Backup Management Full Account Takeover for All-Roles & Privilege-Escalation via Stored DOM Blind XSS

Critical **bertugfahriozer** published **GHSA-85m8-g393-jcxf** 5 days ago

Package

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

Affected versions	Patched versions
<= 0.28.6.0	0.31.0.0

Description

Summary

Vulnerability: Stored DOM Blind XSS via Backup Management Filename (Persistent Payload Injection)

- Stored Cross-Site Scripting (Blind XSS) via Unsanitized Backup Filename in Backup Management

Description

The application fails to properly sanitize user-controlled input when handling backup uploads and processing backup metadata. An attacker can inject a malicious JavaScript payload into the backup filename via the uploaded `xss.sql`, which uses SQL functionality to insert the XSS payload server-side.

This stored payload is later rendered unsafely in multiple backup management views without proper output encoding, leading to stored blind cross-site scripting (Blind XSS).

Affected Functionality

- Backup upload functionality
- Backup processing functionality

- Backup storage and retrieval logic

Attack Scenario

- An attacker uploads `xss.sql` which uses SQL functionality to insert a malicious XSS payload into the backup filename field server-side.
- The application stores this filename without sanitization or encoding.
- The payload persists and executes whenever the backup filename is rendered in affected views.
- The attacker does not see immediate execution, making this a Blind XSS scenario that triggers only when an administrator or privileged user views the backup management panel.

Impact

- Persistent Stored Blind 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/backup/upload`
- `/backend/backup/`
- `/backup/{id}`

Steps To Reproduce (POC)

1. Upload `xss.sql` via the Backup Upload functionality
2. Ensure the SQL executes and inserts an XSS payload into the backup filename field such as: `<img src=x onerror=alert(document.domain)>`
3. Navigate to the Backup Management panel as an administrator
4. View the backup entry via the administrative panel
5. Notice the XSS payload executing automatically (Blind XSS)

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/eNFXgAAA#IETbPcKwr5vVLqJIAdc3uy4qgcVTgyPb_2HhB4zcwAE

Ready File: xss.sql

(You may need to change the database name in the file or the application, in order to execute successfully without any errors)

Severity

Critical 9.1 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-34563

Weaknesses

▶ CWE-79

Credits

bugmithlegend Reporter