



An Introduction to OAuth 2

Aaron Parecki • @aaronpk
O'Reilly Webcast • July 2012

A Brief History

**Login to Twitter below and post this tweet to
get Sky Downloader PRO for FREE!**

www.bnsofts.com

Don't have a Twitter account? [Register Here](#)

Twitter username:

bnsofts

Password:

••••••••



What's happening?

16

Just got the NEW Sky Downloader PRO for FREE (\$49 value) in exchange for this Tweet!
<http://www.skydownloader.com/tweet4pro/>

 **No Thanks**

Post Tweet

Before OAuth

aka the Dark Ages

If a third party wanted access to an account,
you'd give them your password.

Several Problems and Limitations



- Apps store the user's password
- Apps get complete access to a user's account
- Users can't revoke access to an app except by changing their password
- Compromised apps expose the user's password

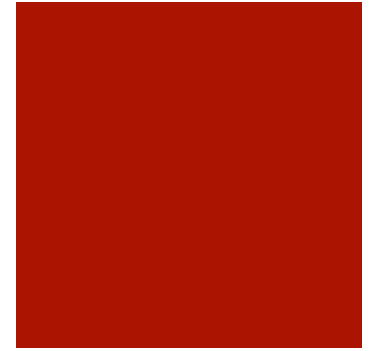
Before OAuth 1.0



- Services recognized the problems with password authentication
- Many services implemented things similar to OAuth 1.0
- Each implementation was slightly different, certainly not compatible with each other

Before OAuth 1.0

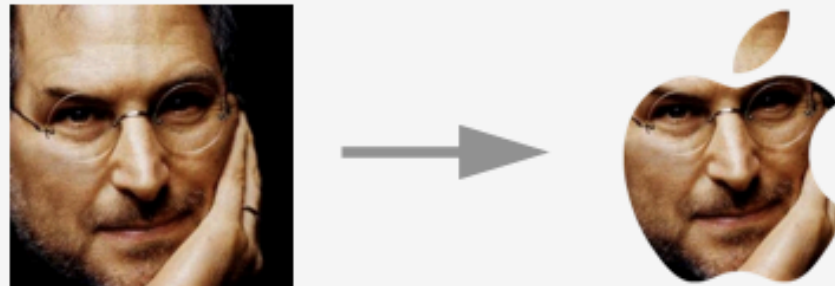
- Flickr: “FlickrAuth” frobs and tokens
- Google: “AuthSub”
- Facebook: requests signed with MD5 hashes
- Yahoo: BBAuth (“Browser-Based Auth”)



“We want something like Flickr Auth / Google AuthSub / Yahoo! BBAuth, but published as an open standard, with common server and client libraries.”

Blaine Cook, April 5th, 2007

OAuth 1.0



Click below to change your profile picture!



This will automatically update your Twitter profile picture! We will not post any tweets without your explicit confirmation.



We will not post anything to your Facebook wall without your explicit confirmation.

Authorize Put an Apple On It to use your account?

This application **will be able to:**

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Authorize app

No, thanks

This application **will not be able to:**

- Access your direct messages.
- See your Twitter password.



Put an Apple On It

putanappleonit.com

Update your Twitter profile image to show your respect to Steve Jobs

← [Cancel, and return to app](#)

You can revoke access to any application at any time from the [Applications tab](#) of your Settings page.

By authorizing an application you continue to operate under [Twitter's Terms of Service](#). In particular, some usage information will be shared back with Twitter. For more, see our [Privacy Policy](#).

Your Twitter profile picture was updated!



RIP Steve Jobs. Show your love by changing your profile picture: <http://putanappleonit.com>



Tweet This!

OAuth 1.0 Signatures

The signature base string is often the most difficult part of OAuth for newcomers to construct. The signature base string is composed of the HTTP method being used, followed by an ampersand ("&") and then the URL-encoded base URL being accessed, complete with path (but not query parameters), followed by an ampersand ("&"). Then, you take all query parameters and POST body parameters (when the POST body is of the URL-encoded type, otherwise the POST body is ignored), including the OAuth parameters necessary for negotiation with the request at hand, and sort them in lexicographical order by first parameter name and then parameter value (for duplicated parameter names) while ensuring that both the key and value are URL encoded in isolation. Each key/value pair is separated by an equals ("=") sign to mark the key/value pair, and the entire string is URL-encoded in the form of "%3D". Each parameter is joined by the URL-escaped ampersand ("&").

```
oauth_nonce="QP70eNmVz8jvdPevU3oJD2AfF7R7o
dC2XJcn4XIZJqk", oauth_callback="http%3A%2F
%2Flocalhost%3A3005%2Fthe_dance
%2Fprocess_callback%3Fservice_provider_id
%3D11", oauth_signature_method="HMAC-SHA1",
oauth_timestamp="1272323042",
oauth_consumer_key="GDdmlQH6jhtmlUypg82g",
oauth_signature="8wUi7m5HFQy76nowoCThusfgB
%2BQ%3D", oauth_version="1.0"
```





OAuth 2:
signatures replaced by https

~~HMAC~~



Some Current Implementers



An application would like to connect to your account

The application **Science Notes** by Science Hack Day would like to connect to your Geoloqi account.

Science Notes wants to:

- see my exact last location
- leave me Geonotes
- subscribe me to layers

Allow Science Notes access?

Deny

Allow



The Windows Blog

Home

Blogs +

Videos

Windows.com

Search Site

Windows Live for Developers

Announcing Support

MAY 04 2011



by Dare Obasanjo

Members of the Windows Live Team as well as the Windows Live for Developers Workshop (IIW) this week in Mountain View, CA, thought leaders in the internet identity space. In the past: Open ID v2, OAuth, Activity

Request for Permission

fred.example@gmail.com | My Account | Sign out



OAuth2 Test



Login with GitHub

f Request for Permission

My Great Website is requesting permission to do the following:



Access my basic information

Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.



