1. **Ứng Dụng .**

Trong thực người ta thường ứng dụng JWT trong 2 kịch bản đó là chứng thực người dùng và trao đổi thông tin.

1. **Ứng dụng của JWT trong chứng thực người dùng.**

Việc phân quyền xác thực người dùng trong các hệ thống là 1 việc rất cần thiết đối với bất cứ 1 hệ thống nào. Có rất nhiều cách để phân quyền cho 1 hệ thống mà ở đó ta tùy vào mức độ và yêu cầu của hệ thống để áp dụng. Một trong những cách truyền thống nhất đó là phân quyền bằng session ( Đã nói ở phần trước *), nhưng với nhiều ưu điểm ,tính linh hoạt cộng thêm với việc các ứng dụng được thiết kế với kiến trúc RESTful , microservice đang ngày càng được ưa dùng thì ứng dụng của JWT trong chứng thực người dùng trong các hệ thống ngày càng được sử dụng mạnh mẽ*.

Một kịch bản như sau : Người dùng đăng nhập vào hệ thống của bạn trong hệ thống của bạn , bạn muốn sử dụng JWT phân ra thành các chức năng theo quyền của người dùng đăng nhập, hãy xem qua cách JWT hoạt động thông qua kịch bản này

Về tư tưởng của bài toán phân quyền này như sau :

Đầu tiên, khi người dùng muốn được quyền truy cập vào một thông tin gì đó việc đầu tiên mà người dùng này phải làm là gửi yêu cầu nên cho server ,server dựa vào những điều kiện yêu cầu của người dùng trên và trả về cho người dùng 1 mã token, ta có thể gói 2 hành động trên của người dùng thành bước yêu cầu token. Sau khi có được mã token người dùng sử dụng các mã này trong các lần truy cấp tiếp theo bằng cách gửi n đi kèm trên header, bước này ta có thể gọi nó là bước truy cập và phân quyền. Về chi tiết các bước như sau :

B1: Yêu cầu token

B1.1. Đầu tiên tại form login yêu cầu người dùng nhập thông tin đăng nhâp ( thường gồm các thông tin : USERNAME , PASSWORD )

B1.2. Server nhận request từ người dùng thực hiện xây dựng mã token theo cấu trúc H.P.S rồi gửi về cho người dùng

B2 : Truy cập và phân quyền – trong các lần truy cập tiếp theo người dùng sẽ gửi kèm theo mã token được nhận nên cho server. Server có trách nghiệm verify chuỗi jwt để xác nhận người dùng Câu hỏi đặt ra ở đây là hệ thống Verify JWT thế nào:

* Chuỗi token có cấu trúc H.P.S được client gửi nên server khi nhận được token server sẽ sử lý tương tự như sau :

Set S1 = S => Set S2 = **HMAC**(SHA256(**H.P**) vỡi secret key của hệ thống) (Giả sử hệ thống sử dụng encryption algorithms SHA256)

Sau đó so sánh S1 vs S2

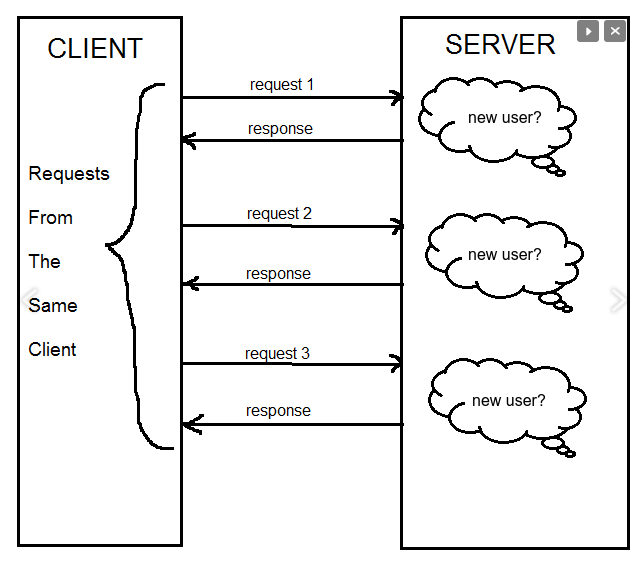
* Nếu S1 và S2 khớp nhau, tức là chữ ký hợp lệ, hệ thống mới tiếp tục thực hiện dao dịch với người dùng

