[Bài đọc] Giới thiệu về JWT

**JWT là gì ?**

JWT là viết tắt của từ JSON Web Token là một tiêu chuẩn mở (([RFC 7519](https://tools.ietf.org/html/rfc7519)) định nghĩa một cách thức truyền tin an toàn giữa các thành viên bằng 1 đối tượng JSON. Thông tin này có thể được xác thực và đánh dấu là tin cậy dựa vào chữ ký của nó và phần chữ ký này của JWT sẽ dược mã hóa với thuật toán **HMAC**hoặc sử dụng cặp khóa public/private key của thuật toán **RSA.**

**Khi nào nên sử dụng JWT ?**

 Dưới đây là một vài tình huống mà chúng ta nên sử dụng đến Json Web Token:

* **Bảo mật**(**Authorization**): Đây là tình huống mà chúng ta hay sử dụng tới JWT nhất. Khi mà người dùng đăng nhập vào hệ thống, với mỗi yêu cầu sử dụng các tài nguyên, các service, đường dẫn của ứng dụng đó sẽ đều cần sử dụng JWT để xác thực quyền truy cập đến những tài nguyên đó. **Single Sign On**(Đăng nhập 1 lần) là tính năng của JWT được sử dụng phổ biến vì chi phí thấp và khả năng tái sử dụng của nó trên các domain khác nhau.
* **Trao đổi thông tin**(**Information Exchange**): JSON Web Token là một cách tốt để trao đổi thông tin giữa các bên với nhau.

**Cấu trúc của JWT**

Đoạn mã JSON Web Token bao gồm 3 thành phần chính được ngăn cách với nhau bởi dấu (.) đó là:

* Header
* Payload
* Signature

Do đó đoạn JWT của chúng ta sẽ có dạng như sau:

xxxx.yyyy.zzzz

Chúng ta cùng tìm hiểu kỹ hơn từng thành phần của JWT ở dưới đây:

**Header**

Phần Header thường bao gồm 2 thành phần như sau:

* Loại token – mặc định sẽ  là JWT.
* Thuật toán của chữ ký đang được sử dụng ví dụ như **HMAC, SHA256, RSA**.

Ta có ví dụ như sau:

{

  "alg": "HS256",

  "typ": "JWT"

}

Sau đó đoạn JSON trên sẽ được mã hóa theo **Base64Url** để tạo ra thành phần đầu tiên của **JWT.**Đoạn JSON trên sau khi được mã hóa Base64URL sẽ trở thành như sau:

String header = "{"alg":"HS256","typ":"JWT"}";

System.out.println(Base64.getUrlEncoder().encodeToString(header.getBytes()));

Và kết quả output sẽ là:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

**Payload**

Thành phần thứ 2 của JWT đó là payload – thành phần này sẽ chứa nôi dung của thông tin (claim). Thông tin có thể là mô tả của 1 thực thể ( thực thể user) hoặc cũng có thể là những thông tin bổ sung thêm cho phần Header. Chúng được chia làm 3 loại thông tin là: registered, public, và private.

* **Registered:** là những thông tin đã được quy định ở trong IANA JSON Web Token Claims registry. Những thông tin này không có cái nào là bắt buộc cả. Tuy nhiên tùy vào từng ưng dụng bạn implement mà hãy ràng buộc yêu cầu bắt buộc đối với những thông tin cần thiết.
  + **iss** (issuer): tổ chức phát hành token (không bắt buộc)
  + **sub** (subject): chủ đề của token (không bắt buộc)
  + **aud** (audience): Đối tượng sử dụng token (không bắt buộc)
  + **exp** (expired time): thời điểm token sẽ hết hạn (không bắt buộc)
  + **nbf** (not before time): token sẽ chưa hợp lệ trước thời điểm này
  + **iat** (issued at): thời điểm token được phát hành, tính theo UNIX time
  + **jti**: id của JWT
* **Public:**Khóa có thể được định nghĩa tùy theo ý muốn của người sử dụng JWT. Tuy nhiên để tránh việc bị trùng lặp thì chúng nên được quy định ở trong IANA JSON Web Token Registry hoặc 1 URI có chứa không gian không bị trùng lặp.
* **Private:**Phần thông tin thêm dùng để truyền qua giữa các client.

