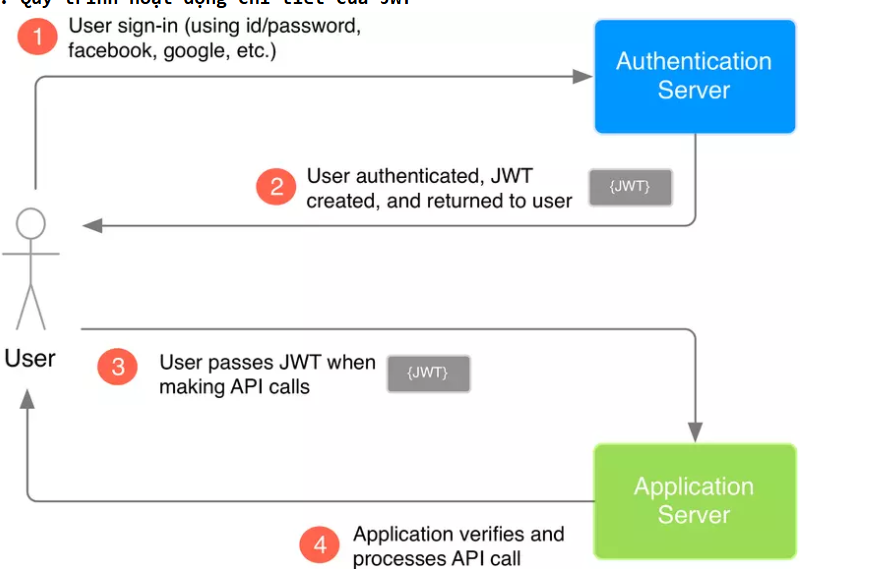
Bài 2:

**Quy trình hoạt động chi tiết của JWT**



**JWT (JSON Web Token)** hoạt động dựa trên việc tạo ra một chuỗi mã hóa chứa thông tin xác thực của người dùng sau khi đăng nhập thành công. Client sẽ lưu trữ token này và sử dụng nó để gửi kèm trong các request tiếp theo, thay vì phải gửi lại thông tin đăng nhập. Server sẽ kiểm tra token này để xác thực người dùng và cấp quyền truy cập tài nguyên.

* **Đăng nhập và xác thực:**
  + Người dùng gửi thông tin đăng nhập (ví dụ: username và password) lên server.
  + Server xác thực thông tin người dùng.
  + Nếu xác thực thành công, server sẽ tạo một JWT và ký nó bằng một private key.
  + JWT bao gồm các thông tin như:
    - **Header:** Chứa thông tin về thuật toán hash được sử dụng.
    - **Payload:** Chứa các thông tin về người dùng (ví dụ: user ID, role, etc.).
    - **Signature:** Được tạo bằng cách sử dụng header, payload và private key.
  + Server trả về JWT cho client.
* **Sử dụng JWT:**
  + Client lưu trữ JWT (thường trong **localStorage**, **sessionStorage**hoặc **cookies**).
  + Khi client muốn truy cập một tài nguyên được bảo vệ, nó sẽ gửi JWT kèm theo request (thường trong header Authorization với prefix Bearer).
  + Ví dụ: Authorization: Bearer <JWT>
* **Xác thực và ủy quyền:**
  + Server nhận request và JWT.
  + Server sử dụng public key (được tạo ra từ private key) để giải mã JWT và xác minh chữ ký.
  + Nếu chữ ký hợp lệ, server xác định rằng JWT chưa bị thay đổi và thông tin trong JWT là chính xác.
  + Server có thể sử dụng thông tin trong payload để xác thực người dùng và cấp quyền truy cập tài nguyên.
  + Nếu JWT không hợp lệ, server sẽ từ chối request.

Bài 3:

**1. Giải mã JWT mẫu**

**Chuỗi JWT mẫu:**

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkxhbiBBbmgiLCJpYXQiOjE1MTYyMzkwMjIsImV4cCI6MTY5Nzk5OTk5OX0.Wn3xU4uSHhEwhNO7oTnZm0a2ZXlfU29lvYzPlyoN8yo

**Kết quả giải mã phần payload:**

{

"sub": "1234567890",

"name": "Lan Anh",

"iat": 1516239022,

"exp": 1697999999

}

**2. Giải thích các trường trong payload**

| **Trường** | **Ý nghĩa** |
| --- | --- |
| sub | **Subject** – ID của người dùng hoặc đối tượng chính của token. |
| name | **Tên người dùng**, có thể hiển thị trên UI hoặc dùng để cá nhân hóa. |
| iat | **Issued At** – Thời điểm phát hành token (giá trị Unix timestamp). |
| exp | **Expiration** – Thời gian token hết hạn (cũng là Unix timestamp). |

