SPA Conference 2014

Keeping Passwords Private with OAuth

Nick Rozanski
nick@rozanski.org.uk
http://www.nick.rozanski.org.uk

Eoin Woods
eoin@copse.org.uk
http://www.eoinwoods.info

Chris Cooper-Bland
Chris.Cooper-Bland@endava.com
http://www.endava.com/

Agenda

09:15 - 09:20	Presenter Introduction
09:20 - 09:30	Problem Statement
09:30 – 10:00	OAuth Overview
10:00 – 10:15	Environment Setup and Break
10:15 –11:00	Exercise 1: Granting Access to Dropbox Programmatically
11:00 – 11:15	Break
11:15 – 11:45	Exercise 2: Using the Dropbox API to Read and Write Files
11:45 – 12:00	Discussion: What Do We Think of OAuth?

Presenters

Nick Rozanski

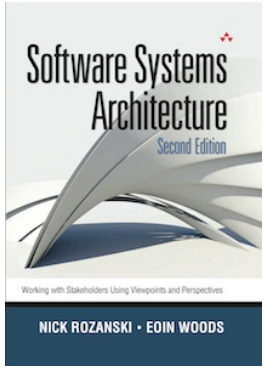
- Architect for Rates IT and Control at Barclays Investment Bank
- 30 years in IT, been working as an architect for about 12 years
- Last wrote code for a living about 15 years ago…

Eoin Woods

- Software architect for securities processing systems at UBS, and head of software engineering for the Operations IT group
- Worked in software engineering since 1990 for companies including Ford, Group Bull, Sybase, InterTrust and BGI

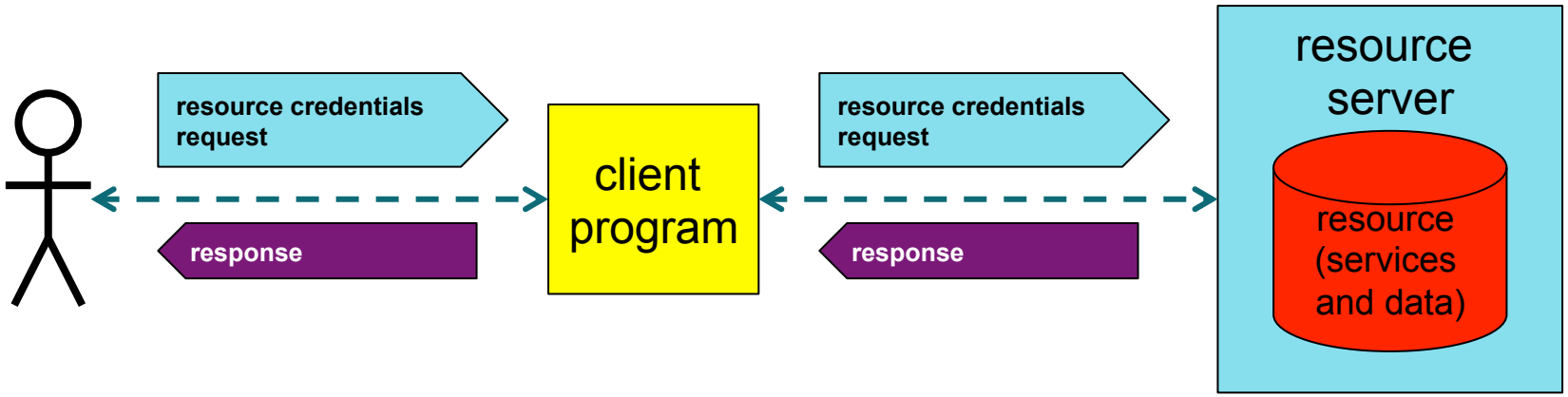
Chris Cooper-Bland

- Head of Architecture and Analysis (UK) at Endava

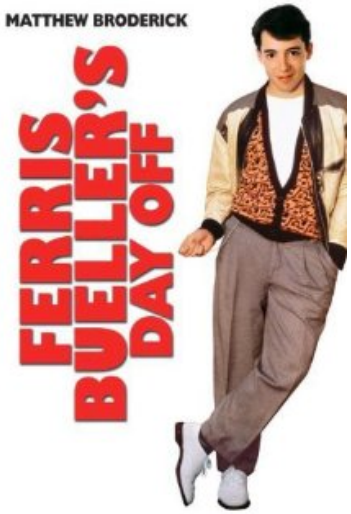


Problem Statement

- There are many third-party programs which access cloud-based services like Dropbox, Amazon or Facebook
  - These cloud-based services are almost always secured using usernames and passwords
- However trusting such a program with your user credentials is risky
  - You have no way of knowing how the program will keep them secure
  - You have no way of preventing the program from using them maliciously
- This problem is known as *cross-domain authentication*



