New issue

AiraHub2 Server-Side Request Forgery via agent and hub URL parameters #39

Open

wing3e opened 3 weeks ago Owner

AiraHub2 Server-Side Request Forgery via agent and hub URL parameters

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: IhateCreatingUserNames2
- Product: AiraHub2
- Repository: https://github.com/IhateCreatingUserNames2/AiraHub2
- Reviewed local source path: `datasets_set/003/IhateCreatingUserNames2-AiraHub2`
- Affected component(s):
- `AiraHub.py`

- `aira_hub.py`

4) Vulnerability Type

- CWE: CWE-918 (Server-Side Request Forgery)
- Short title: AiraHub2 Server-Side Request Forgery via agent and hub URL parameters

5) Affected Versions

- Confirmed affected: 3e4b77fd7d48ed811ffe5b8d222068c17c76495e
- Suspected affected range: versions containing the same input-to-sink flow documented below
- Fixed version: Not available at time of report (April 2, 2026)

6) Vulnerability Description

Multiple endpoints perform outbound HTTP requests to URLs derived from user-controlled input (`agent_url`, `hub_urls`, and registration URL fields) without SSRF defenses such as host allowlists or internal-network blocking. This permits attackers to coerce the server into probing internal services or accessing cloud metadata endpoints.

7) Technical Root Cause

1. `/connect/stream` consumes request parameter `agent_url` and calls `client.stream('GET', agent_url, ...)`.
2. `/admin/sync_agents` iterates request body `hub_urls` and requests `f'{hub_url}/agents'`.
3. Registration logic in `aira_hub.py` fetches `agent_card_url` derived from user-provided agent URL with no network egress policy checks.

```
# AiraHub.py
@app.post("/connect/stream")
async def connect_stream_endpoint(request: Request, agent_url: str, name: str, ...):
    async with httpx.AsyncClient() as client:
        async with client.stream("GET", agent_url, headers={"Accept": "text/event-stream"}) a
            ...

@app.post("/admin/sync_agents")
async def sync_agents(request: Request):
    data = await request.json()
    hub_urls = data.get("hub_urls", [])
    ...
    response = await client.get(f"{hub_url}/agents")

# aira_hub.py
response = await client.get(agent_card_url)
```

8) Attack Prerequisites

- Attacker can send requests to the vulnerable endpoint or MCP tool interface.
- Deployment does not enforce compensating controls that block the demonstrated payload.
- Vulnerable code path remains enabled in runtime configuration.

9) Proof of Concept / Reproduction Guidance

1. Send an admin sync request with attacker-controlled URL target.
2. Observe outbound request from the hub server to the supplied target.

PoC request:

```
curl -sS -X POST http://TARGET_HOST:8000/admin/sync_agents   -H 'Content-Type: applicat js
```

Alternative PoC using stream endpoint:

```
curl -sS -X POST   "http://TARGET_HOST:8000/connect/stream?agent_url=http://127.0.0.1:2 ev
```

Expected behavior:

- The hub performs server-side requests to attacker-supplied URLs.
- Internal-only endpoints become reachable from the hub context.

10) Security Impact

- Confidentiality: High (potential internal service and metadata exposure).
- Integrity: Medium to High (chained attacks may target internal admin APIs).
- Availability: Medium (internal service probing may cause load and instability).
- Scope: Unchanged to Changed depending on surrounding deployment topology.

11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:L (8.8, depending on admin endpoint exposure)`
- Final score should be adjusted by CNA/vendor based on real deployment exposure.

12) Workarounds / Mitigations

- Restrict access to vulnerable interfaces (network ACL, authn/authz, mTLS).
- Reject untrusted metacharacters/URLs and enforce strict server-side allowlists.

- Add regression tests for all PoC payload patterns included in this report.

13) Recommended Fix

- Remove direct execution/fetch of attacker-controlled values.
- Use safe APIs (`subprocess.run` with argument arrays and `shell=False`; strict URL policy checks for outbound requests).
- Enforce endpoint-level authentication and authorization before reaching sensitive sinks.

14) References

- Repository: https://github.com/IhateCreatingUserNames2/AiraHub2
- Reviewed source location: `datasets_set/003/IhateCreatingUserNames2-AiraHub2`
- SARIF evidence references:
- `py/mcp-partial-ssrf` at `AiraHub.py:1298`
- `py/mcp-partial-ssrf` at `AiraHub.py:1718`
- `py/mcp-partial-ssrf` at `aira_hub.py:1333`
- `py/mcp-partial-ssrf` at `aira_hub.py:1864`
- `py/mcp-partial-ssrf` at `aira_hub.py:1964`
- `py/mcp-partial-ssrf` at `aira_hub_OLD.py:1597`

15) Credits

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

16) Additional Notes for Form Mapping

- Issue status at report time: source-code confirmed in local dataset and exploitation path documented with explicit PoC.
- Dynamic verification was represented through deterministic command/URL payloads suitable for lab reproduction.
- Version-range precision should be finalized by maintainer release history before 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