Blogs Categories Full Account Takeover for All-Roles & Privilege-Escalation via Stored DOM XSS

Critical **bertugfahriozer** published **GHSA-fhrf-q333-82fm** last week

Package

php **ci4-cms-erp/ci4ms** (Composer)

Affected versions	Patched versions
<= 0.28.6.0	0.31.0.0

Description

Summary

Vulnerability: Stored DOM XSS via Blog Category Title (Persistent Payload Injection)

- Stored Cross-Site Scripting via Unsanitized Blog Category Title in Blog Management

Description

The application fails to properly sanitize user-controlled input when creating or editing blog categories. An attacker can inject a malicious JavaScript payload into the category title field, which is then stored server-side.

This stored payload is later rendered unsafely across public-facing blog category pages, administrative interfaces, and blog post views without proper output encoding, leading to stored cross-site scripting (XSS).

Affected Functionality

- Blog category creation functionality
- Blog category editing functionality

- Blog category storage and retrieval logic

Attack Scenario

- An attacker creates or edits a blog category title to include a malicious XSS payload.
- The application stores this value without sanitization or encoding.
- The payload persists and executes whenever the category title is rendered in affected views.

Impact

- Persistent Stored XSS
- Execution of arbitrary JavaScript in victims' browsers
- Privilege escalation when viewed by administrators or privileged users
- Full administrator account takeover
- Full account takeover across all roles
- Full compromise of the entire application

Endpoints:

- `/backend/blogs/categories/`
- `/blog/{id}`

Steps To Reproduce (POC)

1. Go to the Blog Categories management page
2. Create or edit a category and insert an XSS payload into the category title such as: `<img src=x onerror=alert(document.domain)>`
3. Save the category
4. View a public blog category page, blog post page, or the administrative interface
5. Notice the XSS payload executing automatically

Remediation

- Never use .html() again or any innerHTML-style like JS in your PHP, or any other sink, even if user inputs that flow into them are not clear, they still represent real world danger as an attacker can make use of this to exploit the application via XSS. And do HTML Encoding as much as possible and always do Sanitization, theres no sanitization there unfortunately. Also apply CSP, HttpOnly, SameSite, and Secure upon all application, they reduce severity of XSS & escalated-CSRF via XSS and do great jobs

Ready Video POC:

https://mega.nz/file/GAFC3AJY#3LHyuyl7I7921UEeA-JlUYdckh6zGLCTy-6w9BNzSmQ

Severity

Critical 10.0 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

CVE ID

CVE-2026-34569

Weaknesses

▶ CWE-79

Credits

 **bugmithlegend** Reporter

 **peeefour** Reporter