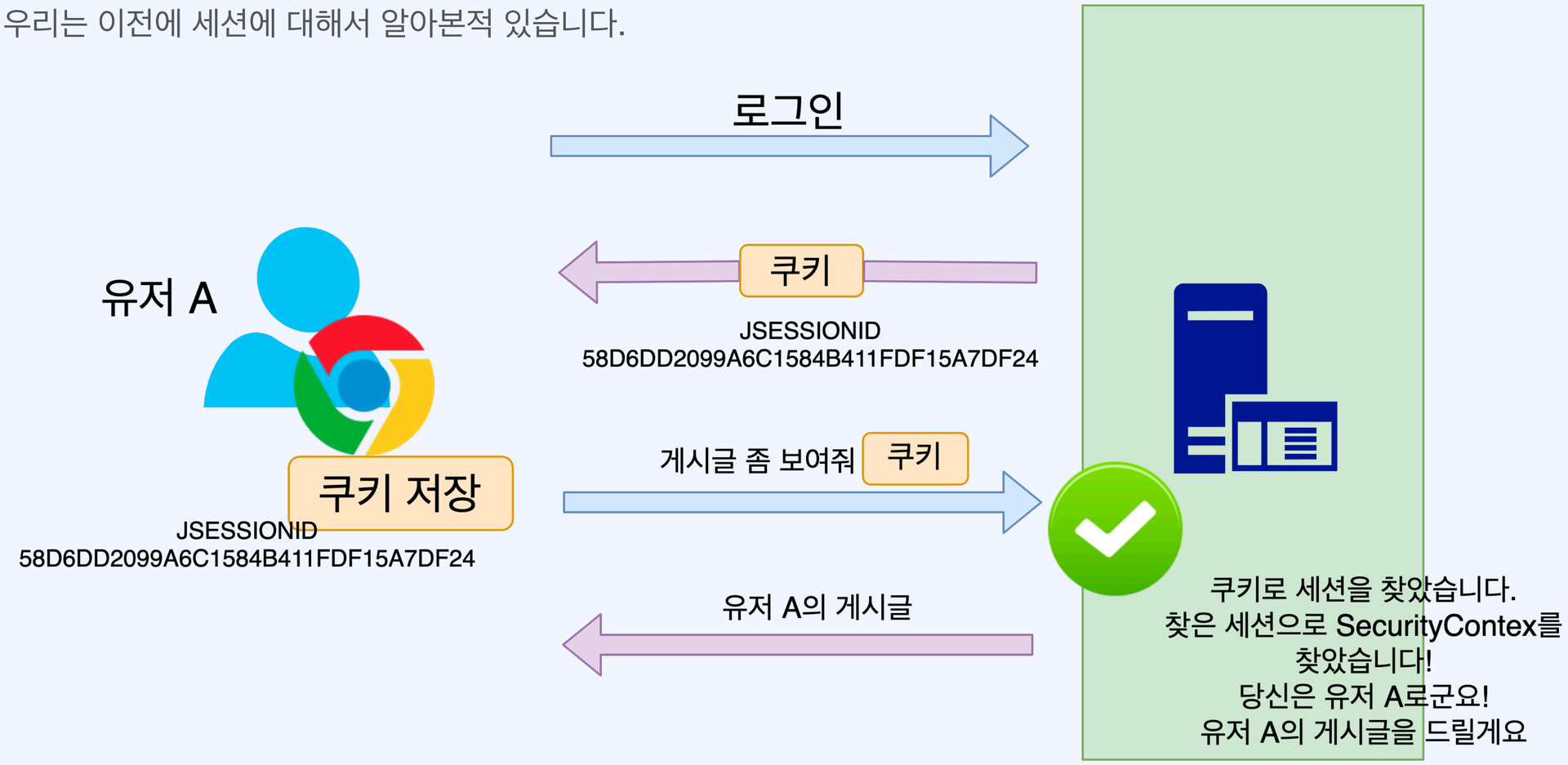
Spring Security

5 JWT 기본개념

토큰으로 인증하기

우리는 이전에 세션에 대해서 알아본적 있습니다.



