JWT와 OAuth 2.0은 서로 다른 개념이지만 주로 OAuth 2.0에서 JWT를 **Access Token**으로 사용하는 방식으로 함께 사용된다. OAuth 2.0은 인증과 권한 부여를 처리하는 **프로토콜**이고 JWT는 권한을 담은 **토큰 형식**으로 사용되기 때문에 두 가지를 결합하면 효율적이고 확장 가능한 인증 시스템을 만들 수 있다.

OAuth 2.0에서 JWT의 역할

OAuth 2.0에서 클라이언트 애플리케이션이 API에 접근할 수 있도록 **Access Token**을 발급받는다. 이 Access Token을 JWT 형식으로 사용할 수 있다. Access Token에는 권한(scope)과 사용자 정보 등이 포함되어 있으며 이를 통해 자원 서버(API 서버)는 클라이언트가 올바른 권한을 가지고 있는지 확인할 수 있다.

사용자 인증 요청

클라이언트 애플리케이션이 사용자를 대신하여 자원 서버(API 서버)에 접근하려고 할 때 먼저 OAuth 2.0의 **Authorization Server**에 인증 요청을 한다. 이 과정에서 사용자는 로그인 절차를 거친다.

Access Token 발급 (JWT 형식)

사용자가 성공적으로 로그인하면 **Authorization Server**는 클라이언트에게 **Access Token**을 발급한다. 이 Access Token이 JWT 형식으로 발급될 수 있다. JWT는 사용자 정보, 권한 범위(scope), 만료 시간(exp) 등의 정보를 담고 있으며 해당 정보는 서버에서 서명되어 보호된다.

클라이언트가 Access Token을 사용하여 API 호출

클라이언트는 JWT 형식의 Access Token을 가지고 자원 서버(API 서버)에 요청을 보낸다. 이때 요청 헤더에 `Authorization: Bearer <JWT 토큰>` 형식으로 Access Token을 포함시킨다.

자원 서버가 JWT를 검증

자원 서버(API 서버)는 전달된 JWT Access Token의 서명(Signature)을 검증하고 만료 시간(exp), 권한(scope) 등의 정보를 확인한다. JWT가 올바르게 서명되었고 만료되지 않았다면 자원 서버는 클라이언트가 요청한 자원에 대한 접근을 허용한다.

JWT와 OAuth 2.0 사용 시 고려 사항

1. 토큰 유효성 검사 : 자원 서버(API 서버)는 JWT 토큰의 서명 검증을 통해 토큰의 무결성을 확인하고 발급된 토큰이 변조되지 않았는지 확인한다.
2. JWT 만료 시간 : JWT에는 만료 시간이 포함되어 있어 만료된 토큰은 더 이상 유효하지 않는다. 만료 시간이 지나면 사용자는 다시 인증을 받아 새로운 토큰을 발급받아야 한다.
3. 리프레시 토큰 사용 : OAuth 2.0에서는 Access Token이 만료되면 **리프레시 토큰**을 사용하여 새로운 Access Token(JWT)을 발급받을 수 있다. 리프레시 토큰은 주로 OAuth 2.0의 **Authorization Code Flow**에서 사용되며 클라이언트는 리프레시 토큰을 통해 사용자가 다시 로그인하지 않아도 토큰을 갱신할 수 있다.

4. Access Token과 Refresh Token의 구분 : Access Token(JWT)은 만료 시간이 짧고 자원 서버에 전달되지만 리프레시 토큰은 좀 더 긴 만료 시간을 갖고 주로 Authorization Server에서 사용된다.

JWT를 사용하는 OAuth 2.0의 장점

1. 무상태(stateless) : JWT는 서버에 세션 정보를 저장할 필요가 없고 토큰만으로도 인증 및 권한 부여가 가능하므로 서버 확장이 용이하다.
2. 보안성 : JWT는 서명을 통해 변조를 방지할 수 있다. RSA나 HMAC 등의 알고리즘을 사용하여 서명을 검증하면 클라이언트에서 토큰이 변경되지 않았음을 확인할 수 있다.
3. 권한 범위 관리 : JWT의 `scope` 필드를 통해 어떤 자원에 접근할 수 있는지 명확하게 정의할 수 있다.

OAuth 2.0과 JWT를 함께 사용할 때 흐름 요약

1. **클라이언트가 사용자 인증 요청** → 사용자가 인증 서버에서 인증.
2. **Authorization Server가 JWT 형식의 Access Token 발급**.
3. **클라이언트가 자원 서버(API 서버)에 Access Token(JWT)을 전송**.
4. **자원 서버는 JWT의 서명 및 권한 정보 검증 후 요청 처리**.

OAuth 2.0은 권한 부여를 위한 프로토콜이고 **JWT**는 그 권한을 증명하는 데 사용되는 **토큰 포맷**이다. OAuth 2.0에서 JWT는 주로 **Access Token**으로 사용되어 클라이언트가 자원 서버에 안전하게 요청을 보낼 수 있게 한다.
OAuth 2.0과 JWT의 결합은 확장성 있고 효율적이며 무상태적인 인증 및 권한 부여 시스템을 제공한다.