📄 Cơ Chế Access Token & Refresh Token

1. Tổng Quan

- 🔑 **Access Token**:
  - Dùng để **xác thực** các request API.
  - Có thời gian sống ngắn (ví dụ: **2 ngày**) nhằm giảm thiểu rủi ro khi bị lộ.
- ♻️ **Refresh Token**:
  - Dùng để **làm mới** Access Token khi nó hết hạn.
  - Có thời gian sống dài hơn (ví dụ: **7 ngày**) và được kiểm soát chặt chẽ.
- 💾 **Lưu trữ**:
  - **Access Token**: Nên lưu trong bộ nhớ (Redux, Vuex) hoặc HttpOnly Cookie (không nên lưu ở localStorage).
  - **Refresh Token**: Nên lưu trong HttpOnly Cookie để tránh bị tấn công XSS.

---

2. Quy Trình Xác Thực & Làm Mới Token

2.1 Đăng Nhập / Đăng Ký

1. 📝 **Yêu cầu từ client**:
   - Người dùng gửi thông tin đăng nhập (email, password) đến endpoint `/shop/login` hoặc đăng ký tại `/shop/signUp`.
2. ✅ **Xác thực thông tin**:
   - Nếu hợp lệ, hệ thống tạo ra cặp **privateKey** (bí mật, chỉ server biết) và **publicKey** (dùng xác minh, lưu trên server & DB).
3. 🔐 **Tạo Token**:
   - Tạo ra **Access Token** (2 ngày) và **Refresh Token** (7 ngày) qua hàm `createKeyPair`.
4. 💾 **Lưu trữ**:
   - Lưu thông tin token (refresh token, privateKey, publicKey) vào collection `keyTokens` trong MongoDB.
5. 📤 **Phản hồi**:
   - Trả về cho client thông tin shop cùng với token.

2.2 Xác Thực Request (Middleware `authentication()`)

1. 🧩 **Thu thập dữ liệu**:
   - Lấy `userId` từ header (ví dụ: `x-client-id`).
   - Tìm **keyStore** trong DB thông qua hàm `findByUserId`.
   - Lấy **accessToken** từ header (`authorization`).
2. 🔍 **Xác minh Access Token**:
   - Sử dụng `jwt.verify()` cùng với `publicKey` từ keyStore để xác thực token.
3. ✅ **Nếu hợp lệ**:
   - Gán thông tin vào `req.keyStore` và cho phép request tiếp tục.
4. ❌ **Nếu không hợp lệ hoặc hết hạn**:
   - Middleware ném lỗi `UnauthorizedError` (HTTP 401).

💡 **Lưu ý**: Middleware chỉ kiểm tra access token. Khi access token hết hạn, nó trả về lỗi 401 và client sẽ phải xử lý gọi endpoint refresh.

2.3 Làm Mới Token (Endpoint `handlerRefreshToken`)

1. 🔄 **Khi nào gọi refresh?**
   - Khi client nhận được lỗi 401 từ middleware (Access Token hết hạn).
2. **Quy trình refresh token**:
   - 🚫 **Kiểm tra Replay Attack**:
     - Xác minh xem refresh token có nằm trong danh sách `refreshTokensUsed` của keyStore.
     - Nếu có, có thể là dấu hiệu tấn công → Xóa toàn bộ token của user và báo lỗi.
   - 🔍 **Tìm kiếm token hợp lệ**:
     - Sử dụng hàm `findByRefreshToken` để lấy keyStore tương ứng với refresh token.
     - Nếu không tìm thấy, trả lỗi `UnauthorizedError`.
   - 🔐 **Xác minh Refresh Token**:
     - Dùng `verityToken` cùng với `privateKey` để xác thực refresh token.
   - 👤 **Kiểm tra user**:
     - Xác nhận sự tồn tại của user qua email.
   - 🚀 **Tạo cặp token mới**:
     - Gọi `createKeyPair` để tạo Access Token & Refresh Token mới.
   - 📝 **Cập nhật keyStore**:

        - Thay thế refresh token cũ bằng token mới.
        - Thêm refresh token cũ vào mảng `refreshTokensUsed` để đánh dấu đã sử dụng.
    - 📤 **Phản hồi**:
        - Trả về thông tin user cùng với cặp token mới.

---

3. Hướng Dẫn Sử Dụng Cho Client

1. 🔔 **Đăng Nhập / Đăng Ký**:
    - Gửi thông tin đăng nhập/đăng ký tới server.
    - Nhận về cặp token: Access Token và Refresh Token.
    - Lưu Access Token để gửi kèm theo mỗi request; lưu Refresh Token an toàn (HttpOnly Cookie).
2. 🔒 **Gọi API Bảo Vệ**:
    - Đính kèm Access Token trong header (ví dụ: `authorization: Bearer <access_token>`).
    - Nếu nhận lỗi 401 (Access Token hết hạn), chuyển sang bước refresh token.
3. ♻️ **Làm Mới Token**:
    - Khi nhận lỗi 401, gọi endpoint `/shop/handlerRefreshToken` kèm Refresh Token.
    - Nếu thành công, nhận về cặp token mới và cập nhật lại cho các request sau.
4. 🚪 **Đăng Xuất**:
    - Khi người dùng đăng xuất, gọi endpoint `/shop/logout` để xóa thông tin token khỏi server và xóa token trên client.

---

4. Các Khuyến Nghị Bảo Mật

- 🔐 **Lưu trữ an toàn**:
    - Dùng HttpOnly Cookie cho Refresh Token để ngăn chặn tấn công XSS.
- 🔒 **Sử dụng HTTPS**:
    - Mã hóa toàn bộ dữ liệu truyền qua mạng.
- 👀 **Giám sát Refresh Token**:
    - Nếu phát hiện token bị tái sử dụng (reused), xóa toàn bộ token của user và yêu cầu đăng nhập lại.
- 🔄 **Phân tách nhiệm vụ**:

    - Middleware chỉ kiểm tra Access Token.
    - Endpoint refresh token chỉ làm nhiệm vụ làm mới token khi có yêu cầu từ client.

---

5. Kết Luận

- 🎯 **Access Token**: Xác thực các request, có thời gian sống ngắn, giảm rủi ro khi bị đánh cắp.
- ♻️ **Refresh Token**: Cho phép làm mới Access Token mà không cần đăng nhập lại, nhưng phải được quản lý cẩn thận.
- 🛡️ **Tách biệt rõ ràng**:
    - Middleware `authentication()` kiểm tra Access Token.
    - Endpoint `/shop/handlerRefreshToken` xử lý việc làm mới token khi cần.

Việc áp dụng cơ chế này giúp hệ thống của bạn đảm bảo tính bảo mật cao, đồng thời cung cấp trải nghiệm người dùng mượt mà khi không phải đăng nhập lại mỗi khi token hết hạn.