Keeping Passwords Private with OAuth

Nick Rozanski and Eoin Woods, SPA2014, Tuesday 1st July, 2014 9:15am – 12:00pm

Problem Statement

Cloud-based services are often accessed by third-party programs (e.g. Dropbox). You don't want to trust the app with your username and password. This problem is called cross-domain authentication.

More interesting problem statement – special car keys that allow the owner to provide limited authorisations to e.g. children, valets… *Getting Started with OAuth 2.0*, Boyd

OAuth

Standard protocol
Client which implements the 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 itself as part of the one-off authorization process.

Benefits and Criticisms

- Reduces need to trust the client
    - Resource server can constrain access to data and services – e.g. a specific folder on Dropbox
- Decouples resource access from password changes
- Allows user to revoke access without changing password
- Helps avoid users becoming desensitised to phishing, password-harvesting etc.

But

- OAuth is about authorisation, not identity (unlike Kerberos, for example) – anyone with the secret and key can access the resources
- OAuth 2 doesn't require signatures to identify endpoints
    - Developer can accidentally send credentials to a malicious endpoint
    - Partly mitigated by using SSL
    - Still under development in the standard
- OAuth 2 is "complicated"

A Brief History

- 1.0 – 2007-2010 (Eran Hammer) – early adopter Twitter
- 2.0 – 2012 – not backward compatible, Hammer very critical of "committee patchwork"
- About 80 service providers use one or other OAuth standard: Amazon, Google, Facebook, Microsoft, Yahoo, Dropbox, Zendesk…

Terminology

- Resource owner – grants access
- Protected resource – data or service to be protected
- Resource server – hosts the protected resource
- Authorisation server – performs resource authorisation actions
- Client – a program that makes OAuth calls to perform actions on protected resources on behalf of the resource owner

- Registration – process whereby a developer sets up key attributes of their application and is given an access token in return
- Workflow

Application Registration

OAuth 2 requires clients (specifically, their developers) to register with the authorization server so that client requests can be properly identified. Service providers normally provide some kind of web console to do this, e.g. Google API 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 URLs or URIs. The client may only redirect to one URI in the "finish" step of authorization.

Authorization Workflow

OAuth defines four workflows, two of which are authorisation workflows. We look at the "no-redirect" authorization token workflow first.

No-Redirect workflow

User accesses client. Client redirects to Dropbox Auth Server. User prompted to log into Dropbox (if not already logged in). Authorisation page displayed by auth server that asks the user whether to allow the access to the specified dropbox resource. If OK, auth code displayed by Dropbox, which user has to copy/paste into the client. Client then requests an access token from auth server, using the code as proof of permission, and if received, redirects to the FINISH URI.

Redirect workflow

Very similar, but instead of displaying the auth code to the user, the auth server redirects the client to a web page that passes the access token to the client (e.g. in a cookie – though for the exercise today, a disk file is used, which the client can read).

Exercise

Part 1: authorise with DropBox using OAuth2 (both workflows)
Part 2: run various commands to display or manipulate DropBox files.

Setting up

- Sign up with DropBox
- Clone Nick's git repo https://github.com/rozanski/bcs_spa2014
  - Demo directory contains full working code
  - Exercise contains skeleton code to fill in
- Follow the instructions in the README and the README for chosen implementation language – Java, Ruby, Python or whatever
- Run unit tests to make sure everything works
- Start coding (follow the TODO tags in the code)

Conclusions

Exercise was very good in teaching the basics rapidly.
OAuth 2 is frequently combined with OpenID. There doesn't seem to be much in the way of alternatives (possibly Kerberos).
GitHub provides some good example code.

DropBox recently introduced two-factor authentication. You can use Google Authenticator or it can send you a text.