The Current State of OAuth 2

Aaron Parecki • @aaronpk
Open Source Bridge • Portland, June 2011

A Brief History

Before OAuth

aka the Dark Ages

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

Before OAuth 1.0

Many sites implemented things similar to OAuth 1.0, with slight differences between them.

- Flickr: “FlickrAuth” frobs and tokens
- Google: “AuthSub”
- Facebook: requests signed with MD5 hashes

OAuth 1.0


twitter
An application would like to connect to your account
The application Gowalla by Alamofire, Inc. would like the ability to access and update your data on Twitter. This application plans to use Twitter for logging you in in the future. Sign out if you want to connect to an account other than andypowell.
Allow Gowalla access?
Deny
Allow
Twitter takes your privacy very seriously.
Please ensure that you trust this website with your information before proceeding!
By clicking "Allow" you continue to operate under Twitter's Terms of Service. You may revoke access to this application at any time by visiting your Settings page.

OAuth 1.0 Signatures

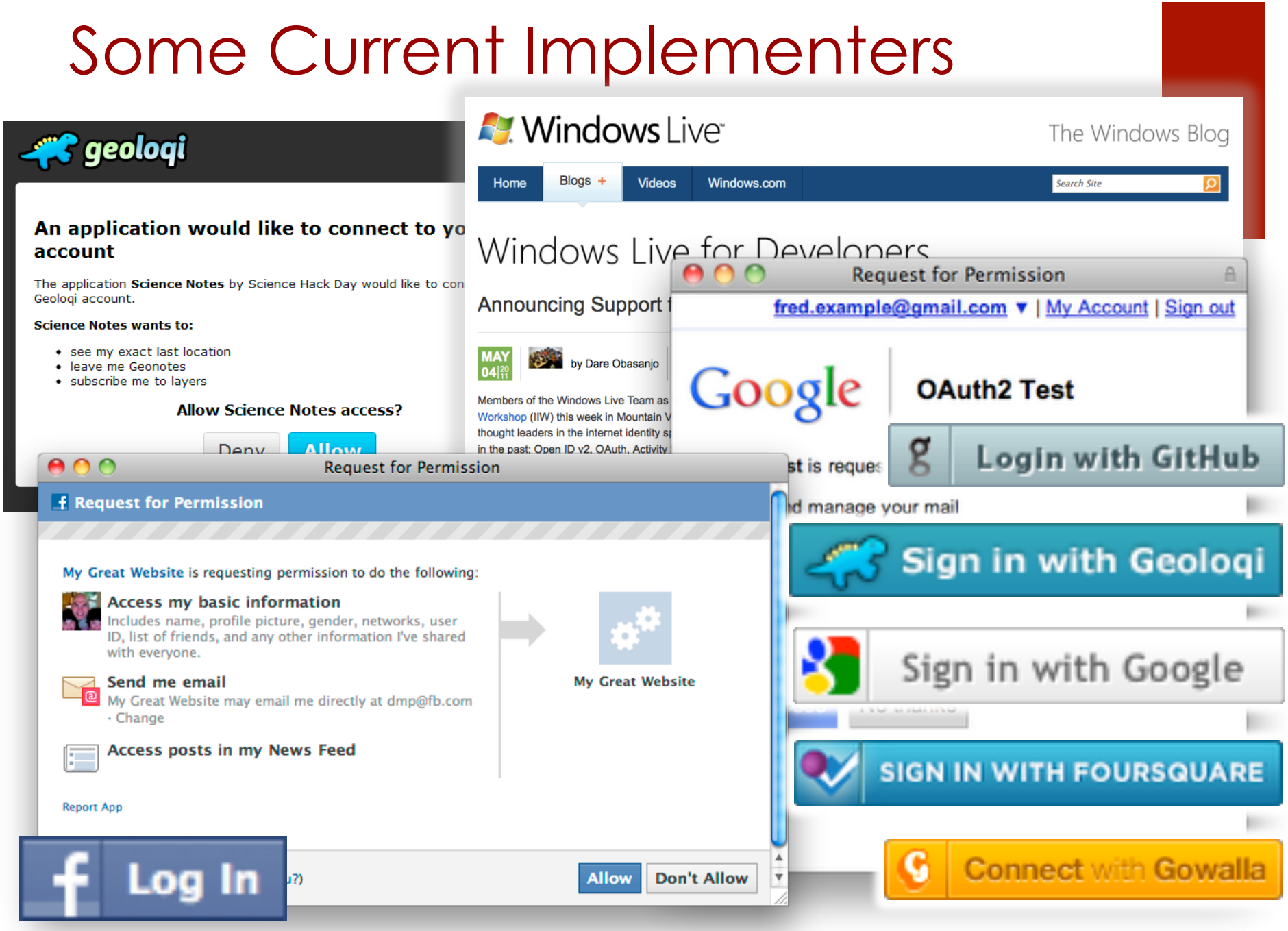
ARGH!!

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 duplica
while ensuring that both the key a
parameter are URL encoded in iso
equals ("=") sign to mark the key/v
the URL-encoded form of "%3D". E
joined by the URL-escaped amper

