New issue

SSRF and Path Traversal Vulnerabilities in xhs_publish_content of xhs-mcp #6

Open

BruceJqs opened 2 weeks ago

SSRF and Path Traversal Vulnerabilities in xhs_publish_content of xhs-mcp

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: Apr 14, 2026

2) Reporter Contact

- Reporter name: `BruceJin`
- Reporter email: `brucejin@zju.edu.cn`
- Permission to share contact with vendor: `Yes`

3) Vendor / Product Identification

- Vendor: Algovate
- Product: xhs-mcp
- Repository: https://github.com/Algovate/xhs-mcp
- Affected component(s):
- `src/shared/image-downloader.ts`
- `src/server/handlers/tool.handlers.ts`

- `src/core/publishing/publish.service.ts`
- `src/core/publishing/publish-image.service.ts`
- `src/core/publishing/publish-video.service.ts`

4) Vulnerability Type

- CWE: CWE-918 (Server-Side Request Forgery)
- CWE: CWE-22 (Improper Limitation of a Pathname to a Restricted Directory)
- Short title: SSRF via image URL download and path traversal via local video path in `xhs_publish_content`

5) Affected Versions

- Confirmed affected: `0.8.11`
- Suspected affected range: revisions containing the same request-to-outbound-fetch and request-to-local-path-access flows listed below
- Fixed version: Not available at time of report

6) Vulnerability Description

A server-side request forgery (SSRF) vulnerability (CWE-918) and a path traversal vulnerability (CWE-22) have been identified in xhs-mcp version 0.8.11, specifically within the xhs_publish_content MCP tool. The SSRF issue arises when a user-supplied media_paths value begins with http:// or https://, causing the server to fetch the URL via fetch() without allowlisting or destination validation, enabling requests to internal or loopback services. The path traversal issue allows an attacker to supply a local video path containing ../ sequences that escape the project directory, leading to unintended filesystem access. An attacker with access to the MCP interface can exploit these flaws to interact with internal network endpoints and access local files outside the intended workspace. No fixed version is available at the time of reporting.

7) Technical Root Cause

1. `js/request-forgery-from-request`
    - Source: `src/server/mcp.server.ts:46` (`request`)
    - Propagation: `src/server/mcp.server.ts:47` (`const { name, arguments: args } = request.params;`)
    - Source: `src/server/handlers/tool.handlers.ts:239` (`case 'xhs_publish_content':`)
    - Source: `src/server/handlers/tool.handlers.ts:244` (`args?.media_paths as string[]`)
    - Propagation: `src/server/handlers/tool.handlers.ts:167` (`this.publishService.publishContent(...)`)
    - Propagation: `src/core/publishing/publish.service.ts:45` (`if (type === 'image')`)

- Propagation: `src/core/publishing/publish.service.ts:46` (`return this.publishNote(title, content, mediaPaths, tags, browserPath);`)
- Propagation: `src/core/publishing/publish-image.service.ts:175` (`validateAndResolveImagePaths(imagePaths)`)
- Propagation: `src/core/publishing/publish-image.service.ts:177` (`this.imageDownloader.processImagePaths(imagePaths)`)
- Propagation: `src/shared/image-downloader.ts:302` (`processImagePaths(imagePaths)`)
- Propagation: `src/shared/image-downloader.ts:307` (`for (const path of imagePaths)`)
- Propagation: `src/shared/image-downloader.ts:309` (`urlsToDownload.push(path);`)
- Propagation: `src/shared/image-downloader.ts:263` (`downloadImages(imageUrls)`)
- Propagation: `src/shared/image-downloader.ts:265` (`imageUrls.map((url) =>`)
- Propagation: `src/shared/image-downloader.ts:266` (`this.downloadImage(url)`)
- Sink: `src/shared/image-downloader.ts:156`
- Sink code: `const response = await fetch(imageUrl, {`

2. `js/file-access-from-request`

- Source: `src/server/handlers/tool.handlers.ts:244` (`args?.media_paths as string[]`)
- Propagation: `src/core/publishing/publish.service.ts:48` (`return this.publishVideo(title, content, mediaPaths[0], tags, browserPath);`)
- Propagation: `src/core/publishing/publish-video.service.ts:207` (`validateAndResolveVideoPath(videoPath)`)
- Propagation: `src/core/publishing/publish-video.service.ts:208` (`const resolvedPath = join(process.cwd(), videoPath);`)
- Sink: `src/core/publishing/publish-video.service.ts:210`
- Sink code: `if (!existsSync(resolvedPath)) {`
- Sink: `src/core/publishing/publish-video.service.ts:214`
- Sink code: `const stats = statSync(resolvedPath);`

8) Attack Prerequisites

- Attacker can invoke MCP tools exposed by the affected xhs-mcp server.
- The `xhs_publish_content` tool is reachable by the attacker.
- For SSRF, the server has network egress to attacker-chosen or internal HTTP/HTTPS targets.
- For SSRF, the target URL must return a valid image payload because the downloader validates image magic bytes after the HTTP request.
- For path traversal, the server process has filesystem access to the attacker-chosen local media path.
- No effective authentication, authorization, egress filtering, URL allowlist, or runtime policy blocks attacker-controlled image URLs before `fetch`.
- No effective workspace/media-root boundary check blocks attacker-controlled traversal paths before `existsSync` and `statSync`.

