Stored XSS via Unrestricted Asset Upload

High **enchant97** published **GHSA-9pr4-rf97-79qh** 5 days ago

Package

note-mark

Affected versions

v0.19.1

Patched versions

v0.19.2

Description

Summary

A stored same-origin XSS vulnerability allows any authenticated user to upload an HTML, SVG, or XHTML file as a note asset and have it executed in a victim's browser under the application's origin. Because the application serves these files inline without a safe content type and without `nosniff`, browsers can sniff and render active content, giving the attacker access to authenticated Note Mark API actions as the victim.

Details

This issue results from three compounding flaws in the asset handling and delivery path.

1. Asset delivery can be used as an attack vector

The asset delivery route can be used to deliver attacker-controlled uploaded content directly to a victim by URL.

Relevant route:

- `handlers/assets.go:40`

```
huma.Get(api, "/api/notes/{noteID}/assets/{assetID}", h.GetNoteAssetContentByID)
```

This makes the uploaded asset reachable by direct navigation, which provides the delivery mechanism for the payload.

2. Text-based active content is served with an empty `Content-Type`

The asset handler relies on `h2non/filetype` for content-type detection:

- `handlers/assets.go:147`

```
kind, _ := filetype.Match(buf)
if kind != filetype.Unknown {
    contentType = kind.MIME.Value
}
```

The detection library uses magic-byte matching and does not identify text-based formats such as HTML, SVG, JavaScript, XML, or XHTML. For those files, `filetype.Match` returns `Unknown`, leaving `Content-Type` unset or empty.

As a result, uploaded active content is served without an authoritative MIME type.

3. Files are rendered inline and sniffed by the browser

The asset response is sent with inline disposition:

- `handlers/assets.go:153`

```
w.Header().Set("Content-Disposition", fmt.Sprintf("inline; filename=\"%s\"", asset)
```

At the same time, the response does not set:

```
X-Content-Type-Options: nosniff
```

This combination is dangerous:

- the uploaded file contains attacker-controlled active markup
- the browser is instructed to render it inline
- the response does not provide a trustworthy content type
- content sniffing is not disabled

Under these conditions, browsers may detect HTML or SVG content and execute embedded JavaScript. Because the asset is served from the application's own origin, the script runs with same-origin access to the application and its authenticated APIs.

This turns an uploaded asset into a stored XSS payload that executes when a victim opens the asset URL.

PoC

The issue can be reproduced by uploading a text-based active content file such as HTML or SVG as a note asset, then opening the served asset URL in a browser and observing that script executes in the context of the application origin.

Impact

- **Type:** Stored same-origin cross-site scripting (XSS)
- **Who is impacted:** Any user who can be induced to open a malicious asset URL, and any deployment allowing asset uploads
- **Security impact:** An attacker can execute JavaScript in the victim's authenticated application context, allowing access to private notes, books, profile data, and authenticated API actions
- **Privileges required:** A valid low-privilege user account capable of uploading note assets
- **User interaction:** Required, because the victim must navigate to the malicious asset URL
- **Scope:** Changed, because attacker-controlled content executes in the victim's origin and impacts other users rather than remaining confined to the attacker's own account

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-2026-40262

Weaknesses

- CWE-79
- CWE-434

Credits

QiaoNPC	Reporter
Across-Verticals-Malaysia	Reporter
enchant97	Remediation developer