

OAuth 1.0 Versus OAuth 2.0 and Use Case

Cyrille Fauvel
Philippe Leefsma
Autodesk Developer Network

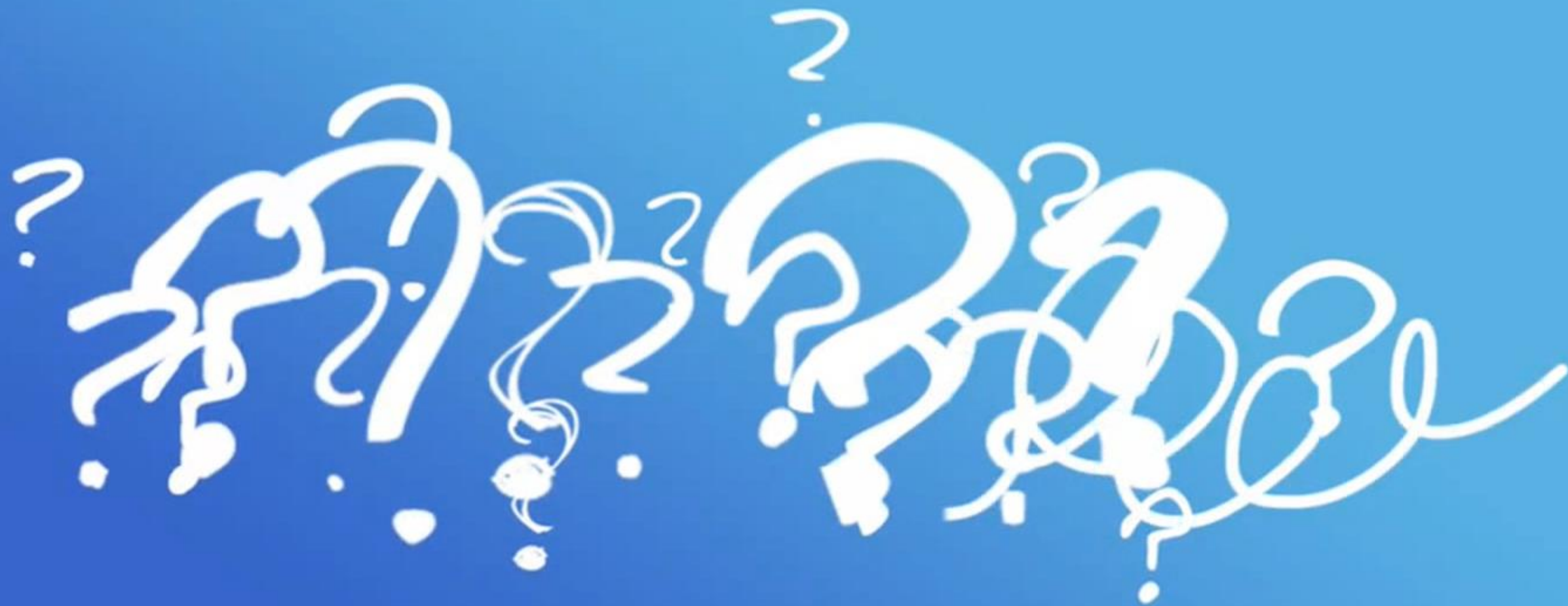
Class summary

Many services such as Facebook, GitHub, and Google have already deployed OAuth 2.0 servers. The OAuth 2.0 spec leaves many decisions up to the implementer. Instead of describing the possible decisions that need to be made to successfully implement OAuth 2.0, this lecture will explain most of the appropriate decisions to make for most implementations. This lecture is an attempt to explain OAuth 2.0 in a simplified format to help developers and service providers implement the protocol. Attendees will also discover how they can use other people's OAuth servers instead of implementing their own.

Key learning objectives

At the end of this class, you will be able to:

- Understand differences between OAuth versions
- Understand requirements for implementing your own OAuth server
- Learn why you should use OAuth and how to use OAuth when exposing an API or using a third-party API
- Learn how to use third-party OAuth server in your infrastructure



- OAuth is an open standard for authorization. OAuth provides client applications a 'secure delegated access' to server resources on behalf of a resource owner.
 - Delegated Authorization
- <http://en.wikipedia.org/wiki/OAuth>



Rough Timeline of WEB technologies

- 2006 – Twitter OpenID
- 2007 – OAuth 1.0 ([RFC 5849](#))
- 2008 - IETF normalization started in 2008
- 2010 – WRAP (Web Resource Authorization Profiles)
proposed by Microsoft, Yahoo! And Google
- 2012 – OAuth 2.0 ([RFC 6749](#))
Bearer Token ([RFC 6750](#))

OAuth 1.0 vs. 2.0

- Not compatible
- OAuth 1.0 – OAuth 1.0a
 - http
 - Encryption and signature
- OAuth 2.0
 - https
 - No signature – relies on SSL