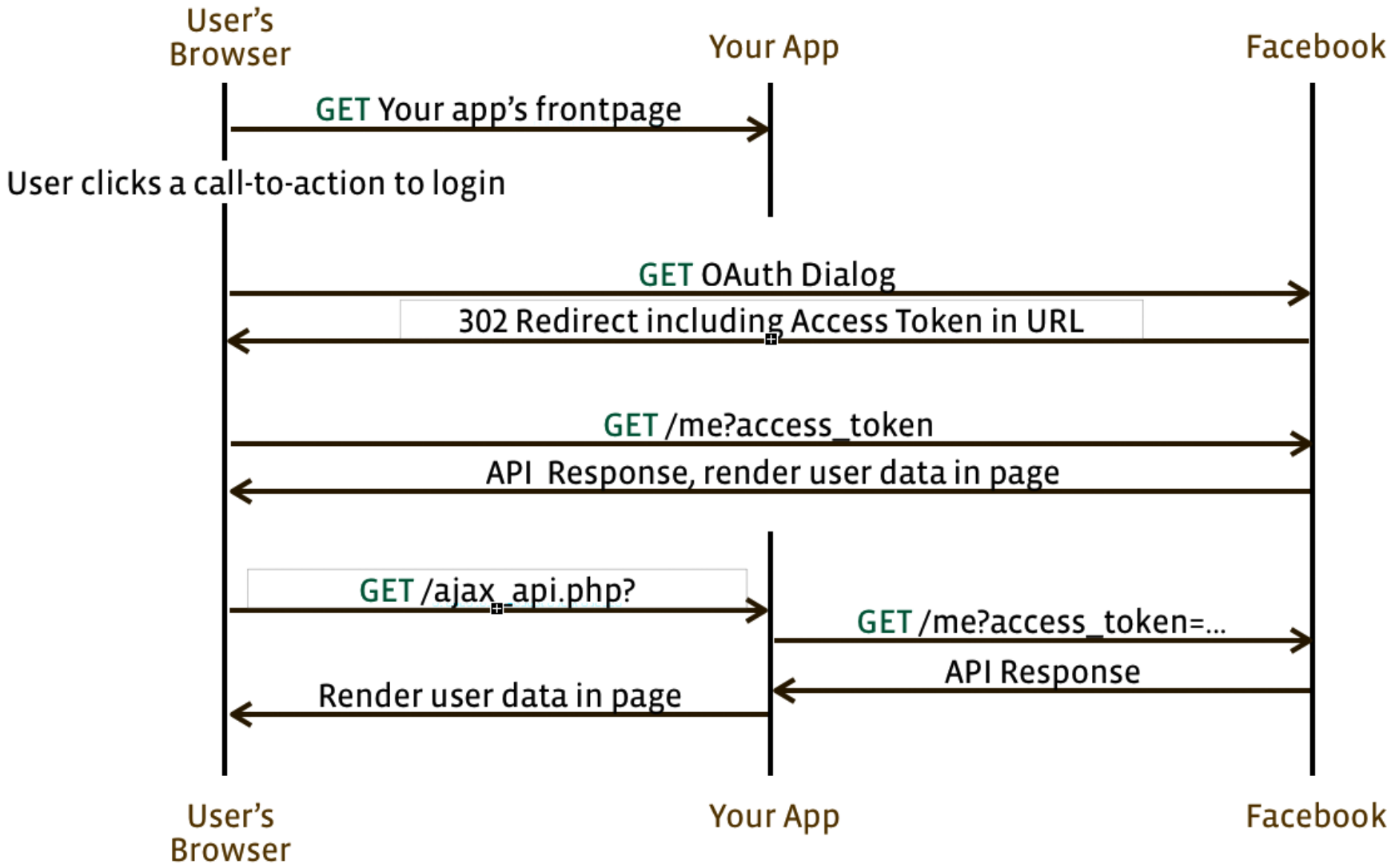
oauth_nonce="QP70eNmVz8jvdPevU3oJD2AfF7R7odC2XJcn4XlZJqk", oauth_callback="http%3A%2F%2Flocalhost%3A3005%2Fthe_dance%2Fprocess_callback%3Fservice_provider_id%3D11", oauth_signature_method="HMAC-SHA1", oauth_timestamp="1272323042", oauth_consumer_key="GDdmIQH6jhtmLUypg82g", oauth_signature="8wUi7m5HFQy76nowoCThusfgB%2BQ%3D", oauth_version="1.0"