A More Interesting Problem Statement

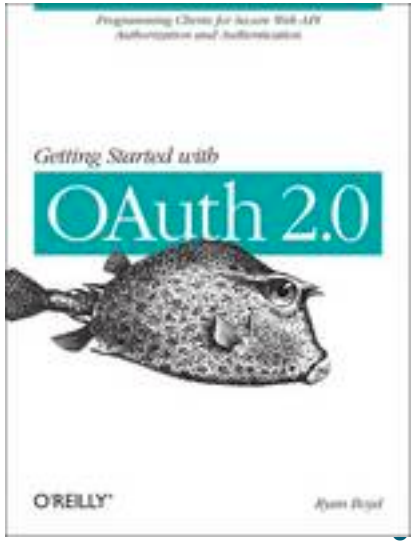


"In the movie Ferris Bueller's Day Off, a valet attendant takes a fully restored 1961 Ferrari out for a joyride.

How do you prevent the same thing from happening to your brand-new Mustang?

Some cars now come with special keys that allow the owner to provide limited authorization to valet attendants (or kids!) and prevent activities such as opening the trunk and driving at excessive speeds.

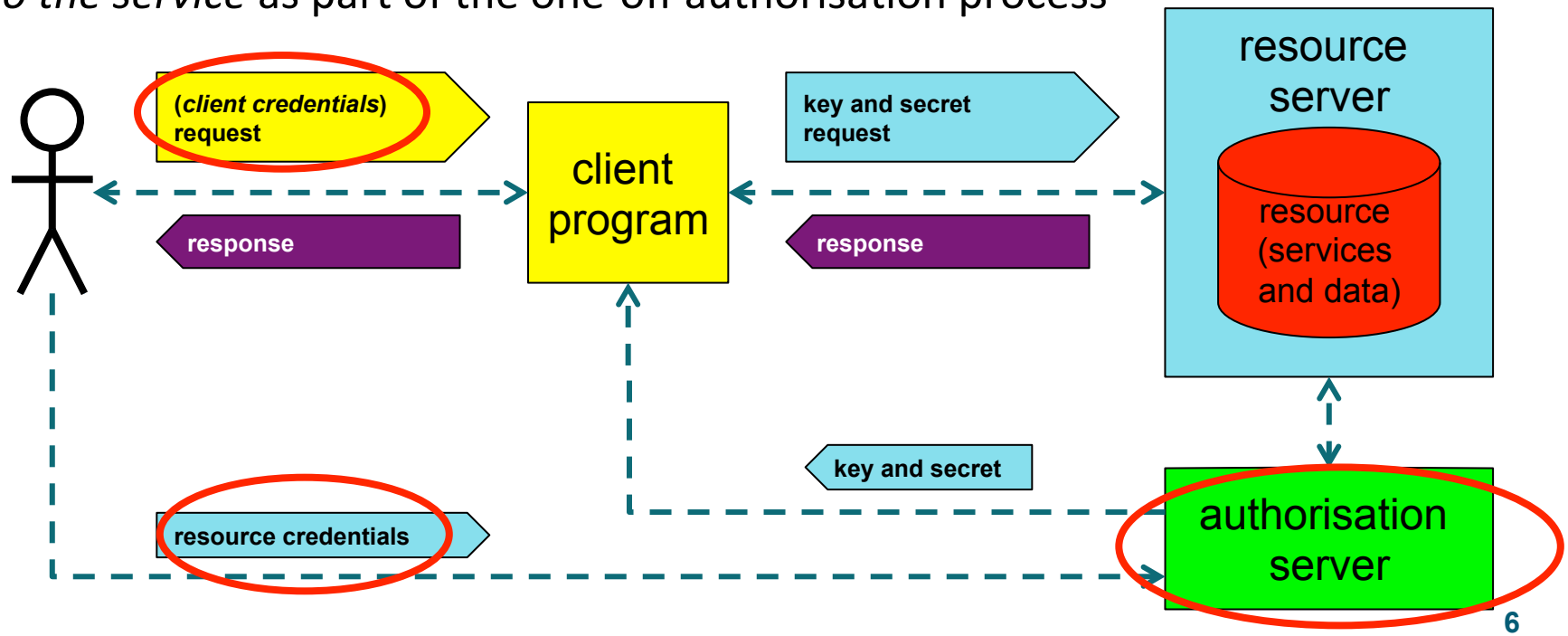
OAuth was created to solve the same core issue online."