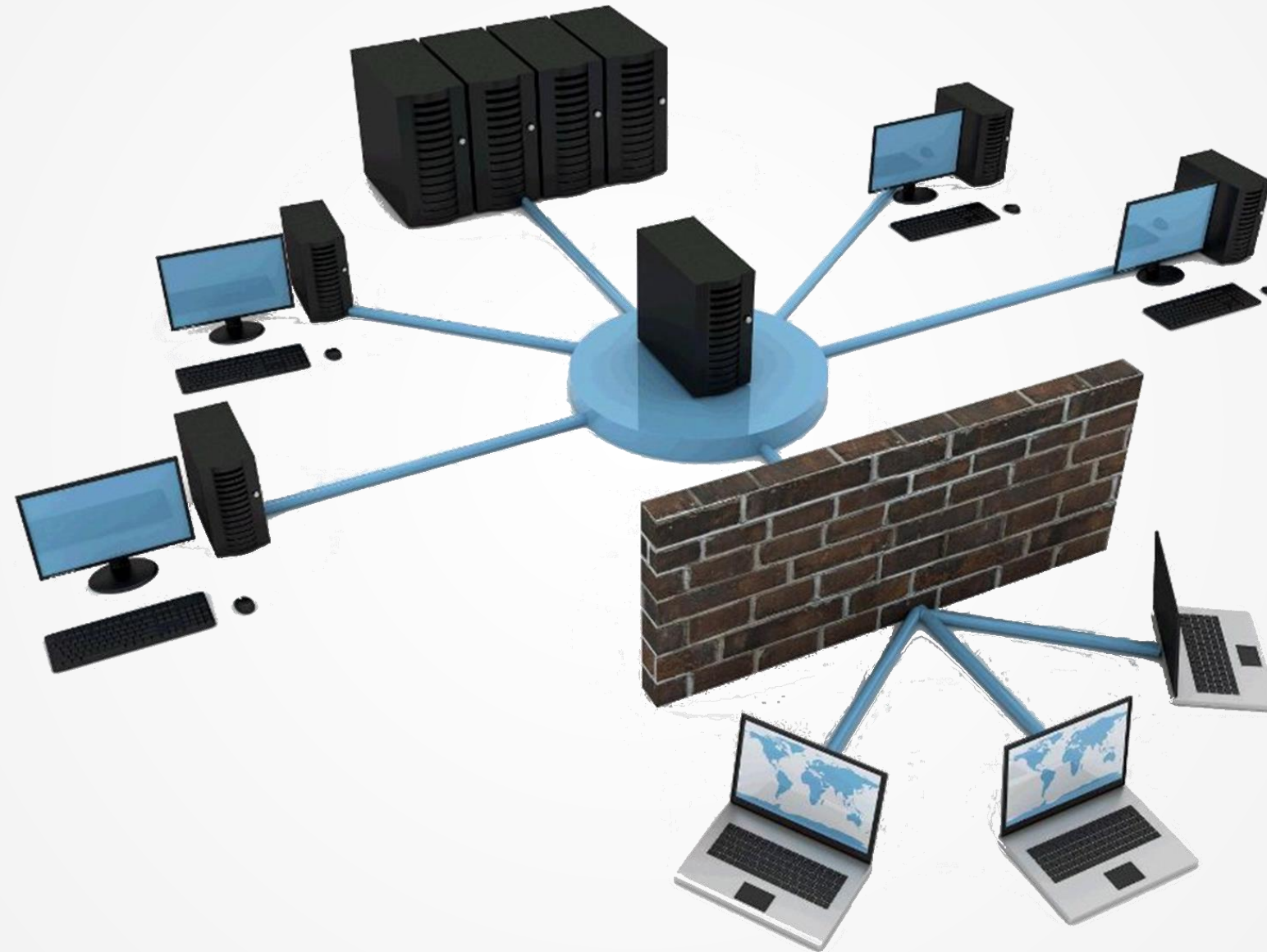
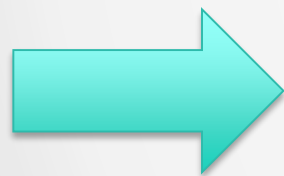
- The new security stack for modern applications
 - OAuth 1.0a
 - OAuth 2.0
- Security discussion
- Resources

The Security Stack for Modern Applications

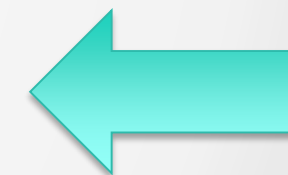
Enterprise Security

VPN

SOAP
WS*



XML
SAML



The mobile Revolution

No SOAP
No SAML
No WS*



HTTP
JSON

OAuth

What is OAuth

- OAuth is Authorization
- OAuth is not Authentication
- For Authentication take a look to OpenID Connect

Why do you need OAuth?

- If you control both side, you don't
 - Both side are: application, and back-end
 - What if the application is written by someone else?
 - How much trust can you put on the application accessing resources on your back-end
- (or the other way around - you need to access someone else back-end)

FAKING OUT PARENTS

- 1. Fake a Stomach Cramp**
- 2. Moan and Wail**
- 3. Lick Palms**







No Problem – Trust me



Still confused?



OAuth 1.0(a)

Simulation

- Jane Doe (resource owner)
Stores pictures on mypictures.com
- print.photos.net (client)
Prints photos for user and collaborate with mypictures.com
- mypictures.com (the resource server)
Exposes a protected API via OAuth



Simulation (1)



- Jane wants to print the photos stored on mypictures.com using print.photos.net services.

- ask print.photos.net to do the job



- print.photos.net needs to access the resources from mypictures.com server

- send a request for a temporary request token



- mypictures.com validates the client request and sends it a temporary request token

- return a request token

Simulation (2)



- print.photos.net redirects Jane to mypictures.com for login

- either redirect or give the preformatted URL to sign on



- mypictures.com requests Jane to sign in using her credentials that authenticate her with the server and asks her approval to grant permission to the client to access her resources.
 - log on mypictures.com using login/password to approve giving access to client to access her resources

Simulation (3)



- print.photos.net is informed when Jane finishes granting authorization to the client
 - via a callback or ...



- print.photos.net asks mypictures.com for an access token using its approved temporary request token
 - send a request for a definitive access token



