0-DAY NO CVE HIGH PUBLIC

phpMyAdmin 5.2.3, Designer VQB history XSS

phpMyAdmin 5.2.3 Designer visual-query history concatenates table names into an h3 with innerHTML. Column names in the same block are escaped. TableName still allows < and >. A CREATE TABLE plant becomes markup in the victim origin. Labbed. No CVE yet.

NAME phpMyAdmin 5.2.3, Designer VQB history XSS	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR phpMyAdmin
AFFECTED phpMyAdmin 5.2.3 tag RELEASE_5_2_3 (962857e). Authenticated CREATE TABLE on a database the victim opens in Designer. PMASA-2025-1 did not touch this path.	PUBLISHED 12 Oct 2026
UPDATED 12 Oct 2026	TAGS 0-day, phpmyadmin, xss, designer, cwe-79, authenticated

TL;DR

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

01 The column helper never met the table name

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

This is **phpMyAdmin 5.2.3** (`962857e4f63d42e38f11ff4d63f5e722018add76`). No CVE yet. CWE-79. 8.0 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:N`). Remote stored XSS. Authenticated `CREATE TABLE`. The victim opens Designer.

PMASA-2025-1 escaped Check-tables index-problem headings. It did not touch Designer visual-query history. Column names in that history already go through `$('<div/>').text(...).html()`. Table names are concatenated into an `<h3>` and assigned with `innerHTML`. `TableName` still allows `<` and `>`. MySQL accepts them in a quoted identifier. The leftover is that mismatch, not a new class of unauthenticated RCE.

02 Why phpMyAdmin

I had already published the CustomServer WebAuthn leftover on this pin. The rest of the 5.2.3 tree was still sitting there: Designer, import plugins, privilege oracles, preference writers, pmadb, setup scripts that outlive `config.inc.php`. phpMyAdmin is a large PHP UI over MySQL with a long PMASA history. Cookie auth. A control user. JavaScript that still builds HTML by hand in 2026.

I only wanted leftovers that survived 5.2.2. Local is in-scope. Medium stays unless it chains. Unauthenticated RCE and OS LPE stayed empty on this pin.

03 Pinning RELEASE_5_2_3

Latest 5.2.x GA. Tag `RELEASE_5_2_3`. Commit `962857e4f63d42e38f11ff4d63f5e722018add76`. `Version::VERSION = 5.2.3`. Official image `phpmyadmin:5.2.3`. That is the pin.

6.x / master already quotes some pmadb blobs that 5.2.3 still concatenates. This leftover is not one of those. `history.js` is still the hand-built string on 5.2.3.

04 The skip list is half the hunt

Before you look for leftovers, you write down what is already public so you do not rediscover a PMASA with a press release.

- **PMASA-2025-1.** XSS in Check-tables index-problem headings. Escaped in 5.2.2. Not Designer VQB history.
- **PMASA-2025-2.** XSS on insert. 5.2.2. Insert BIT / `pma_type` leftovers looked incomplete. Confidence too low to treat as a new critical.
- **PMASA-2025-3.** iconv. 5.2.2.

Closed on this pin from the rest of the tree: classic `goto` LFI (allowlist is `index.php` only), `uri` / version-check HTTP (hardcoded phpmyadmin.net), external transformation `proc_open` (empty `$allowed_programs` by default). Unauthenticated RCE and OS LPE were not found.

Other 5.2.3 leftovers from the same review sit in their own packs: WebAuthn CustomServer (already public), unquoted `authentication_plugin`, `isSuperUser()` catalog SELECT, `/config/set` 2FA strip, ImportLdi `TERMINATED BY`. This post is the Designer history `<h3>`.

05 Surfaces that returned nothing

I split Designer into the pieces that look like XSS and the pieces that already learned. Relation lines, PDF schema, the table-list filter, and the VQB **OK** button path were the first pass. The option-button attribute is pre-escaped in Twig and then autoescaped. A click currently feeds entities into `getTab()`. That looked, for a minute, like the whole feature was safe.

It is not a fix of `history.js`. `.small_tab_pref_1` still copies `designerTable.getTableName()` into `table_name`. The live script still concatenates `getTab()` into an `<h3>`. One attribute got double-encoded. The sink did not.

