OAuth(Open Authorization)는 제3자 애플리케이션이 사용자 자격 증명을 직접 알지 못하면서 사용자 리소스에 접근할 수 있도록 허용하는 인증 및 권한 부여하는 방식이다. 주로 소셜 미디어나 웹 서비스에서 많이 사용되며 보안이 중요한 API와 애플리케이션 간의 데이터 접근을 제어하는 데 활용된다.

OAuth는 주로 두 가지 버전이 존재하는데 OAuth 1.0과 OAuth 2.0이 있다. 현재는 더 간단하고 보안이 강화된 OAuth 2.0이 표준으로 사용된다.

OAuth 2.0 구성 요소

1. 클라이언트(Client)
    1. 정의 : 사용자로부터 권한을 받아 리소스 서버에 접근하려는 애플리케이션이다.
    2. 예시 : 모바일 앱, 웹 애플리케이션, 데스크톱 애플리케이션 등
2. 리소스 소유자(Resource Owner)
    1. 정의 : 자신의 데이터에 대한 권한을 가지고 있는 사용자이다. 사용자가 자신의 데이터를 제3자에게 제공할 수 있는 권한을 부여한다.
    2. 예시 : 구글 계정 사용자, 페이스북 계정 사용자 등
3. 인증 서버(Authorization Server)
   3. 정의 : 사용자의 인증을 처리하고 클라이언트가 적절한 권한을 부여받았는지 검증하는 서버이다. 인증 서버는 액세스 토큰을 발급한다.
   4. 예시 : 구글 OAuth 서버, 페이스북 OAuth 서버 등
4. 리소스 서버(Resource Server)
    1. 정의 : 사용자의 데이터가 저장된 서버로, 액세스 토큰을 통해 요청된 데이터를 클라이언트에 제공합니다. 이 서버는 클라이언트가 제출한 액세스 토큰의 유효성을 확인한 후 데이터 접근을 허용한다.
    2. 예시 : 구글 API 서버, 페이스북 API 서버 등
5. 액세스 토큰(Access Token)
   3. 정의 : 클라이언트가 리소스 서버에 대한 접근 권한을 증명하는 증표이다.
   4. 특징 :
   1. 비밀 유지가 중요하며 일반적으로 만료 시간이 설정된다.
   2. 리소스 서버는 액세스 토큰을 검증하여 클라이언트의 권한을 확인한다.

OAuth 2.0의 개념과 동작 방식

OAuth에는 다양한 흐름이 있지만 주로 권한 부여 코드 그랜트(Authorization Code Grant) 방식을

사용된다.

1. 사용자(리소스 오너)가 클라이언트(앱)에 로그인을 요청한다.
2. 클라이언트는 인증 서버로 사용자를 리디렉션한다.
3. 사용자는 인증 서버에서 자신의 아이디와 비밀번호를 입력하고 클라이언트에게 특정 정보에 대한 접근을 허용한다.
4. 인증 서버는 클라이언트에게 권한 부여 코드를 발급한다.
5. 클라이언트는 권한 부여 코드를 이용하여 인증 서버에 액세스 토큰을 요청한다.
6. 인증 서버는 액세스 토큰을 발급하고 클라이언트는 이 토큰을 이용하여 리소스 서버에 접근한다.
7. 리소스 서버는 액세스 토큰의 유효성을 검증하고 요청된 리소스(예: 사용자 정보, 파일 등)를 클라이언트에게 제공한다.

예시 :

1. **A 앱**에서 **구글 드라이브**의 파일을 열려고 한다고 가정해 보자.
2. 사용자가 A 앱에서 구글 로그인을 클릭하면 A 앱은 사용자를 구글의 인증 서버로 이동시킨다.
3. 사용자가 구글 계정으로 로그인하고 A 앱에 자신의 구글 드라이브에 접근할 수 있는 권한을 부여한다.
4. 구글은 A 앱에게 **권한 부여 코드**를 발급한다.
5. A 앱은 이 코드를 이용하여 구글에 **액세스 토큰**을 요청한다.
6. 구글은 A 앱에게 액세스 토큰을 발급하고 A 앱은 이 토큰을 이용하여 구글 드라이브에 접근하여 파일을 열 수 있다.

추가 설명

1. 액세스 토큰 : 일정 시간이 지나면 만료되며 비밀 유지가 매우 중요하다.
2. 리프레시 토큰 : 액세스 토큰이 만료되었을 때 새로운 액세스 토큰을 발급받기 위해 사용하는 토큰이다.
3. 스코프 : 클라이언트가 요청할 수 있는 권한의 범위를 정의한다. 예시 사용자의 프로필만 읽을 수 있는 권한, 이메일을 읽고 보낼 수 있는 권한 등이 있다.

