Senior Platform Engineer - IAM

Home Assignment: Secure API Authentication & Authorization

Objective:

This assignment aims to evaluate your ability to design and implement a basic yet secure API endpoint with authentication and authorization logic. You'll focus on using industry-standard techniques to manage access to a protected resource.

Scenario:

You are building a simple "User Management Service" that has a single endpoint: `/users`. This endpoint is intended to allow authorized clients to retrieve a list of users (or their basic information), but only after successful authentication.

Task:

1. API Endpoint Design:
    - Create a REST API endpoint at `/users`.
    - This endpoint should (for this exercise) return a hardcoded JSON list of user objects if authorization is successful. Example:

Unset
```
[
    {"id": "user1", "username": "john.doe"},
    {"id": "user2", "username": "jane.smith"}
]
```

2. Authentication:
    - Implement a basic authentication mechanism. You can use a simple API Key authentication model. The API Key will be passed in the `Authorization` header with `Bearer` authentication scheme. Example: `Authorization: Bearer some-api-key`.
    - For this exercise, you can have a hardcoded "valid" API Key value.

   - If authentication fails (invalid API key), the API should return a `401 Unauthorized` status code and a basic error message, such as `{"error": "Invalid API Key"}`.

3. Authorization:
   - Implement basic role-based authorization: For this scenario, assume that the API endpoint should only be accessed by clients with a `viewer` role.
   - Associate roles with API Keys: The `some-api-key` used above should be associated with the `viewer` role, and that role should be checked against before allowing access. For simplicity, you can assume there are no other roles for the moment.
   - If authorization fails (e.g. wrong role), the API should return a `403 Forbidden` status code and a basic error message such as `{"error": "Forbidden. Insufficient Permissions"}`.

4. Language/Framework:
   - Choose one language from Go, Java, or C# for this assignment.
   - You can use any lightweight framework/library you are comfortable with for creating a simple HTTP server.

5. Documentation:
   - Include a `README.md` file that explains:
     - How to run your application.
     - How to test the API (include example `curl` or similar commands).
     - How your chosen authentication and authorization approach works.
     - Any design or implementation choices that you think are relevant.

Deliverables:

1. **Required: A Git repository containing:**
   a. **All source code.**
   b. **A `README.md` file, with at least:**
      i. **Instruction to run the code**
      ii. **A list of dependencies used (if any)**
   c. **No Dockerfile or Kubernetes manifests are required, but greatly appreciated (in particular to test the API easily)**
2. **Optional:**
   a. **Docker image**
   b. **A running example hosted somewhere ready to be queried**