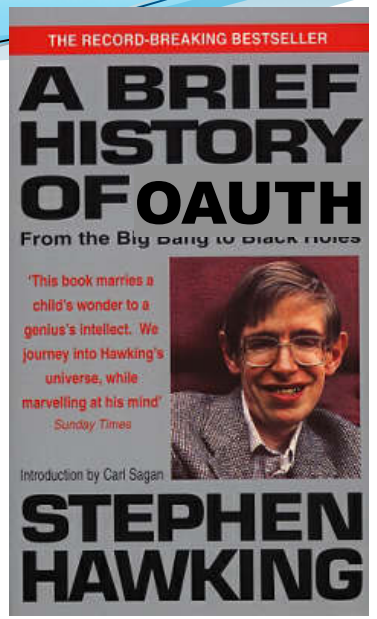
Getting Started with OAuth 2.0,
Ryan Boyd, O'Reilly Books 2012

A Solution: OAuth

- One way of solving the cross-domain authentication problem is to use an open standard called **OAuth**
- A client which implements the OAuth protocol uses a specially-negotiated *key and secret* to gain access to secured services or data
- The client program never sees the user's credentials, which are submitted *only to the service* as part of the one-off authorisation process



A Brief History of OAuth



- OAuth 1.0: 2007-2010
  - development co-ordinated by Eran Hammer
  - Twitter was an early adopter (mandated for all Twitter clients since 2010)
- OAuth 2.0 2012:
  - new authorisation model, not backwards-compatible with OAuth 1.0
- Adoption
  - Wikipedia now lists about 70-80 service providers who support OAuth 1 or 2
  - from Amazon to Zendesk via Google, Facebook, Microsoft, Yahoo, Dropbox…
- Controversy and Criticism
  - Eran Hammer eventually resigned his role of lead author, withdrew from the IETF working group, and removed his name from the specification
  - He describes Oauth 2 as “a designed-by-committee patchwork of compromises”

OAuth Benefits (and Criticisms)

- ☑ **Lessens the need to trust the client**
  - the resource server can constrain access to data and services
- ☑ **Decouples resource access from password changes**
  - the key and secret do not need to change if the user changes their password
- ☑ **Allows the user to revoke access without having to change their password**
  - can revoke access by one client without affecting others
- ☑ **Helps avoid users becoming "desensitised" to phishing and password-harvesting**
  - Authorisation is done in the context of the service (eg on the Dropbox website) rather than of the client

- ☐ **OAuth is about authorisation, not identity**
  - possession = authorisation
  - anyone can use the valet keys
  - User can use OpenId to authenticate themselves with the resource server
- ☐ **OAuth 2 does not require signatures to identify endpoints**
  - easy for developer to accidentally send credentials to a malicious endpoint
  - partly mitigated by the use of SSL (client must validate endpoint)
  - this aspect of the standard is still being developed
- ☐ **OAuth 2 is "complicated!"**
  - easy for developers to get something wrong

