***Bai 1***

**1. JWT là gì?**

**JWT (JSON Web Token)** là một chuẩn mở (RFC 7519) để truyền tải thông tin an toàn giữa các bên dưới dạng **JSON** đối tượng. Thông tin này có thể được xác minh và tin cậy vì nó được **ký (signed)** bằng chữ ký số (thường dùng HMAC hoặc RSA/ECDSA).

**Ưu điểm của JWT:**

* **Không trạng thái (stateless)**: Thông tin người dùng được lưu trữ trực tiếp trong token nên không cần lưu session phía server.
* **Khả năng mở rộng cao**: Phù hợp với hệ thống microservices và API Gateway.
* **Dễ dàng xác thực**: Token chứa thông tin xác thực, chỉ cần kiểm tra chữ ký là đủ.

**Nhược điểm của JWT:**

* **Không thể thu hồi dễ dàng**: Sau khi phát hành, token có thể bị lạm dụng nếu chưa hết hạn.
* **Dung lượng lớn hơn session ID**: Vì chứa nhiều dữ liệu hơn.
* **Phức tạp khi xử lý bảo mật**: Nếu không sử dụng HTTPS hoặc lưu trữ an toàn có thể gây rò rỉ thông tin.

**2. Cấu trúc của JWT**

JWT có **3 phần**, được ngăn cách bằng dấu chấm .:

xxxxx.yyyyy.zzzzz

↓ ↓ ↓

Header Payload Signature

Ví dụ một JWT:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkJp5buDbmggVsO6IiwiYWRtaW4iOnRydWV9.

SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

**a. Header**

{

"alg": "HS256",

"typ": "JWT"

}

* alg: Thuật toán dùng để ký token (VD: HS256, RS256).
* typ: Loại token, thường là JWT.

**Mục đích**: Cho biết cách giải mã chữ ký.

**b. Payload**

{

"sub": "1234567890",

"name": "Bình Vũ",

"admin": true

}

* Chứa **claims (khẳng định)** – thông tin định danh người dùng hoặc các dữ liệu cần truyền.
* Có 3 loại claims:
  + **Registered** (chuẩn): iss, exp, sub, aud, ...
  + **Public**: Định nghĩa công khai, mọi hệ thống đều có thể hiểu.
  + **Private**: Do hệ thống định nghĩa riêng.

**Mục đích**: Truyền dữ liệu cần xác thực, định danh người dùng.

**c. Signature**

HMACSHA256(

base64UrlEncode(header) + "." + base64UrlEncode(payload),

secret

)

* Được tạo bằng cách mã hóa header + payload với một **secret key** hoặc **private key**.
* Giúp đảm bảo **tính toàn vẹn** và xác thực của token.

**Mục đích**: Xác minh rằng token không bị thay đổi và được phát hành bởi bên đáng tin cậy.

**3. JWT dùng trong những trường hợp nào?**

**Các ứng dụng phổ biến:**

| **Trường hợp sử dụng** | **Mô tả** |
| --- | --- |
| **Xác thực (Authentication)** | Gửi token trong header (Authorization: Bearer ...) sau khi đăng nhập. |
| **Phân quyền (Authorization)** | Kiểm tra role/permission trong payload của JWT. |
| **Trao đổi thông tin giữa dịch vụ (API)** | Gửi JWT giữa các microservice để xác thực nhanh. |
| **Ứng dụng di động/SPAs** | Lưu JWT trên client (localStorage, sessionStorage). |
| **Single Sign-On (SSO)** | JWT giúp chia sẻ thông tin đăng nhập giữa các hệ thống. |

**Tóm tắt**

* **JWT** là cách truyền thông tin an toàn, có chữ ký xác minh.
* Gồm 3 phần: **Header**, **Payload**, **Signature**.
* Dùng trong **xác thực**, **phân quyền**, và **truyền dữ liệu bảo mật**.
* Tuy tiện lợi nhưng cần xử lý bảo mật cẩn thận (HTTPS, thời gian hết hạn, blacklist,...).

***Bai 2***

**1. Mô tả tổng quan quy trình xác thực JWT**

Quy trình gồm **2 giai đoạn chính**:

* Giai đoạn 1: **Đăng nhập và nhận JWT**
* Giai đoạn 2: **Sử dụng JWT để truy cập tài nguyên bảo vệ**

**2. Sơ đồ quy trình xác thực JWT**

Client Server (REST API)

| |

| 1. Gửi thông tin đăng nhập |

| --------------------------------> |

| |

| 2. Xác thực thông tin |

| -> Tạo JWT |

| -> Trả JWT về client |

| <-------------------------------- |

| |

| 3. Lưu token (localStorage...) |

| |

| 4. Gửi request + token (Bearer) |

| --------------------------------> |

| |

| 5. Backend kiểm tra JWT |

| -> Giải mã + xác thực chữ ký |

| -> Check thời hạn, role,... |

| |

| 6. Nếu hợp lệ => cho truy cập |

| <-------------------------------- |

**3. Chi tiết từng bước**

**Bước 1: Client gửi thông tin đăng nhập**

* Gửi POST /login hoặc /authenticate với:
* {
* "username": "user",
* "password": "123456"
* }

**Bước 2: Server xác thực và phát JWT**

* Server kiểm tra thông tin đăng nhập (so sánh DB).
* Nếu hợp lệ:
  + Tạo **JWT** chứa thông tin như: username, role, exp.
  + Ký token bằng secret/private key.
* Trả lại JWT:
* {
* "token": "eyJhbGciOiJIUzI1NiJ9..."
* }

**Bước 3: Client lưu JWT**

* Lưu token vào:
  + Trình duyệt: localStorage, sessionStorage.
  + Mobile: SecureStorage hoặc tương đương.