OAuth 2:
signatures replaced by https

Some Current Implementers

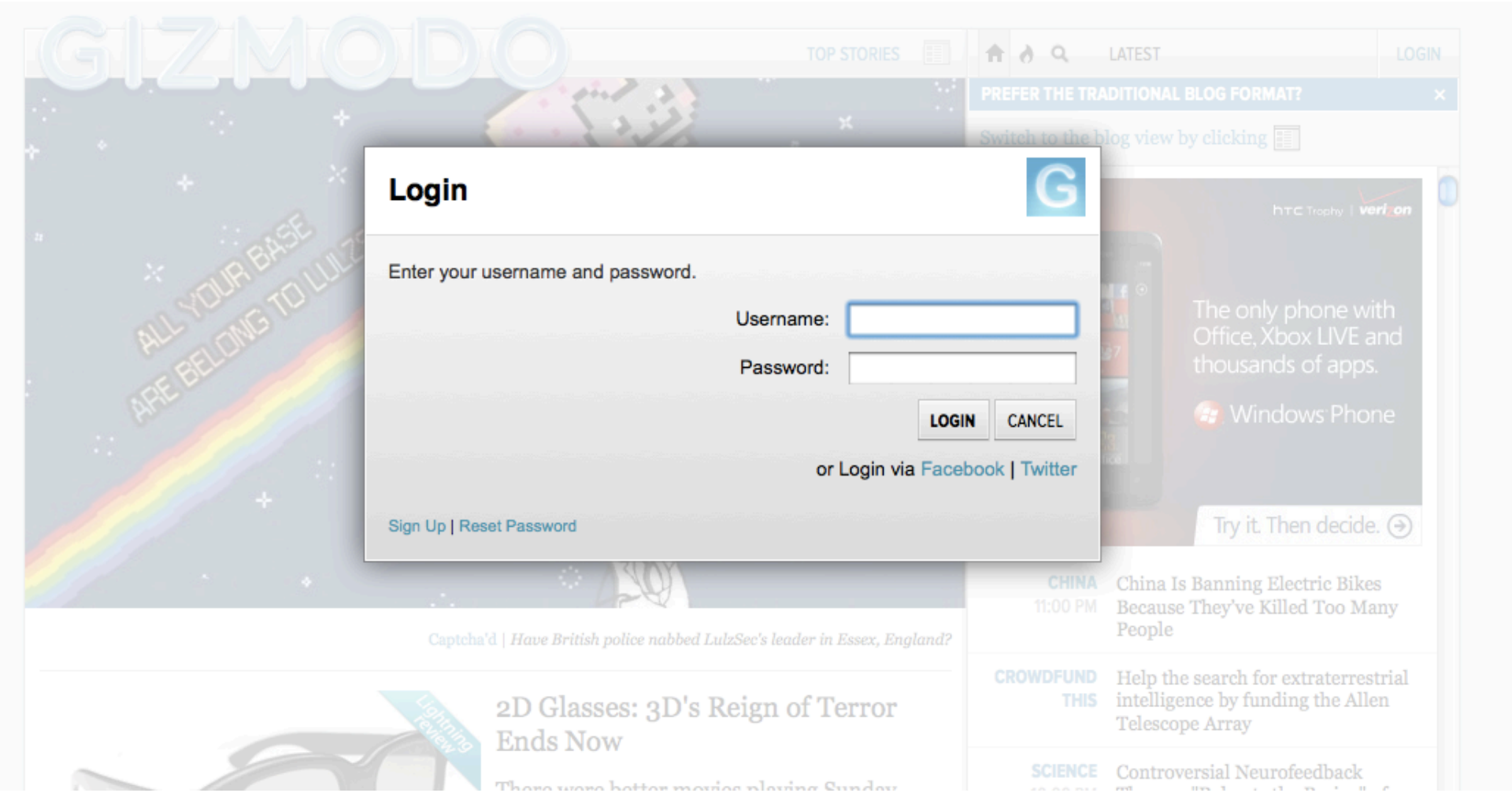


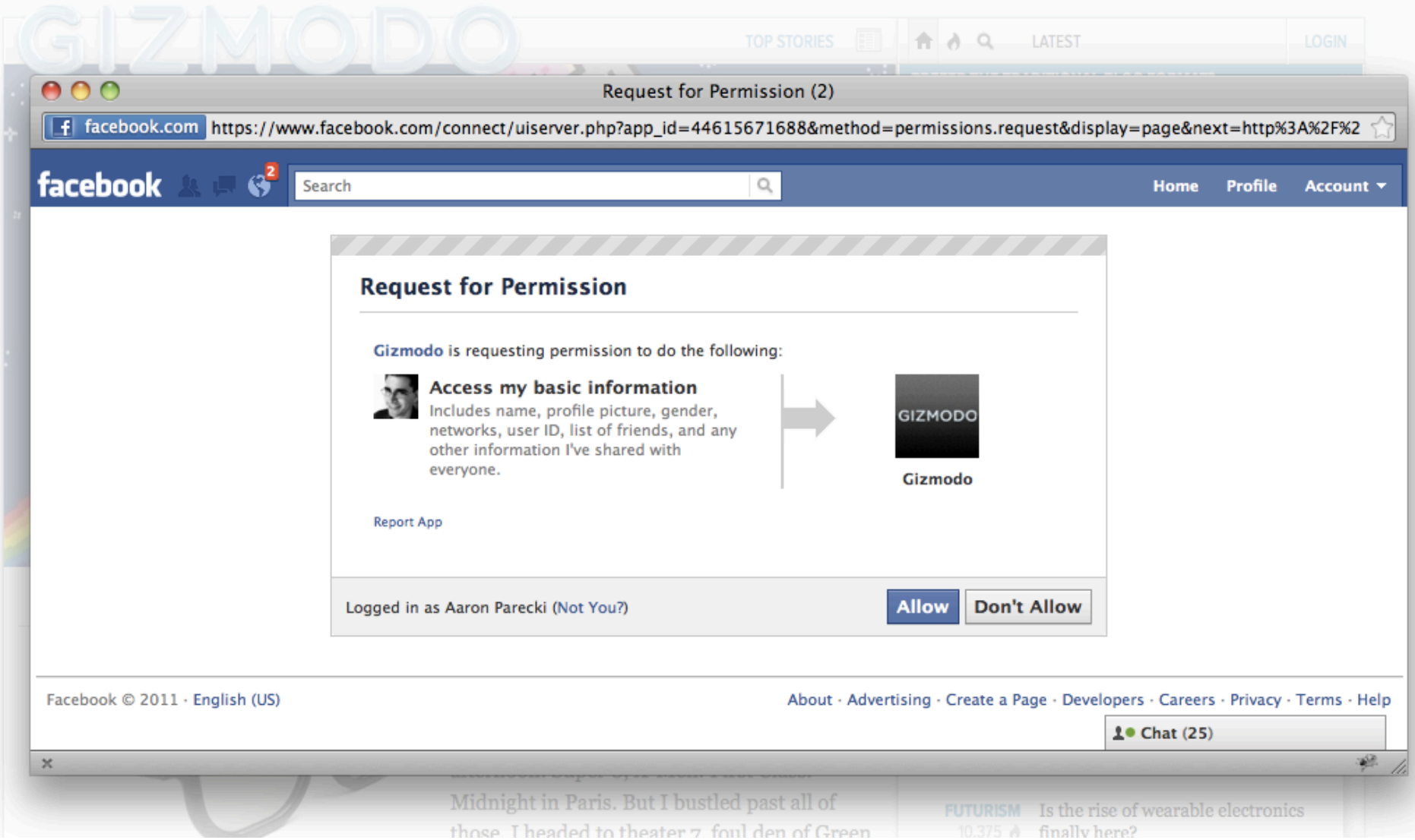
Facebook's OAuth Flow



Source: https://developers.facebook.com/docs/authentication/

@aaronpk


GIZMODO
TOP STORIES
LATEST
LOGIN
PREFER THE TRADITIONAL BLOG FORMAT?
Switch to the blog view by clicking
Login
Enter your username and password.
Username:
Password:
LOGIN
CANCEL
or Login via Facebook | Twitter
Sign Up | Reset Password
htc Trophy | verizon
The only phone with Office, Xbox LIVE and thousands of apps.
Windows Phone
Try it. Then decide.
CHINA
11:00 PM
China Is Banning Electric Bikes Because They've Killed Too Many People
CROWDFUND THIS
Help the search for extraterrestrial intelligence by funding the Allen Telescope Array
SCIENCE
Controversial Neurofeedback
Captcha'd | Have British police nabbed LulzSec's leader in Essex, England?
2D Glasses: 3D's Reign of Terror Ends Now
Lightning Review


GIZMODO
TOP STORIES
LATEST
LOGIN
Request for Permission (2)
facebook.com
https://www.facebook.com/connect/uiserver.php?app_id=4461567688&method=permissions.request&display=page&next=http%3A%2F%2
facebook
Search
Home
Profile
Account
Request for Permission
Gizmodo 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.
GIZMODO
Gizmodo
Report App
Logged in as Aaron Parecki (Not You?)
Allow
Don't Allow
Facebook © 2011 · English (US)
About · Advertising · Create a Page · Developers · Careers · Privacy · Terms · Help
Chat (25)
Midnight in Paris. But I bustled past all of
FUTURISM
Is the rise of wearable electronics
finally here?