06 The leftover

`js/src/designer/history.js` (shipped as `js/dist/designer/history.js`):

JAVASCRIPT Copy

```
temp = historyArray[i].getTab();
str += '<h3 class="tiger"><a href="#">' + temp + '</a></h3>';
// ...
'<td width="175" style="padding-left: 5px">' + $('<div/>').text(historyArray[i].g
```

Someone looked at column names and reached for the jQuery text helper. Someone looked at table names and concatenated. `TableName.php` only checks non-empty, max 64, no trailing space. `<` and `>` are legal.

`templates/database/designer/database_tables.twig` copies the raw MySQL name onto `.small_tab_pref_1`. Chromium does not have to invent the identifier. It is already in the DOM. `DesignerHistory.display()` writes it into `#ab`.

07 Wrong turns in the order they happened

I treated the double-escaped option-button attribute as the fix. It is not. The OK click currently feeds entities. The history sink is still `getTab()` plus `innerHTML`.

I treated the column helper as covering table names. Same function, two fields, one escaped. That is the leftover.

I considered claiming unauthenticated XSS. Designer is behind cookie login. The plant is `CREATE TABLE` on a shared database. UI:R is the higher-priv user opening Designer.

Theatre: a reverse shell. The oracle is an `<i id="pma-vqb">` node in the history `<h3>`, and a column cell that stays text.

08 The lab

phpmyadmin:5.2.3 + mysql:8.0 on loopback. Two users, shared `shareddb`. Playwright Chromium in the `poc` service because the sink is a browser `innerHTML`. Host bind **127.0.0.1:19250**. Compose project `phpmyadmin-vqb-xss`.

Low-priv `creator` runs `CREATE TABLE \`t\x\`(\`cy\` INT)`. Higher-priv ***victim*** `logs into phpMyAdmin and opens Designer with` ***query=1***`. Live` ***history.js*** `is the copy from the container (` ***h3-unescaped=yes***`, column helper present). Chromium calls` ***DesignerHistory.display()*** `with the table name from` ***getTableName()`*** ***on the page.***

Result: `#ab h3.tiger` ***innerHTML is*** `<a href="#">t<i id="pma-vqb">x</i></a>`***. Column td text is*** `c<b id=pma-col>y` ***with no*** `b#pma-col` ***node. Markup only. Not a script payload.***

PLAIN TEXT Copy

```
python3 phpmyadmin-vqb-xss-Abraxas-Labs.py
# or: cd lab && ./run.sh
```

Last line:

PLAIN TEXT Copy

```
SUCCESS phpmyadmin-vqb-xss h3-markup=yes col-escaped=yes how=live PMA-VQB-XSS-WI
```

09 What an attacker can do

Plant an HTML table name, then wait for a higher-privilege user to open Designer on that database and render visual-query history.

PRIVILEGE	METHOD / SINK
Inject markup into the phpMyAdmin origin	`DesignerHistory.display()` *concatenates* `getTab()` *int* `#ab` *innerHTML*
Break out of a designer `table_name` *attribute*	`database_tables.twig` `small_tab_pref_1` *uses* `designerTable.getTableNam`
Read the victim session cookie and CSRF `token`	*Same-origin script once his renders*
Run SQL, manage users, export, as the victim MySQL account	*Token-backed POST* `/sql` *,* `/server/privileges` *, expo*

PRIVILEGE	METHOD / SINK
Change 2FA / userconfig	POST `/config/set` as the victim

Closed on this leftover: unauthenticated access, OS RCE, and the VQB OK button path. Column cells stay text-escaped. This is not a PHP shell on the host. It is the session of the victim MySQL user through the phpMyAdmin origin.

10 The fix

Escape `getTab()` with `$('<div/>').text(temp).html()` or `Functions.escapeHtml` before concatenating into the `<h3>`. Do not rely on Twig double-encoding of one attribute. Keep the column-name helper as the model.

11 Contact

Abraxas Labs - abraxaslabs.tech - github.com/abraxas - @abraxas_null - abraxas.null@proton.me

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**

Home Research Open Source Reading Room About GitHub @abraxas_null Console

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

© 2026 Abraxas Labs