Send me email

My Great Website may email me directly at dmp@fb.com · Change



Access posts in my News Feed

Report App

My Great Website

Allow

Don't Allow



Log In



Sign in with Geoloqi



Sign in with Google



SIGN IN WITH FOURSQUARE



Connect with Gowalla

<u>1.</u>	Introduction	<u>4</u>
<u>1.1.</u>	Notational Conventions	<u>5</u>
<u>1.2.</u>	Terminology	<u>5</u>
<u>1.3.</u>	Overview	<u>7</u>
<u>1.4.</u>	Client Profiles	<u>10</u>
<u>1.4.1.</u>	Web Server	<u>10</u>
<u>1.4.2.</u>	User-Agent	<u>11</u>
<u>1.4.3.</u>	Native Application	<u>13</u>
<u>1.4.4.</u>	Autonomous	<u>14</u>
<u>2.</u>	Client Credentials	<u>14</u>
<u>2.1.</u>	Client Password Credentials	<u>15</u>
<u>3.</u>	Obtaining End-User Authorization	<u>16</u>
<u>3.1.</u>	Authorization Response	<u>18</u>
<u>3.2.</u>	Error Response	<u>20</u>
<u>3.2.1.</u>	Error Codes	<u>21</u>
<u>4.</u>	Obtaining an Access Token	<u>21</u>
<u>4.1.</u>	Access Grant Types	<u>23</u>
<u>4.1.1.</u>	Authorization Code	<u>23</u>
<u>4.1.2.</u>	Resource Owner Password Credentials	<u>24</u>
<u>4.1.3.</u>	Assertion	<u>24</u>
<u>4.1.4.</u>	Refresh Token	<u>25</u>
<u>4.2.</u>	Access Token Response	<u>26</u>
<u>4.3.</u>	Error Response	<u>27</u>
<u>4.3.1.</u>	Error Codes	<u>28</u>
<u>5.</u>	Accessing a Protected Resource	<u>29</u>
<u>5.1.</u>	Authenticated Requests	<u>29</u>
<u>5.1.1.</u>	The Authorization Request Header Field	<u>30</u>
<u>5.1.2.</u>	URI Query Parameter	<u>30</u>
<u>5.1.3.</u>	Form-Encoded Body Parameter	<u>31</u>
<u>5.2.</u>	The WWW-Authenticate Response Header Field	<u>32</u>
<u>5.2.1.</u>	Error Codes	<u>33</u>
<u>6.</u>	Extensibility	<u>34</u>

The OAuth 2 Spec

<http://oauth.net/2/>

OAuth 2?!

There are 29 versions!

Versions: [00](#) [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#)
[12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#)
[24](#) [25](#) [26](#) [27](#) [28](#) [29](#)

Currently Implemented Drafts

Provider	Draft	Reference
Foursquare	-10	http://aaron.pk/2YS
Google	-10	http://code.google.com/apis/accounts/docs/OAuth2.html
Facebook	-10 (ish)	https://developers.facebook.com/docs/authentication/oauth2_updates/
Windows Live	-10	http://aaron.pk/2YV
Salesforce	-10	http://aaron.pk/2YW
Github	-07	http://develop.github.com/p/oauth.html
Geoloqi	-10	http://developers.geoloqi.com/api

So how does it work?

Definitions

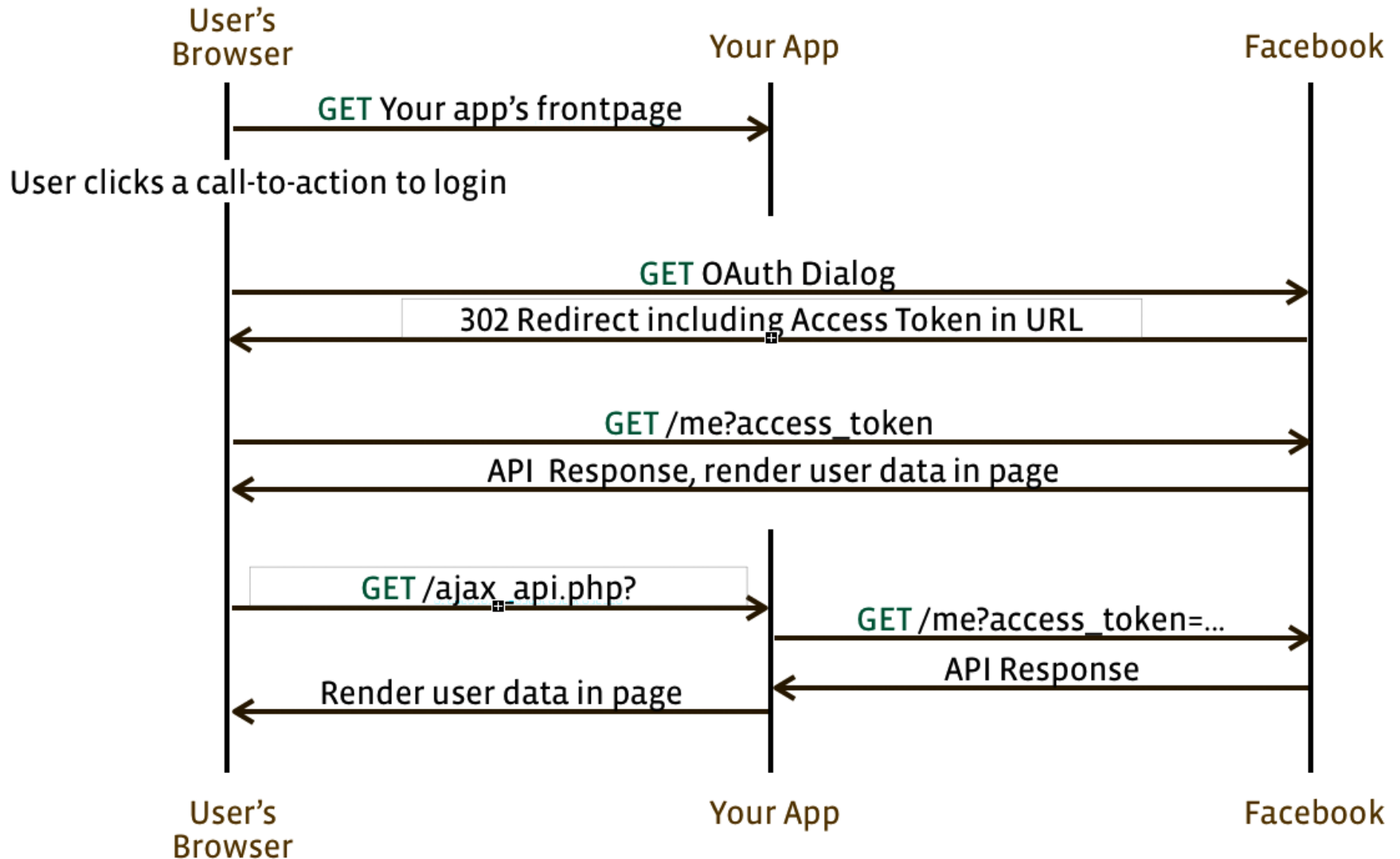


- **Resource Owner:** The User
- **Resource Server:** The API
- **Authorization Server:** Often the same as the API server
- **Client:** The Third-Party Application

Use Cases

- Web-server apps
- Browser-based apps
- Username/password access
- Application access
- Mobile apps

Facebook's OAuth Flow



Web Server Apps

Authorization Code Grant

Create a “Log In” link

Link to:

`https://facebook.com/dialog/oauth?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI&scope=email`



Create a “Log In” link

Link to:

[https://facebook.com/dialog/oauth?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI&scope=email](https://facebook.com/dialog/oauth?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email)



Create a “Log In” link

Link to:

[https://facebook.com/dialog/oauth?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI&scope=email](https://facebook.com/dialog/oauth?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email)



Create a “Log In” link

Link to:

[https://facebook.com/dialog/oauth?
response_type=code&client_id=YOUR_CLIENT_ID
&**redirect_uri=REDIRECT_URI**&scope=email](https://facebook.com/dialog/oauth?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email)



Create a “Log In” link

Link to:

[https://facebook.com/dialog/oauth?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI&scope=email](https://facebook.com/dialog/oauth?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email)



User visits the authorization page

```
https://facebook.com/dialog/oauth?  
response_type=code&client_id=28653682475872  
&redirect_uri=everydaycity.com&scope=email
```



Everyday City


Go to App

Cancel

3 people use this app

ABOUT THIS APP

Who can see posts this app makes for you on your Facebook timeline: [?]