(DEMO)

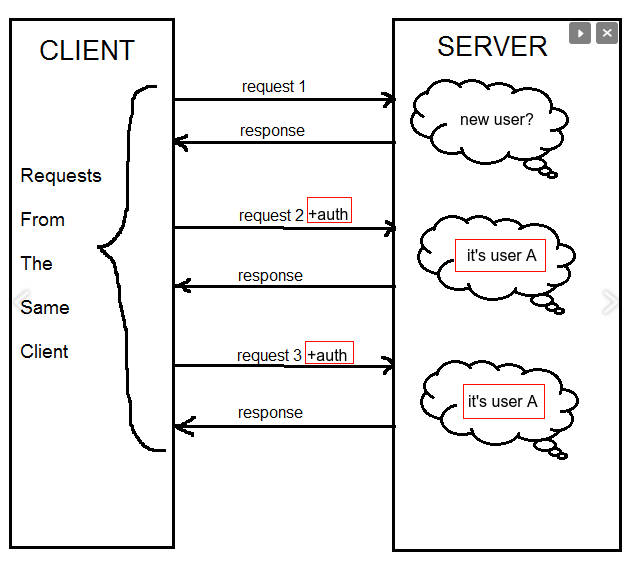
Giải thích tại sao lại nói “*các ứng dụng được thiết kế với kiến trúc RESTful , microservice đang ngày càng được ưa dùng thì ứng dụng của JWT trong chứng thực người dùng trong các hệ thống ngày càng được sử dụng mạnh mẽ*”

Như đã biết, RESTful web service sử dụng HTTP protocol như một phương tiện giao tiếp và HTTP request là stateless protocol. Tức là server không lưu giữ bất kỳ thông tin nào của client, server xử lý các request một cách độc lập, không phụ thuộc vào trạng thái hay kết quả của request trước.

Như hình bên dưới, do server không lưu giữ bất kỳ thông tin nào của request trước. Nên mỗi request gửi lên server đều phải phải chứng thực lại, mặc dù là request của cùng một user đã được chứng thực.



Một trong những cách để giải quyết vấn đề này là mỗi request client gửi lên đều gửi kèm thông tin đã chứng thực trước đó.



Và JWT là 1 cách ứng dụng có thể thỏa mãn yêu cầu trên, đó là lý do nói các ứng dụng được thiết kế với kiến trúc RESTful , microservice đang ngày càng được ưa dùng thì ứng dụng của JWT trong chứng thực người dùng trong các hệ thống ngày càng được sử dụng mạnh mẽ

1. **Ứng dụng của JWT trao đổi thông tin.**

JWT là 1 cách thức không tồi để truyền tin an toàn giữa các thành viên với nhau, nhờ vào phần “chữ ký” của nó. Phía người nhận có thể biết được người gửi là ai thông qua phần chữ ký. Ngoài ra, 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.

