chenhouser2025 / **ISSUE-Github-REPORT-ssrf-remote-schema.md** Secret

Last active 3 months ago

Code Revisions 2

Blind Server-Side Request Forgery (SSRF) in API Tool Remote Schema Fetch

ISSUE-Github-REPORT-ssrf-remote-schema.md

Advisory Details

Title: Blind Server-Side Request Forgery (SSRF) in API Tool Remote Schema Fetch

Description:

Summary

A Blind Server-Side Request Forgery (SSRF) vulnerability exists in the `get_api_tool_provider_remote_schema` method of the `ApiToolManageService`. The application attempts to fetch an OpenAPI schema from a user-provided URL using `httpx.get` directly, completely bypassing the internal `ssrf_proxy` protection mechanism. This creates a security gap allowing authenticated users (even with low privileges) to induce the server to make arbitrary HTTP GET requests to internal network resources or cloud metadata services.

Details

The vulnerability is located in the file `api/services/tools/api_tools_manage_service.py`.

The function `get_api_tool_provider_remote_schema` takes a `url` parameter from the user input. It is intended to fetch a remote OpenAPI JSON schema definition. However, unlike other parts of the codebase that use `core.helper.ssrf_proxy` to safely fetch external resources, this function imports and uses `httpx.get` directly.

Vulnerable Code Snippet:

```
# api/services/tools/api_tools_manage_service.py

from httpx import get  # Direct import, bypassing proxy middleware

class APIToolManageService:
    # ...
    @staticmethod
    def get_api_tool_provider_remote_schema(url: str, fail_if_empty: bool =
True):
        # ...
        try:
            # VULNERABILITY: User-controlled 'url' is accessed directly
            response = get(url, headers=headers, timeout=10)
        except Exception as e:
            # ...
```

There is no validation on the `url` to ensure it targets an external address, effectively allowing the server to act as an open HTTP proxy for GET requests.

PoC

We have developed a functional Proof of Concept (`real_poc_ssrf.py`) that demonstrates this on a live instance.

Reproduction Steps:

1. **Authenticate** to the Dify API to obtain a valid `access_token` and `csrf_token`.
2. **Target Endpoint**: `/console/api/workspaces/current/tool-provider/api/remote`
3. **Inject Payload**: Pass an internal URL (e.g., `http://docker_redis_1:6379/`) as the `url` query parameter.

Manual Reproduction (Curl):

```
# Export your cookies and token first
export COOKIE="csrf_token=...; session=..."
export TOKEN="eyJhbG..."

# Attempt to scan internal Redis port (6379)
curl -v "http://localhost/console/api/workspaces/current/tool-
provider/api/remote?url=http://docker_redis_1:6379/" \
  -H "Cookie: $COOKIE" \
  -H "X-CSRF-Token: <CSRF_TOKEN_FROM_COOKIE>" \
  -H "Authorization: Bearer $TOKEN"
```

Observation:

- If the internal port is **OPEN** (e.g., Redis): The server connects, receives data (which is not valid JSON schema), and returns a 400 error `invalid_param`, confirming the connection was established.
- If the internal port is **CLOSED**: The server returns a Connection Refused error or times out, distinguishable from the open port response.

Impact

- **Network Reconnaissance**: Attackers can map the internal network infrastructure (IPs, open ports) by analyzing response times and error messages.
- **Cloud Metadata Access**: In cloud environments (AWS, GCP, Azure), attackers can retrieve sensitive instance metadata (IAM credentials, configuration) via `http://169.254.169.254/`.
- **Internal Service Interaction**: Attackers can trigger GET-based actions on internal APIs (e.g., Kubelet, management interfaces) that assume trust based on network position.

Affected products

- **Ecosystem**: PyPI / Docker
- **Package name**: langgenius/dify
- **Affected versions**: <= 0.6.9 (Verified on commit `ee662e54b1999eff0db66fe57de6eb461f40b214`)
- **Patched versions**:

Severity

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

Weaknesses

- **CWE**: CWE-918: Server-Side Request Forgery (SSRF)

Occurrences

Permalink

https://github.com/langgenius/dify/blob/main/api/services/tools/api_tools_manage_service