 **Everyone** ▼

THIS APP WILL RECEIVE:

- Your basic info [?]
- Your email address (aaron@parecki.com)
- Your location

By proceeding, you will be taken to everydaycity.com · [Report App](#)



On success, user is redirected
back to your site with auth code

`https://example.com/auth?code=AUTH_CODE_HERE`

On error, user is redirected back
to your site with error code

`https://example.com/auth?error=access_denied`

Server exchanges auth code for an access token

Your server makes the following request

POST `https://graph.facebook.com/oauth/access_token`

Post Body:

```
grant_type=authorization_code  
&code=CODE_FROM_QUERY_STRING  
&redirect_uri=REDIRECT_URI  
&client_id=YOUR_CLIENT_ID  
&client_secret=YOUR_CLIENT_SECRET
```

Server exchanges auth code for an access token

Your server gets a response like the following

```
{  
  "access_token": "RsT5OjbzRn430zqMLgV3Ia",  
  "token_type": "bearer",  
  "expires_in": 3600,  
  "refresh_token": "e1qoXg7Ik2RRua48lXIV"  
}
```

or if there was an error

```
{  
  "error": "invalid_request"  
}
```

Browser-Based Apps

Implicit Grant

Create a “Log In” link

Link to:

[https://facebook.com/dialog/oauth?
response_type=token&client_id=CLIENT_ID
&redirect_uri=REDIRECT_URI&scope=email](https://facebook.com/dialog/oauth?response_type=token&client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email)



User visits the authorization page

```
https://facebook.com/dialog/oauth?  
response_type=token&client_id=2865368247587  
&redirect_uri=everydaycity.com&scope=email
```



Everyday City


Go to App

Cancel

3 people use this app

ABOUT THIS APP


Who can see posts this app makes for you on your Facebook timeline: [?]

 **Everyone** ▼

THIS APP WILL RECEIVE:

- Your basic info [?]
- Your email address (aaron@parecki.com)
- Your location

By proceeding, you will be taken to everydaycity.com · [Report App](#)



On success, user is redirected
back to your site with the access
token in the fragment

https://example.com/auth#token=ACCESS_TOKEN

On error, user is redirected back
to your site with error code

https://example.com/auth#error=access_denied

Browser-Based Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Browser makes API requests directly

Username/Password

Password Grant

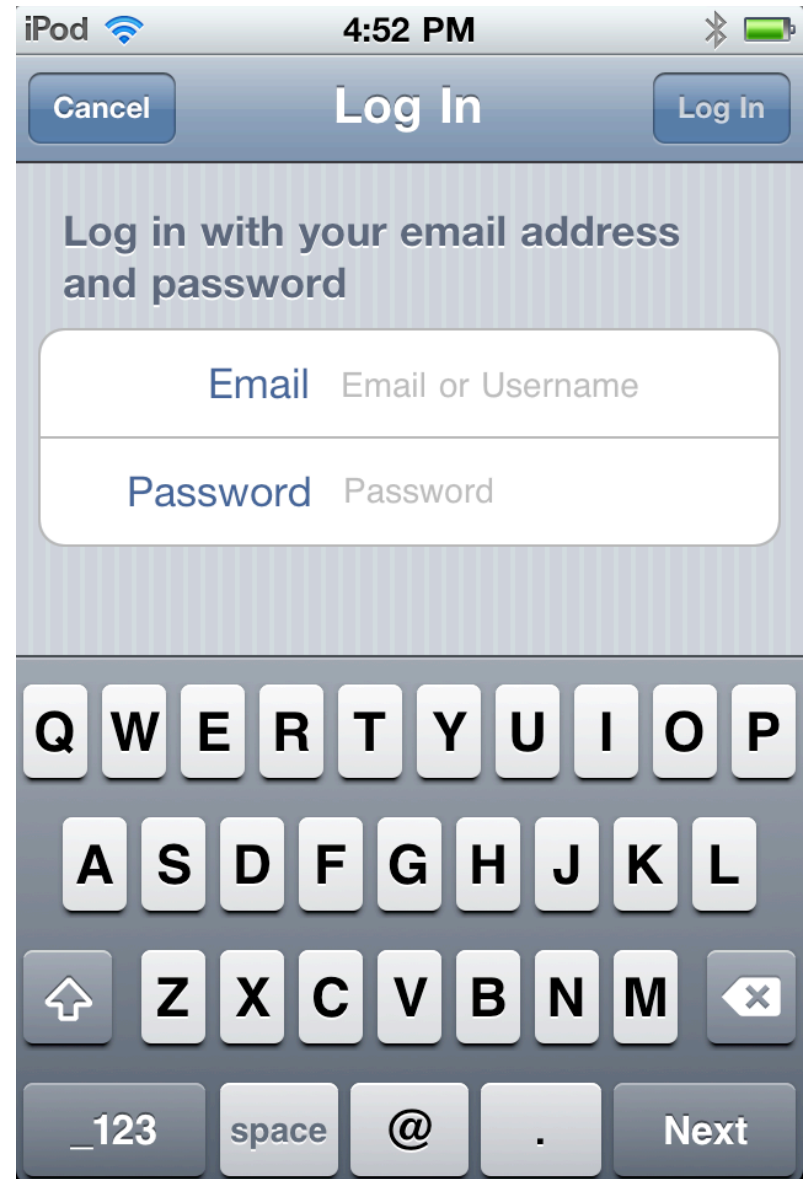
Password Grant

Password grant is only appropriate for trusted clients, most likely first-party apps only.

If you build your own website as a client of your API, then this is a great way to handle logging in.

Password Grant Type

Only appropriate for your service's website or your service's mobile apps.



