1. **RESTFUL API 인증방식**

**가.API Key 방식**

가장 기초적인 방법은 API Key를 이용하는 방법이다. API Key란 특정 사용자만 알 수 있는 일종의 문자열이다. API를 사용하고자 할 때, 개발자는 API 제공사의 포탈 페이등 등에서 API Key를 발급 받고, API를 호출할 때 API Key를 메시지 안에 넣어 호출한다. 서버는 메시지 안에서 API Key를 읽어 이 API가 누가 호출한 API인지를 인증하는 흐름이다.

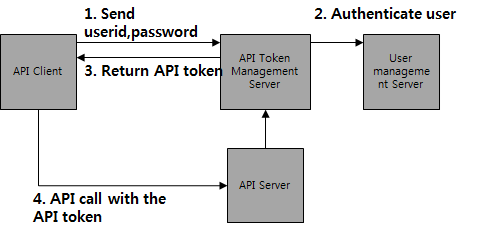
모든 클라이언트들이 같은 API Key를 공유하기 때문에 한번 API Key가 노출이 되면 전체 API가 뚫려 버리는 문제가 있기 때문에 높은 보안 인증을 요구 하는 경우에는 권장하지 않는다.

**나. API Token 방식**

다른 방식으로는 API Token을 발급하는 방식이 있는데, 사용자 ID,PASSWD등으로 사용자를 인증한 후에, 그 사용자가 API 호출에 사용할 기간이 유효한 API Token을 발급해서 API Token으로 사용자를 인증하는 방식이다.

매번 API 호출시 사용자 ID,PASSWD를 보내지 않고, API Token을 사용하는 이유는 사용자 PASSWD는 주기적으로 바뀔 수 있기 때문이고, 매번 네트워크를 통해서 사용자 ID와 PASSWD를 보내는 것은 보안적으로 사용자 계정 정보를 탈취 당할 가능성이 높기 때문에 API Token을 별도로 발급해서 사용하는 것이다.

API Token을 탈취 당하면 API를 호출할 수 는 있지만, 반대로 사용자 ID와 PASSWD는 탈취 당하지 않는다. 사용자PASSWD를 탈취당하면 일반적으로 사용자들은 다른 서비스에도 같은 PASSWD를 사용하는 경우가 많기 때문에 연쇄적으로 다른 서비스에 대해서도 공격을 당할 수 있는 가능성이 높아지기 때문이다. 예를 들어서 매번 호출시마다 사용자 ID,PASSWD를 보내서 페이스북의 계정과 비밀번호를 탈취 당한 경우, 해커가 이 계정과 비밀 번호를 이용해서 GMail이나 트위터와 같은 다른 서비스까지 해킹 할 수 있기 때문에, 이러한 가능성을 최소화하기 위함이다.



흐름을 설명하면 위의 그림과 같다.

1. API Client가 사용자 ID,PASSWD를 보내서 API호출을 위한 API Token을 요청한다.

2. API 인증 서버는 사용자 ID,PASSWD를 가지고, 사용자 정보를 바탕으로 사용자를 인증한다.

3. 인증된 사용자에 대해서 API Token을 발급한다. (유효 기간을 가지고 있다.)

4. API Client는 이 API Token으로 API를 호출한다. API Server는 API Token이 유효한지를 API Token 관리 서버에 문의하고, API Token이 유효하면 API 호출을 받아 들인다.

1. **RESTFUL API 인증시 사용자 인증방식**

가. **HTTP Basic Auth**

※ 상세 : http://en.wikipedia.org/wiki/Basic\_access\_authentication

가장 기본적이고 단순한 형태의 인증 방식으로 사용자 ID와 PASSWD를 HTTP Header에 Base64 인코딩 형태로 넣어서 인증을 요청한다.

예를 들어 사용자 ID가 terry이고 PASSWD가 hello world일 때, 다음과 같이 HTTP 헤더에 “terry:hello world”라는 문자열을 Base64 인코딩을해서 “Authorization”이라는 이름의 헤더로 서버에 전송하여 인증을 요청한다.

