Integrating services with

Luca Mearelli
Web2Expo - Berlin 2008
Web 2.0 means sharing data, through API
Users want to access their data using many services
Developers want to satisfy their users (and make it easy for them)
Service providers need to keep their users data secure.
welcome
the password antipattern
Are your friends on Twitter?

We can check if anyone in your email contacts already has a Twitter account.

Search Web Email (Hotmail, Yahoo, Gmail, Etc.)

Your Email

Email Password

Email Security
We don't store your login, your password is submitted securely, and we don't email without your permission.
Find your friends

SlideShare is more fun when your friends are on it
Use the form below to sync up with your friends, or invite them to join it.

Import email addresses
(from Gmail, Yahoo, Hotmail, and AOL address books)

Email Id: [blank]
Password: [blank]

Import Friends

*SlideShare respects your privacy. We will not store your username and password, or send any unsolicited emails to your contacts.
Find contacts that are already on LinkedIn and invite them to your professional network.

**Find Web email contacts**
Check your address book to find contacts who are on LinkedIn. We will only invite the ones you select.

- Windows Live Hotmail
- Gmail
- Other
- Yahoo!
- AOL

**Username:** luca.mearelli@gmail.com @hotmail.com

**Password:** ....

**Address book contacts**
Outlook, Apple Mail, etc.

**Current & past colleagues**

**Former classmates**

Your privacy is our top concern.
Your contacts are your private information. Only you have access to your contacts, and LinkedIn will not send them any email. For more information please see the LinkedIn Privacy Statement.
<table>
<thead>
<tr>
<th>Email Address</th>
<th>Account</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>njestates</td>
<td>Account</td>
<td>Username: njestates Password:</td>
<td></td>
</tr>
<tr>
<td>leodias2007</td>
<td>Account</td>
<td>Username: leodias2007 Password:</td>
<td></td>
</tr>
<tr>
<td>strongholdtextiles</td>
<td>Account</td>
<td>Username: strongholdtextiles Password:</td>
<td></td>
</tr>
<tr>
<td>abhinavgupta85</td>
<td>Account</td>
<td>Username: abhinavgupta85 Password:</td>
<td></td>
</tr>
<tr>
<td>christianbelleau</td>
<td>Account</td>
<td>Username: christian.belleau Password:</td>
<td></td>
</tr>
<tr>
<td>markelliott4</td>
<td>Account</td>
<td>Username: markelliott4 Password:</td>
<td></td>
</tr>
<tr>
<td>zygelis</td>
<td>Account</td>
<td>Username: zygelis Password:</td>
<td></td>
</tr>
<tr>
<td>strongholdtextiles</td>
<td>Account</td>
<td>Username: strongholdtextiles Password:</td>
<td></td>
</tr>
<tr>
<td>oswaldogrimaldi</td>
<td>Account</td>
<td>Username: oswaldogrimaldi Password:</td>
<td></td>
</tr>
<tr>
<td>igslog</td>
<td>Account</td>
<td>Username: igslog Password:</td>
<td></td>
</tr>
</tbody>
</table>
Passwords are precious
Stop asking them
Stop the antipattern
How to delegate access?
Your valet key for the web
a play in 3 acts
(to exchange authorization)
Actors on the scene

User
Consumer
Service Provider
Prologue

Where the Consumer presents himself to the Service Provider
Consumer (to Service Provider): here i am, this is what i do
Consumer key
Consumer secret

consumer → service provider
## Register New Application

Register a new application for the Pownce API. You may update this information at any time.

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Application Type</th>
<th>Application URL</th>
<th>Developer Email</th>
<th>Support Email</th>
<th>Description</th>
<th>Display Name (optional)</th>
<th>Version (optional)</th>
<th>Callback URL (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><a href="http://mypownceapp.com/">http://mypownceapp.com/</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
First act
Where the Consumer obtains an unauthorized Request Token
Consumer (to Service Provider):
give me a request token
consumer → service provider

oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
[additional parameters]
Service Provider (to consumer):
here is the request token
(you can use it only once!)
oauth_token (request token)
oauth_token_secret
[additional parameters]
Integrate your Fire Eagle account

"Fire Eagle is the secure and stylish way to share your location with sites and services online."