Password Grant

POST `https://api.example.com/oauth/token`

Post Body:

grant_type=password

&username=USERNAME

&password=PASSWORD

&client_id=YOUR_CLIENT_ID

&client_secret=YOUR_CLIENT_SECRET

Response:

```
{
  "access_token": "RsT5OjzbzRn430zqMLgV3Ia",
  "token_type": "bearer",
  "expires_in": 3600,
  "refresh_token": "e1qoXg7Ik2RRua48lXIV"
}
```


Application Access

Client Credentials Grant

Client Credentials Grant

POST `https://api.example.com/1/oauth/token`

Post Body:

grant_type=client_credentials

&client_id=YOUR_CLIENT_ID

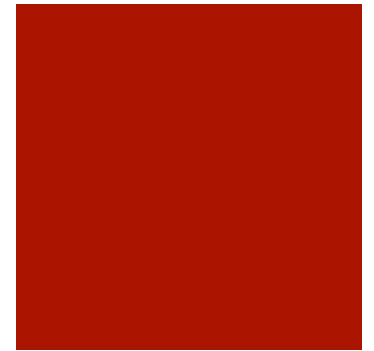
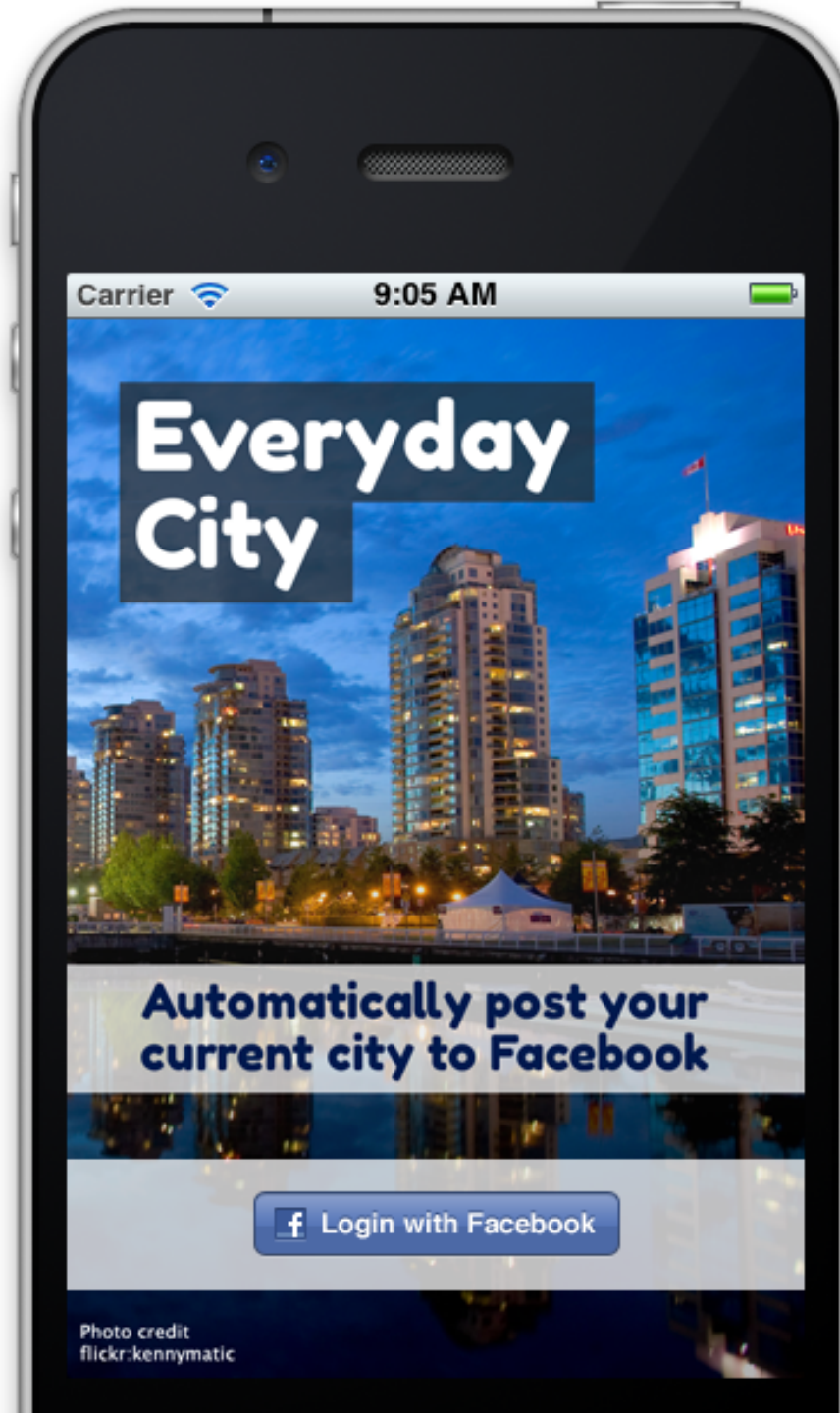
&client_secret=YOUR_CLIENT_SECRET

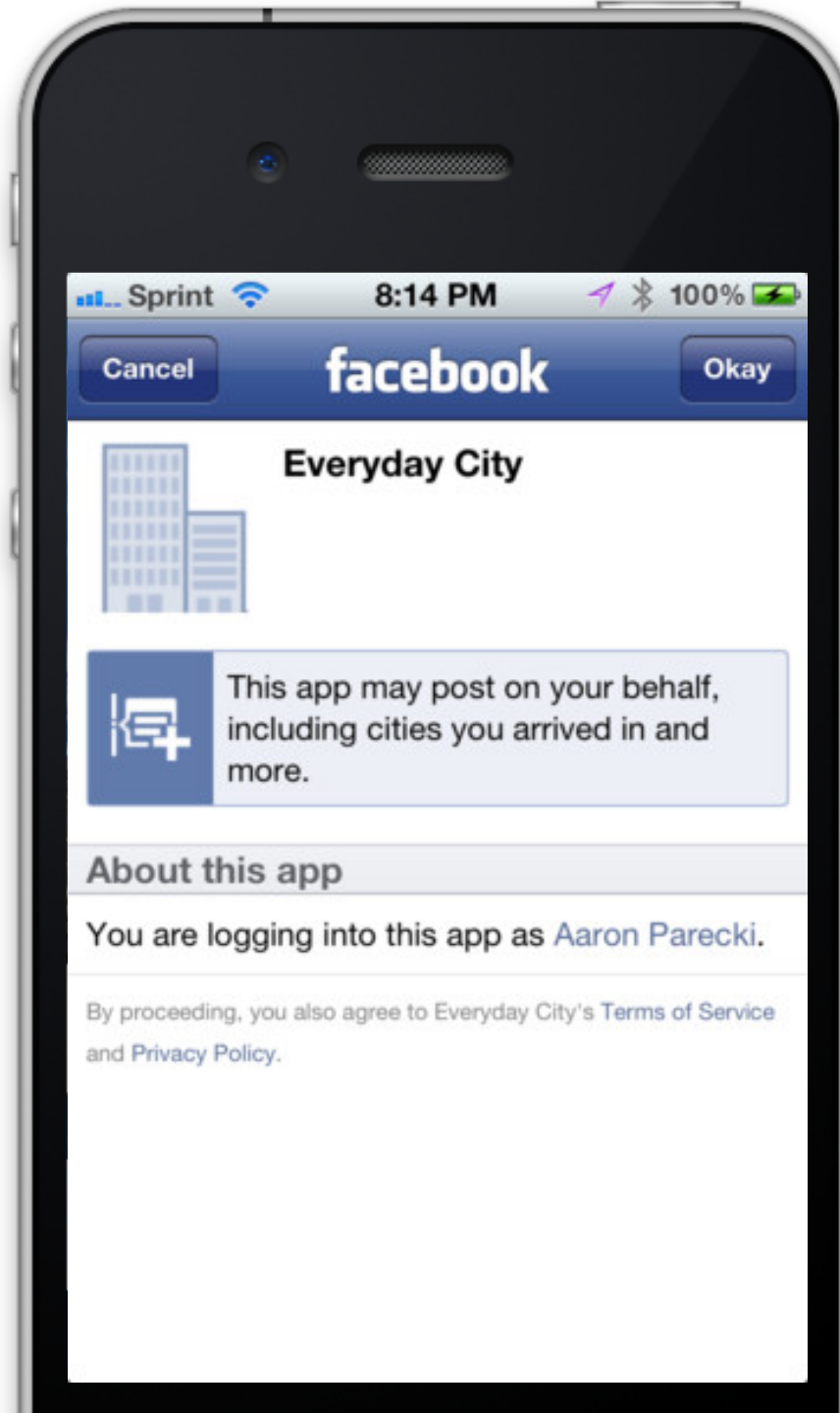
Response:

```
{
  "access_token": "RsT5OjzbzRn430zqMLgV3Ia",
  "token_type": "bearer",
  "expires_in": 3600,
  "refresh_token": "e1qoXg7Ik2RRua48lXIV"
}
```