Authorization: Basic VGVycnk6aGVsbG8gd29ybGQ=

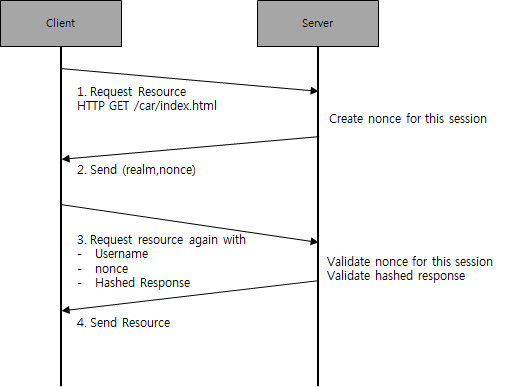
중간에 패킷을 가로채서 이 헤더를 Base64로 디코딩하면 사용자 ID와 PASSWD가 그대로 노출되기 때문에 반드시 HTTPS 프로토콜을 사용해야 한다.

나. **Digest access Authentication**

※ 상세 : http://en.wikipedia.org/wiki/Digest\_access\_authentication

HTTP Basic Auth가 Base64 형태로 PASSWD를 실어서 보내는 단점을 보강하여 나온 인증 프로토콜이 Digest access Authentication 이라는 방법으로, 기본 원리는 클라이언트가 인증을 요청할 때, 클라이언트가 서버로부터 nonce 라는 일종의 난수값을 받은 후에, (서버와 클라이언트는 이 난수 값을 서로 알고 있음), 사용자 ID와 PASSWD를 이 난수값을 이용해서 HASH화하여 서버로 전송하는 방식이다.

이 경우에는 직접 ID와 PASSWD가 평문 형태로 날아가지 않기 때문에, 해커가 중간에 PASSWD를 탈취할 수 없고, 설령 HASH 알고리즘을 알고 있다고 하더라도, HASH된 값에서 반대로 PASSWD를 추출하기가 어렵기 때문에, Basic Auth 방식에 비해서 향상된 보안을 제공한다. 전체적인 흐름을 보자

  
다. **클라이언트 인증 추가**

추가적인 보안 강화를 위해서 사용자 인증 뿐만 아니라, 클라이언트 인증 방식을 추가할 수 있다. 페이스북의 경우 API Token을 발급 받기 위해서, 사용자 ID,PASSWD 뿐만 아니라 client Id와 Client Secret이라는 것을 같이 입력 받도록 하는데,

Client Id는 특정 앱에 대한 등록 Id이고, Client Secret은 특정 앱에 대한 비밀 번호로 페이스북 개바자 포털에서 앱을 등록하면 앱 별로 발급 되는 일종의 비밀 번호이다.

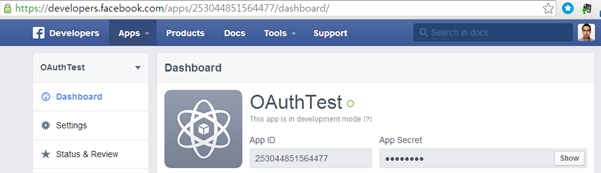


그림. 페이스북 개발자 포탈에서 등록된 client Id(appId)와 client secret(App Secret)을 확인하는 화면

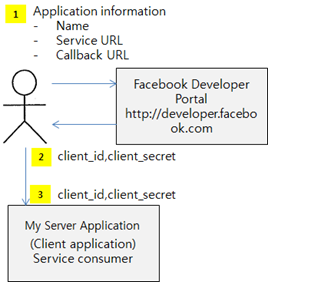
API Token을 발급 받을 때, Client Id와 Client Secret 을 이용하여, 클라이언트 앱을 인증하고 사용자 ID와 PASSWD를 추가로 받아서 사용자를 인증하여 API access Token을 발급한다

라. **제3자 인증 방식(OAuth방식)**

3자 인증 방식은 페이스북이나 트위터와 같은 API 서비스 제공자들이 파트너 애플리케이션에 많이 적용하는 방법으로 만약 내가 My Server Application라는 서비를 Facebook 계정을 이용하여 인증을 하는 경우이다.