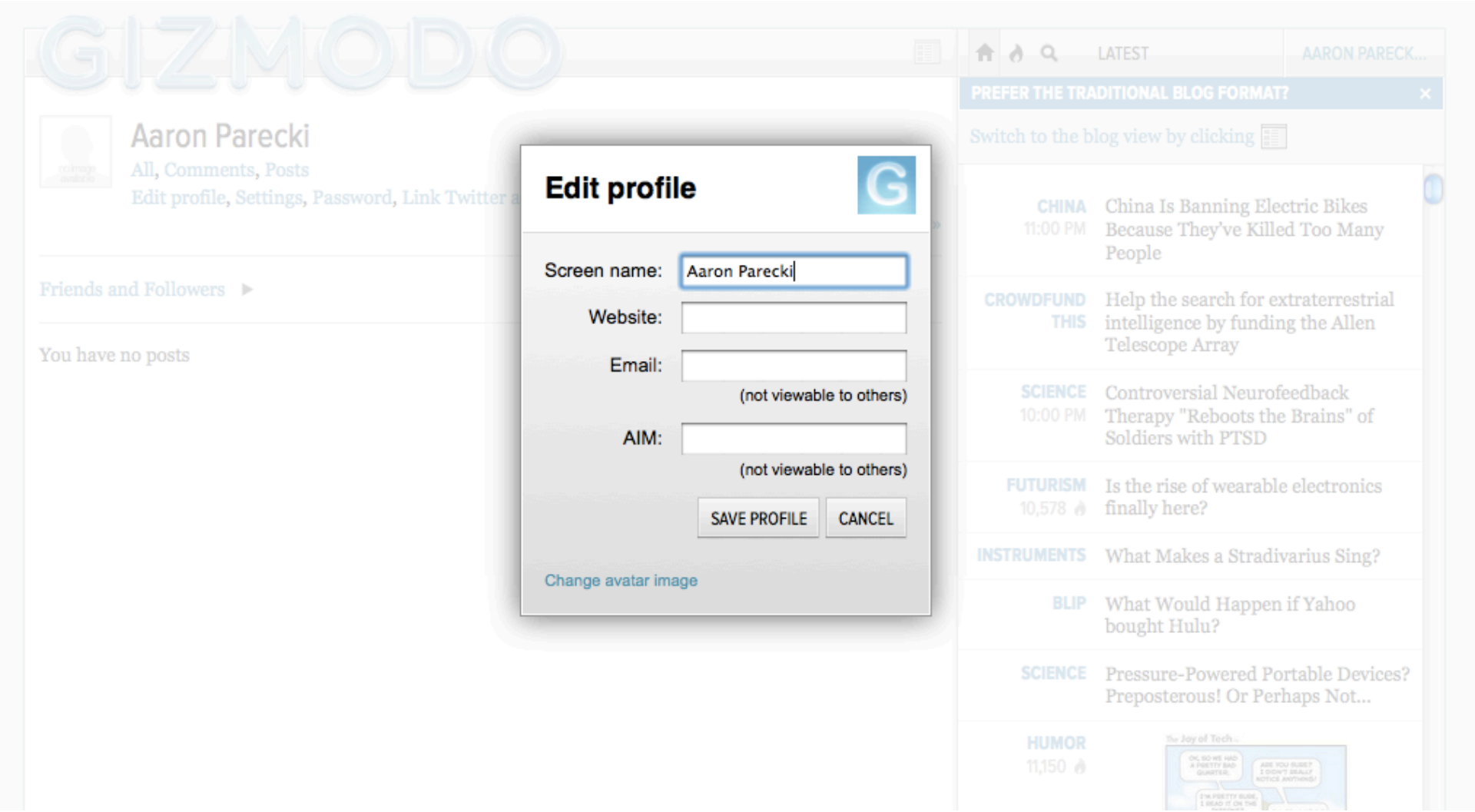

GIZMODO
LATEST
AARON PARECK...
PREFER THE TRADITIONAL BLOG FORMAT?
Switch to the blog view by clicking
Aaron Parecki
All, Comments, Posts
Edit profile, Settings, Password, Link Twitter a
Friends and Followers
You have no posts
Edit profile
Screen name:
Aaron Parecki
Website:
Email:
(not viewable to others)
AIM:
(not viewable to others)
SAVE PROFILE
CANCEL
Change avatar image
CHINA
11:00 PM
China Is Banning Electric Bikes Because They've Killed Too Many People
CROWDFUND THIS
Help the search for extraterrestrial intelligence by funding the Allen Telescope Array
SCIENCE
10:00 PM
Controversial Neurofeedback Therapy "Reboots the Brains" of Soldiers with PTSD
FUTURISM
10,578
Is the rise of wearable electronics finally here?
INSTRUMENTS
What Makes a Stradivarius Sing?
BLIP
What Would Happen if Yahoo bought Hulu?
SCIENCE
Pressure-Powered Portable Devices? Preposterous! Or Perhaps Not...
HUMOR
11,150

Table of Contents

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

The Spec

Read it.

It's not that bad, I promise.

aaron.pk/oauth2-10

But which draft? There are 16!!

Versions: 00 01 02 03 04 05 06 07 08 09 10 11
12 13 14 15 16

Currently Implemented Drafts

Provider	Draft	Reference
Foursquare	-10	http://aaron.pk/2YS
Google	-10	http://code.google.com/apis/accounts/docs/OAuth2.html
Gowalla	-8	http://gowalla.com/api/docs/oauth
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://geoloqi.org/API

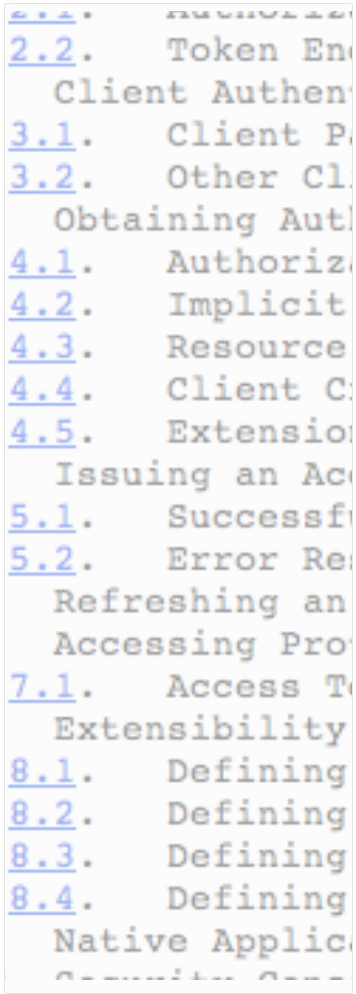
Abstract Protocol Flow

```
+--------+                                +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |         Authorization Grant & +---------------+
|        |--(C)--- Client Credentials -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Definitions

```
resource owner
   An entity capable of granting access to a protected resource.
   When the resource owner is a person, it is referred to as an end-
   user.
resource server
   The server hosting the protected resources, capable of accepting
   and responding to protected resource requests using access tokens.
client
   An application making protected resource requests on behalf of the
   resource owner and with its authorization.
authorization server
   The server issuing access tokens to the client after successfully
   authenticating the resource owner and obtaining authorization.
```

1. Authorization



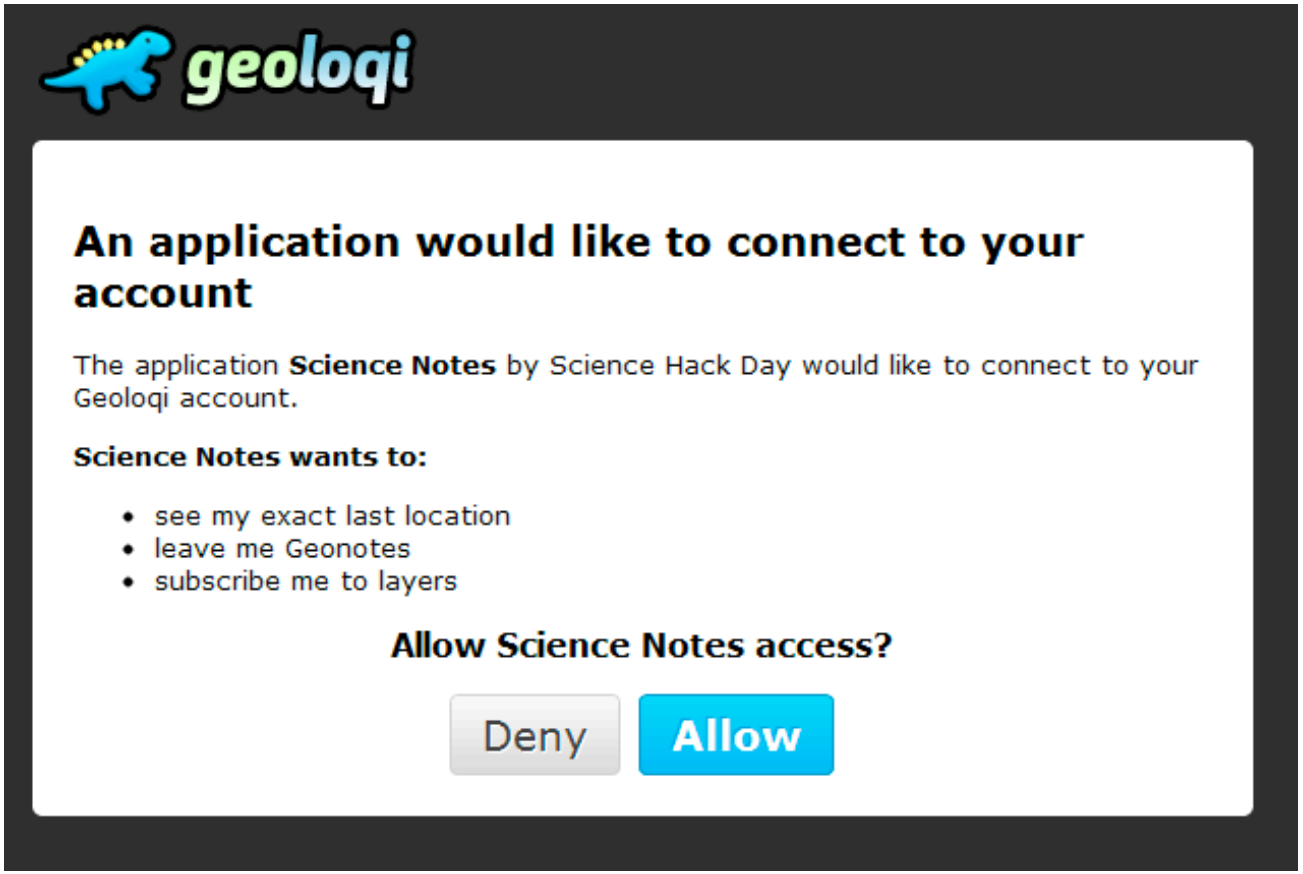
Create a “Log In” link

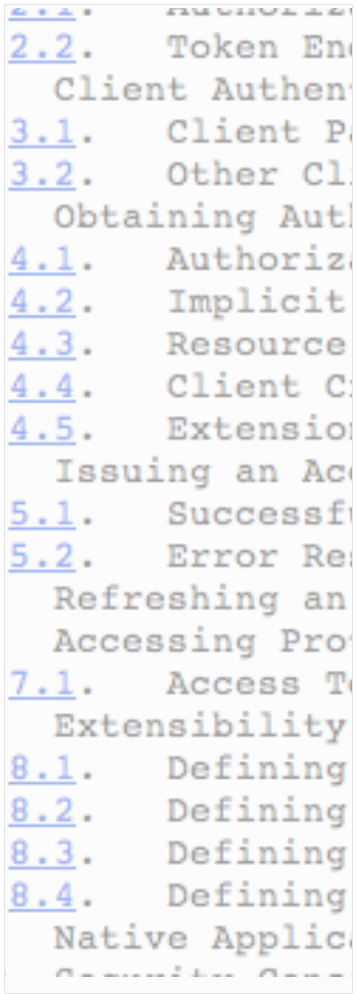
Link to:

```
https://geoloqi.com/oauth/authorize?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI
```

Send the user to the auth page

```
https://geoloqi.com/oauth/authorize?
response_type=code&client_id=YOUR_CLIENT_ID
&redirect_uri=REDIRECT_URI
```





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`

Exchange auth code for an access token

Your server makes the following request

```
POST https://api.geoloqi.com/1/oauth/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
```

Exchange auth code for an access token (response)

2.2. Token En
Client Authen
3.1. Client P
3.2. Other Cl
Obtaining Aut
4.1. Authoriz
4.2. Implicit
4.3. Resource
4.4. Client C
4.5. Extensio
Issuing an Ac
5.1. Successf
5.2. Error Re
Refreshing an
Accessing Pro
7.1. Access T
Extensibility
8.1. Defining
8.2. Defining
8.3. Defining
8.4. Defining
Native Applic

Your server gets a response like the following

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

or if there was an error

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

2. Accessing Resources

Use the access token to make requests

2.2. Token En
Client Authen
3.1. Client P
3.2. Other Cl
Obtaining Aut
4.1. Authoriz
4.2. Implicit
4.3. Resource
4.4. Client C
4.5. Extensio
Issuing an Ac
5.1. Successf
5.2. Error Re
Refreshing an
Accessing Pro
7.1. Access T
Extensibility
8.1. Defining
8.2. Defining
8.3. Defining
8.4. Defining
Native Applic

Now you can make requests using the access token.

```
GET https://api.geoloqi.com/1/account/profile
Authorization: OAuth RsT5OjbzRn430zqMLgV3Ia
```

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

```
https://api.geoloqi.com/1/account/profile?
oauth_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

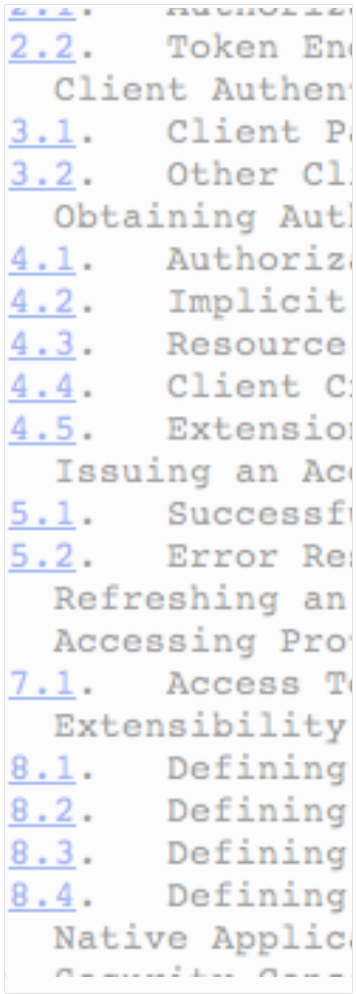
```
POST https://api.geoloqi.com/1/oauth/token

grant_type=refresh_token
&reresh_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":"RsT5OjbzRn430zqMLgV3Ia",
  "expires_in":3600,
  "refresh_token":"e1qoXg7Ik2RRua48lXIV"
}
```