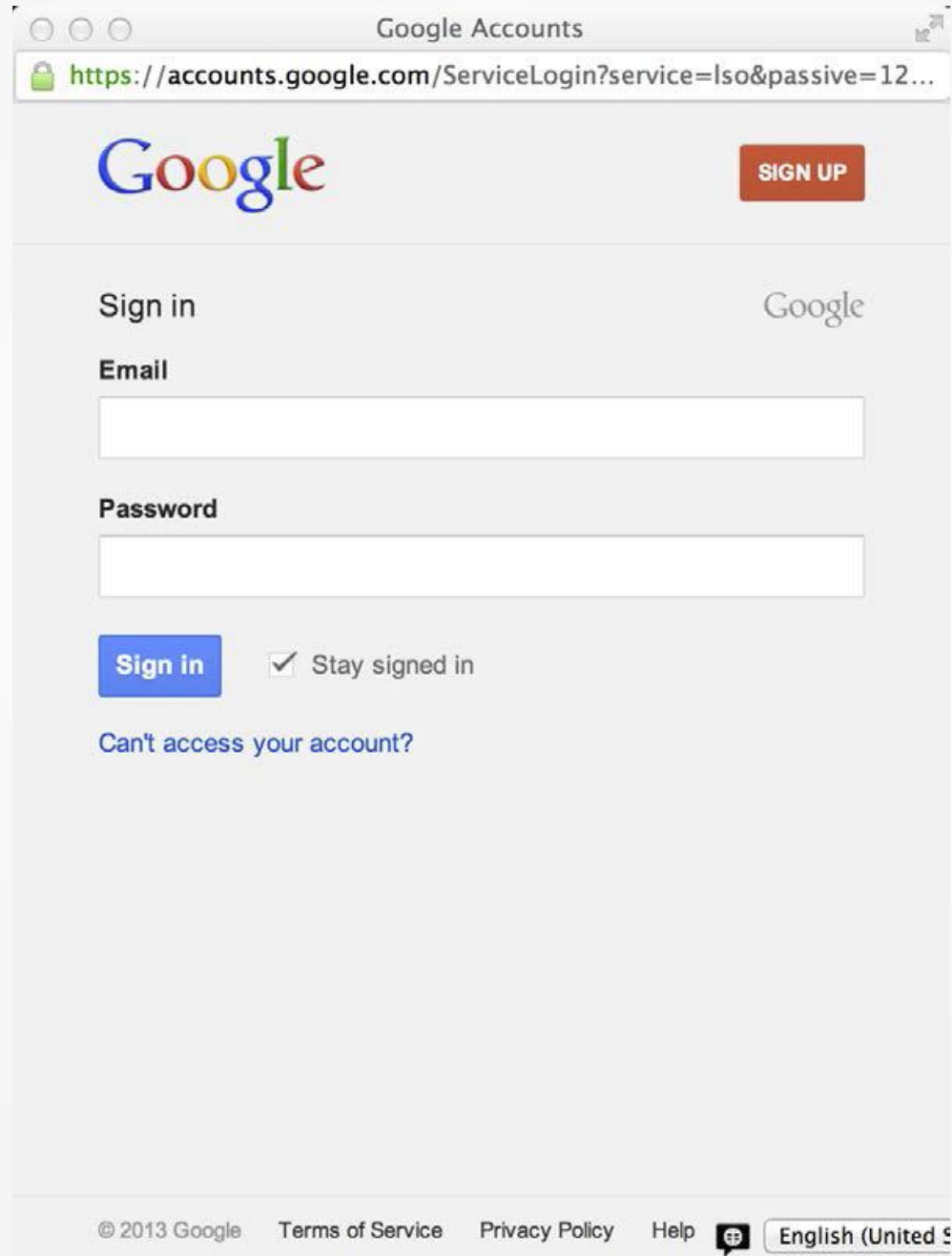
- mypictures.com validates the request and sends back an access token
 - return an access token

Simulation (4)



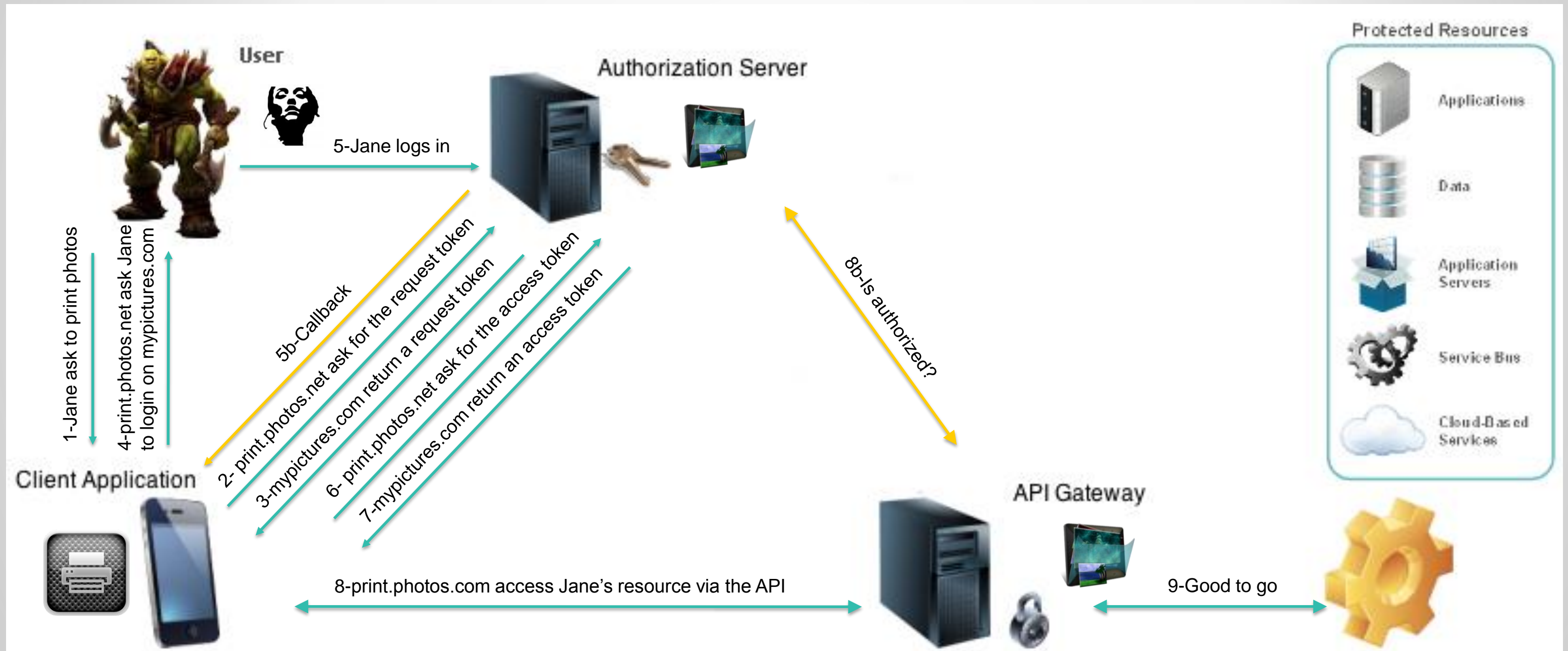
- print.photos.net now access Janes photos from the mypictures.com server with this access token
 - do its work without knowing Jane's credential

Authentication



A screenshot of a web browser window titled "Google Accounts". The address bar shows the URL <https://accounts.google.com/ServiceLogin?service=iso&passive=12...>. The page features the Google logo at the top left and a red "SIGN UP" button at the top right. Below the logo, the text "Sign in" is displayed, followed by the word "Email" and a text input field. Underneath the email field is the label "Password" and another text input field. A blue "Sign in" button is positioned below the password field, accompanied by a checked checkbox and the text "Stay signed in". A link that says "Can't access your account?" is located below the "Sign in" button. At the bottom of the page, there is a footer containing copyright information "© 2013 Google", links for "Terms of Service", "Privacy Policy", and "Help", a language selector showing "English (United S", and a small chat icon.

Summary



OAuth 2.0

Top Differences between OAuth 1.0 and OAuth 2.0

- SSL is required for all the communications required to generate the token. This is a huge decrease in complexity because those complex signatures are no longer required.
- Signatures are not required for the actual API calls once the token has been generated - SSL is also strongly recommended here.