1. **Tính an toàn của JWT và hiệu suất của JWT và các ý tưởng phát triển.**
2. **Tính an toàn**

PhầnSigning JWTs đã nhằm mục đích ngăn chặn sự giả mạo ở phía máy khách, nhưng chúng cũng có thể được mã hóa để đảm bảo rằng token rằng mã thông báo mang rất an toàn. Hiện nay JWT chủ yếu được lưu trữ trực tiếp trong bộ lưu trữ web (local/session storage) hoặc ở cookies. Nói chung, giống như việc sử dụng Cookied-based authentication, việc bạn để mất token (JWT hay SessionID) vào tay người khác cũng đồng nghĩa với việc trao quyền truy cập cho họ. Giống như việc bạn mang chìa khóa nhà mình đi đánh mới cho người khác. Việc của bạn lúc này là phải thay ngay cái ổ khóa mới mà thôi, nghĩa là làm cho cái khóa đang mở được cửa nhà bạn trở nên vô nghĩa. Với JWT, điều này có thể được hạn chế bằng việc rút ngắn thời gian valid (giảm expiration date) để tăng cường độ thay đổi JWT của bạn (giống như việc bạn thay đổi ổ khóa)

Chú ý rằng: Để ngăn chặn việc bị đánh cắp token, sử dụng HTTPS/SSL để đảm bảo rằng Cookies và JWTs luôn được encrypted trong quá trình trao đổi client-sever. (Ví dụ như ngăn ai đó dùng Wireshark theo đổi request đọc request của bạn)

việc phân tích này là cực kì cần thiết. Khi tạo một request từ client tới server, nếu một lượng lớn thông tin đc mã hóa trong JWT, nó sẽ tạo ra một luợng lưu trữ đáng kể trong mỗi HTTP request. Tuy nhiên, với session, thì chỉ có lượng vô cùng nhỏ bởi SESSION ID thực tế chỉ là một chuỗi mã hóa unique để định danh mà ko mang theo thông tin gì khác.

Tuy có nhiều điểm mạnh nhưng vấn đề về tính an toàn vẫn luôn là 1 vấn đề khiến những người phát triển hệ thống có JWT đau đầu. **Cross-site Scripting (XSS) attack** là hình thức tấn công thường gặp nhất khi sử dụng **Javascript** để giải quyết các vấn đề bảo mật. Các hacker sẽ lợi dụng các dữ liệu input của người dùng để chèn các đoạn javascript code độc hại nhằm lấy cắp JWT token của nạn nhân, sau đó sử dụng để mạo danh người dùng.

Để có được 1 hệ thống đảm bảo tốt về mặt an toàn , chỉ có JWT không là không đủ nhưng nếu chúng ta áp dụng các kỹ thuật khác (Sử dụng https ,không truyền param nhạy cảm vào URL, kiểm soát đầu vào ,...) vào trong dự án hoặc hệ thống thì để phát huy tối đa tác dụng của nó thì thực sự là rất tốt

1. **Hiệu suất của JWT**

Khi được mã hóa, dung lượng của JWT này sẽ gấp nhiều lần 1 SESSION\_ID (chỉ dùng để định danh), do đó việc tạo ra JWT sẽ nặng hơn đáng kể ở mỗi request. Với session, những thông tin này sẽ có server tìm kiếm dựa vào SESSION ID mà mình gửi lên.

JWT lưu giữ thông tin dưới client (bản thân JWT là một chuỗi chứa thông tin, sau đó dc mã hóa để server side giải mã đọc thông tin). **Điều này giúp mỗi request gửi lên không phải truy vấn quá nhiều và liên tục vào cơ sở dữ liệu** (Như các bạn thấy trong mục **3.2 Hệ thống Verify chuỗi JWT thế nào?** , hệ thống sẽ chỉ cần kiểm tra chữ ký Signature và 1 số thông tin trong claims là đủ). Hãy tưởng tự 1 web page angular với hàng loạt các component thực hiện hàng loạt request tới server. Mỗi request đều cần xác thực để cấp quyền truy cập tài nguyên thì sẽ có bấy nhiêu lần phải truy vấn DB. Với số lượng người dùng lớn đồng thời thì thực sự là có vấn đề đó đúng ko các bạn). Tuy nhiên hãy cẩn trọng đừng đặt quá nhiều claims trong JWT để tránh request quá lớn. (Payload càng nhiều thông tin thì chuỗi JWT càng dài).