Flex Beta-Engine Security Implementation Overview

1. Current Architecture & Setup

Website

- **Steam OpenID Authentication:**
  - Users log in via Steam OpenID (OpenID 2.0), which verifies their identity and retrieves a unique Steam ID.
- **Email Collection:**
  - After login, users are prompted to enter an email address. This email is used for further communication and identification.
- **Token Considerations:**
  - Currently, the website does not pass a user-managed token to Beta Engine; it relies on the fact that the Steam login already verifies the user.

Beta Engine

- **Input Handling:**
  - Endpoints like /user and /user/details receive a Steam ID (and optionally a name) and process the request.
- **Protected Endpoints:**
  - Endpoints such as /user/traits and /user/skills are protected using an x-api-key mechanism.
- **Stateless Design:**
  - Beta Engine does not maintain per-user sessions; all persistent data is stored in a database.
- **Internal Processing:**
  - The API calls internal helper functions (e.g., internal.add_games_data(), internal.add_achievements_data()) that retrieve and process user data based on the provided Steam ID.

Models API

- **JWT-Protected Service:**
  - Beta Engine calls the Models API (which uses JWT for its endpoints) to fetch predictions and configuration data.
- **Secure Communication:**
  - Beta Engine obtains a JWT from the Models API's /login endpoint (using stored credentials) and includes it in the Authorization header when making API calls.

---

2. Security Gaps

- **Trusting the Steam ID Input:**
  - Beta Engine currently accepts any Steam ID provided by the website. Without further verification, an attacker could supply arbitrary IDs. (spoofing)
- **Static API Key:**
  - The x-api-key used for protecting endpoints is static
- **Lack of End-to-End Token Verification:**
  - Beta Engine relies solely on the website's authentication via Steam OpenID. There is no additional token or session check to ensure that the request originates from a verified source.

---

3. Proposed Security Enhancements

Option A: Strengthen Existing x-api-key Mechanism

- **Key Rotation & Secure Storage:**
  - Regularly rotate the API key and ensure it is stored only as secure environment variables in Render.
- **Network-Level Restrictions:**
  - Consider restricting access via IP whitelisting or a reverse proxy so that only requests from the official website are accepted.

Option B: Introduce an Internal JWT Flow

- **Website Issues Its Own JWT:**
  - After a successful Steam OpenID login, the website could generate a JWT that includes the verified Steam ID (and possibly the email) as claims.
- **Token Attachment to Requests:**
  - The website attaches this JWT to requests made to Beta Engine.
  - Beta Engine uses middleware (e.g., Flask-JWT-Extended) to verify the token, ensuring that the request came from an authenticated source.
- **Advantages:**
  - This method verifies that the Steam ID was authenticated on the website without requiring customers to manage tokens themselves.
  - It provides end-to-end integrity and non-repudiation without exposing additional complexity to external clients.

Option C: Validate the Steam ID with the Steam API

- **Direct Verification:**
  - As an extra layer, Beta Engine could call the Steam API to verify that the provided Steam ID exists and is valid.
- **Considerations:**
  - This adds network overhead and might contribute to reaching the Steam rate limit (100,000 calls per day), so it should be used sparingly or cached.

---

4. Why Not Use Steam OpenID Directly for Token Generation?

- **OpenID 2.0 vs. OpenID Connect (OIDC):**
  - Steam uses the older OpenID 2.0 specification, which does not provide a JWT-like ID token. OpenID Connect (OIDC) offers standardized tokens that are easier to validate and use in a JWT flow.
- **Implementation Complexity:**
  - Re-implementing the OpenID 2.0 flow for direct token generation in Beta Engine would be complex and error-prone.
- **Rate Limit Concerns:**
  - Steam's API rate limit (100,000 calls per day) makes it impractical to verify every request by calling Steam directly.

---

6. OAuth for Email – Possibility & Considerations

Possibility of Adding OAuth

- **Option:**
  - You could add an OAuth flow (using a provider like Google) specifically to authenticate the email address.
- **Advantages:**
  - This would provide an additional layer of verification for the email address.
  - It may allow users to sign in using their email if they prefer, or to verify that their email is indeed valid.
- **Redundancy Considerations:**

    - Since the website already uses Steam OpenID to verify the user and then collects an email, adding a separate OAuth flow solely for email might be redundant.
    - If most users are coming through Steam and the email is used only for communication purposes, the current process may be sufficient.
- **Recommendation:**
    - Maintain the current process where the email is collected after Steam authentication.

---

7. Recommendations

Long-Term Recommendations

- **Consider Internal JWT Flow:**
    - Evaluate the feasibility of having the website issue a JWT after Steam OpenID login.
    - Use this JWT for all internal communications with Beta Engine to provide stronger verification without additional customer friction.
- **Evaluate OAuth for Email:**
    - If your user base expands to non-steam users or if email verification becomes critical, consider adding a minimal OAuth flow for email.
    - For now, it may be redundant given the current workflow, but it should be documented as a potential enhancement.