Mobile Apps

Implicit Grant





Redirect back to your app



Facebook app redirects back to your app using a custom URI scheme.

Access token is included in the redirect, just like browser-based apps.

```
fb2865://authorize/#access_token=BAAEEemo2nocQBaff0eRTd
```



Mobile Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Mobile app makes API requests directly

Accessing Resources

So you have an access token.
Now what?

Use the access token to make requests

Now you can make requests using the access token.

```
GET https://api.example.com/me
Authorization: Bearer RsT5OjbzRn430zqMLgV3Ia
```

Access token can be in an HTTP header or a query string parameter

```
https://api.example.com/me?
access_token=RsT5OjbzRn430zqMLgV3Ia
```

Eventually the access token will expire

When you make a request with an expired token, you will get this response

```
{  
  "error": "expired_token"  
}
```

Now you need to get a new access token!

Get a new access token using a refresh token

Your server makes the following request

POST `https://api.example.com/oauth/token`

grant_type=refresh_token

&refresh_token=e1qoXg7Ik2RRua48lXIV

&client_id=YOUR_CLIENT_ID

&client_secret=YOUR_CLIENT_SECRET

Your server gets a similar response as the original call to `oauth/token` with new tokens.

```
{
  "access_token": "RsT5OjzbzRn430zqMLgV3Ia",
  "expires_in": 3600,
  "refresh_token": "e1qoXg7Ik2RRua48lXIV"
}
```

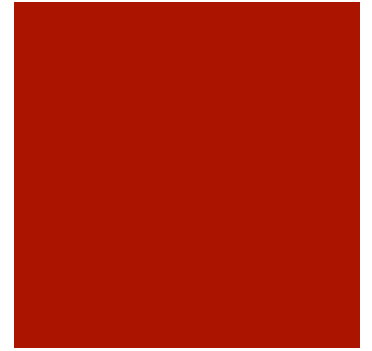
`aaron.pk/oauth2`

`@aaronpk`

Moving access into separate specs

Bearer tokens vs MAC
authentication

Bearer Tokens



```
GET /1/profile HTTP/1.1  
Host: api.example.com  
Authorization: Bearer B2mpLsHWhuVFw3YeLFW3f2
```

Bearer tokens are a cryptography-free way to access protected resources.

Relies on the security present in the HTTPS connection, since the request itself is not signed.

Security Recommendations for Clients Using Bearer Tokens

. Summary of Recommendations

Safeguard bearer tokens
bearer tokens are not
be able to use them t
is the primary securi
underlies all the mor

Validate SSL certificate
certificate chain whe
Failing to do so may
token and gain uninte

Always use TLS (https)
when making requests
the token to numerous
access.

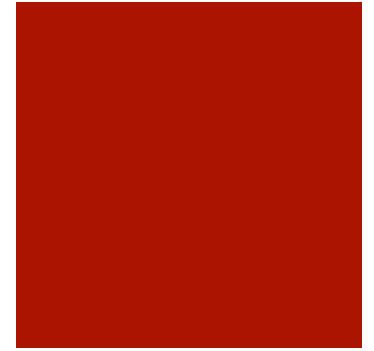
Don't store bearer token
bearer tokens within
is the default transm

Issue short-lived bearer
bearer tokens can red
In particular, only s
clients that run with
information leakage m

Don't pass bearer tokens
other software may no
history, web server l
tokens are passed in
parameters), attacker
history data, logs, o
bearer tokens in HTTP
confidentiality measu

- Safeguard bearer tokens
- Validate SSL certificates
- Always use https
- Don't store bearer tokens in plaintext cookies
- Issue short-lived bearer tokens
- Don't pass bearer tokens in page URLs

MAC Tokens



```
GET /1/profile HTTP/1.1
Host: api.example.com
Authorization: MAC id="jd93dh9dh39D",
               nonce="273156:di3hvdf8",
               bodyhash="k9kbtCIyI3/FEfpS/oIDjk6k=",
               mac="W7bdMZbv9UWOTadASIQHagZyirA="
```

MAC tokens provide a way to make authenticated requests with cryptographic verification of the request.

Similar to the original OAuth 1.0 method of using signatures.