Ta có ví dụ cho phần Payload như sau:

{

  "sub": "nhs3108",

  "exp": 1558065420

}

Đoạn JSON trên sau khi được mã hóa base64url sẽ trở thành như sau

String payload = "{"sub":"nhs3108","exp":1558063837}";

System.out.println(Base64.getUrlEncoder().encodeToString(payload.getBytes()));

Kết quả của đoạn mã trên sẽ là

eyJzdWIiOiJuaHMzMTA4IiwiZXhwIjoxNTU4MDYzODM3fQ

**Signature**

Phần chữ ký của JWT được tạo từ phần mã hóa của Header và phần mã hóa của Payload với một mã bí mật (secrect) cùng với một thuật toán mã hóa bất kỳ ví dụ như HMAC SHA-256

Ví dụ:

HMACSHA256(

  base64UrlEncode(header) + "." +

  base64UrlEncode(payload),

  secret)

Giả sử chúng ta mã hóa chữ ký với header và payload ở trên ta sẽ thu được như sau:

String header = "{"alg":"HS256","typ":"JWT"}";

String encodedHeader = Base64.getUrlEncoder().encodeToString(header.getBytes());

String payload = "{"sub":"nhs3108","exp":1558063837}";

String encodedPayload = Base64.getUrlEncoder().encodeToString(payload.getBytes());

String signature = encodedHeader + "." + encodedPayload;

String encodedSignature = HMACSHA256.encode(signature, secretKey);

System.out.println(encodedSignature);

Kết quả hiển thị màn hình sẽ là

449KVmOFWcpsa3OUjnYGm-f1QWhY8N-DerKDfTK0JQm1Nc

Và sau khi đã thu đủ 3 thành phần Header, Payload và Signature chúng ta có đoạn mã JWT như sau:

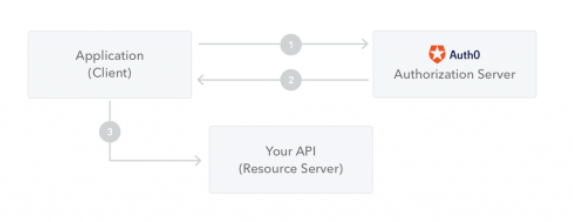
eyJhbGciOiJIUzI1NiIsxz4InR5cCI6IkpXVCJ9.eyJzdWIi54OiJuaHMzMTA4IiwiZXhwIjoxNTU4MDYzODM3fQ.449KVmOFWcpOUjnYGm-f1QWh54Y8N-DerKDfTK0JQm1Nc

**Làm thế nào để sử dụng JWT ?**

Trong bảo mật, khi người dùng đăng nhập thành công vào hệ thống, ứng dụng sẽ trả về một đoạn JWT. Và người dùng muốn sử dụng đoạn JWT đó để có thể truy cập tài nguyên của ứng dụng thì người dùng cần gửi đoạn JWT đó vào trong Header của mỗi request với thuộc tính **Authorization**và sử dụng **Bearer**và nội dung của header sẽ có dạng như sau:

Authorization: Bearer <token>

**Lưu đồ cách hoạt động của JWT**



Nhìn vào hình trên chúng ta có thể thấy cách hoạt động của JWT như sau:

* User thực hiện login bằng cách gửi id/password hay sử dụng các tài khoản mạng xã hội lên phía Authentication Server (Server xác thực)
* Authentication Server tiếp nhận các dữ liệu mà User gửi lên để phục vụ cho việc xác thực người dùng. Trong trường hợp thành công, Authentication Server sẽ tạo một JWT và trả về cho người dùng thông qua response.
* Người dùng nhận được JWT do Authentication Server vừa mới trả về làm “chìa khóa” để thực hiện các “lệnh” tiếp theo đối với Application Server.
* Application Server trước khi thực hiện yêu cầu được gọi từ phía User, sẽ verify JWT gửi lên. Nếu OK, tiếp tục thực hiện yêu cầu được gọi.