Top Differences between OAuth 1.0 and OAuth 2.0

- Once the token was generated, OAuth 1.0 required that the client use/send two security tokens on every API call. OAuth 2.0 has only one security token, and no signature is required.
- It is clearly specified which parts of the protocol are implemented by the "resource owner," which is the actual server that implements the API, and which parts may be implemented by a separate "authorization server."

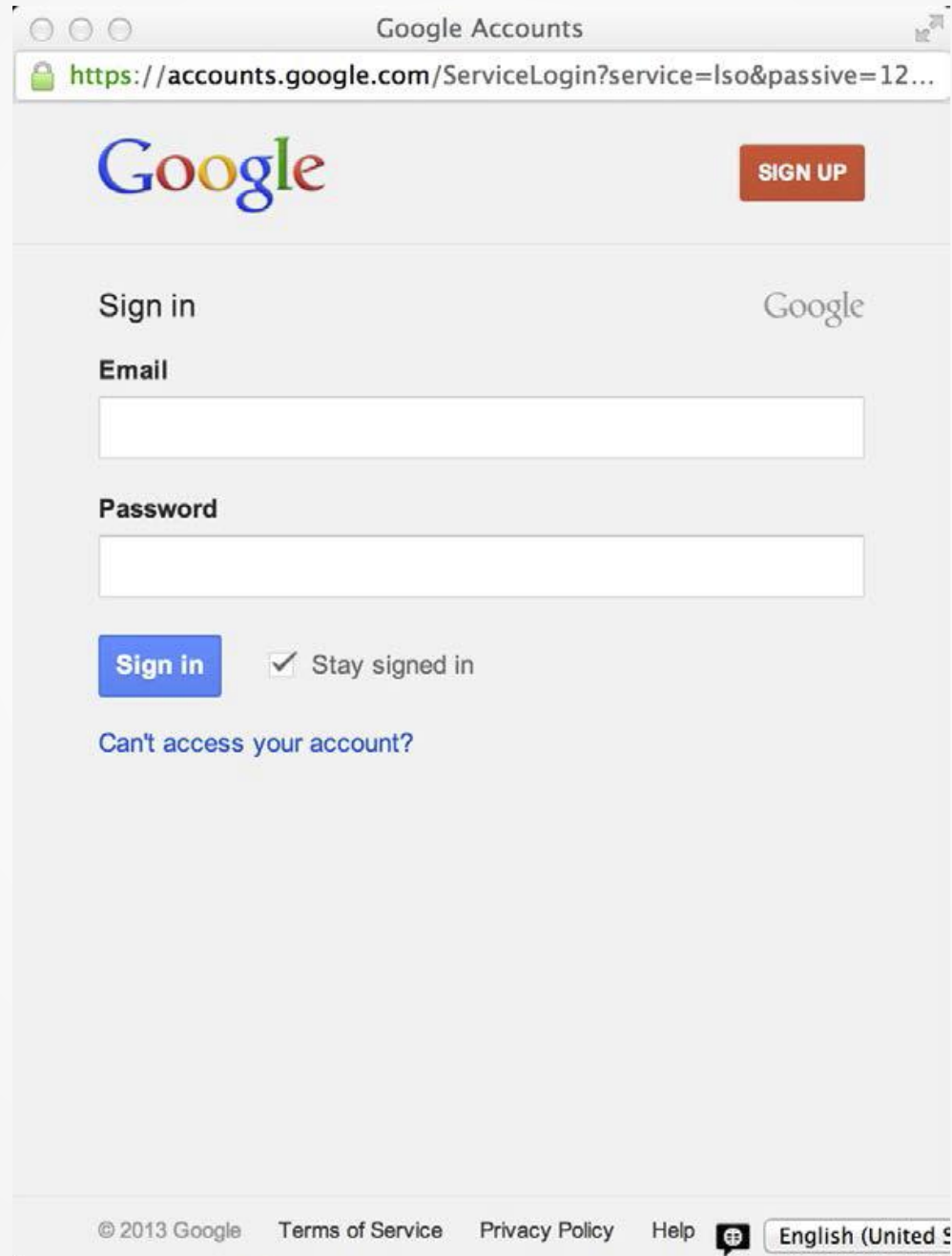
OAuth 2.0 Flows with User Interaction

- Authorization Code Flow
 - Web application clients
 - Request authorization
 - Request token
 - Access resource
- Implicit Flow
 - Native / local clients
 - Request authorization & token
 - Access resource

