the login page. It POSTed `route=/` with a signature that did not even satisfy the structural checks. You stay on 2FA. `webauthn_request_response` is still in the HTML.

A few other ways to lose without learning anything:

- `Webauthn\Server` present. Wrong server class.
- Challenge not the one from this session. `hash_equals` fails.
- Origin host mismatch. Use the same host you logged into.

- User-present flag clear.
- Credential id not in `allowCredentials`. The lab seed enrolls one id; the assertion has to use it.
- Unauthenticated, no MySQL password. This is not that bug.
- A reverse shell. Theatre. The witness is: after the second POST, `still_2fa=False` and you landed in the normal UI.

The tell is the 2FA field gone and a phpMyAdmin navigation page, not a tiny JSON body. This is a form POST, not `admin-ajax`.

06 What I actually did

Treat the on-disk product as the spec. I read `createServer`, then `parseAndValidateAssertionResponse`, then the unit test that does not pass a public key. Proof of concept: PMA-5.2.3-WebAuthn-2FA-Abraxas-Labs.py.

Password, then forged assertion. POST username/password/token. Pull the session challenge from the 2FA page. POST `webauthn_request_response` with matching challenge, origin, rpIdHash, user-present, enrolled credential id, and a garbage signature. I am not printing that JSON here.

Witness. The second response is not the WebAuthn form. `landed=True`. stdout line `SUCCESS PMA-WEBAUTHN-BYPASS WebAuthn 2FA accepted with garbage signature`.

Last lab run, trimmed:

PLAIN TEXT Copy

```
password-login status=200 webauthn=True
2fa-post status=200 still_2fa=False landed=True
SUCCESS PMA-WEBAUTHN-BYPASS WebAuthn 2FA accepted with garbage signature
```

The client that produced it is on GitHub.

Wrong turns already recorded: `Webauthn\Server` present (wrong server class); challenge not the one from this session; origin host mismatch; user-present flag clear; credential id not in `allowCredentials`; unauthenticated, no MySQL password; a reverse shell. Theatre. The tell is the 2FA field gone and a phpMyAdmin navigation page.

07 What this is not

It is not "WebAuthn is broken." The assertion signature is the point of the protocol. This server class never asks the stored public key. It is not unauthenticated phpMyAdmin. You still need the MySQL password.

No CVE id yet. Apply whatever phpMyAdmin ships, or put `web-auth/webauthn-lib` in production so `CustomServer` is not selected. Re-run the loopback client against a patched build: the garbage signature must not land.

I am not going to print a PublicKeyCredential you can paste at someone else's `/index.php?route=/`. The missing COSE verify is the useful part. If you own the box, run the proof of concept against loopback.

Client bugs look like product bugs. A helper that shadows `http.client` never sends a packet. `.recv()` on an `HTTPResponse` is not `.read()`. I mention that once because it cost time and looked, for a minute, like phpMyAdmin was fine.

08 References

- Proof of concept: abraxas/PMA-5.2.3-WebAuthn-2FA · PMA-5.2.3-WebAuthn-2FA-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · init.sql · config.inc.php

- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-347 · CWE-287 · WebAuthn-3 verifying assertion
- 5.2.3: `CustomServer::parseAndValidateAssertionResponse`, `WebAuthn::createServer`, `CustomServerTest`
- Product: phpMyAdmin · github.com/phpmyadmin/phpmyadmin

© 2026 Abraxas Labs