OAuth 2 Clients

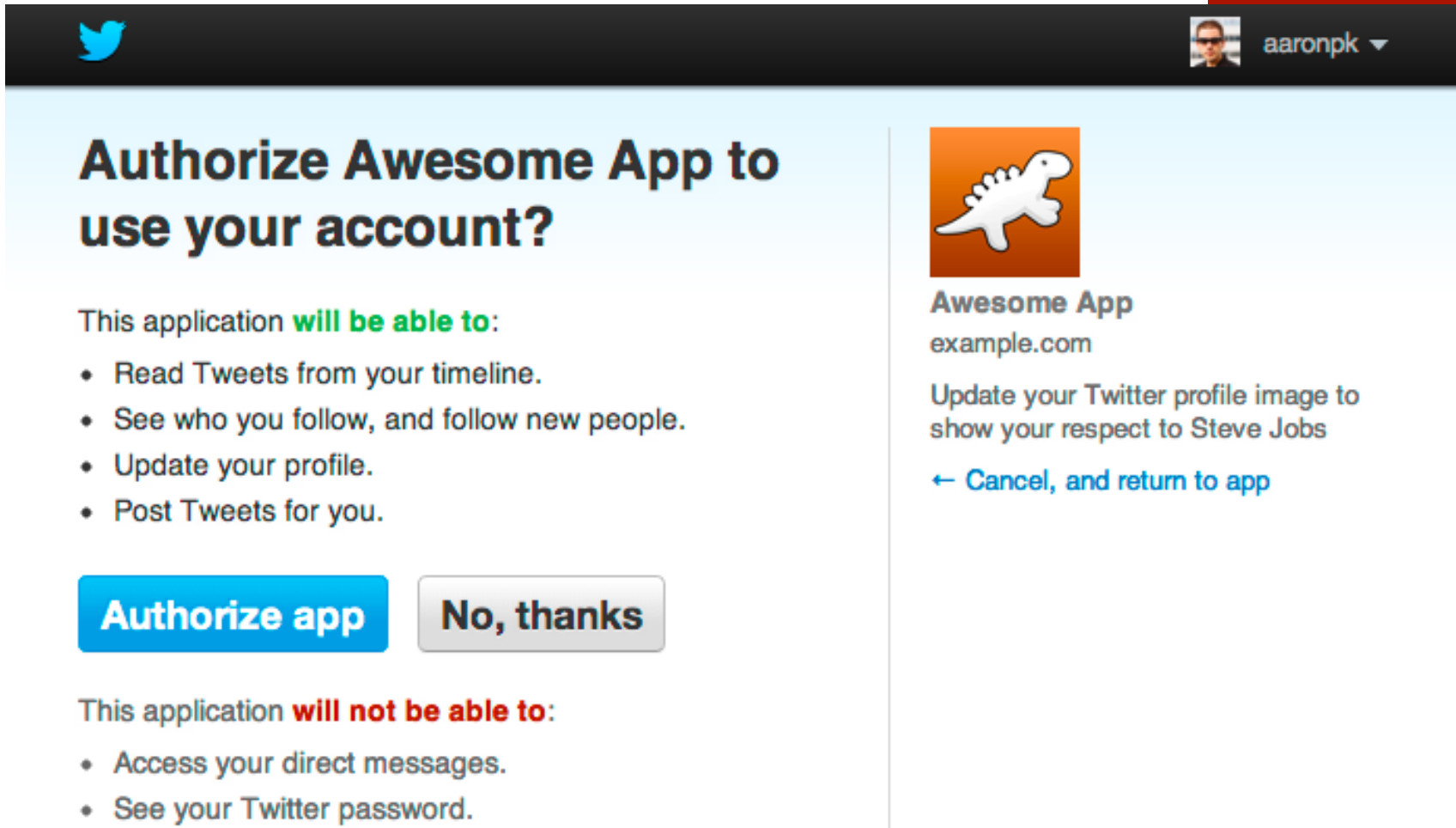
Client libraries should handle refreshing the token automatically behind the scenes.



Scope

Limiting access to resources

Limiting Access to Third Parties



The image shows a Twitter interface for a user named 'aaronpk'. The main heading is 'Authorize Awesome App to use your account?'. Below this, it states 'This application will be able to:' followed by a list of permissions: 'Read Tweets from your timeline.', 'See who you follow, and follow new people.', 'Update your profile.', and 'Post Tweets for you.' There are two buttons: 'Authorize app' (blue) and 'No, thanks' (grey). Below these, it states 'This application will not be able to:' followed by a list of permissions it cannot access: 'Access your direct messages.' and 'See your Twitter password.' On the right side, there is a profile picture of a dinosaur, the name 'Awesome App', the email 'example.com', and a message: 'Update your Twitter profile image to show your respect to Steve Jobs'. At the bottom right, there is a link: '← Cancel, and return to app'.

Authorize Awesome App to use your account?

This application **will be able to**:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Authorize app **No, thanks**

This application **will not be able to**:

- Access your direct messages.
- See your Twitter password.

Awesome App
example.com


Update your Twitter profile image to show your respect to Steve Jobs

← [Cancel, and return to app](#)

You can revoke access to any application at any time from the [Applications tab](#) of your Settings page.

By authorizing an application you continue to operate under [Twitter's Terms of Service](#). In particular, some usage information will be shared back with Twitter. For more, see our [Privacy Policy](#).

Limiting Access to Third Parties



The screenshot shows a Twitter interface for a user named 'aaronpk'. The main heading is 'Authorize Awesome App to use your account'. A large red arrow points to the text 'will be able to:' in the permissions list. The permissions list includes: Read Tweets from your timeline, See who you follow, and follow new people, Update your profile, and Post Tweets for you. Below the list are two buttons: 'Authorize app' (blue) and 'No, thanks' (gray). To the right, there is a section for 'Awesome App' with a dinosaur icon, the URL 'example.com', and a message: 'Update your Twitter profile image to show your respect to Steve Jobs'. Below this message is a link: '← Cancel, and return to app'.

Authorize Awesome App to use your account

This application **will be able to:**

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Authorize app **No, thanks**

This application **will not be able to:**

- Access your direct messages.
- See your Twitter password.


Awesome App
example.com

Update your Twitter profile image to show your respect to Steve Jobs

← [Cancel, and return to app](#)

You can revoke access to any application at any time from the [Applications tab](#) of your Settings page.

Limiting Access to Third Parties



The screenshot shows a Twitter interface for user 'aaronpk'. The main heading is 'Authorize Awesome App to use your account?'. Below this, it states 'This application will be able to:' followed by a list of permissions: 'Read Tweets from your timeline.', 'See who you follow, and follow new people.', 'Update your profile.', and 'Post Tweets for you.' There are two buttons: a blue 'Authorize app' button and a grey 'No thanks' button. A large red arrow points to the 'No thanks' button. To the right, there is a section for 'Awesome App' with a dinosaur icon, the URL 'example.com', and a message: 'Update your Twitter profile image to show your respect to Steve Jobs'. Below this is a link: '← Cancel, and return to app'.

Authorize Awesome App to use your account?

This application **will be able to**:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Authorize app **No thanks**

This application **will not be able to**:

- Access your direct messages.
- See your Twitter password.

Awesome App
example.com

Update your Twitter profile image to show your respect to Steve Jobs

← [Cancel, and return to app](#)

You can revoke access to any application at any time from the [Applications tab](#) of your Settings page.

By authorizing an application you continue to operate under [Twitter's Terms of Service](#). In particular, some usage information will be shared back with Twitter. For more, see our [Privacy Policy](#).