OAuth 2.0 Flows with no User Interaction

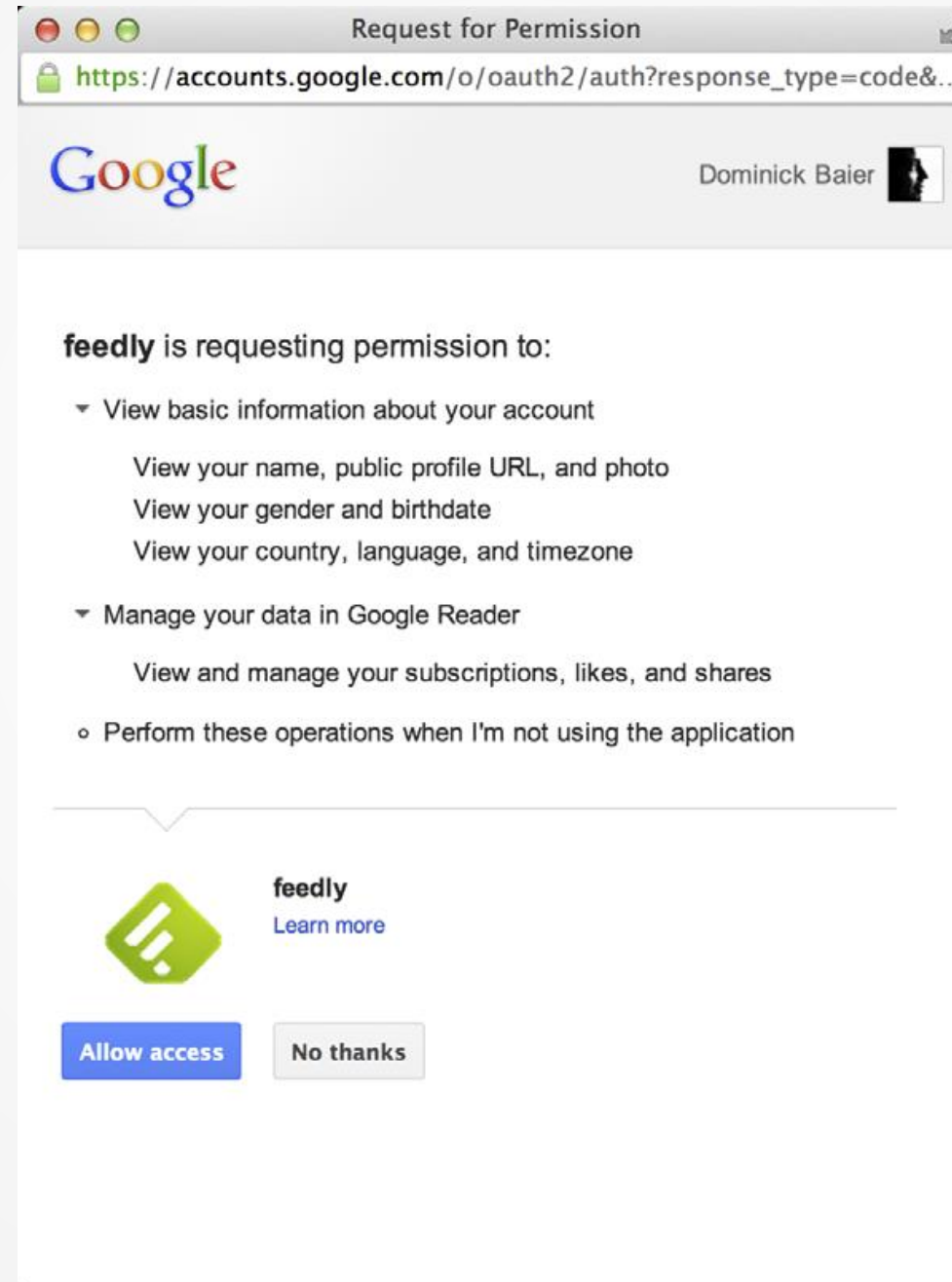
- Resource Owner Password Credential Flow
 - "Trusted clients"
 - Request token with resource owner credentials
 - Access resource
- Client Credential Flow
 - Client to Service communication
 - Request token with client credentials
 - Access resource

Authentication



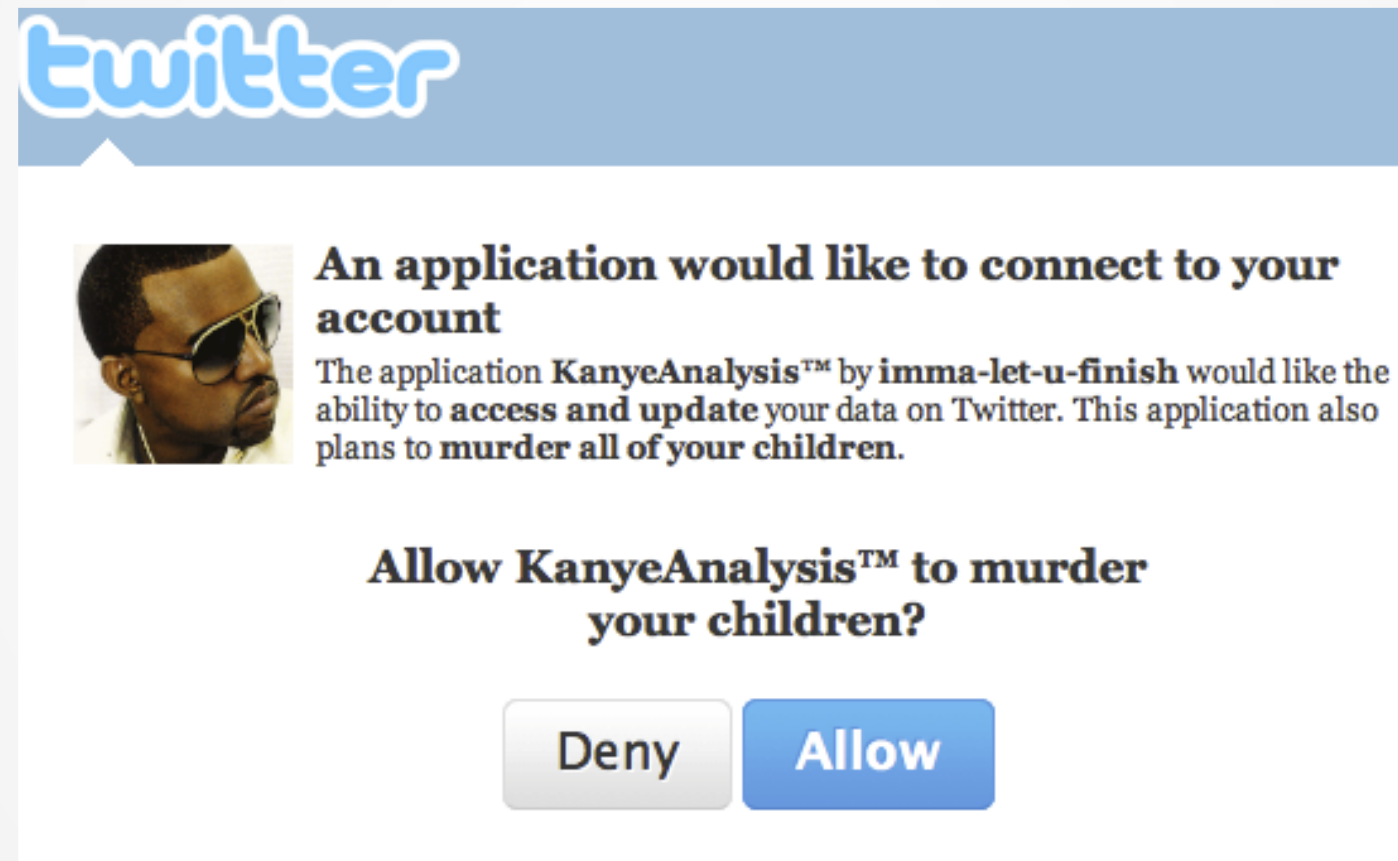
A screenshot of a web browser window titled "Google Accounts". The address bar shows the URL <https://accounts.google.com/ServiceLogin?service=iso&passive=12...>. The page features the Google logo at the top left and a red "SIGN UP" button at the top right. Below the logo, the text "Sign in" is displayed. To the right of "Sign in" is the word "Google". Underneath, there are two input fields: "Email" and "Password". Below the "Password" field is a blue "Sign in" button and a checkbox labeled "Stay signed in" which is checked. Below the checkbox is a link that says "Can't access your account?". At the bottom of the page, there is a footer with copyright information "© 2013 Google", links for "Terms of Service", "Privacy Policy", and "Help", and a language selector set to "English (United S)".