Some OAuth Terminology

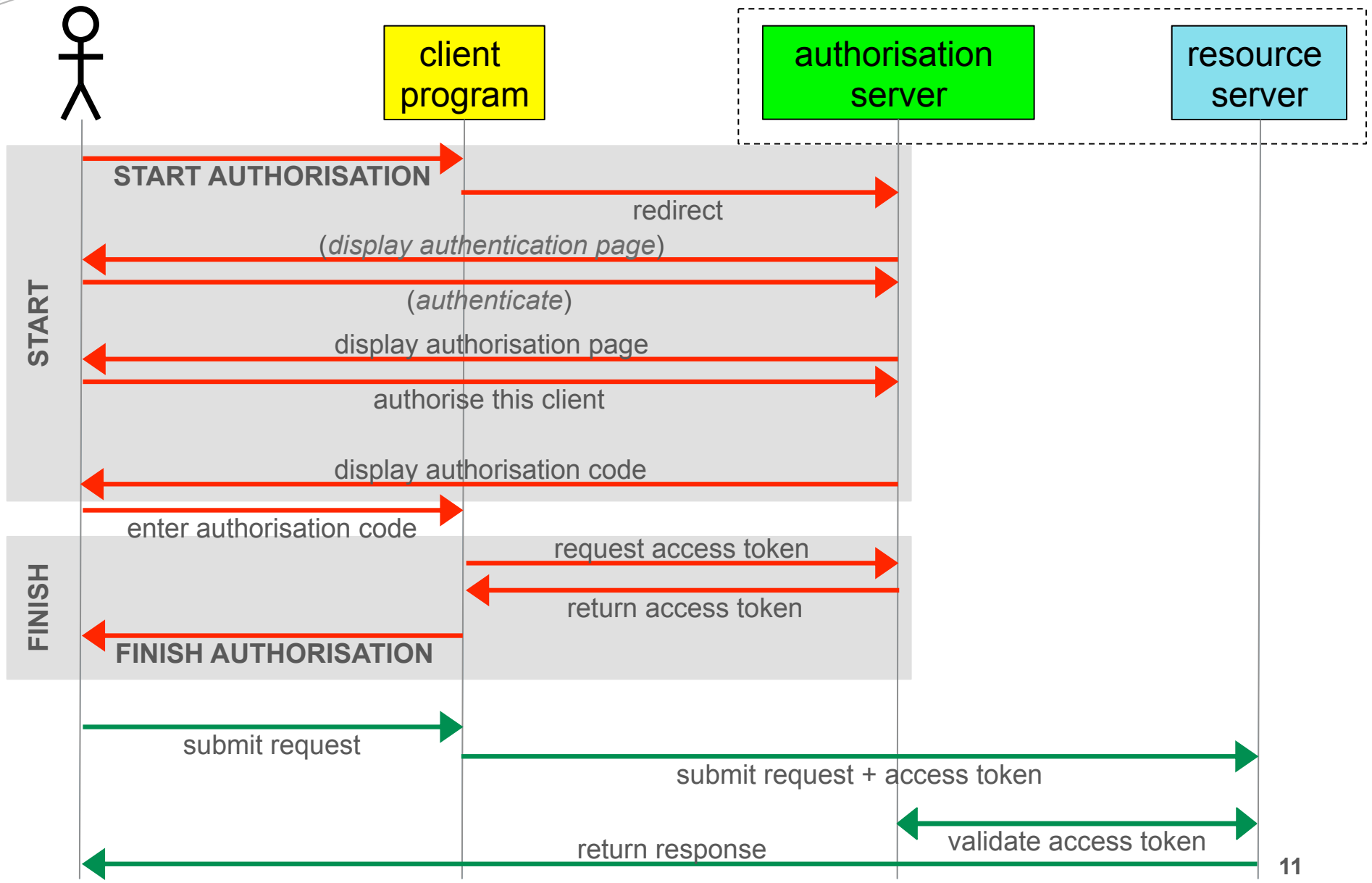
Term	Description	Example
Resource owner	Person who has authority to grant access to data and services	You
Protected resource	Data and / or services which the resource owner wants to keep secure	Dropbox files / Dropbox actions (eg upload file)
Resource server	The server which hosts the resources that are protected by OAuth	data server in the Dropbox data centre
Authorisation server	Server which performs resource authorisation actions, including: • receives access consent from user • issues access token to client • validates access tokens provided by clients	authorisation server in the Dropbox data centre
Client	A program which makes Oauth calls to perform actions on protected resources on behalf of the resource owner	Desktop program, mobile app or web server which stores its data in Dropbox
Authorisation workflow	A sequence of steps in which the resource owner authorises a client to access a protected resource	*See following slides*
Registration	The process whereby a developer sets up key attributes of their application and is given an access token in return	*See following slides*