OAuth 2 Clients

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



Authorization Methods

- Auth Code
- Refresh Token
- Password

Draft 10 also has

- Assertion

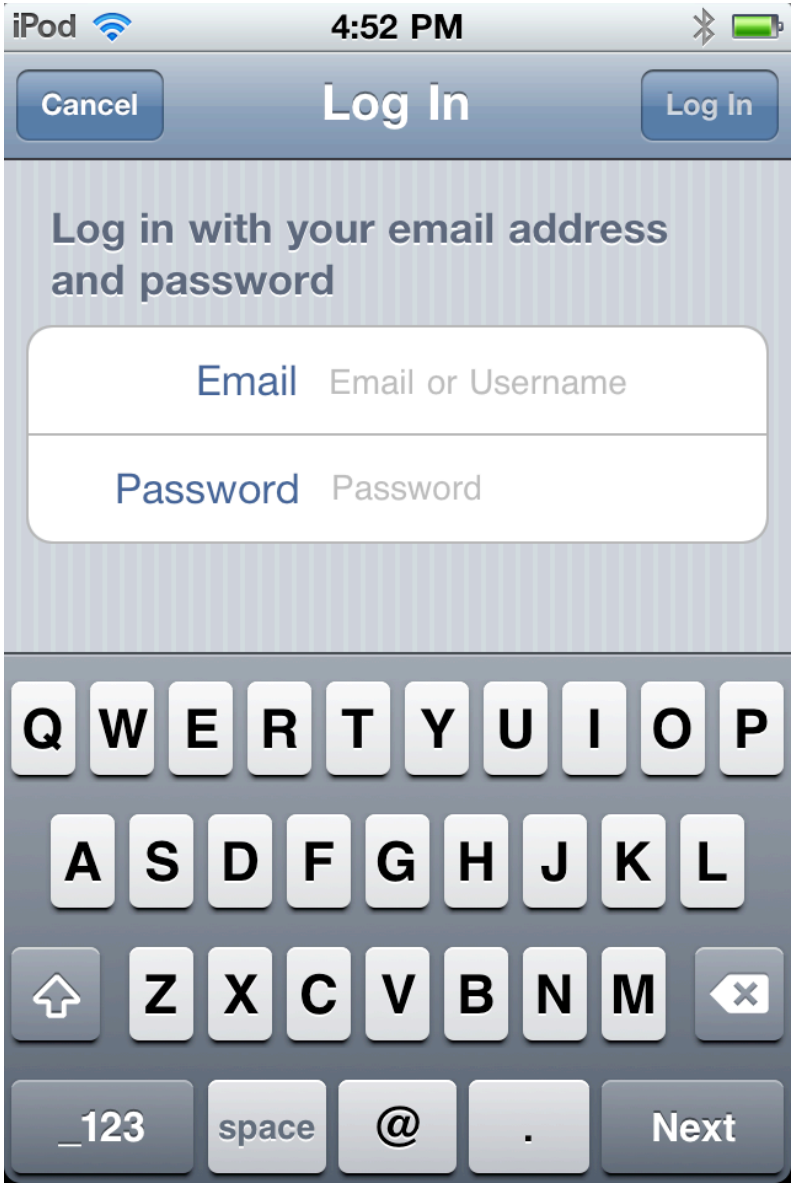
Draft 16 also has

- Implicit (for browser-based apps)
- Extensions (for defining custom grant types)

Password Grant Type

Suitable for mobile or native desktop apps where a web browser flow would be awkward.

This breaks the fundamental benefit of OAuth (not giving your password to third parties), so should probably be limited to your own apps.



Password Grant

Your server makes the following request

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

grant_type=password
&username=USERNAME
&password=PASSWORD
&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":"RsT5OjbzRn430zqMLgV3Ia",
  "expires_in":3600,
  "refresh_token":"e1qoXg7Ik2RRua48lXIV"
}
```

Implicit Grant (-16)

For clients who can't store a client secret in a secure way, typically Javascript-based apps.

No concept of refresh tokens, and auth codes are not used either.

The redirection back to your app will include an access token in the URL fragment.

```
https://example.com/auth#access_token=FJQbwq9
```

Implementing an OAuth Server

github SOCIAL CODING

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
```