토큰으로 인증하기

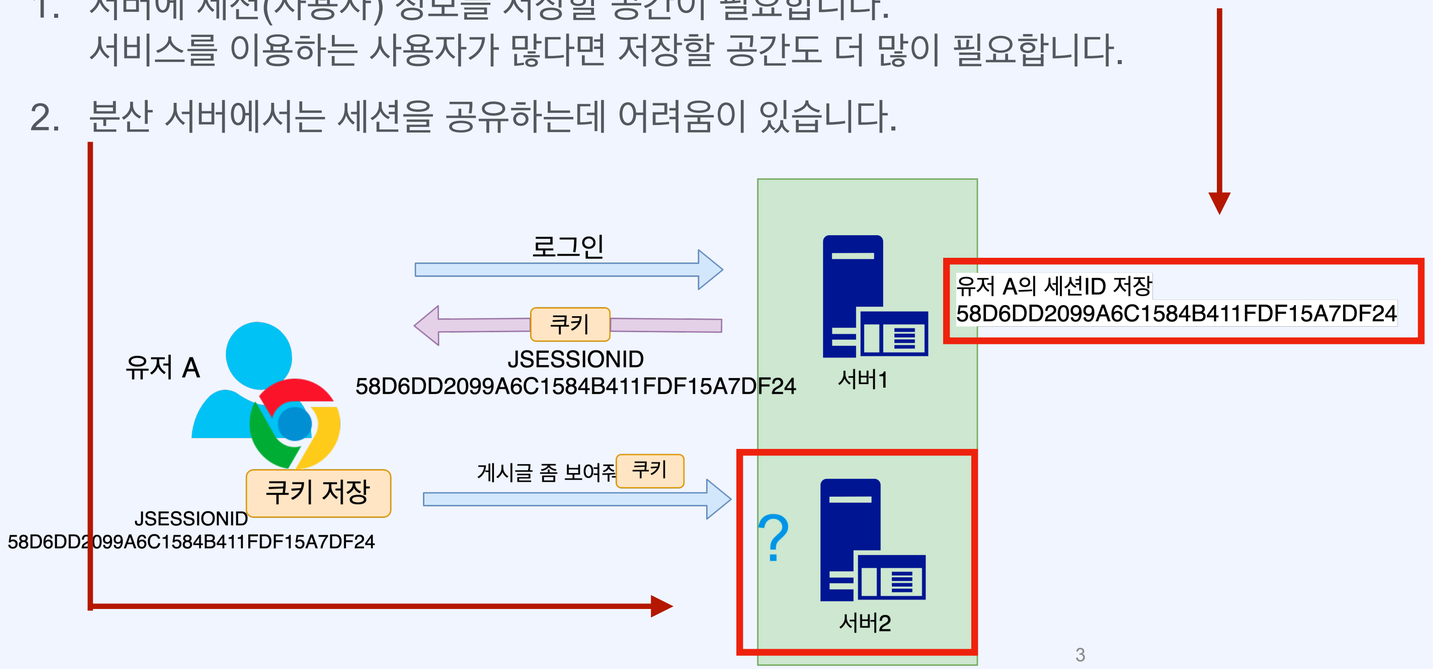
세션은 이런 장단점을 가지고 있습니다.

장점

1. JSESSIONID는 유의미한 값이 아니라 서버에서 세션(사용자) 정보를 찾는 Key로만 활용합니다.
   따라서 탈취되었다고 해서 개인정보가 탈취된건 아닙니다.
   (세션하이제킹 공격을 당할수 있기 때문에 절대적으로 안전하다는 뜻은 아닙니다.)

단점

1. 서버에 세션(사용자) 정보를 저장할 공간이 필요합니다.
   서비스를 이용하는 사용자가 많다면 저장할 공간도 더 많이 필요합니다.
2. 분산 서버에서는 세션을 공유하는데 어려움이 있습니다.