**Giải thích cụ thể:**

* sub: "1234567890": Đây là ID người dùng. Có thể là UUID hoặc ID từ hệ thống database.
* name: "Lan Anh": Thông tin người dùng, dùng để hiển thị giao diện.
* iat: 1516239022 → tương đương **Thứ Hai, 18 tháng 1, 2018**.
* exp: 1697999999 → tương đương **Thứ Bảy, 22 tháng 10, 2023**.

**3. Rủi ro bảo mật nếu lộ thông tin nhạy cảm trong payload**

**Những rủi ro tiềm ẩn:**

| **Vấn đề** | **Mô tả** |
| --- | --- |
| **Payload không được mã hóa** | Ai cũng có thể **giải mã** nội dung (dù không sửa được nếu có chữ ký). |
| **Lộ thông tin nhạy cảm** | Nếu chứa email, số điện thoại, vai trò admin,... có thể bị khai thác. |
| **Tấn công giả mạo (phishing)** | Hacker có thể dùng JWT giả để đánh lừa hệ thống nếu thiếu xác thực chặt chẽ. |
| **Lộ thời gian hết hạn** | Hacker biết được khi nào token sẽ hết hiệu lực để tấn công trước đó. |

**Khuyến nghị bảo mật:**

* **Không lưu thông tin nhạy cảm** (mật khẩu, mã OTP, số thẻ, v.v.) trong payload.
* **Sử dụng HTTPS** để truyền token.
* **Mã hóa riêng payload nếu thực sự cần bảo mật nội dung**.
* Dùng short expiration (thời hạn token ngắn).
* Dùng **refresh token** để lấy lại access token mới khi hết hạn.

Bài 6:

**1. JWT HẾT HẠN (ExpiredJwtException)**

* **Tình huống:** Token có exp (expiration time) đã qua.
* **Server phản ứng:**

{

"error": "Token expired",

"status": 401,

"message": "JWT token has expired. Please login again."

}

**2. JWT BỊ SỬA ĐỔI (SignatureException)**

* **Tình huống:** Chữ ký JWT không khớp – có thể do token bị chỉnh sửa thủ công hoặc bị giả mạo.
* **Server phản ứng:**

{

"error": "Invalid signature",

"status": 401,

"message": "JWT signature is invalid. Possible tampering detected."

}

**3. JWT KHÔNG HỢP LỆ (MalformedJwtException)**

* **Tình huống:** Token không đúng cấu trúc chuẩn (thiếu phần header/payload/signature).
* **Server phản ứng:**

{

"error": "Malformed token",

"status": 400,

"message": "JWT token structure is invalid."

}

**4. JWT RỖNG / MẤT TOKEN (IllegalArgumentException)**

* **Tình huống:** Không có token trong header hoặc token là chuỗi rỗng.
* **Server phản ứng:**

{

"error": "Missing token",

"status": 400,

"message": "Authorization token is missing or empty."

}

**5. JWT KHÔNG HỖ TRỢ (UnsupportedJwtException)**

* **Tình huống:** Token được tạo không đúng chuẩn (ví dụ không dùng JWT compact).
* **Server phản ứng:**

{

"error": "Unsupported token",

"status": 400,

"message": "JWT token type is unsupported."

}

**6. MISSING "Bearer" TRONG HEADER**

* **Tình huống:** Client gửi token nhưng thiếu prefix "Bearer " trong Authorization header.
* **Server phản ứng:**

{

"error": "Invalid authorization format",

"status": 400,

"message": "Authorization header must start with Bearer."

}

**KẾT LUẬN: ĐỀ XUẤT HÀNH ĐỘNG CHUNG CHO CLIENT**

| **Loại lỗi** | **Client nên làm** |
| --- | --- |
| Token hết hạn | Logout, redirect login |
| Token bị giả mạo | Xoá token, thông báo vi phạm bảo mật |
| Token không hợp lệ | Đăng nhập lại, kiểm tra luồng xác thực |
| Không có token | Chặn truy cập tài nguyên, hiển thị cảnh báo |
| Header sai định dạng | Sửa lỗi kỹ thuật phía frontend |