9) Proof of Concept / Reproduction Guidance

This proof of concept provides a concise, CVE-style reproduction example for the reported issue.

1. Start a local HTTP service that returns a valid PNG image

```
mkdir -p /tmp/xhs-ssrf
printf 'iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAQAAAC1HAwCAAAAC0lEQVR42mP8/x8AAwMCAO+/p9sAAAA...OR
python3 -m http.server 8765 --directory /tmp/xhs-ssrf
```

2. Start the affected server with MCP Inspector

```
cd xhs-mcp
npm install
npm run build
npx @modelcontextprotocol/inspector node dist/xhs-mcp.cjs mcp
```

3. Reproduction request
   Call the `xhs_publish_content` tool in MCP Inspector with the following arguments:

```
{
  "type": "image",
  "title": "test",
  "content": "test",
  "media_paths": ["http://127.0.0.1:8765/poc.png"],
  "tags": "poc"
}
```

4. Validation

- Confirm that the Python HTTP server logs a request similar to `GET /poc.png HTTP/1.1`.
- The later publishing workflow may fail because of browser setup, login state, or XiaoHongShu page state. That does not affect the SSRF result, because the server-side request to the attacker-controlled URL occurs before browser-based publication.

5. Path traversal validation
   Create a file outside the project directory:

```
mkfile -n 501m /tmp/xhs-traversal-too-large.mp4
```

If `mkfile` is unavailable:

```
truncate -s 501M /tmp/xhs-traversal-too-large.mp4
```

Call `xhs_publish_content` with:

```
{
  "type": "video",
  "title": "test",
  "content": "test",
  "media_paths": ["../../../../../tmp/xhs-traversal-too-large.mp4"],
  "tags": "poc"
}
```

Confirm that the response reports a file size validation error such as `Video file too large: 501.00MB. Maximum allowed: 500MB`, demonstrating that the server accessed a file outside the project directory.

10) Security Impact

- Confidentiality: High (SSRF may allow interaction with internal or loopback HTTP services reachable from the server process; path traversal may expose filesystem metadata and allow unintended local media files to enter the upload workflow).
- Integrity: Low (attacker can cause server-side requests, influence local image cache contents with valid image responses, and select local media files outside the intended project directory for subsequent processing).
- Availability: Low (attacker can induce outbound requests and filesystem access, bounded by timeout and file size checks).
- Scope: Changed.

11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:L`
- Suggested base score: 9.3 (Critical)
- Adjust `PR` upward if the vulnerable MCP tool is strictly authenticated and only available to trusted local users.

12) Workarounds / Mitigations

- Disable URL-based media downloads in `xhs_publish_content` when the MCP server is reachable by untrusted callers.
- Disable or restrict local media path support when the MCP server is reachable by untrusted callers.
- Enforce strict URL allowlists for permitted image hosts.
- Block loopback, link-local, RFC1918, IPv6 local, and cloud metadata destinations after DNS resolution and redirect handling.
- Limit redirects or re-validate every redirect target before following it.
- Add authentication and authorization for publishing tools.
- Run the MCP server with least-privilege permissions and network egress controls.

- For local media paths, restrict file access to an explicit workspace or media directory.

13) Recommended Fix

- Eliminate the request-to-arbitrary-fetch data flow documented above.
- Validate image URLs with an allowlist of schemes and hosts, not only a `http://` or `https://` prefix check.
- Resolve DNS and reject private, loopback, link-local, multicast, and metadata IP ranges before issuing the request.
- Re-validate URL targets after redirects.
- Consider requiring users to provide local media files only, or proxy image downloads through a hardened downloader with explicit egress policy.
- For local video and image paths, resolve paths against a trusted media directory and verify that the resolved path remains inside that directory.
- Add regression tests proving URLs such as `http://127.0.0.1:8765/poc.png` and paths such as `../../../../../tmp/xhs-traversal-too-large.mp4` are rejected before network or filesystem access.
- Publish a maintainer security advisory once a patch is released.

14) References

- Repository: https://github.com/Algovate/xhs-mcp
- Reviewed source file: `src/shared/image-downloader.ts`
- Reviewed source file: `src/server/handlers/tool.handlers.ts`
- Reviewed source file: `src/core/publishing/publish.service.ts`
- Reviewed source file: `src/core/publishing/publish-image.service.ts`
- Reviewed source file: `src/core/publishing/publish-video.service.ts`
- CWE-918: https://cwe.mitre.org/data/definitions/918.html
- CWE-22: https://cwe.mitre.org/data/definitions/22.html

15) Credits

- Discoverer: `BruceJin`
- Discovery method: Static analysis (CodeQL) plus repository source-code audit and dynamic reproduction with MCP Inspector

16) Additional Notes for Form Mapping

- Audit verdict: Exploitable: attacker-controlled `media_paths` image URLs can reach a server-side outbound `fetch` sink, and attacker-controlled local video paths can escape the project directory before filesystem access.

- Dynamic exploit replay status: completed successfully with MCP Inspector using a loopback image URL and an external `/tmp` video path.
- Maintainer should validate release mapping before coordinated disclosure.

For furthermore information, please refer to BruceJqs/public_exp#21

Sign up for free **to join this conversation on GitHub.** Already have an account? Sign in to comment

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants