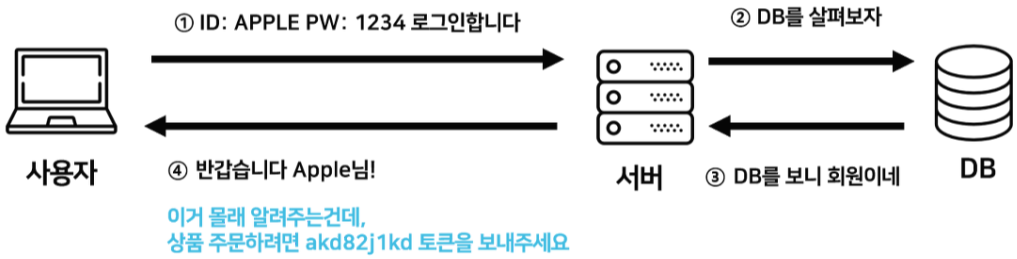
JWT 토큰

발표일	6/14
분류	Security Spring
사람	강희 이

공부한 내용

-

JWT (JSON Web Token) 이란



JWT(JSON Web Token)란 인증에 필요한 정보들을 암호화시킨 JSON 토큰을 의미한다. 그리고 JWT 기반 인증은 JWT 토큰(Access Token)을 HTTP 헤더에 실어 서버가 클라이언트를 식별하는 방식이다

JWT는 JSON 데이터를 Base64 URL-safe Encode 를 통해 인코딩하여 직렬화한 것이며, 토큰 내부에는 위변조 방지를 위해 개인키를 통한 전자서명도 들어있다. 따라서 사용자가 JWT 를 서버로 전송하면 서버는 서명을 검증하는 과정을 거치게 되며 검증이 완료되면 요청한 응답을 돌려준다.

> Info
>
> Base64 URL-safe Encode 는 일반적인 Base64 Encode 에서 URL 에서 오류없이 사용하도록 '+', '/' 를 각각 '-', '_' 로 표현한 것이다.

JWT 구조

JWT는 **.** 을 구분자로 나누어지는 세 가지 문자열의 조합이다.

. 을 기준으로 Header, Payload, Signature를 의미한다.

Header 에는 JWT 에서 사용할 타입과 해시 알고리즘의 종류가 담겨있으며, Payload 는 서버에서 첨부한 사용자 권한 정보와 데이터가 담겨있다. 마지막으로 Signature 에는 Header, Payload 를 Base64 URL-safe Encode 를 한 이후 Header 에 명시된 해시함수를 적용하고, Private Key로 서명한 전자서명이 담겨있다.

실제 디코딩된 JWT는 다음과 같은 구조를 지닌다.

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

- alg : 서명 암호화 알고리즘(ex: HMAC SHA256, RSA)
- typ : 토큰 유형

Payload

토큰에서 사용할 실제 JWT 를 통해서 알 수 있는 데이터인 Claim 이 담겨있다.

즉, 서버와 클라이언트가 주고받는 시스템에서 실제로 사용될 정보에 대한 내용을 담고 있는 섹션이다.

> Tip
>
> key-value 형식으로 이루어진 한 쌍의 정보를 Claim이라고 칭한다.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

페이로드는 정해진 데이터 타입은 없지만, 대표적으로 Registered claims, Public claims, Private claims 이렇게 세 가지로 나뉜다.

```
{
    "jti": "1000",                          // Registered Claim
    "exp": "1521430000000",                 // Registered Claim
    "https://kevin.tistory.com": true, // Public Claim
    "username": "kevin"                     // Private Claim
}
```

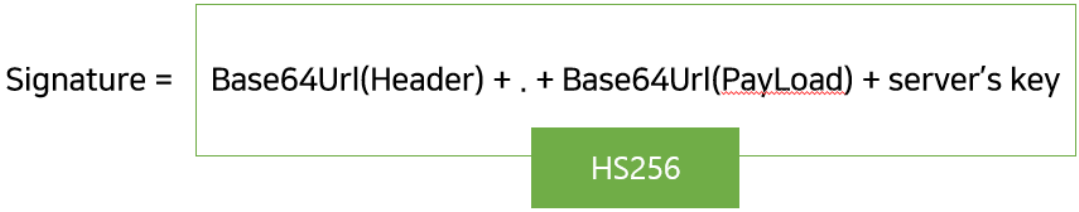
- Registed claims : 미리 정의된 클레임.
  - iss(issuer; 발행자),
  - exp(expireation time; 만료 시간),
  - sub(subject; 제목),
  - iat(issued At; 발행 시간),
  - jti(JWI ID)
- Public claims : 사용자가 정의할 수 있는 클레임 공개용 정보 전달을 위해 사용.
- Private claims : 해당하는 당사자들 간에 정보를 공유하기 위해 만들어진 사용자 지정 클레임. 외부에 공개되도 상관없지만 해당 유저를 특정할 수 있는 정보들을 담는다

Signature

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

시그니처에서 사용하는 알고리즘은 헤더에서 정의한 알고리즘 방식(alg)을 활용한다.

시그니처의 구조는 (헤더 + 페이로드)와 서버가 갖고 있는 유일한 key 값을 합친 것을 헤더에서 정의한 알고리즘으로 암호화를 한다.



JWT 장점

1. Header와 Payload를 가지고 Signature를 생성하므로

   데이터 위변조를 막을 수 있다.

2. 인증 정보에 대한

   별도의 저장소가 필요없다.

3. JWT는 토큰에 대한 기본 정보와 전달할 정보 및 토큰이 검증됬음을 증명하는 서명 등 필요한 모든 정보를 자체적으로 지니고 있다.

4. 클라이언트 인증 정보를 저장하는 세션과 다르게, 가 되어 서버 확장성이 우수해질 수 있다.

   서버는 무상태(StateLess)

5. 토큰 기반으로

   다른 로그인 시스템에 접근 및 권한 공유가 가능하다. (쿠키와 차이)

JWT 단점

1. Self-contained : 토큰 자체에 정보를 담고 있으므로 양날의 검이 될 수 있다.
2. 토큰 길이 : 토큰의 Payload에 3종류의 클레임을 저장하기 때문에, 정보가 많아질수록 토큰의 길이가 늘어나 네트워크에 부하를 줄 수 있다.
3. Payload 인코딩 : payload 자체는 암호화 된 것이 아니라 BASE64로 인코딩 된 것이기 때문에, 중간에 Payload를 탈취하여 디코딩하면 데이터를 볼 수 있으므로, payload에 중요 데이터를 넣지 않아야 한다.
4. Store Token : stateless 특징을 가지기 때문에, 토큰은 클라이언트 측에서 관리하고 저장한다. 때문에 토큰 자체를 탈취당하면 대처하기가 어렵게 된다.

더 궁금한 점

- 쿠키와 세션에 대해서도고 싶다

같이 얘기하고 싶은 점

- typ에 왜 e가 빠졌을까 궁금합니다(**진짜 궁금함**) 제일 궁금해요.
    - 오타인가?….
        - 그냥 그렇게 하고 싶었던거 아닐까..
- 단점의 대한 해결책
    - refresh token