

1. Jwt (JSON Web Token) là gì?

* JWT là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server , các thông tin trong chuỗi JWT được định dạng bằng JSON . Trong đó chuỗi Token phải có 3 phần là header , phần payload và phần signature được ngăn bằng dấu “.”

**Header**

* Phần header sẽ chứa kiểu dữ liệu , và thuật toán sử dụng để mã hóa ra chuỗi JWT

{

"typ": "JWT",

"alg": "HS256"

}

“typ” (type) chỉ ra rằng đối tượng là một JWT

“alg” (algorithm) xác định thuật toán mã hóa cho chuỗi là HS256

**Payload**

* Phần payload sẽ chứa các thông tin mình muốn đặt trong chuỗi Token như username , userId , author , … ví dụ:

{

"user\_name": "admin",

"user\_id": "1513717410",

"authorities": "ADMIN\_USER",

"jti": "474cb37f-2c9c-44e4-8f5c-1ea5e4cc4d18"

}

**Signature**

* Phần chử ký này sẽ được tạo ra bằng cách mã hóa phần header , payload kèm theo một chuỗi secret (khóa bí mật) , ví dụ:

data = base64urlEncode( header ) + "." + base64urlEncode( payload )

signature = Hash( data, secret );

* base64UrlEncoder : thuật toán mã hóa header và payload

Đoạn code trên sau khi mã hóa **header** và **payload** bằng thuật toán **base64UrlEncode** ta sẽ có chuỗi như sau

// header

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

// payload

eyJhdWQiOlsidGVzdGp3dHJlc291cmNlaWQiXSwidXNlcl9uYW1lIjoiYWRtaW4iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNTEzNzE

Sau đó mã hóa 2 chuỗi trên kèm theo **secret (khóa bí mật)** bằng thuật toán **HS256** ta sẽ có chuỗi **signature** như sau:

9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q

Cuối cùng

Kết hợp 3 chuỗi trên lại ta sẽ có được một chuỗi **JWT** hoàn chỉnh

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsidGVzdGp3dHJlc291cmNlaWQiXSwidXNlcl9uYW1lIjoiYWRtaW4iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNTEzNzE.9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q

## **Khi nào nên dùng JSON Web Token?**

* **Authentication**: Đây là trường hợp phổ biến nhất thường sử dụng JWT. Khi người dùng đã đăng nhập vào hệ thống thì những request tiếp theo từ phía người dùng sẽ chứa thêm mã JWT. Điều này cho phép người dùng được cấp quyền truy cập vào các url, service, và resource mà mã Token đó cho phép. Phương pháp này không bị ảnh hưởng bởi Cross-Origin Resource Sharing (CORS) do nó không sử dụng cookie.
* **Trao đổi thông tin**: [JSON](https://topdev.vn/viec-lam-it/json-kt35) Web Token là 1 cách thức khá hay để truyền thông tin an toàn giữa các thành viên với nhau, nhờ vào phần **signature** của nó. Phía người nhận có thể biết được người gửi là ai thông qua phần **signature**. Và chữ ký được tạo ra bằng việc kết hợp cả phần header, payload lại nên thông qua đó ta có thể xác nhận được chữ ký có bị giả mạo hay không.

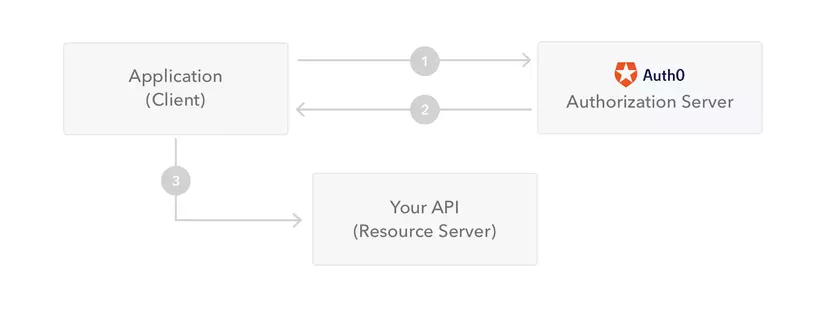
## JSON Web Tokens hoạt động như thế nào?

* Trong xác thực, khi người dùng đăng nhập thành công bằng thông tin đăng nhập của họ, JSON Web Token sẽ được trả về. Vì token là thông tin xác thực, cần phải hết sức cẩn thận để ngăn chặn các vấn đề bảo mật. Nói chung, bạn không nên giữ token lâu hơn yêu cầu.
* Bạn cũng không nên lưu trữ dữ liệu nhạy cảm trên session trong bộ nhớ trình duyệt do thiếu bảo mật.
* Bất cứ khi nào người dùng muốn truy cập route hoặc resource được bảo vệ, tác nhân người dùng nên gửi JWT, thêm Authorization trong header với nội dung là Bearer + token. Nội dung của header sẽ trông như sau:

Authorization: Bearer <token>

* Máy chủ server sẽ kiểm tra tính hợp lệ của JWT trong header mỗi khi nhận request, nếu hợp lệ người dùng sẽ được phép truy cập các resource được bảo vệ. Nếu JWT chứa dữ liệu cần thiết, nhu cầu truy vấn cơ sở dữ liệu cho các hoạt động nhất định có thể bị giảm, mặc dù điều này có thể không phải luôn luôn như vậy.
* Nếu token được gửi trong Authorization header, Chia sẻ tài nguyên nguồn gốc chéo ( Cross-Origin Resource Sharing - CORS) sẽ không thành vấn đề vì nó không sử dụng cookie.

Sơ đồ sau đây cho thấy cách JWT được lấy và sử dụng để truy cập API hoặc resource:



1. Application hoặc client requests authorization đến authorization server. Điều này được thực hiện thông qua một trong các luồng authorization khác nhau. Ví dụ: một ứng dụng web tuân thủ OpenID Connect điển hình sẽ đi qua / oauth / ủy quyền điểm cuối bằng cách sử dụng luồng mã authorization.
2. Khi authorization được cấp, authorization server sẽ trả lại access token cho application.
3. Application sẽ sử dụng access token để truy cập vào resource (như API).

Xin lưu ý rằng với các signed tokens, tất cả thông tin trong token vẫn được hiển thị cho người dùng hoặc các bên khác, mặc dù họ không thể thay đổi thông tin đó. Điều này có nghĩa là bạn không nên đặt thông tin bảo mật trong token.