Consent

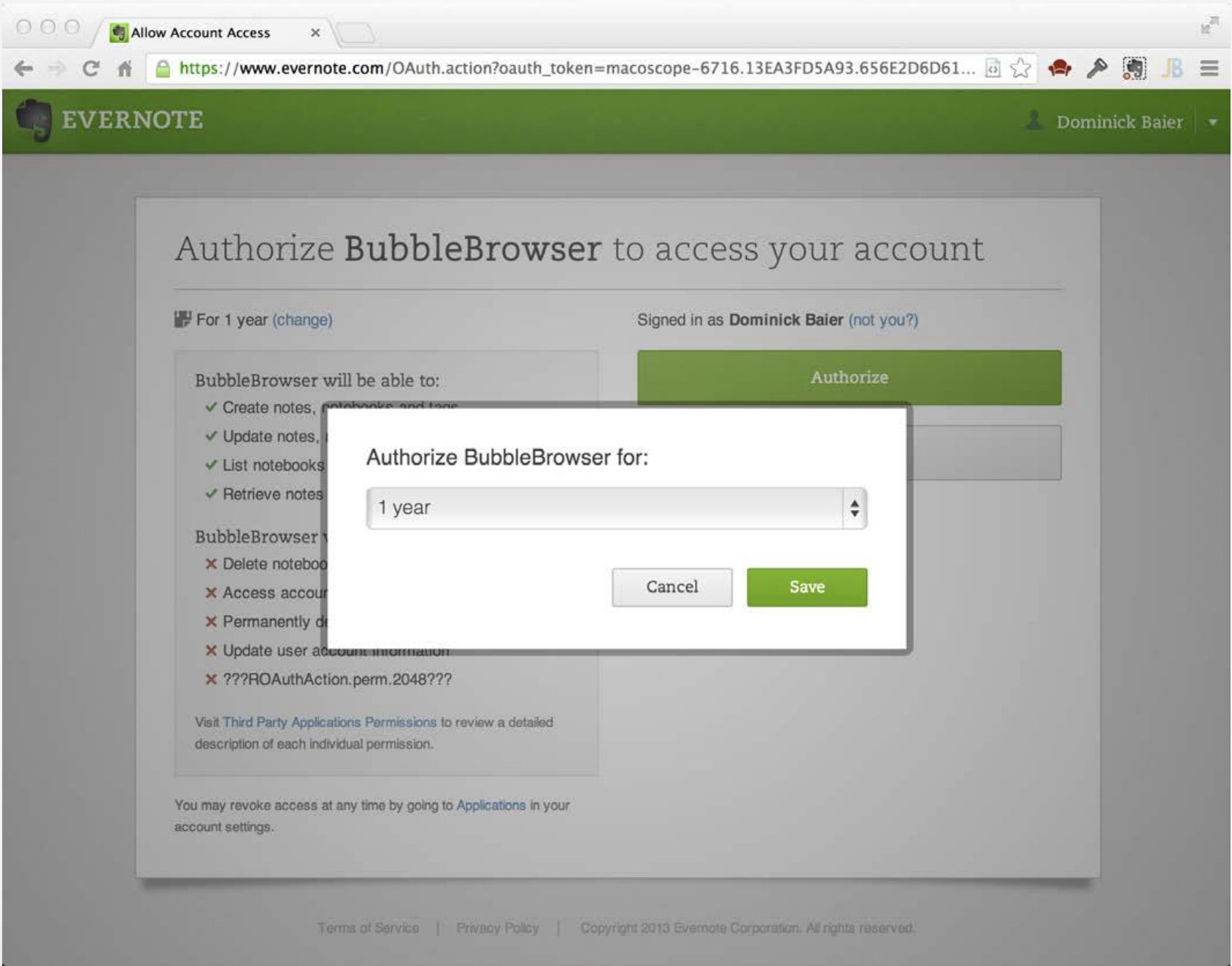


Consent

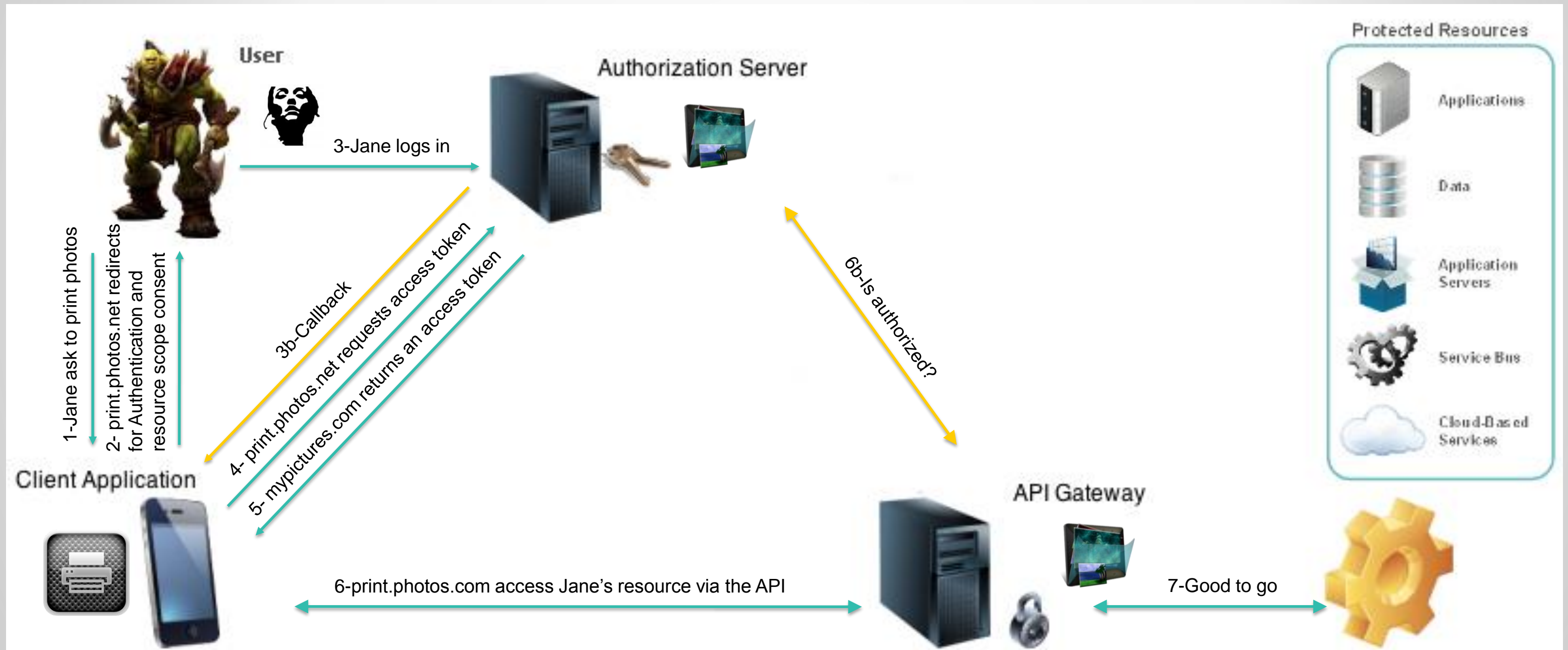
The Consent Screen is important!