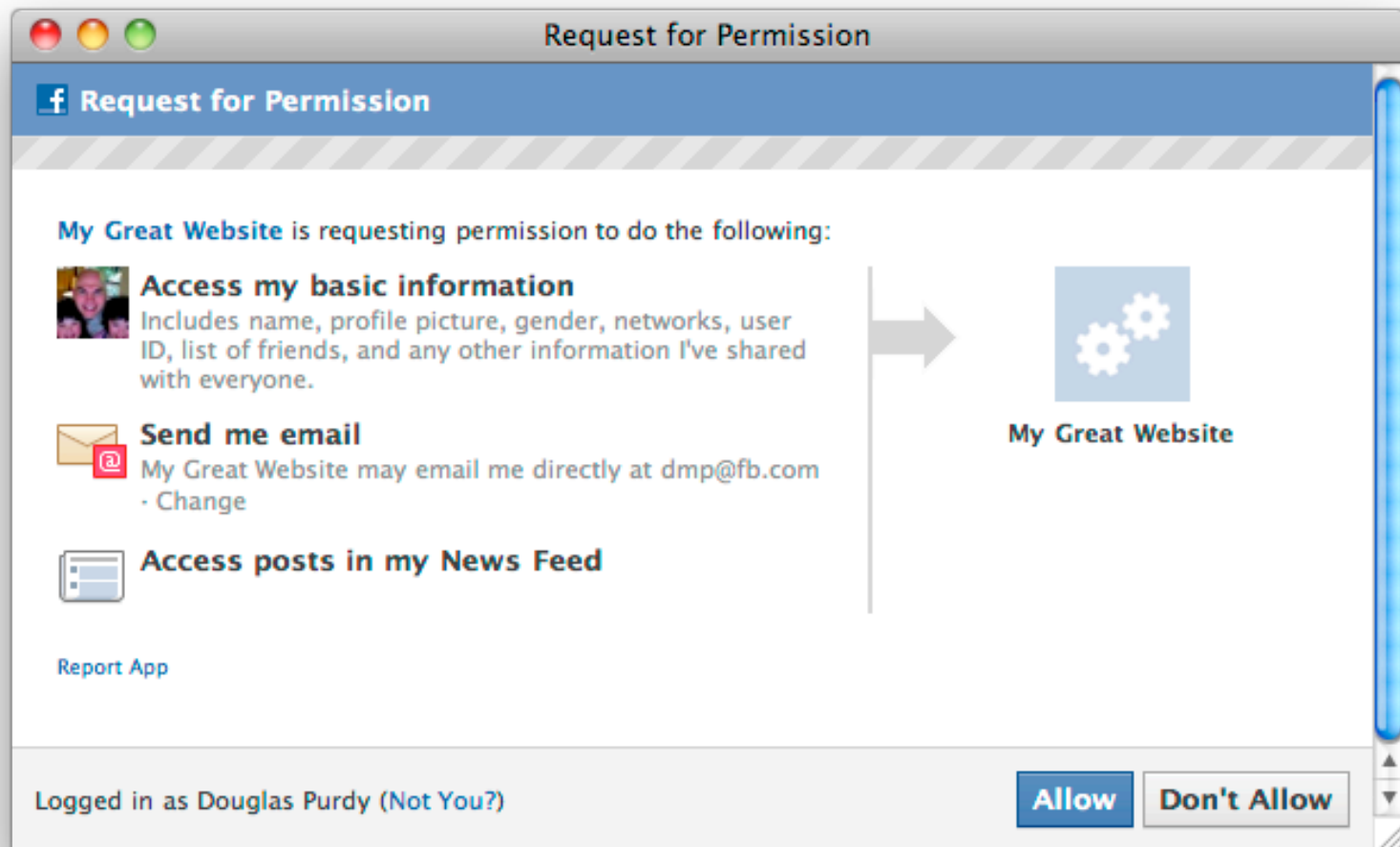
OAuth 2 scope



- Created to limit access to the third party.
- The scope of the access request expressed as a list of space-delimited strings.
 - In practice, many people use comma-separators instead.
- The spec does not define any values, it's left up to the implementor.
- If the value contains multiple strings, their order does not matter, and each string adds an additional access range to the requested scope.

OAuth 2 **scope** on Facebook

`https://www.facebook.com/dialog/oauth?
client_id=YOUR_APP_ID&redirect_uri=YOUR_URL
&scope=email,read_stream`



OAuth 2 **scope** on Facebook



User permission	Friends permission	Description
<code>user_about_me</code>	<code>friends_about_me</code>	Provides access to the "About Me" section of the profile in the <code>about</code> property
<code>user_activities</code>	<code>friends_activities</code>	Provides access to the user's list of activities as the <code>activities</code> connection
<code>user_birthday</code>	<code>friends_birthday</code>	Provides access to the birthday with year as the <code>birthday</code> property
<code>user_checkins</code>	<code>friends_checkins</code>	Provides read access to the authorized user's check-ins or a friend's check-ins that the user can see. This permission is superseded by <code>user_status</code> for new applications as of March, 2012.
<code>user_education_history</code>	<code>friends_education_history</code>	Provides access to education history as the <code>education</code> property
<code>user_events</code>	<code>friends_events</code>	Provides access to the list of events the user is attending as the <code>events</code> connection
<code>user_groups</code>	<code>friends_groups</code>	Provides access to the list of groups the user is a member of as the <code>groups</code> connection
<code>user_hometown</code>	<code>friends_hometown</code>	Provides access to the user's hometown in the <code>hometown</code> property

OAuth 2 **scope** on Github



`https://github.com/login/oauth/authorize?
client_id=...&scope=user,public_repo`

user

- Read/write access to profile info only.

public_repo

- Read/write access to public repos and organizations.

repo

- Read/write access to public and private repos and organizations.

delete_repo

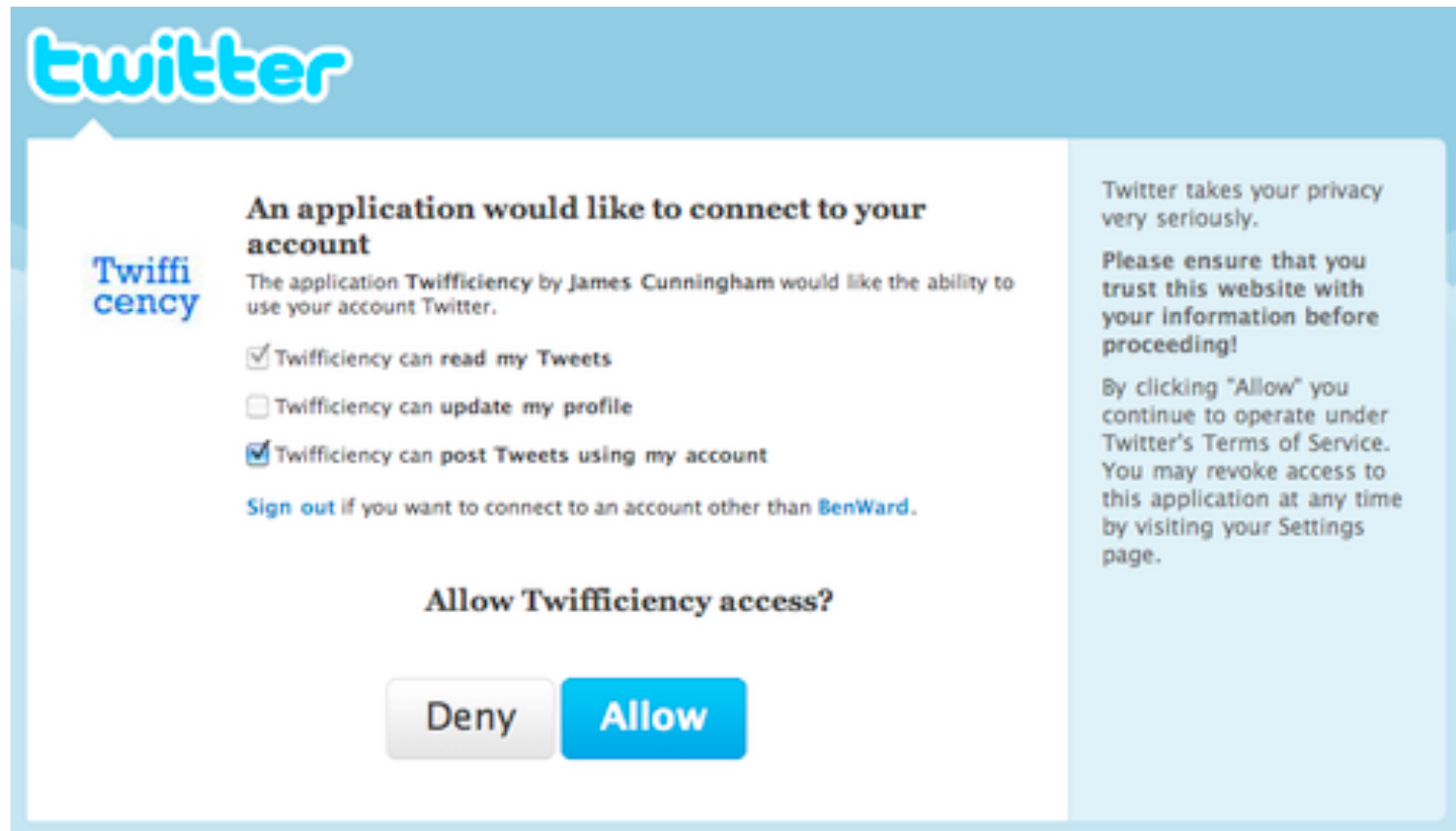
- Delete access to adminable repositories.