OAuth Workflow

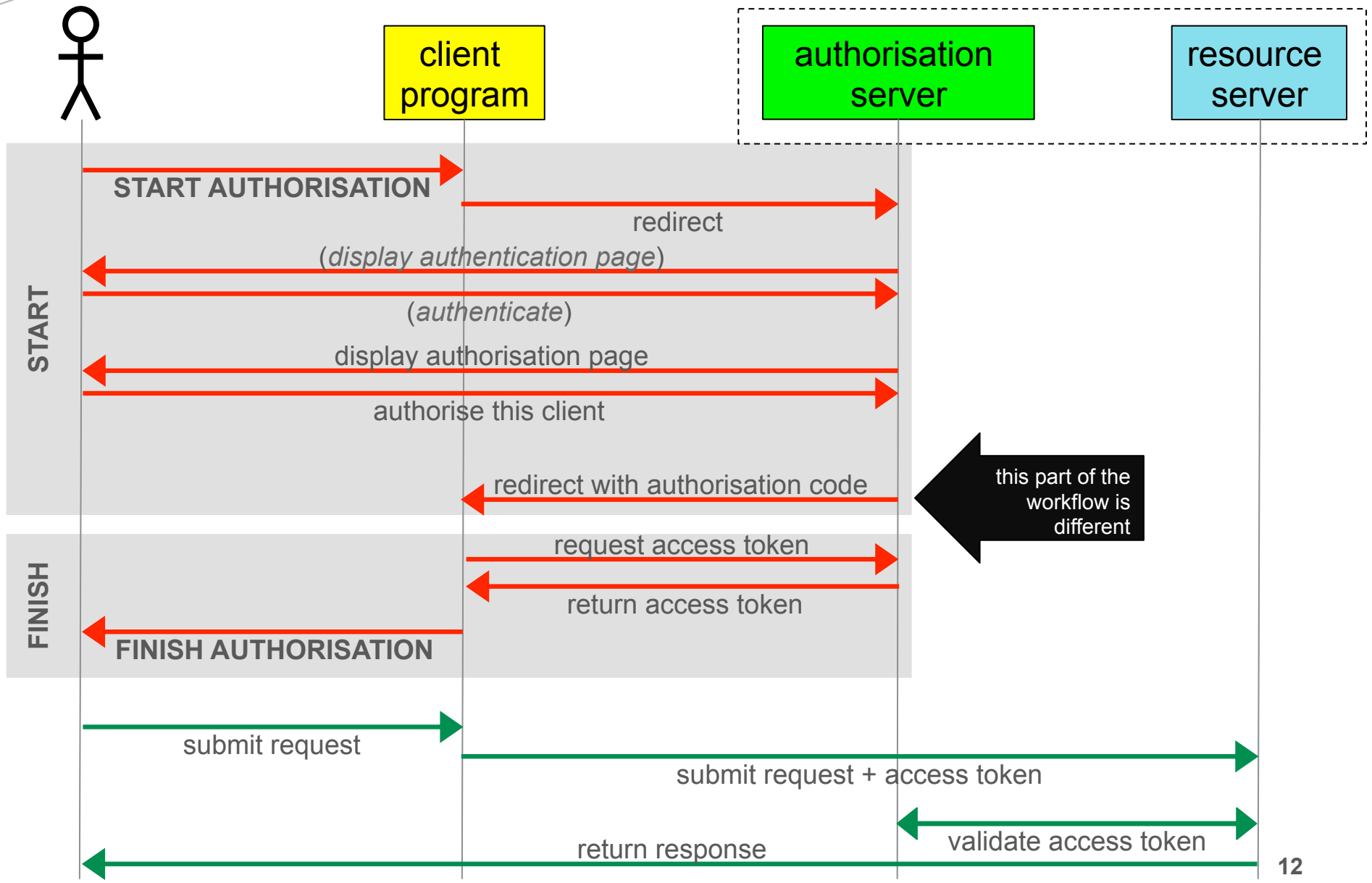
- The steps to authorise a client are referred to as **authorisation workflow**
- The OAuth standard defines four types of authorisation workflow
- We will be looking at one of these today ("authorization code" workflow)
  - The demo illustrates two flavours of this workflow, with or without a final redirect step

Workflow	Description	Used For
Authorization Code	After authorisation, the client is redirected back to a web URL (can be a local URL or URI scheme)	Server-side web applications Clients who can run a web server or use URI schemes
Implicit Grant	*Access token passed back in a #fragment in the URL*	*Client-side web applications running in a browser*
Password-Based Grant	*Client is given and forwards username and password*	*Highly-trusted client (eg provider's own mobile app)*
Client Credentials	*Authorisation is previously arranged*	*Where the client is accessing an API (such as a storage service or database) on behalf of itself rather than the user*

