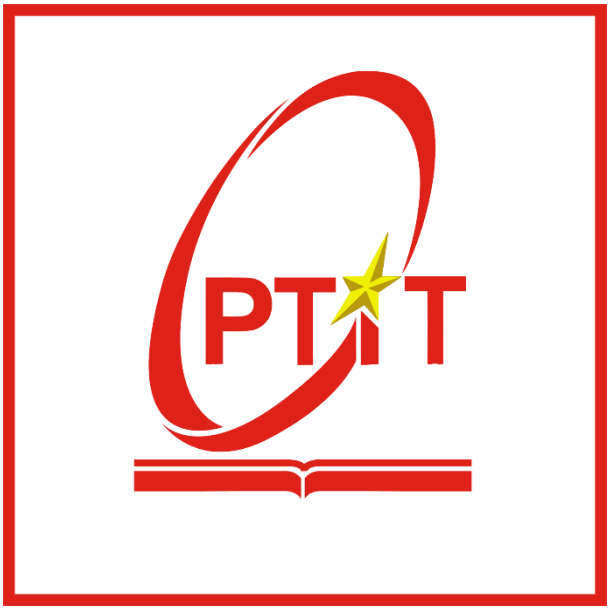
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 6

JWT – JSON Web Tokens và Session Cookies trong việc Authentication

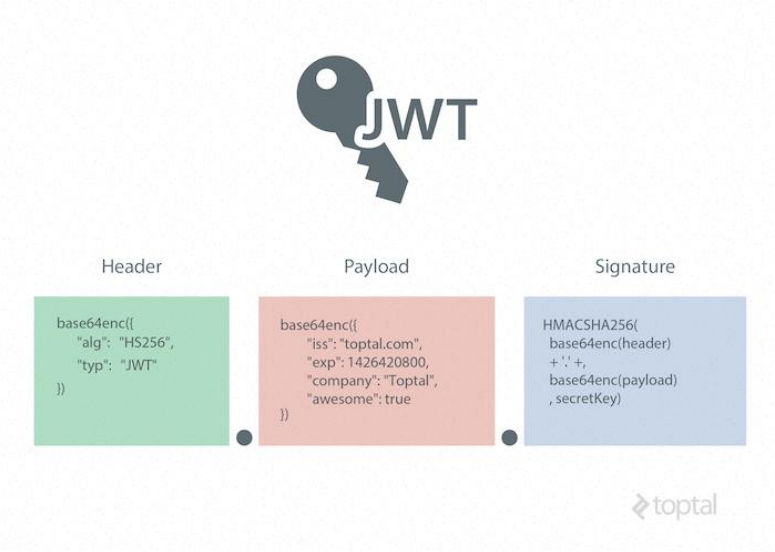
Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Lê Quang Thanh – B22DCVT509

I. JWT 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 “.”



II. Cấu trúc của JSON Web Token:

- JSON Web Token bao gồm 3 phần, được ngăn cách nhau bởi dấu chấm (.):

1. Header
2. Payload
3. Signature (chữ ký)

1. 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.

2. 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"

}

- Lưu ý không nên đặt quá nhiều thông tin trong chuỗi payload vì nó sẽ ảnh hưởng đến độ trể khi Server phải xác nhận một Token quá dài.

3. 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 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 sẽ có chuỗi signature như sau:

9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q

- Cuối cùng, kết hợp 3 chuỗi trên lại sẽ có được một chuỗi JWT hoàn chỉnh:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsidGVzdGp3dHJlc291cmNlaWQiXSwidXNlcl9uYW1lIjoiYWRtaW4iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNTEzNzE.9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q

III. 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

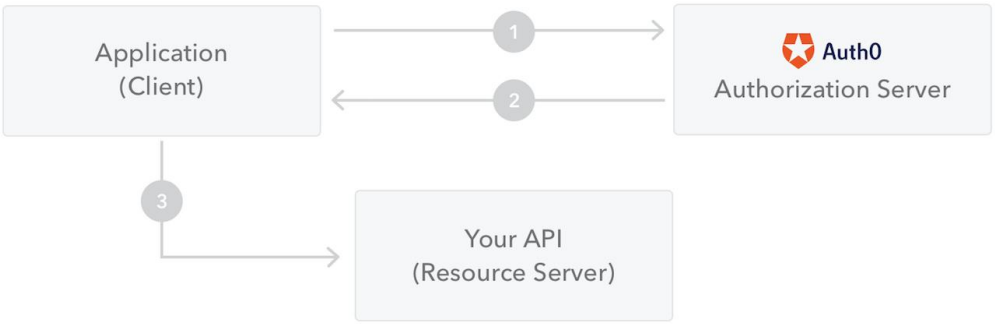
ngăn chặn các vấn đề bảo mật. Nói chung, không nên giữ token lâu hơn yêu cầu và cũng không nên lưu trữ dữ liệu nhạy cảm trên session trong bộ nhớ trình duyệt do thiếu bảo mật.