Consent

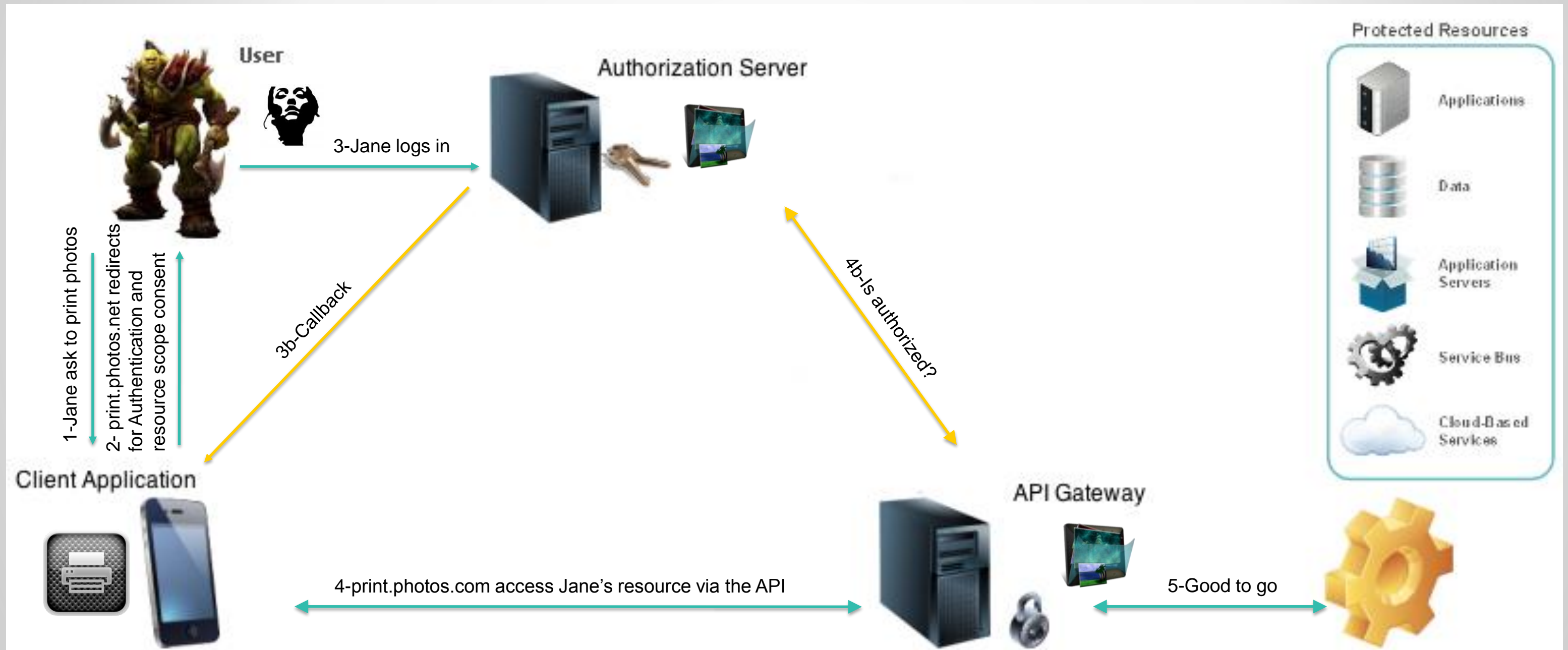


Authorization Code Flow (WEB Application Clients)

