# 차량 매칭 서비스에서 JWT를 활용한 효율적인 인증방법

*김언동
계명대학교
kimud6003@gmail.com

**박요한
계명대학교
yhpark@kmu.ac.kr

# Efficient Authentication Scheme using JWT for Ride Sharing Services

*UnDong Kim
College of Engineering,
Dept. of Computer
Engineering,
Keimyung University

**YoHan Park
College of Engineering,
Dept. of Computer
Engineering,
Keimyung University

## 요약

최근 4차 산업혁명이 가속화되면서 IT 기반 산업들이 제품 중심 사업에서 서비스 위주의 사업으로 변화하는 양상을 보인다. 그에 뒤 받침 하듯이 세계 스타트업 기업 가치를 측정해 보았을 때 1위부터 4위까지 모두 IT 기반의 사업이라는 것을 확인할 수 있다. 그중 750억 달라 가치를 가진 Uber, 560억 가치를 가진 디디추싱은 모두 차량 공유 시스템을 서비스를 기업으로, 세계 각국의 사람들의 현재 차량 매칭 서비스에 관심이 집중 되어있다. 이러한 차량 매칭 서비스는 사용자들 실시간 매칭을 해주기 때문에 많은 사람들이 서버에 접속하여 인증을 요청하게 된다. 또한 기업은 많은 사람들의 실시간 인증을 처리해야하기 때문에 다수의 요청을 처리하는 것은 시스템 사양을 결정하게 되는 중요한 요소가 된다. 이러한 서버 기반 인증 방식은 메모리에 부하가 걸리는 문제가 발생한다. 본 논문에서는 다수의 사람들이 들어오는 차량 매칭 서비스에서 토큰 인증 기반 방식인 JWT를 활용하여 기존의 서버에 사용자의 정보를 저장하는 세션 인증 방식보다 좀 더 효율적인 인증 방식을 제안하고자 한다.

## 1. 소개

4차 산업 시대에 들어섬에 따라 대중들은 생활 전반의 수많은 정보와 서비스 들을 인터넷을 통해 접하고 있다. 또한 많은 IT 기반 산업들 역시 제품 중심에서 서비스 중심으로 변화하는 양상을 보인다[1]. 이러한 서비스 중심의 사업에서 미국의 승차 공유 서비스를 기반으로 한 uber는 스타트업 5년 만에 기업 가치 75조 원을 넘는 것을 보여주며 세계의 많은 사람들이 차량 매칭 서비스에 관심이 많다는 것을 보여주었다[2,3].

차량 매칭 서비스는 승객과 차량을 실시간 매칭해주는 것이기 때문에 많은 사람들이 서버에 접속하여 인증을 요청하게 되고 사용자들의 인증 요청을 처리하는 것은 시스템의 사양을 결정하는 요소가 된다. 하지만 세션을 활용한 서버 기반의 인증 방식에서는 사용자가 인증을 위해 서버에 인증 요청을 하게 되며 서버는 사용자의 정보를 담은 세션을 메모리에 저장한다. 따라서 로그인 중인 사용자가 늘어날 경우에는 서버의 메모리에 부하가 걸리게 되는 문제가 발생한다.

본 논문에서는 이러한 서버 기반의 인증의 단점을 보완하는 JSON Web Token (JWT)을 사용하여 차량 매칭 서비스에서 보다 효율적인 인증을 하고자 한다.

## 2. 토큰 인증 기반

그림1은 토큰 기반의 인증 방식에 대한 설명이다[4]. 토큰 기반의 인증 방식은 인증을 요청하는 사용자에게 토큰을 발급한다. 그 후 사용자가 서버에 요청을 할 때 토큰을 전송하여 유효성 검사를 한다. 이러한 방식으로 서버는 사용자의 인증 정보를 서버나 세션에 유지하지 않고 클라이언트에서 들어오는 요청만으로 작업을 처리할 수 있다. 즉, 서버 기반의 인증 시스템과 달리 상태를 유지하지 않으므로 Stateless한 구조를 갖는다. 이러한 토큰 기반의 인증 방식을 통해 수많은 문제점들을 해결할 수 있는데, 대표적으로 사용자가 로그인이 되어있는지 안 되어 있는지 신경 쓰지 않고 손쉽게 시스템을 확장할 수 있다. 또한 보안성으로 클라이언트가 서버로 요청을 보낼 때 더 이상 쿠키를 전달하지 않으므로

** 교신저자

쿠키 사용에 의한 취약점이 사라지게 된다.