토큰으로 인증하기

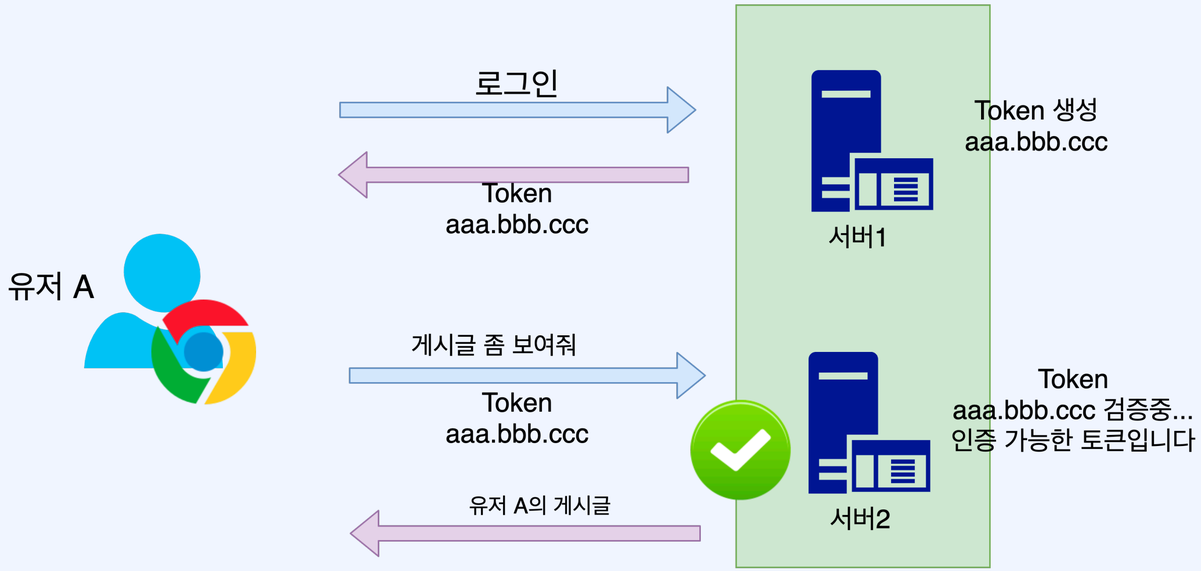
세션의 단점을 해결하기 위해서 토큰으로 인증하는 방법을 사용할 수 있습니다.

유저가 로그인을 하면 서버에서는 토큰을 생성한 뒤에 저장하지 않고 (stateless) 토큰값을 내려줍니다.

이 토큰 값을 유저가 게시글 조회 요청을 할 때 함께 보내고 서버2(서버1이여도 상관없음)에서 이 토큰을 의미있는 값으로 해석합니다. 그리고 그 값을 토대로 유저를 인증합니다.

세션방식에서의 JSESSIONID은 key로만 활용될 수 잇는 의미없는 값이 들어있습니다.

그러나 토큰은 유저를 설명할 수 있는 데이터를(ex username)포함합니다.



토큰으로 인증하기

토큰 방식은 다음과 같은 장단점이 있습니다.

장점

1. 세션관리를 할필요가 없어 별도의 저장소가 필요하지 않습니다.
2. 서버 분산&클러스터 환경과 같은 확장성에 좋습니다.

단점

1. 한 번 제공된 토큰은 회수가 어렵습니다.
   세션의 경우에는 서버에서 세션을 삭제해버리면 브라우저의 JSESSIONID는 무용지물이 되어버립니다.
   그러나 토큰은 세션을 저장하지 않기 때문에 한번 제공된 토큰을 회수할수 없습니다.
   그래서 보통 토큰의 유효기간을 짧게 합니다.
2. 토큰에는 유저의 정보가 있기 때문에 상대적으로 안정성이 우려됩니다.
   따라서 민감정보를 토큰에 포함시키면 안됩니다. (ex 패스워드, 개인정보)

JWT의 구조

토큰 방식에서 가장 잘 알려진 JWT가 있습니다.

Json Web Token의 줄임말입니다.

JWT구조는 아래와 같습니다.