What does this mean? After you connect your Dopplr account with your Fire Eagle account, we'll send a location update to Fire Eagle on any day that you're travelling away from your home city. You can choose to share that location with other services that use Fire Eagle, and revoke that permission at any time. You can revoke Dopplr's authorisation to update your location at any time from your Fire Eagle account page.

You can read more about Fire Eagle and sign up for an account at the Fire Eagle website.

Visit Fire Eagle and authorise
Second act
Where the User authorizes the Request Token
Consumer (to the User):
Please go to the Service Provider and authorize this request
`oauth_token` *(request token)*

`oauth_callback`

[additional parameters]
Service Provider (to the User):
Do you **authorize consumer** to **access your data**?
An application would like to connect to your account

Dopplr would like to access & update your location on Fire Eagle. If you agree, Dopplr will be able to access & update your location until you tell us otherwise...

Dopplr

Allow them to access: my exact location

Allow them to update my location?  ○ Yes  ○ No

Confirm  Cancel

We’ll never share your password with anyone. If you change your mind, you can change these preferences at any time at http://fireeagle.yahoo.net. If you don’t know why Dopplr is asking for your permission, or how you got here, you should click “Cancel”. We take your privacy seriously, learn more.
User (to the Service Provider): **YES!**
(or maybe NO :-) )
Service Provider (to the User): You can go back to the consumer
service provider → user → consumer

oauth_token (request token)
Third act
Where the Consumer exchanges the Request Token for an Access Token
Consumer (to the Service Provider): Please give me the **access token** for the user.
oauth_consumer_key
oauth_token  (request token)
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version  (optional)
Service Provider (to the Consumer): here is the **access token** for the user
oauth_token (access token)
oauth_token_secret
[additional parameters]
Dopplr integrated your Fire Eagle account

Now your location will be updated on Fire Eagle once a day when you're travelling, according to your trip schedule.

The Fire Eagle last spotted you 3 minutes ago in Berlin, Deutschland.

Any time you like, you can have us forget your Fire Eagle connection.
Epilogue
Where the consumer accesses the resources
Consumer (to the Service Provider): Here i am again on behalf of the user
oauth_consumer_key
oauth_token (access token)
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
[additional parameters]
The details...
OAuth parameters exchange

HTTP Authorization header
HTTP Post body
URL query parameters
Request signing algorithm

plaintext

HMAC-SHA1

RSA-SHA1
Signature base string

a consistent reproducible concatenation of
the request elements into a single string
Other security measures

Nonces

Timestamps
Service setup

OAuth discovery (XRDS)
Benefits

Granular authorization
Easy grant and revoke
Tracking of use
Many open/free libraries use them, contribute to them
Challenges
UI/UX for the Service Provider
Provide basic informations to the user
Ease the user’s choice
Link / enable getting deeper info
UI/UX for the **Consumer**
Explain what’s happening
Educate the user
Use the right language
Security considerations

Confidentiality of Requests
Spoofing, Proxying, Phishing
Secrecy of credentials
Cryptographic issues
Denial of Service / Resource Exhaustion
Beyond the browser
Mobile devices
Installable applications
The sequel

OAuth over XMPP
<iq from='travelbot@findmenow.tld/bot' id='sub1' to='feeds.worldgps.tld' type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe node='bard_geoloc'/>
    <oauth xmlns='urn:xmpp:tmp:oauth'>
      <oauth_consumer_key>0685bd9184jfhq22</oauth_consumer_key>
      <oauth_nonce>4572616e48616d6d6572c61686176</oauth_nonce>
      <oauth_signature>wOJIO9A2W5mFwDgiDvZbTSMK%2FPY%3D</oauth_signature>
      <oauth_signature_method>HMAC-SHA1</oauth_signature_method>
      <oauth_timestamp>1218137833</oauth_timestamp>
      <oauth_token>ad180jjd733klru7</oauth_token>
      <oauth_version>1.0</oauth_version>
    </oauth>
  </pubsub>
</iq>
The morale

Integrating services can be done without asking or storing the user’s credentials while at the same time gaining flexibility and control.
Links!

http://oauth.net
http://oauth.net/core/1.0/
http://code.google.com/p/oauth/
http://groups.google.com/group/oauth/
http://oauth.net/discovery/1.0
Thanks for listening!!

Luca Mearelli

http://spazidigitali.com
l.mearelli@spazidigitali.com