Authorization Code Workflow (No-Redirect)



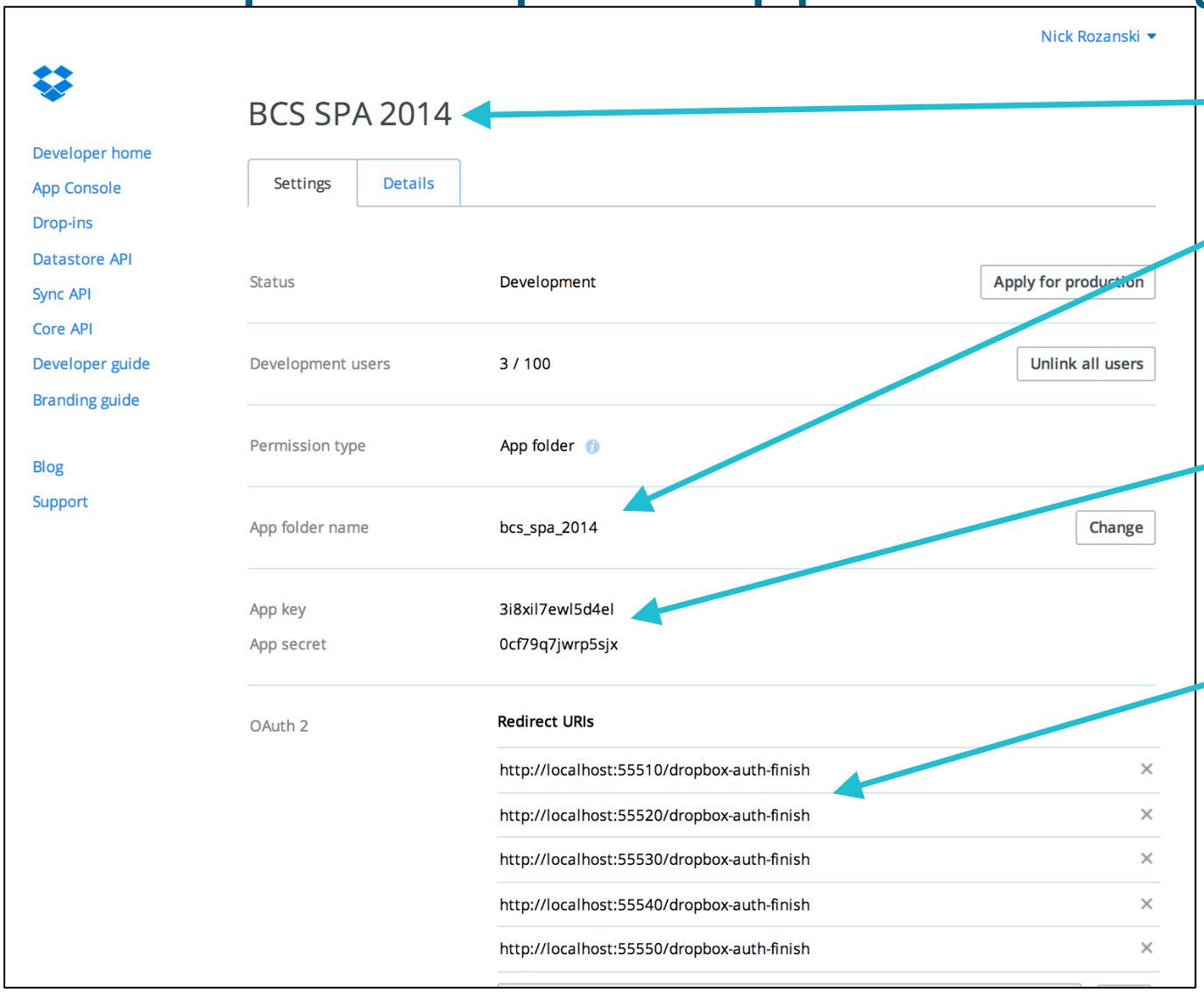
Authorization Code Workflow (Redirect)