gist

- write access to gists.

Proposed New UI for Twitter

by Ben Ward



<http://blog.benward.me/post/968515729>

Implementing an OAuth Server



aaronpk 44

Dashboard

Inbox 3

Account Settings

Log Out

Explore GitHub

Gist

Blog

Help



Search...

geoloqi / oauth2-php

Watch

Fork

1

3

Source

Commits

Network

Pull Requests (0)

Graphs

Tree: a502060

Switch Branches (1)

Switch Tags (0)

Branch List

PHP OAuth 2 Server

Downloads

HTTP

Git Read-Only

https://github.com/geoloqi/oauth2-php.git



Read-Only access

* Updates server library to revision 10 of the OAuth 2.0 spec



aaronpk (author)

August 03, 2010

commit a502060e3370f184e37a
tree 970953f954b2924c86a3
parent 1a6d7c0824ef6c2ea2ee

oauth2-php /

name	age	message	history
lib/	August 03, 2010	* Updates server library to revision 10 of the OAu... [aaronpk]	
server/	August 03, 2010	* Updates server library to revision 10 of the OAu... [aaronpk]	

Implementing an OAuth Server



- Find a server library already written:
 - A short list available here: <http://oauth.net/2/>
- Read the spec of your chosen draft, *in its entirety*.
 - These people didn't write the spec for you to ignore it.
 - Each word is chosen carefully.
- Ultimately, each implementation is somewhat different, since in many cases the spec says SHOULD and leaves the choice up to the implementer.
- Understand the security implications of the implementation choices you make.

Implementing an OAuth Server



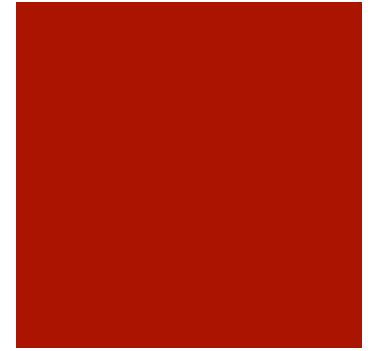
- Choose which grant types you want to support
 - Authorization Code – for traditional web apps
 - Implicit – for browser-based apps and mobile apps
 - Password – for your own website or mobile apps
 - Client Credentials – if applications can access resources on their own
- Choose whether to support Bearer tokens, MAC or both
- Define appropriate scopes for your service

OAuth 2 **scope** on your service



- Think about what scopes you might offer
- Don't over-complicate it for your users
- Read vs write is a good start

Mobile Applications



- External user agents are best
 - Use the service's primary app for authentication, like Facebook
 - Or open native Safari on iPhone rather than use an embedded browser
- Auth code or implicit grant type
 - In both cases, the client secret should never be used, since it is possible to decompile the app which would reveal the secret

Staying Involved

Join the Mailing List!

- <https://www.ietf.org/mailman/listinfo/oauth>
- People talk about OAuth
- Keep up to date on changes
- People argue about OAuth
- It's fun!





An **open protocol** to allow **secure authorization** in a **simple** and **standard** method from web, mobile and desktop applications.

[Read the OAuth 2 specification »](#)

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service.

For Consumer developers...

If you're building...

- web applications
- desktop applications
- mobile applications
- Javascript or browser-based apps
- webpage widgets

OAuth is a simple way to publish and interact with protected data. It's also a safer and more secure way for people to give you access. We've kept it simple to save you time.

For Service Provider developers...

If you're supporting...

- web applications
- mobile applications
- server-side APIs
- mashups

If you're storing protected data on your users' behalf, they shouldn't be spreading their passwords around the web to get access to it. Use OAuth to give your users access to their data while protecting their account credentials.

Get started...

oauth.net Website



- <http://oauth.net>
- Source code available on Github
 - github.com/aaronpk/oauth.net
- Please feel free to contribute to the website
- Contribute new lists of libraries, or help update information
- OAuth is community-driven!

github.com/aaronpk/oauth.net



github



Explore Gist Blog Help

aaronpk



177

PUBLIC



aaronpk / oauth.net

Admin

Pull Request

Unwatch

7

Fork

5

Code

Network

Pull Requests 0

Issues 0

Wiki

Graphs

The <http://oauth.net> website. Feel free to send pull requests with updates.

<http://oauth.net>

Clone in Mac

ZIP

HTTP

SSH

Git Read-Only

`git@github.com:aaronpk/oauth.net.git`

Read+Write access

branch: master

Files

Commits

Branches 1

Tags

Downloads

Latest commit to the **master** branch

cleaned up server/client section, added new links and added a section...

aaronpk authored 5 hours ago

commit b10247cb17

oauth.net /

name	age	message	history
2	5 hours ago	cleaned up server/client section, added new links and added a section... [aaronpk]	
about	a month ago	Update require() lines to avoid sending output before the header file... [aaronpk]	
advisories	a month ago	Update require() lines to avoid sending output before the header file... [aaronpk]	
code	a month ago	Update require() lines to avoid sending output before the header file... [aaronpk]	

More Info, Slides & Code Samples:

aaron.pk/oauth2

Thanks.

Aaron Parecki

@aaronpk

aaronparecki.com

github.com/aaronpk