**Bước 4: Client gửi request có token**

* Khi gọi API cần bảo vệ, thêm vào header:
* Authorization: Bearer eyJhbGciOiJIUzI1NiJ9...

**Bước 5: Backend kiểm tra JWT**

* Kiểm tra Authorization header:
  + Nếu có token → tách token.
  + Giải mã token → xác minh chữ ký bằng secret key.
  + Kiểm tra:
    - Hết hạn chưa (exp)?
    - Định danh user (sub/username)?
    - Role có hợp lệ không?

**Bước 6: Xử lý truy cập tài nguyên**

* Nếu hợp lệ:
  + Cho phép truy cập tài nguyên.
* Nếu **token không hợp lệ** (sai chữ ký, hết hạn, giả mạo):
  + Trả về HTTP **401 Unauthorized**.

**4. Vòng đời của JWT**

| **Giai đoạn** | **Mô tả** |
| --- | --- |
| **Tạo token** | Khi người dùng đăng nhập thành công. |
| **Gửi token** | Client gửi token kèm request truy cập tài nguyên. |
| **Xác thực token** | Server giải mã, xác minh và cho phép nếu hợp lệ. |
| **Hết hạn** | Token tự mất hiệu lực khi exp hết hạn (không cần lưu trạng thái). |
| **Tạo mới (nếu có)** | Có thể dùng **refresh token** để tạo lại JWT nếu cần. |

**Lưu ý bảo mật**

* **Luôn dùng HTTPS** để tránh bị đánh cắp token.
* **Không lưu JWT trong cookie nếu không cấu hình bảo mật kỹ (HttpOnly, Secure)**.
* Cân nhắc thêm **refresh token** để giảm rủi ro khi access token bị lộ.

***Bai 3***

Chúng ta sẽ thực hành **giải mã một chuỗi JWT**, phân tích phần **payload** và đánh giá các **rủi ro bảo mật** nếu để lộ thông tin nhạy cảm.

**1. Chuỗi JWT mẫu**

Ví dụ chuỗi JWT sau (được mã hóa bằng thuật toán HMAC SHA256):

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkJp5buDbmggVsO6IiwiYWRtaW4iOnRydWUsImV4cCI6MTcwMDAwMDAwMH0.

TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

**2. Giải mã JWT**

JWT gồm 3 phần:  
Header.Payload.Signature (dạng Base64URL)

Ta giải mã thủ công hoặc dùng công cụ như:

[**https://jwt.io**](https://jwt.io/) (Paste vào ô “Encoded”)

**Kết quả giải mã:**

**Header (Thông tin thuật toán):**

{

"alg": "HS256",

"typ": "JWT"

}

**Payload (Thông tin người dùng):**

{

"sub": "1234567890",

"name": "Bình Vũ",

"admin": true,

"exp": 1700000000

}

**Signature (Dùng để xác thực):**

HMACSHA256(

base64UrlEncode(header) + "." + base64UrlEncode(payload),

secret

)

**3. Phân tích các trường trong Payload**

| **Trường** | **Ý nghĩa** |
| --- | --- |
| sub | (Subject) Định danh duy nhất của người dùng – thường là ID trong hệ thống |
| name | Tên người dùng – dùng để hiển thị hoặc gợi nhớ |
| admin | Quyền truy cập (true/false) – có thể dùng để phân quyền |
| exp | Thời điểm hết hạn token (UNIX timestamp) |

Với exp: 1700000000 tương ứng khoảng **14/11/2023 22:13:20 UTC**

**4. Rủi ro bảo mật nếu để lộ thông tin trong Payload**

JWT **không mã hóa payload**, chỉ **mã hóa chữ ký để kiểm tra tính toàn vẹn**. Vì thế **bất kỳ ai có token đều có thể đọc payload** bằng cách decode Base64URL.

**Ví dụ rủi ro:**

* Nếu payload chứa:
  + password, email, phone, số CCCD, bank\_account,...
  + Hoặc quyền cao (admin: true)  
    ⟶ Dễ bị **lộ thông tin nhạy cảm** và có thể bị **tấn công leo thang quyền (privilege escalation)**.

**Giải pháp an toàn**

| **Rủi ro** | **Giải pháp** |
| --- | --- |
| Lộ thông tin người dùng | Không lưu thông tin nhạy cảm trong payload |
| Giả mạo JWT | Luôn kiểm tra chữ ký JWT bằng secret hoặc public key |
| Token bị chặn hoặc đánh cắp | Dùng HTTPS và thời hạn token ngắn |
| Không thể thu hồi token | Dùng Refresh Token và cơ chế blacklist nếu cần |

**Tóm lại**

* Payload JWT dễ dàng giải mã, KHÔNG nên chứa thông tin nhạy cảm.
* Các trường sub, exp, role, username là an toàn và phổ biến.
* Nên dùng JWT chỉ để định danh, không để truyền thông tin bí mật.