

thorsten / phpMyFAQ Public[Code](#) [Issues](#) 11 [Pull requests](#) 1 [Discussions](#) [Actions](#) [Projects](#)

# Stored XSS via Unsanitized Email Field in Admin FAQ Editor

High thorsten published **GHSA-98gw-w575-h2ph** 5 days ago

## Package

**phpMyFAQ**

### Affected versions

&lt; 4.1.0

### Patched versions

4.1.1

## Description

### Summary

An unauthenticated attacker can submit a guest FAQ with an email address that is syntactically valid per RFC 5321 (quoted local part) yet contains raw HTML — for example "`<script>alert(1)</script>`"@evil.com. PHP's `FILTER_VALIDATE_EMAIL` accepts this email as valid. The email is stored in the database without HTML sanitization and later rendered in the admin FAQ editor template using Twig's `|raw` filter, which bypasses auto-escaping entirely.

### Details

1. PHP `FILTER_VALIDATE_EMAIL` accepts RFC-valid quoted local parts with dangerous characters

```
phpmyfaq/src/phpMyFAQ/Controller/Frontend/Api/FaqController.php:99
```

```
$email = trim((string) Filter::filterVar($data->email, FILTER_VALIDATE_EMAIL));
```

PHP accepts "`<script>alert(1)</script>`"@evil.com as a valid email (RFC 5321 allows `<`, `>` inside quoted local parts). Confirmed:

```
"<script>alert(1)</script>"@evil.com => string (valid, not false)
```

2. Email stored raw without HTML sanitization

```
phpmyfaq/src/phpMyFAQ/Faq.php — email retrieved directly as $row->email from the database.
```

3. Admin Twig template renders email with `|raw`

```
phpmyfaq/assets/templates/admin/content/faq.editor.twig:296
```

Affected version: 4.2.0-alpha, commit [f0dc86c](#)

## PoC

The reproduction of the vulnerability was implemented with the help of AI while reviewing the source code to generate the proof-of-concept. Please kindly note this for reference. Since the vulnerability has already been confirmed directly in the source code, the proof-of-concept code may be considered as a reference only.

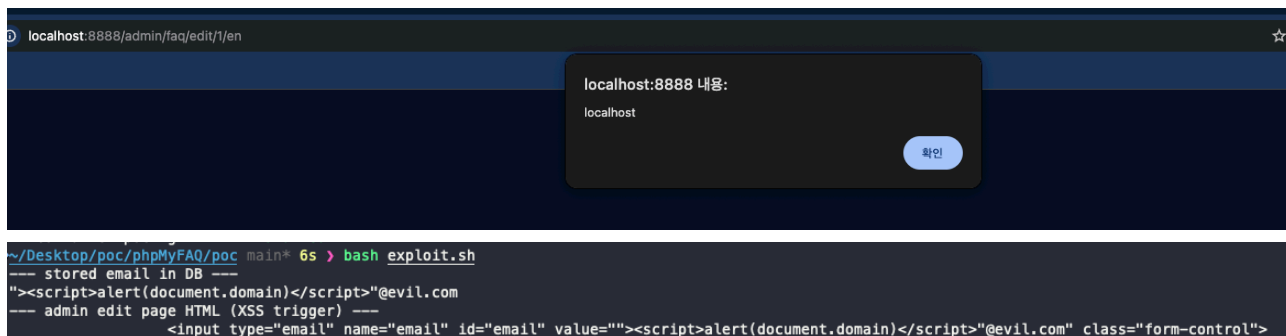
Please extract the attached compressed file and proceed.

[poc.zip](#)

0. (docker compose -f docker-compose.yml down -v)
1. docker compose -f docker-compose.yml up -d mariadb php-fpm nginx
2. bash exploit.sh

1. Access <http://localhost:8888/admin/>
  2. Log in with admin / Admin1234!
  3. After logging in, check whether the URL remains <http://localhost:8888/admin/>
  4. Go to Content → FAQ Administration → edit "poc" → alert popup should appear
- If it does not appear, you can also access it directly via:

<http://localhost:8888/admin/faq/edit/1/en>



## Impact

When an administrator opens `/admin/faq/edit/{id}/{lang}` to review the pending FAQ, the injected script executes in the admin's browser context. This allows an attacker to:

- Steal the administrator's session cookie → full admin account takeover
- Perform arbitrary admin actions (create users, modify content, change configuration)
- Pivot to further attacks on the server

The attack chain requires no authentication. By default, `records.allowNewFaqForGuests=true` allows unauthenticated FAQ submission, and `records.defaultActivation=false` guarantees the administrator must visit the edit page to review it.

Note on captcha: The built-in captcha is enabled by default when the PHP gd extension is present (spam.enableCaptchaCode=true). This prevents fully automated exploitation but does not prevent a targeted manual attack — an attacker can solve the captcha once and submit the payload.

## Credits

wooseokdotkim

## Severity

High

## CVE ID

CVE-2026-32629

## Weaknesses

- ▶ CWE-20
- ▶ CWE-79