OAuth 1.0 구성 요소

1. 소비자 (Consumer)

    1. 정의 : 사용자의 자원에 접근하려는 애플리케이션이다.
    2. 예시 : 트위터 클라이언트 애플리케이션, 타사 웹 애플리케이션 등.
2. 서비스 제공자 (Service Provider)
    1. 정의 : 사용자의 자원을 호스팅하고 이를 보호하는 서비스다. Consumer에게 인증 및 권한을 부여한다.
    2. 예시 : 트위터, 구글, 페이스북 등의 API 제공자
3. Request Token
    1. 정의 : Consumer가 사용자의 권한을 얻기 전에 발급받는 임시 토큰이다. 이 토큰은 사용자의 승인을 받기 위한 과정에서 사용된다.
    2. 특징 :
        1. 임시 토큰으로, 사용자가 승인하면 Access Token으로 변환된다.
        2. 서비스 제공자가 발급한다.
4. Access Token
    1. 정의 : 사용자가 승인 절차를 거친 후 Consumer에게 발급되는 최종 토큰으로 이를 통해 서비스 제공자의 API에 접근할 수 있다.
    2. 특징 :
        1. Request Token을 승인 후 교환하여 발급된다.
        2. 각 요청마다 서명과 함께 사용되며, 만료되거나 재사용이 제한된다.
5. 서명 (Signature)
    1. 정의 : 각 API 요청마다 암호화된 서명을 포함하여 Consumer가 요청의 무결성을 보장하는 방식이다.
    2. 특징 :
        1. 요청의 변조를 방지하며, Consumer Key, Secret, Request Token, 요청 메서드, URL 등 다양한 요소를 조합해 생성된다.
        2. 서명을 통해 서비스 제공자는 요청이 올바른지 확인할 수 있다.

OAuth 1.0의 개념과 동작 방식

1. Request Token 요청
    1. Consumer는 Service Provider에 Request Token을 요청한다. 이 과정에서 Consumer Key와 Consumer Secret이 함께 전송된다.
2. 사용자 승인
    1. Request Token을 발급받은 후, 사용자는 Consumer가 자신의 자원에 접근할 수 있도록 Service Provider를 통해 승인을 요청받는다.
3. Access Token 발급
    1. 사용자가 승인을 완료하면 Service Provider는 Request Token을 Access Token으로 변환하여 Consumer에게 전달한다.
4. 서명을 통한 요청

    1. Consumer는 Access Token을 사용하여 Service Provider의 자원에 접근할 때마다 서명된 요청을 전송한다. 서명은 요청의 변조 여부를 확인하기 위해 사용된다.
5. 자원 접근
    1. Access Token을 소유한 Consumer는 서비스 제공자의 API를 통해 사용자의 자원에 안전하게 접근할 수 있다.

예시 :

1. 트위터 애플리케이션이 사용자 A의 트윗 목록에 접근하려고 한다고 가정하자.
2. 트위터 애플리케이션은 트위터 서버(Service Provider)에 Request Token을 요청한다.
3. 사용자 A는 트위터 로그인 화면을 통해 애플리케이션에 권한을 부여한다.
4. 트위터 서버는 애플리케이션에 Access Token을 발급한다.
5. 애플리케이션은 Access Token을 사용하여 사용자 A의 트윗 목록을 불러온다.

추가 설명

1. OAuth 1.0은 요청의 무결성을 보장하기 위해 서명을 필수적으로 요구하며 이를 통해 높은 보안성을 제공한다.
2. 하지만 서명 방식과 Request/Access Token 처리 과정이 복잡해서 OAuth 2.0에서는 이를 간소화한 방식이 도입되었다.

OAuth 1.0과 OAuth 2.0 차이

OAuth 1.0과 OAuth 2.0은 웹 애플리케이션과 API 간의 인증을 위한 프로토콜로 사용자 자격 증명 없이 안전하게 권한을 부여할 수 있도록 돕는 방식이다. 하지만 두 버전 사이에는 중요한 차이점들이 있다.