Application Registration

- OAuth 2 requires that clients *register* with the authorisation server so that client requests can be properly identified
- Service providers normally provide some sort of web console to do this, eg:
  - Google API Console
  - Windows Line Application Management site
  - Facebook Developer site
  - Dropbox App Console
- Once registered, the client is issued with a *client id* and *client secret*
- During registration you have to specify one or more allowable *redirect URL*s or *URI*s (Uniform Resource Identifiers)
  - The client can only redirect to one of these URIs
- Registration helps ensure workflow requests are authentic, and enables the service provider to tailor its response according to the privileges granted to that client

Example: Dropbox Application Registration



The Dropbox app is called **BCS SPA 2014**

Application files are stored in the Dropbox folder **apps/bcs_spa_2014**

The **client key** and **secret** are allocated by Dropbox at registration time

The app owner enters one or more **redirect URIs** the client can only redirect to one of these in the "finish" step of authorisation

Exercise

- The demo has two parts:
  1. Authorise with Dropbox using OAuth
  2. Run various commands to display or manipulate Dropbox files (to demonstrate that authorisation was successful)
- Once you have authorised with Dropbox, the access token is saved to a file on disk and used in subsequent calls to Dropbox functions
  - You can authorise / deauthorise as many times as you want
- The demo supports both Dropbox authorisation modes
  - In no-redirect mode, the Dropbox authorisation webpage displays an authorisation code which the user copies and pastes into the demo client when prompted
  - In redirect mode, the Dropbox authorisation webpage automatically redirects the client back to a “finish” webpage served by the demo HTTP server

Setting up the Exercise

1. Sign up to Dropbox if you don't already have an account
2. Download all the files in my git repository (https://github.com/rozanski/bcs_spa14)
   - You can download the files directly from the website (click **Download Zip**) or retrieve them using a git tool
3. Download the Dropbox Core API software (Python or Java)
4. Follow the other instructions in the project README and the README for your chosen language
5. Run the unit tests to make sure everything is working
6. You are ready to start coding!

What's in the Repository (*see Appendix*)

- The **demo** directory contains two complete implementations of the code needed for the session, one in Python and one in Java
  - They are functionally equivalent, so you can use whichever language you prefer
  - The demo code should run on Windows, Mac and Linux
  - Each of these directories has its own README with instructions on running the demo and unit tests
- The **exercise** directory contains code skeletons created from the demo directory
  - This is the code you will be editing
- The exercise code contains comments which guide you to the various API calls you need to make
  - If you ever get stuck, you can refer to the corresponding code in the demo directory

Keeping Passwords Private with OAuth

Exercise 1: Granting Access to Dropbox Programmatically

Exercise 1: Granting Access to Dropbox Programmatically

- Complete the code in the following files in the **exercise** directory
- Each source file identifies the code blocks you need to write using **TODO** tags
- The tag is preceded by comments which explain what you have to do
- If you get stuck, look at the same file in the **demo** directory

Tasks	Python	Java
optional: define OAuth access data load / save data from token and session files	common_oauth.py	
start the no-redirect and redirect workflows	oauth_client.py	OauthClient.java
implementation of the no-redirect and redirect workflows	dropbox_workflow.py	DropboxWorkflowNoRedirect.java DropboxWorkflowRedirect.java

Keeping Passwords Private with OAuth

Exercise 2: Using the Dropbox API to Read and Write Files

Exercise 2: Using the Dropbox API to Read and Write Files

- Complete the code in the following files in the **exercise** directory
- The source file identifies the code blocks you need to write using **TODO** tags
- The tag is preceded by comments which explain what you have to do
- If you get stuck, look at the same file in the **demo** directory
- You can also make Dropbox calls in the interpreter using the module variable **dropbox_client** (a **DropboxClient()**)
  - see https://www.dropbox.com/developers/core/docs/python

Tasks	Python	Java
display, create, delete files	dropbox_tools.py	DropboxTools.java

Conclusion

Conclusion

- What do we think of OAuth?
  - How easy is it to understand?
  - How easy is it to use?
  - How well does it work?