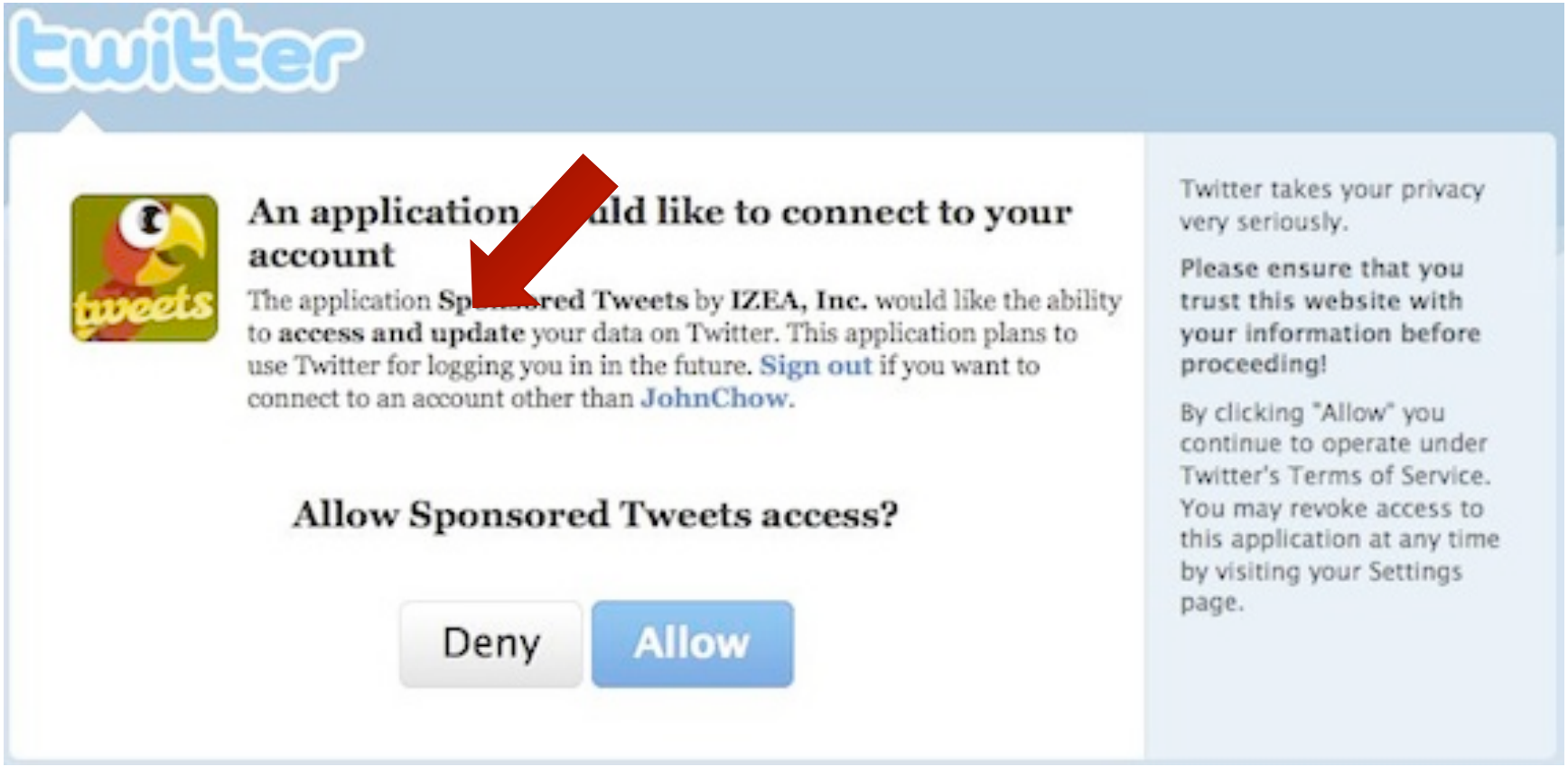
oauth2-php /

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

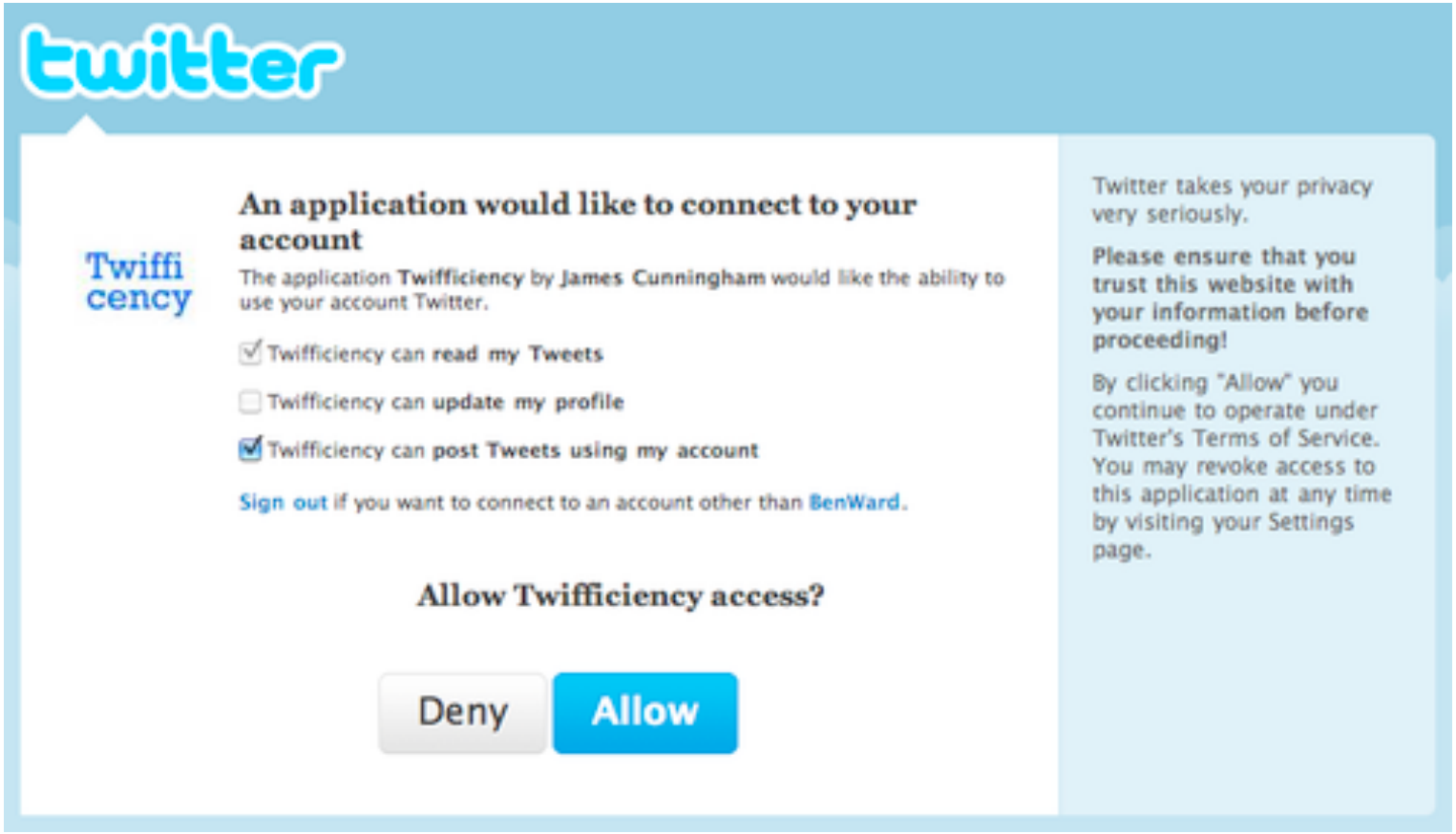
- Find a server library already written
- Choose a draft to implement
  - Draft 10 – more likely that your users will be familiar with it
  - Latest Draft – if you want to show you're bleeding edge ;)
- Read the spec of your chosen draft, *in its entirety*.
  - These people didn't write the spec for 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.

Limiting Access to Third Parties



Proposed New UI for Twitter

by Ben Ward



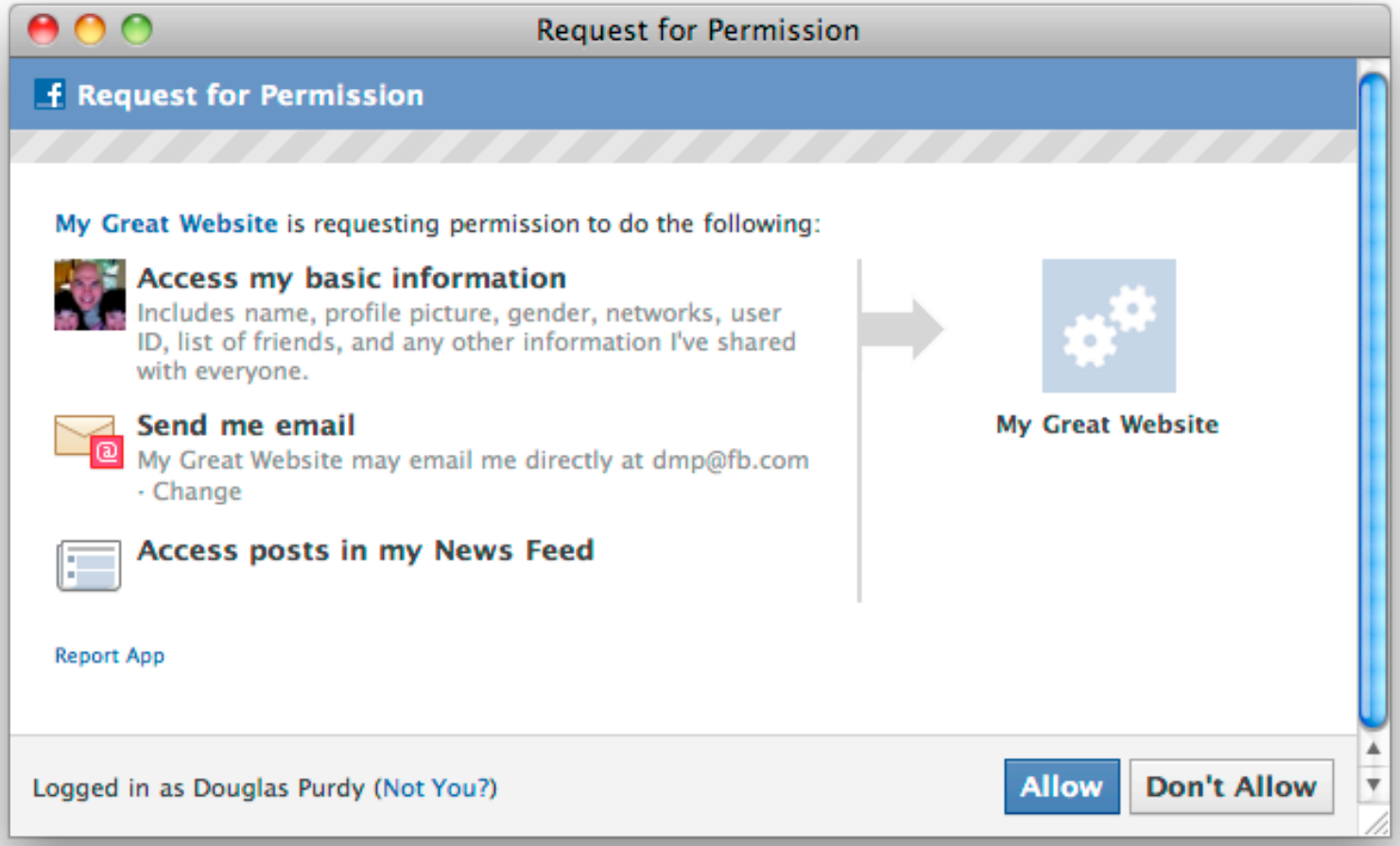
http://blog.benward.me/post/968515729

OAuth 2 **scope**

- Created to limit access to the third party.
- The scope of the access request expressed as a list of space-delimited, case sensitive strings.
- The value is defined by the authorization server.
- If the value contains multiple space-delimited 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 Github

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

- (no scope) - public read-only access (includes user profile info, public repo info, and gists).
- `user` - DB read/write access to profile info only.
- `public_repo` - DB read/write access, and Git read access to public repos.
- `repo` - DB read/write access, and Git read access to public and private repos.
- `gist` - write access to gists.