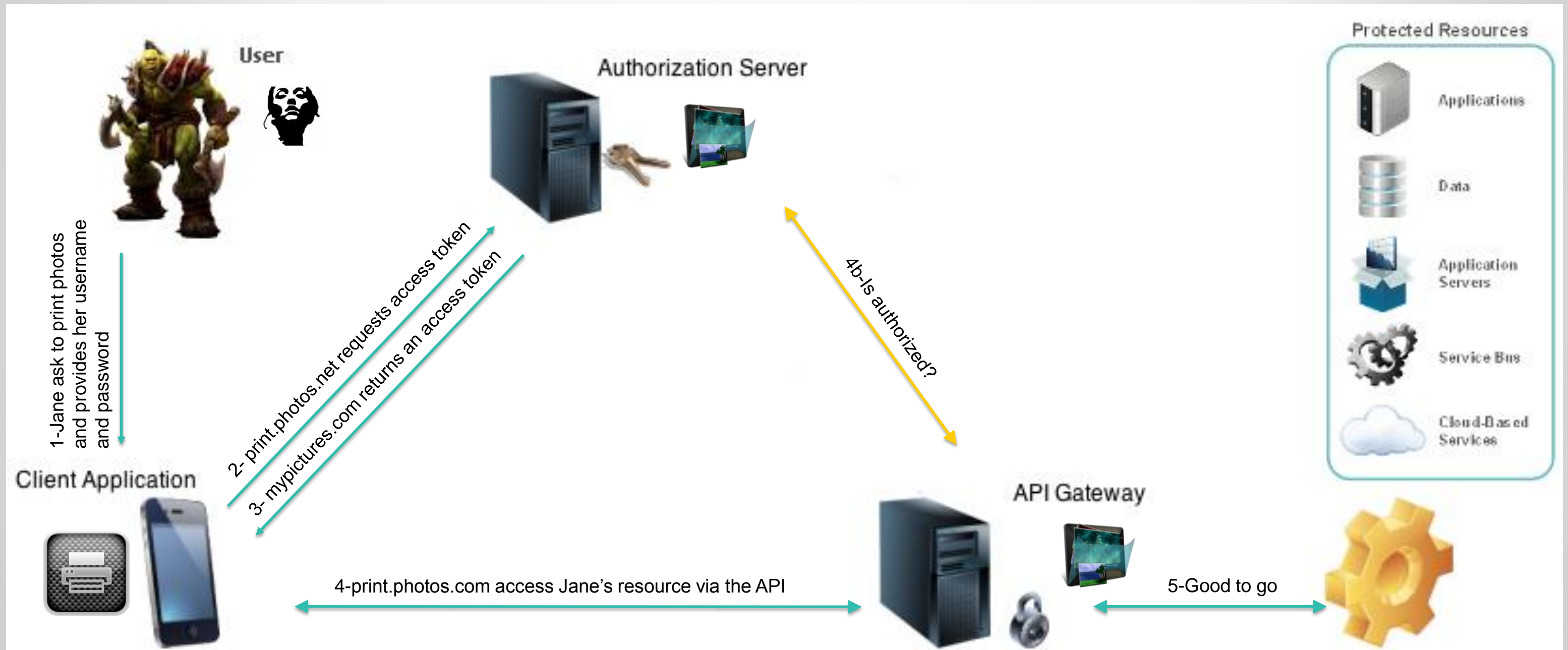


Authorization Code Flow

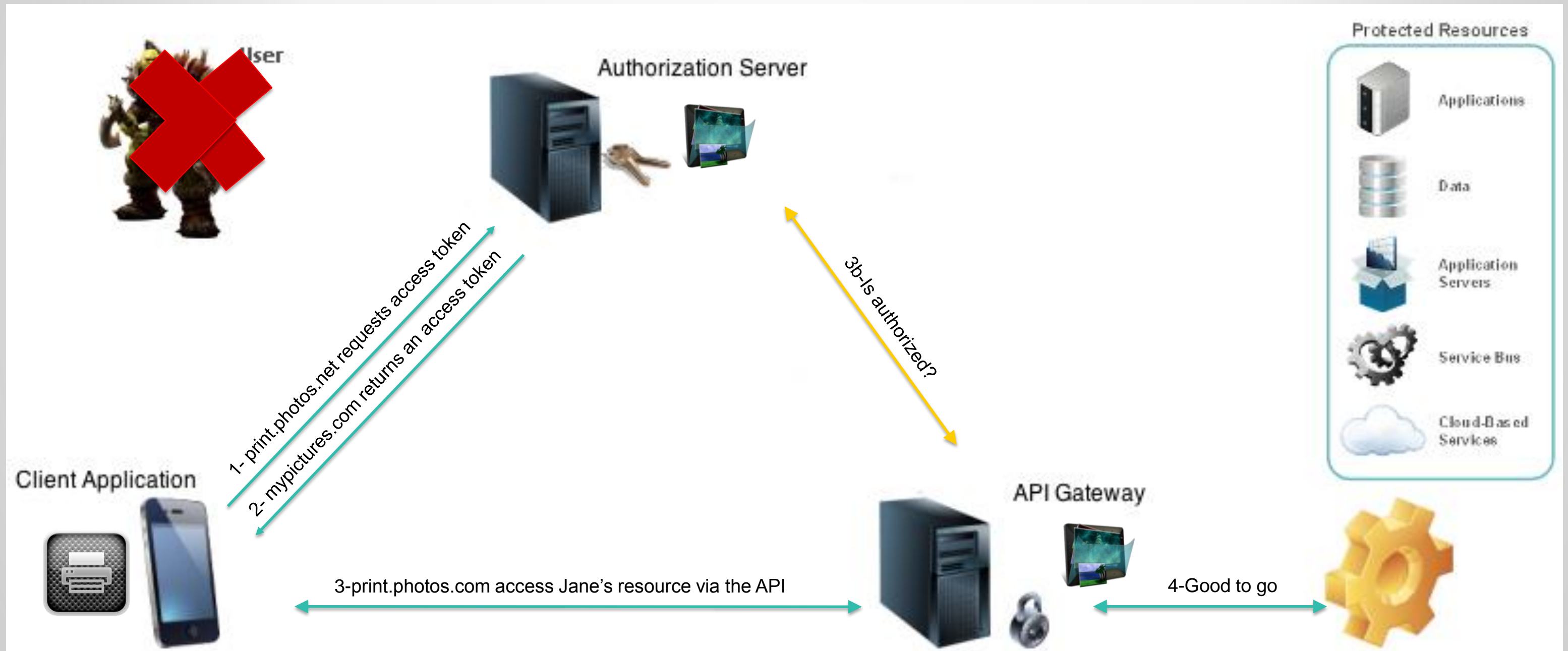
Implicit Flow (Native / Local Clients)



Resource Owner Password Credential Flow (Trusted Application)



Client Credentials Flow (No human involved at all)



Security discussion

Eran Hammer

- <http://hueniverse.com/2010/09/oauth-bearer-tokens-are-a-terrible-idea/>
- <http://hueniverse.com/2010/09/oauth-2-0-without-signatures-is-bad-for-the-web/>
- <http://hueniverse.com/2012/07/oauth-2-0-and-the-road-to-hell/>
- **OAuth2: Looking back and moving on**
 - <https://vimeo.com/52882780>

Autodesk oAuth implementation

Autodesk OAuth implementation

- Use OAuth 1.0a
- No username/password support
- <https://account.autodesk.com/>
- <https://account-staging.autodesk.com/>
- ReCap Photo API

Autodesk OAuth implementation

- Use OAuth 2.0
- No username/password support
- <https://developer.api.autodesk.com>
- Autodesk 360 Viewer, AutoCAD i/o

Examples & Demos

Resources

Additional Resources

- Autodesk ADN Samples for clients
 - <https://github.com/ADN-DevTech>
 - <https://github.com/ADN-DevTech/AutodeskOAuthSamples>
 - <https://github.com/ADN-DevTech/Autodesk-ReCap-Samples>
 - <https://github.com/developer-autodesk>
 - <https://github.com/Developer-Autodesk/autodesk-view-and-data-api-samples>
 - <https://github.com/Developer-Autodesk/AutoCAD.io>

Additional Resources

- Thinktecture.IdentityModel
 - <https://github.com/thinktecture/Thinktecture.IdentityModel.45>
- Thinktecture.IdentityServer
 - <https://github.com/thinktecture/Thinktecture.IdentityServer.v2>
- DotNetOpenAuth
 - <http://dotnetopenauth.net/>

Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2015 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time