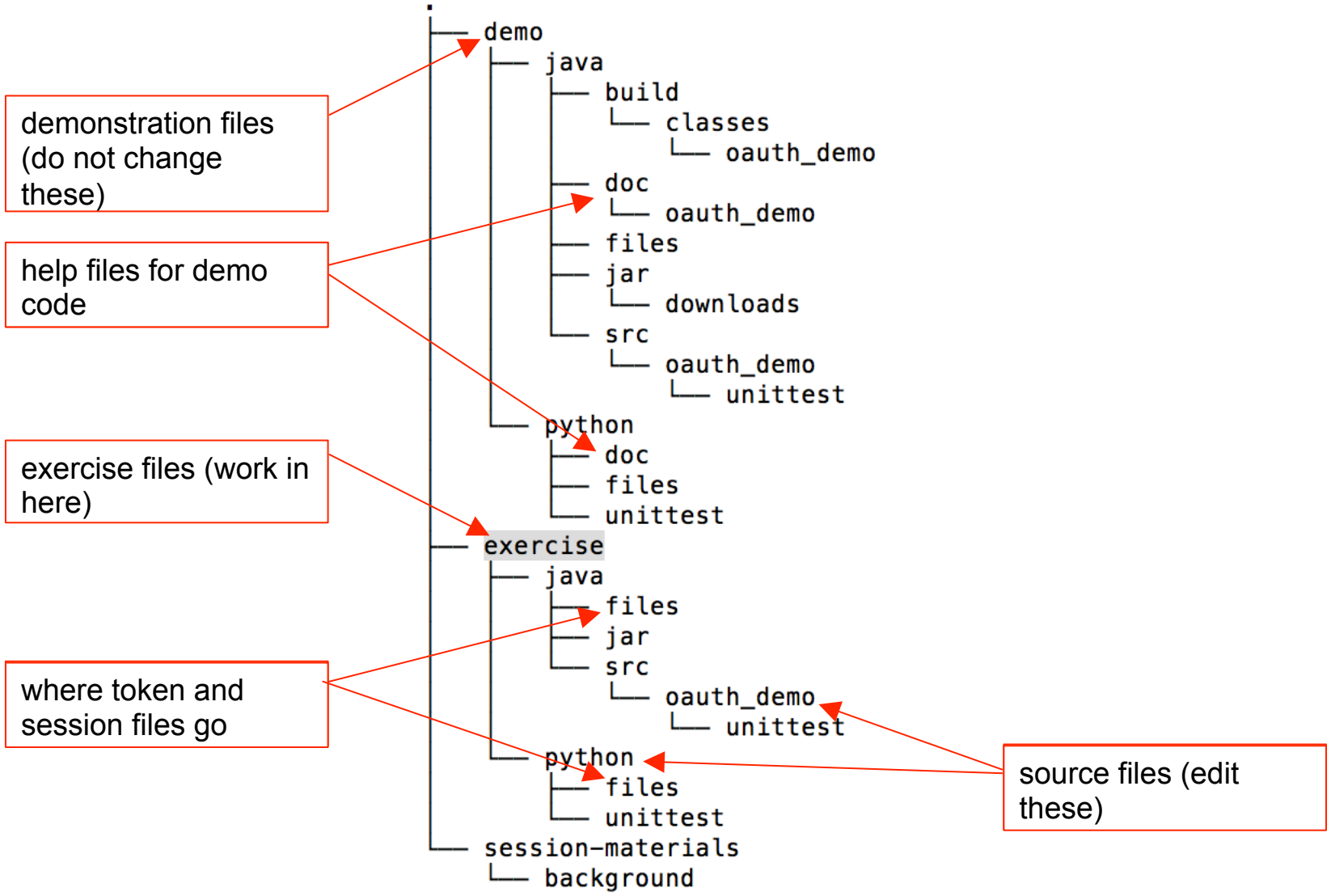
- Was Eran Hammer right to resign as lead author, withdrew from the IETF working group, and remove his name from the specification?
- Or is OAuth good enough to solve the Fuerris Bueller problem?

Eran Hammer

Keeping Passwords Private with OAuth

Appendix

Appendix: Directory Structure



Further Information

OAuth Website

- http://oauth.net

OAuth 2 Specification

- http://www.rfc-editor.org/info/rfc6749

OAuth 2 Workflow Diagrams

- https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server/raw/master/docs/OAuth2.0_Diagrams.pdf

APIs

Dropbox Core API

- https://www.dropbox.com/developers/core (*Python, Java, Ruby, Android, iOS etc*)

Dropbox REST API

- https://www.dropbox.com/developers/core/docs

Dropbox Tutorials

- https://www.dropbox.com/developers/core/start/python (also Java, Ruby, …)

Eran Hammer

- http://hueniverse.com/2012/07/26/oauth-2-0-and-the-road-to-hell/