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

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!

Raging Debates

- March and April 2011
- Summary Email: http://aaron.pk/2YX

The question:

Require or **recommend** using TLS on the auth redirect?

Require TLS on auth redirect?

- When the auth server redirects back to the client with an auth code in the query string, an MITM attack is possible unless the client's server is using https
- Problem: It is not always practical to run your site with https
- Certificates aren't the only barrier
- Loading off-site content (banner ads, site statistics, etc) would also have to use https, otherwise the visitor sees terrible warning messages
- https is often slower, since browsers don't like to cache things

Require TLS on auth redirect?

- Many large providers (Google, Facebook) won't require it, because they don't want to force developers to have SSL certificates (would lead to lower adoption)
- Many enterprise users do want to require it, because they are more concerned with security than high adoption of their API
- The question is whether the spec should require it or just recommend, if it does, then Google/Facebook/others won't be compliant

Require TLS on auth redirect?

"This debate is just like talking about highway safety. We know how to make driving 100% safe. Everyone will drive tanks at no faster than 10mph, or alternatively, no one will drive and we'll all climb on conveyor belts to get anywhere (each bubble wrapped and put into a medically induced coma to make sure we don't move around). But in the real world, practical considerations trump 100% guarantees.

By just offering services online you put users at some risk. If you want 100% safe online banking, just don't offer any."

Excerpt of an email to the oauth-wg list from Eran Hammer-Lahav: http://aaron.pk/2Ya

Re: [OAUTH-WG] Authorization code security issue (reframed)

- *From*: Blaine Cook <romeda at gmail.com>
- *To*: Eran Hammer-Lahav <eran at hueniverse.com>
- *Cc*: OAuth WG <oauth at ietf.org>
- *Date*: Tue, 5 Apr 2011 08:28:39 +0100

```
<chair>

DO NOT REPLY TO THIS EMAIL.

To Eran's point, before reaching the end of this thread after limited
access to email over the weekend, I was going to shut this thread down
anyways.

I'm not going to issue a call for consensus on this issue, because I
don't believe anyone on the list (except for a small handful of active
participants) have read even a fraction of the thread. Furthermore,
I'm sure that we understand this threat, and we will ensure that it is
properly documented in the Security Considerations.

If you are reading this, have not had your voice heard on this matter,
and think that this issue is unrepresented in the Security
Considerations, please email the chairs directly.

b.

nb: The OAuth working group's job is to ship OAuth 2.0; while this is
includes ensuring a secure protocol, "secure" does not equal
"impenetrable". If you believe that cookies and security in HTTP is
flawed, please go talk to the HTTP working group and lobby for a MUST
of TLS over all HTTP connections. I also urge you to consider the
implications of Comodogate — $100 certs ain't what they used to be.

</chair>
```

draft-ietf-oauth-v2-16

10.9. Authorization Codes

The transmission of authorization codes SHOULD be made over a secure channel, and the client SHOULD implement TLS for use with its redirection URI if the URI identifies a network resource. Authorization codes MUST be kept confidential. Since authorization codes are transmitted via user-agent redirections, they could potentially be disclosed through user-agent history and HTTP referrer headers.

Authorization codes operate as plaintext bearer credentials, used to verify that the end-user who granted authorization at the authorization server, is the same end-user returning to the client to complete the process. Therefore, if the client relies on the authorization code for its own end-user authentication, the client redirection endpoint MUST require TLS.

Authorization codes SHOULD be short lived and MUST be single use. If the authorization server observes multiple attempts to exchange an authorization code for an access token, the authorization server SHOULD revoke all access tokens already granted based on the compromised authorization code.

If the client can be authenticated, the authorization servers MUST authenticate the client and ensure that the authorization code was issued to the same client.

Moving access into separate specs

Bearer tokens vs MAC authentication

Draft 10

```
GET /1/profile HTTP/1.1
Host: api.example.com
Authorization: OAuth vF9dft4qmT
```

Draft 16

```
GET /1/profile HTTP/1.1
Host: api.example.com
Authorization: Bearer vF9dft4qmT
```

or

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

http://tools.ietf.org/html/draft-ietf-oauth-v2-10#section-5
http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer-04#section-2
http://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-00#section-3

Security Recommendations for Clients Using Bearer Tokens

- 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

```
.  Summary of Recommenda

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
```

@blaine
Blaine Cook

Hurrah! Four years after OAuth was modelled on FlickrAuth, Flickr supports OAuth. Sometimes standards work. :-D code.flickr.com/blog/2011/06/2...

7:18 AM Jun 22nd 2011 via web

http://code.flickr.com/blog/2011/06/21/flickr-now-supports-oauth-1-0a/

code.flickr

Home | DevBlog | Forums | API | Jobs

Flickr now Supports OAuth 1.0a

Posted by jamal on June 21st, 2011

We're happy to announce that Flickr now supports OAuth! This is an open standard for authentication, which is now fully supported by the Flickr API. You can get started by going to our OAuth documentation. As part of this announcement, we would also like to note that the old Flickr

OAuth is very
token (frob in
token which c
you will be ab
started.

In addition to
web, and have
Desktop flow, which is no longer necessary.

Currently, we only support OAuth 1.0a, but we have plans to eventually support OAuth 2.0. The decision was based on the fact that OAuth 2.0 is still an evolving definition that is rapidly changing.

Currently, we only support OAuth 1.0a, but we have plans to eventually support OAuth 2.0. The decision was based on the fact that OAuth 2.0 is still an evolving definition that is rapidly changing.

DEVBLOG: RECENT POSTS
Flickr development team talks nerdy.

Flickr now Supports OAuth 1.0a
Posted by jamal on Jun 21, 2011

Refreshing The API Explorer Posted by paulmison on Jun 1, 2011

d by

lickr API
b 23, 2011

ws Posted

tions

api clustr conference dopplr exif externalinterface fireeagle flash flickr flickr.photos.search frontend geo geotags i18n international internationalization interview

Nothing is perfect.

Everything changes.

Go build stuff.

Thanks.

Aaron Parecki

@aaronpk

aaronparecki.com

github.com/aaronpk