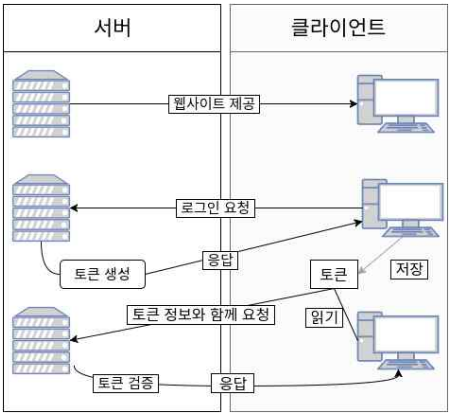


그림 1. 토큰 인증 방식.

## 3. JWT

JSON Web Token (JWT) 은 웹 표준 (RFC 7519)으로서 두 개체에서 JSON 객체를 사용하여 가볍고 정보를 안전성 있게 전달해 주는 토큰 인증방법이다. 또한 JWT 역시 토큰 인증 기반 방식이기에 앞서 언급한 토큰 인증 기반 방식 그림1과 같이 진행된다. JWT의 특징으로는 자가 수용적(self-contained)이라는 것인데 발급된 토큰에 유저 정보, 토큰 자체에 대한 정보, signature를 포함하고 있다[6]. 그림 2는 JWT의 구조에 대한 설명이다.


그림 2 JWT 구조.

구조는 크게 3가지로 나누어지며 각 구조에 대해 설명 하도록 하겠다. 먼저 Header는 두 가지의 정보를 지니고 있는데, 첫 번째는 토큰의 타입, 두 번째는 해싱 알고리즘을 지정한다.

해싱 알고리즘은 보통 HMAC SHA256 혹은 RSA 가 사용되며, 이 알고리즘은, 토큰을 검증할 때 사용되는 signature 부분에서 사용된다. 다음은 payload로 담을 정보가 들어가 있는 부분이다. 여기에 담는 정보의 한 '조각을 클레임(claim)이라고 부르고, 이는 name/value의 한 쌍으로 이뤄져 있다. 클레임의 종류에는 크게 3가지가 있는데 registered claim, public claim, private claim이 있다.

registered claim은 서비스에서 필요한 정보들이 아닌, 토큰에 대한 정보들을 담기 위하여 이름이 이미 정해진 클레임들이다. 등록된 클레임의 사용은 모두 선택적 (optional)이며, 이에 포함된 클레임 이름들은 다음 표 1과 같다.

다음 public claim은 충돌이 방지된 (collision-resistant) 이름을 가지고 있고, 충돌을 방지하기 위해서 클레임 이름을 URI 형식으로 저장한다.

표 1. payload에 들어가는 클레임의 종류

| 속성 | 설명 |
|---|---|
| iss | 토큰 발급자 (issuer) |
| sub | 토큰 제목 (subject) |
| aud | 토큰 대상자 (audience) |
| exp | 토큰의 만료시간 (expiration) |
| nbf | Not Before |
| iat | 토큰이 발급된 시간 (issued at) |
| jti | JWT의 고유 식별자 |

마지막으로 private claim은 두 참여자(클라이언트 <->서버) 사이에 협의 하에 사용되는 클레임 이름이다. 간단한 payload의 형식은 다음 그림 3과 같다.

```
{
    "iss": testl.com",
    "exp": "11244230000000",
    "https://test.com/jwt_claims/is_admin": true,
    "userId": "10044230000000",
    "username": "test"
}
```

그림 3. private claim에 들어가는 클레임의 종류.

마지막 signature은 헤더의 인코딩 값과, 정보의 인코딩 값을 합친 후 주어진 비밀키로 해시를 하여 생성하는 구조로 이루어진다. 이러한 JWT의 주요한 이점은 사용자 인증에 필요한 모든 정보는 토큰 자체에 포함하기 때문에 별도의 인증 저장소가 필요 없다는 것이다. 그렇기 때문에 서버의 확장성이나 쿠키를 사용하는 방식보다 안전하다.

## 4. JWT 기반 시스템 구현 및 결과

JWT를 이용한 차량 매칭 서비스를 웹으로 구현하기 위해 javascript의 프레임워크 Reactjs를 사용하여 웹 프론트를 구성했다. 또한 백엔드 역시 같은 언어인 javascript의 프레임 워크 Node.js를 이용하여 서버를 구축했다. 다음 표 2는 시스템 구현 환경에 대한 설명이다.

표 2. 시스템 구현 환경

| 항목 | 환경 |
|---|---|
| 웹 프론트 | Web-Front : Reactjs |
| 백엔드 | Web-Server : Node.js |
| 데이터베이스 | Data-Base : Postgresql |
| 개발 환경 | Mac os |
| 개발 도구 | visual studio cod |

사용자가 웹서비스에 로그인을 하기 위해 핸드폰 번호를 입력하게 되면 서버에서 해당 핸드폰 번호에 인증번호를 날리게 된다. 사용자는 자신의 문자에 온 인증번호를 웹 사이트에 다시 입력하여 로그인을 요청하게 된다. 서버는 사용자의 인증번호와 서버에서 만든 인증번호를 확인하여 두 번호가 동일하였을 경우 token을 생성 하는 함수에서 토큰을 반환 하게 된다. 토큰을 생성하는 코드는 아래 그림 4와 같다.

```
import jwt from "jsonwebtoken";
const createJWT = (id: number): string => {
  const token = jwt.sign({
      id
    },
    process.env.JWT_TOKEN(비밀키)
  );
  return token;
};
export default createJWT;
```

그림 4. 토큰 생성 코드.