이때 중요한 점은 서비스 My Server Application에 대해서 해당 사용자가 페이스북 사용자임을 인증을 해주지만, 서비스 My Server Application는 사용자의 비밀번호를 받지 않고, 페이스북이 사용자를 인증하고 서비스 My Server Application에게 알려주는 방식이다. 즉 파트너 서비스에 페이스북 사용자의 비밀번호가 노출되지 않는 방식이다.

전체적인 흐름을 보면 다음과 같다.

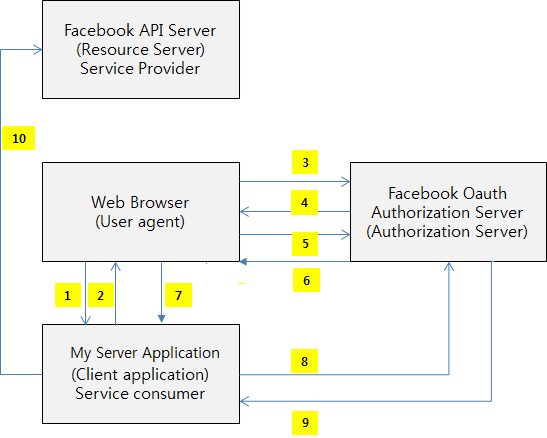


1. 먼저 페이스북의 Developer Portal에 접속을 하여, 페이스북 인증을 사용하고자 하는 애플리케이션 정보를 등록한다. (서비스 명, 서비스 URL,그리고 인증이 성공했을 때 인증 성공 정보를 받을 CallBack URL)

2. 페이스북 Developer Portal은 등록된 정보를 기준으로 해당 애플리케이션에 대한 client\_id와 client\_secret을 발급한다. 이 값은 앞에서 설명한 클라이언트 인증에 사용된다.

3. 다음으로, 개발하고자 하는 애플리케이션에, 이 client\_id와 client\_secret등을 넣고, 페이스북 인증 페이지 정보등을 넣어서 애플리케이션을 개발한다.

애플리케이션이 개발되서 실행이 되면, 아래와 같은 흐름에 따라서 사용자 인증을 수행하게 된다.



1. 웹브라우져에서 사용자가 My Server Application 서비스를 접근하려고 요청한다.

2. My Server Application은 사용자가 인증이되어 있지 않기 때문에, 페이스북 로그인 페이지 URL을 HTTP Redirection으로 URL을 브라우져에게 보낸다. 이때 이 URL에 페이스북에 이 로그인 요청이 My Server Application에 대한 사용자 인증 요청임을 알려주기 위해서, client\_id등의 추가 정보와 함께, 페이스북의 정보 접근 권한 (사용자 정보, 그룹 정보등)을 scope라는 필드를 통해서 요청한다.

3. 브라우져는 페이스북 로그인 페이지로 이동하여, 2단계에서 받은 추가적인 정보와 함께 로그인을 요청한다.

4. 페이스북은 사용자에게 로그인 창을 보낸다.

5. 사용자는 로그인창에 ID/PASSWD를 입력한다.

6. 페이스북은 사용자를 인증하고, 인증 관련 정보과 함께 브라우져로 전달하면서, My Server Application의 로그인 완료 페이지로 Redirection을 요청한다.

7. My Server Application을 6에서 온 인증 관련 정보를 받는다.

8. My Server Application은 이 정보를 가지고, 페이스북에, 이 사용자가 제대로 인증을 받은 사용자인지를 문의한다.

9. 페이스북은 해당 정보를 보고 사용자가 제대로 인증된 사용자임을 확인해주고, API Access Token을 발급한다.