1. **복잡성 및 설계 철학**
    1. OAuth 1.0 : 복잡한 인증 프로세스를 사용한다. OAuth 1.0은 주로 암호화된 서명을 사용해 요청의 무결성을 보장하며 각 요청에 서명을 생성하고 이를 검증하는 과정이 포함되어 있다. 따라서 구현이 복잡하고 관리할 요소가 많다.
    2. OAuth 2.0 : OAuth 2.0은 사용하기 쉽게 설계되었으며 보안은 HTTPS(SSL/TLS)에 의존한다. 암호화된 서명보다는 HTTPS로 트래픽을 보호하는 방식을 사용하기 때문에 OAuth 1.0보다 더 단순하고 개발자들이 쉽게 사용할 수 있다.
2. **서명(Signature) 방식**
    1. OAuth 1.0 : OAuth 1.0에서는 각 요청이 반드시 서명되어야 한다. 이 서명은 클라이언트 비밀키를 사용하여 암호화 알고리즘을 통해 생성되며 요청 내용과 매개변수 등이

포함된다. 서명 생성 과정이 복잡하고 이를 검증하는 데 추가적인 계산이 필요하다.

2. OAuth 2.0 : OAuth 2.0은 서명을 사용하지 않는다. 대신 HTTPS를 사용하여 데이터 전송 중 보안을 유지하며 클라이언트 인증에는 액세스 토큰을 사용한다. 클라이언트 비밀키는 토큰을 발급받을 때만 사용된다. 그 외의 요청은 단순히 액세스 토큰을 포함하여 서버와 통신한다.

3. **액세스 토큰 발급 방식**
    1. OAuth 1.0 : OAuth 1.0에서는 **Request Token**과 **Access Token**이라는 두 단계의 토큰을 사용한다. 먼저 리소스 소유자가 인증 서버에서 요청 토큰을 발급받고 이 토큰을 사용해 액세스 토큰을 교환하는 방식이다. 이러한 이중 토큰 방식은 좀 더 복잡하다.
    2. OAuth 2.0 : OAuth 2.0에서는 권한 부여 흐름에 따라 **Access Token**을 직접 발급받는다. 토큰 발급 과정이 단순화되었으며 Refresh Token을 사용하여 만료된 Access Token을 새로 발급받을 수 있다.
4. **토큰 유형 및 갱신**
    1. OAuth 1.0 : OAuth 1.0에서는 토큰 갱신(refresh)이 없다. 액세스 토큰이 만료되면 새로운 토큰을 발급받아야 한다.
    2. OAuth 2.0 : OAuth 2.0에서는 **Refresh Token**이 도입되었다. 이는 액세스 토큰이 만료되었을 때 새로 인증을 하지 않고도 액세스 토큰을 다시 발급받을 수 있도록 한다. 따라서 사용자 경험이 더 매끄럽다.
5. **권한 부여 방식**
    1. OAuth 1.0 : OAuth 1.0은 기본적으로 인증 코드 흐름만을 사용한다. 사용자 자격 증명을 활용해 인증 서버로부터 권한을 부여받고 그에 따라 토큰을 발급받는 단일 방식만을 지원한다.
    2. OAuth 2.0 : OAuth 2.0에서는 다양한 권한 부여 방식을 지원한다. 각 상황에 맞는 권한 부여 방식을 선택할 수 있다.
        1. Authorization Code Grant : 일반적인 웹 애플리케이션 인증에 사용.
        2. Implicit Grant : 브라우저 기반 애플리케이션에 사용.
        3. Resource Owner Password Credentials Grant : 사용자가 애플리케이션에 직접 아이디와 비밀번호를 제공하는 방식.
        4. Client Credentials Grant : 서버 간 통신에 사용.
        5. Device Code Grant : 제한된 입력을 허용하는 장치(예: TV, IoT 기기)에서 사용.
6. **보안**

1. OAuth 1.0 : OAuth 1.0은 모든 요청에 서명이 포함되므로 트래픽이 평문으로 전달되더라도 서명 검증을 통해 요청이 위조되지 않았는지 확인할 수 있다. 그러나 암호화 서명을 사용하는 방식이 복잡하여 개발자가 구현하는 데 어려움이 있다.

2. OAuth 2.0 : OAuth 2.0은 모든 트래픽을 HTTPS로 보호하는 것을 전제로 한다. HTTPS를 통해 기밀성(허가되지 않은 사람이나 시스템이 정보에 접근하는 것을 방지하는 것)과 무결성(정보가 생성되고 전송되는 과정에서 변조나 손상 없이 원본 그대로 유지되는 것)을 보장하며 토큰 발급 및 사용 과정에서 추가적인 보안 메커니즘을 도입할 수 있다. 그러나 OAuth 1.0과 비교했을 때 보안이 HTTPS에만 의존하므로 HTTPS 설정이 제대로 되어 있지 않으면

보안 문제가 발생할 수 있다.

결론

OAuth 1.0 : 보안은 강력하지만 복잡하고 구현이 어렵다.

OAuth 2.0 : 더 단순하고 유연하며, HTTPS 보안에 의존하여 더 광범위한 사용을 지원한다.