Paulo Cesar · há 2 dias · 2 min de leitura

CVE-2025-29573: Persistent XSS in Mezzanine CMS 6.0.0 via Malicious Filename

Atualizado: há 17 horas

Status: CVE published

Date of discovery: March 5, 2025

Researcher: Paulo Cesar (PC) – Squad AppSec

Last update: No response received from project maintainers

Description

A **Persistent Cross-Site Scripting (XSS)** vulnerability has been identified in **Mezzanine CMS v6.0.0**, specifically in the **"View Entries"** feature of the **Forms** module.

The issue occurs when an administrator views form submission results that include a **File Upload field**. The name of the uploaded file is not properly sanitized, allowing an attacker to include a **malicious JavaScript payload** in the filename. This code executes in the administrator's browser whenever the file is viewed.

The root cause is the **unsafe rendering** of the filename through direct HTML interpolation without proper escaping.

Affected Product

- **Component:** Mezzanine CMS (GitHub)
- **Version:** 6.0.0
- **Affected module:** Forms
- **Vulnerable route: /admin/forms/entries/** (admin interface for viewing form entries)

Vulnerable Code

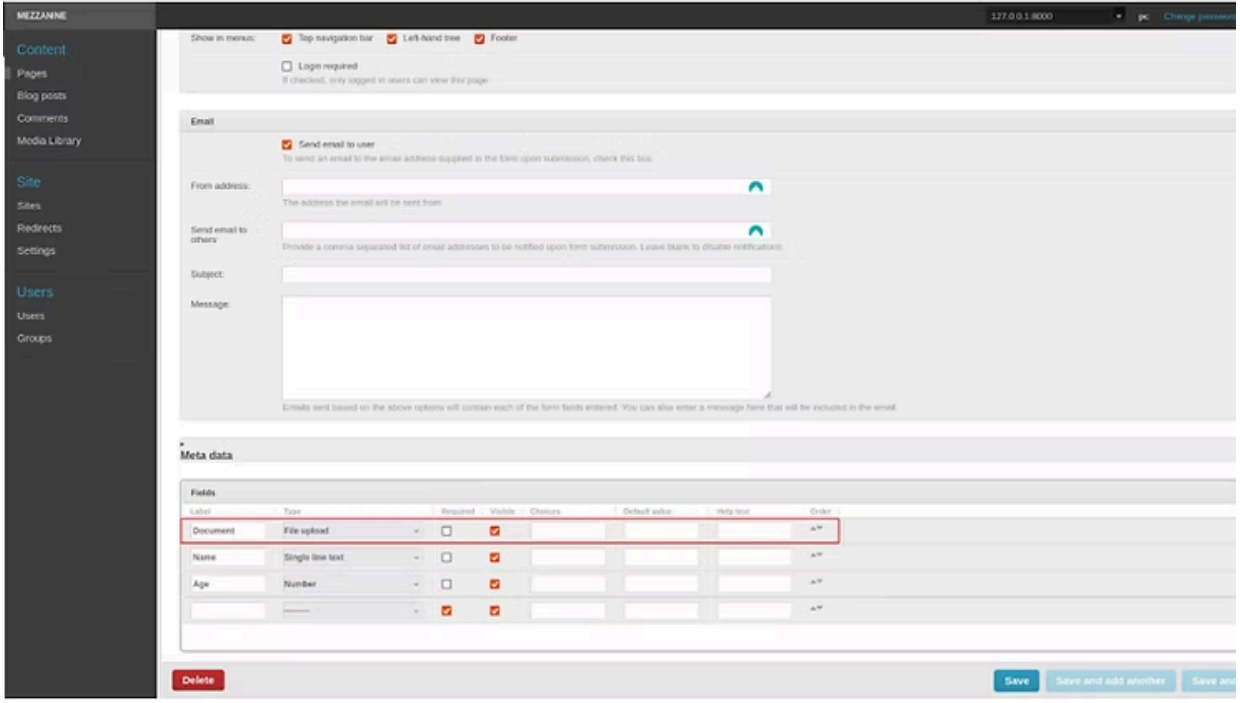
The vulnerability is related to the following line in **mezzanine/forms/forms.py**, line 435:
field_value = mark_safe('%s' % parts)

```
429             # Create download URL for file fields.
430             if field_entry.value and field_id in file_field_ids:
431                 url = reverse( viewname: "admin:form_file", args=(field_entry.id,))
432                 field_value = self.request.build_absolute_uri(url)
433                 if not csv:
434                     parts = (field_value, split(field_entry.value)[1])
435                     field_value = mark_safe('<a href="%s">%s</a>' % parts)
436             # Only use values for fields that were selected.
```

The **mark_safe** function in Django instructs the template engine not to escape the content, assuming it is safe. When combined with untrusted input, such as filenames submitted by users, this function can lead to malicious code execution on the client side.

Proof of Concept (PoC)

1. The admin creates a form in Mezzanine CMS containing a **File Upload** field.



2. The attacker fills out the form and uploads a file with a **malicious filename**, such as:

<img src='x' onerror='alert(document.cookie);'>

3. The administrator accesses the form entries view page in the Mezzanine admin panel. (Note that the malicious JavaScript code is executed when the file is displayed.)

Impact

This vulnerability is considered **high severity**, as it affects the **administrative interface** of Mezzanine CMS — an area with elevated privileges and access to sensitive data.

By exploiting this flaw, an attacker can execute **arbitrary JavaScript** in the browser of authenticated admins, leading to:

- **Session hijacking** (e.g., stealing authentication cookies or CSRF tokens)
- **Privilege escalation**, if internal APIs are abused
- **UI manipulation**, such as injecting rogue forms or phishing links
- **Unauthorized actions**, such as deleting or altering content via session riding

Since this is a **persistent XSS**, the payload remains stored in the system and is executed every time an admin views the affected entry, increasing both exposure time and risk.
In multi-admin environments, this could propagate across accounts, making detection difficult and potentially compromising the entire application.

Mitigation & Recommendations

- Apply a patch if and when it becomes available from the project maintainers.
- Always **sanitize file names** and other user inputs before rendering them as HTML.
- **Avoid using mark_safe** on untrusted content unless properly escaped beforehand.

CVE Details

- **ID:** CVE-2025-29573

- **Type:** Persistent Cross-Site Scripting (XSS)
- **Severity:** High
- **Public reference:** This article
- **Reported by:** Paulo Cesar – Squad AppSec
- **Status:** Reserved (not yet published on cve.org)

Contact

For questions, coordinated disclosure, or partnerships:

📧 contato@squadappsec.com

🌐 https://squadappsec.com