10.  My Server Application은 9에서 받은 API Access Token으로 페이스북 API 서비스에 접근한다.

앞에서 설명했듯이, 이러한 방식은 자사가 아닌 파트너 서비스에게 자사 서비스 사용자의 인증을 거쳐서 API의 접근 권한을 전달하는 방식이다.

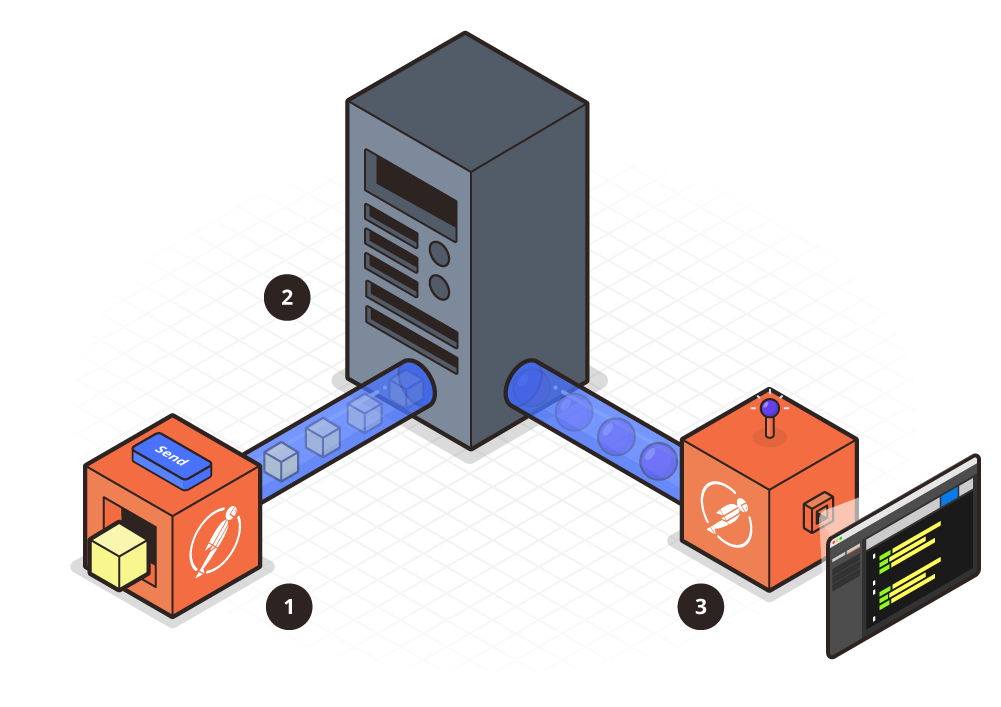
이러한 인증 방식의 대표적인 구현체는 OAuth 2.0으로, 이와 같은 제3자 인증뿐만 아니라, 직접 자사의 애플리케이션을 인증하기 위해서, 클라이언트로부터 직접 ID/PASSWD를 입력 받는 등.

클라이언트 타입(웹,서버,모바일 애플리케이션)에 대한 다양한 시나리오를 제공한다.

※ OAuth 2.0에 대한 자세한 설명은 PACKT 출판사의 OAuth 2.0 Identity and Access Management Patterns (by Martin Spasovski) 책을 참고하기를 추천한다.

이러한 3자 인증 방식은 일반적인 서비스에서는 필요하지 않지만, 자사의 API를 파트너등 외부 시스템에 제공하면서 사용자의 ID/PASSWD를 보호하는데는 필요한 서비스이기 때문에, API 를 외부에 적용하는 경우에는 고려를 해야 한다.

1. **POSTMAN사용 (api client)**

****

1. Enter your request details (URL: postman-echo.com/get) in Postman, and hit the **Send** button.
2. The request is received by the API server (postman-echo.com), and it returns a response.
3. The response is received by Postman, and the response is visualized in the interface