- 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. 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).

IV. 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 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.

V. Cơ chế xác thực đăng nhập bằng Session và Cookies (Session-Based Authentication)

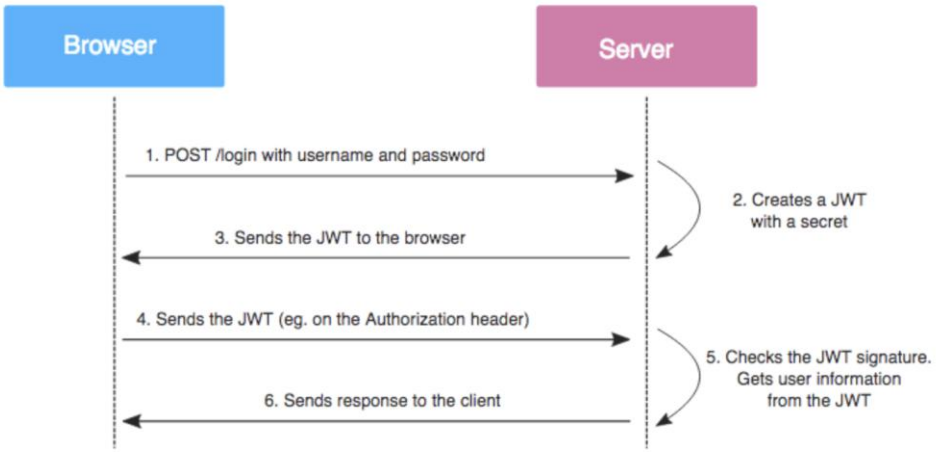
- Với cơ chế xác thực đăng nhập bằng Session và Cookies thì sau khi đăng nhập, server sẽ tạo ra session cho user và lưu vào đâu đó (có thể là file, memory, database,…). Sau đó một session ID sẽ được lưu vào trong cookies của trình duyệt. Trong khi user truy cập vào website thì session ID đó sẽ được trình duyệt lấy ra và gửi kèm theo trong request. Nhờ vậy mà server có thể tham chiếu đến session đã lưu trên server để biết user này đã đăng nhập hay chưa. Sau khi user log-out (đăng xuất) thì session sẽ bị xóa đi.

VI. Cơ chế xác thực đăng nhập bằng Token – (Token-Based Authentication)

- Khi nhắc đến cơ chế xác thực đăng nhập bằng Token (Token-Based Authentication) thì đa phần người ta thường nhắc đến JWT. Trước đây thì các trang web chỉ sử dụng cơ chế Session và Cookies tuy vậy hiện nay cơ chế sử dụng Token vô cùng phổ biến và dần trở thành một tiêu chuẩn cho việc xác thực đăng nhập.

- Cơ chế của JWT là như thế nào? Khi user đăng nhập thì server sẽ tạo ra một đoạn token được mã hóa và gửi lại nó cho client. Khi nhận được token này thì client sẽ lưu trữ vào bộ nhớ (thường sẽ là local storage). Sau đó mỗi khi client request lên server thì sẽ gửi kèm theo token. Từ token này server sẽ biết được user này là ai.

- Đối với việc lưu trữ JWT ở client, sẽ có vài trường hợp, nếu như là trình duyệt web thì JWT có thể lưu vào Local Storage, IOS app thì sẽ là Keychain và Android app sẽ lưu vào SharedPreferences.



VII. Kết luận

- Vậy cuối cùng cách xác thực nào là hiệu quả và bảo mật nhất? Xác thực bằng JWT là phương pháp sẽ có thể trở thành 1 tiêu chuẩn trong tương lai để dần thay thế cho Session, Cookies khi mà SPA đang trở thành xu thế. Ngoài ra nó có thể scale tốt hơn so với Session Cookies vì token sẽ được lưu trữ ở phía client trong khi Session thì phải lưu ở máy chủ và đây có thể là vấn đề khi một số lượng lớn người dùng sử dụng hệ thống cùng một lúc.

- Tuy nhiên, một nhược điểm với JWT là có kích thước lớn hơn nhiều so với session ID vì JWT chứa nhiều thông tin người dùng hơn. Ngoài ra chúng ta cần phải cần thận trong việc cho thông tin người dùng vào JWT vì nó có thể bị decode.