HEADER.PAYLOAD.SIGNATURE

예시)

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyIiwicm9sZXMiOI.hHkvviDzMIx8EkXUTZsNLWGy51p2
bVnJknEC0HYxMaRkXGQJdIWqIX1Rv8rGK6bMq6mYyGES3jxNJVPz33wvEQ

JWT의 구조

Header

HEADER는 JWT를 검증하는데 필요한 정보를 가진 객체입니다.
Signature에 사용한 암호화 알고리즘이 무엇인지, Key의 ID가 무엇인지 정보를 담고 있습니다.
이 정보를 Json으로 변환해서 UTF-8로 인코딩한 뒤 Base64 URL-Safe로 인코딩한 값이 들어가 있습니다.
결과 값이 난해한 문자로 보이지만 암호화된 값은 아닙니다.

```
{
    "alg": "HS512",
    "kid": "key1"
}
```

eyJhbGciOiJIUzUxMiIsImtpZCI6ImtleTEifQ

JWT의 구조

Payload

실질적으로 인증에 필요한 데이터를 저장합니다.
데이터의 각각 필드들을 Claim이라고 합니다.
대부분의 경우에 Claim에 username을 포함합니다.
인증할 때 payload에 있는 username을 가져와서 유저 정보를 조회할 때 사용해야하기 때문입니다.
또한, 토큰 발행시간(iat)와 토큰 만료시간(exp)를 포함합니다.
그 외에도 원하는 Claim을 얼마든지 추가할수 있지만 민감정보는 포함시켜서는 안됩니다.
Payload 역시 Header와 마찬가지로 암호화되지 않습니다.
Json으로 바꾼뒤 UTF8로 인코딩하고 Base64로 변경한 데이터일 뿐입니다.

```
{
  "sub": "user",
  "iat": 1629626974,
  "exp": 1629627574
}
```

eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNjI5NjI2OTc0LCJleHAiOjE2Mjk2Mjc1NzR9

JWT의 구조

Signature

앞선 Header와 Payload는 암호화하지 않았고 json -> utf8 -> base64로 변환한 데이터입니다.
이렇게 Header와 Payload 생성 메커니즘은 너무 쉽고 누구나 만들 수 있는 데이터입니다.
따라서 저 두개의 데이터만 있다면 토큰에 대한 진위여부 판단은 전혀 이루어질수 없습니다.
그래서 JWT의 구조에서 가장 마지막에 있는 Signature는 토큰 자체의 진위여부를 판단하는 용도로 사용합니다.
Signature는 Header와 Payload를 합친뒤 비밀키로 Hash를 생성하여 암호화합니다.

헤더.페이로드 값을 SecretKey로 Hashing하고 Base64로 변경합니다.

Header Payload

```
eyJraWQiOiJrZXkxIiwiYWxnIjoiSFM1MTIifQ.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNjI5NjI2OTc0LCJleHAiOjE2Mjk2Mjc1NzR9
```

HS512 + Secret Key

```
TrK2IFel3Rv5ftoPXtLTEMFwHq8o-tTLMVUTR51zwP48VfOvkIJ4GLHzBipfxrCLYz0QBZosVHkR4xjpn2bIlA
```

JWT의 구조

Key Rolling

JWT의 토큰 생성 메커니즘을 보다보면 Secret Key가 노출되면 사실상 모든 데이터가 유출될 수 있다는 걸 알수 있습니다.
이런 문제를 방지하기 위해서 Secret Key를 여러개 두고 Key 노출에 대비할 수 있습니다.
Secret Key를 여러개를 사용하고 수시로 추가하고 삭제해줘서 변경한다면
SecretKey 중에 1개가 노출되어도 다른 Secret Key와 데이터는 안전한 상태가 됩니다.
이걸 바로 Key Rolling이라고 합니다. (Key Rolling이 필수는 아닙니다)

Key Rolling에서는 여러개의 Secret Key가 존재합니다.
Secret Key 1개에 Unique한 ID (kid 혹은 key id라고 부름) 를 연결시켜 둡니다.
JWT 토큰을 만들 때 헤더에 kid를 포함하여 제공하고 서버에서 토큰을 해석할 때 kid로 Secret Key를 찾아서 Signature를 검증합니다.