또한 토큰을 발급하면 토큰을 통해 인증해야 한다. 사용자가 자신이 받은 토큰을 통해 인증 정보를 서버에게 전달하고 서버는 매번 이 인증 정보를 확인해야 함으로 토큰을 해독하는 함수도 필요하다. 해당 토큰을 해독하는 함수는 다음 그림 5와 같다.

```
import jwt from "jsonwebtoken";
import User from "../entities/User";
const decodeJWT = async (token: string):
Promise <User |undefined >=> {
  try {
    const decoded: any =
   jwt.verify(
          token, process.env.JWT_TOKEN
          );
    const { id } = decoded;
    const user = await User.findOne({ id });
    return user;
  } catch (error) {
    return undefined;
  }
};
export default decodeJWT;
```

그림 5. 토큰 해독 코드.

이렇게 JWT를 생성하고 클라이언트에 발급하게 되면 클라이언트 그림 6과 같은 고유의 토큰을 받을 수 있다.

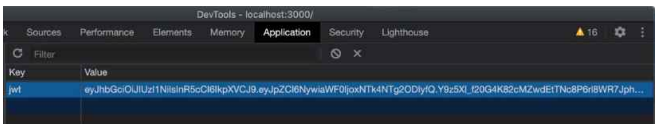


그림 6. Chrome 개발 툴을 통해 찾은 JWT토큰.

최종 구현된 차량 매칭 서비스의 시뮬레이션 화면은 그림 7과 같다.

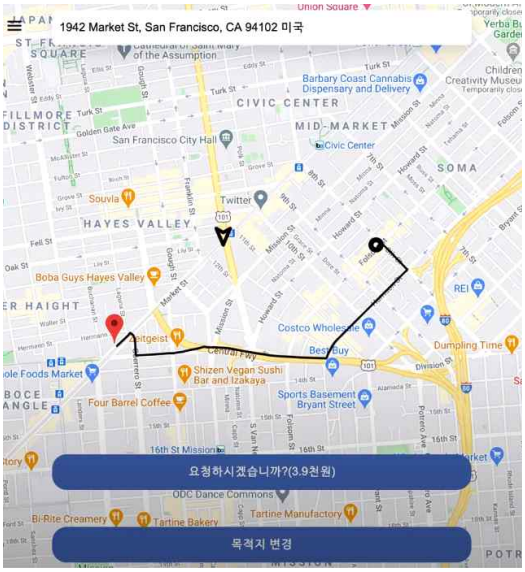


그림 7. 차량 매칭 서비스.

## 5. 결론

4차 산업으로 인한 IT 기술의 급격한 발전은 사용자에게 많은 서비스를 제공해준다. 특히, 차량 매칭 서비스에 많은 기업의 관심이 집중 되어있다. 이러한 차량 매칭 서비스는 사용자들에게 실시간 매칭을 해주기 때문에 많은 사람들이 서버에 접속하여 인증을 요청하게 되고, 이것을 효과적으로 처리하는 문제가 발생한다.

본 논문에서는 토근 인증 기반 방식은 JWT를 활용하여 기존의 세션 기반 인증 방식보다 효율성이 높은 인증 방식을 구현하였다. 차량 매칭 웹 서비스를 만들고 클라이언트에서 JWT이 발급되었는지 chrome 웹 개발자 도구를 통해 토큰이 무사히 오는 것을 확인하였다. 추후 연구로 안전성과 효율성에 대한 비교 시뮬레이션을 수행할 계획이다.

## 감사의 글

본 '연구논문지원 프로그램'은 교육부와 한국교육재단의 계명대학교 대학혁신지원사업비를 지원받아 수행된 것입니다.

## 6. 참고문헌

[1] https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=JAKO201817968656971&SITE=CLICK
[2] https://biz.chosun.com/site/data/html_dir/2018/10/02/2018100200241.html
[3] https://www.forbes.com/sites/ellenhuet/2014/06/06/at-18-2-billion-uber-is-worth-more-than-hertz-united-airlines/?sh=2d48b1c61155
[4] Jones, Michael B.; Bradley, Bradley; Sakimura, Sakimura (May 2015). JSON Web Token (JWT)
[5] https://velopert.com/2350
[6] https://jwt.io/introduction/