Identity Remote Authentication

Jeff Disher (Copyright 2013 Open Autonomy Inc.)
Date: 2013-08-22
Version: 1

OpenAutonomy, while mostly focussed on how multiple applications can interoperate across different servers, also describes a way for a user to create a “login state” on multiple servers, based solely on their ability to provide real authentication credentials to one.

Why not OpenID?

OpenID does solve exactly this problem but there are a few issues which make it inappropriate as the solution to this problem within OpenAutonomy:

- There are too many versions of the spec and supporting all of them would add undue complexity to each deployed server
- Determining if the user provided an OpenAutonomy Identity or a vanilla OpenID would add an additional requirement to the process, in any case, since these are very different in that OpenAutonomy can’t interact with an identity which does not expose the identity RPC interface (OpenID is lighter-weight than this as it has no such requirement)
- In the future, it would be possible for an OpenAutonomy Identity to present itself as an OpenID to servers which already support that but this is not yet implemented on any deployed instances (it would be easy to do, however)

Summary of Flow:

USER := The user accessing the system via a web browser (or equivalent agent)

IDENTITY := The URL of the USER’s identity

HOME := The server where the IDENTITY lives (cookie root)

LOGIN := The URL of the login form (not on HOME) where the user is trying to enter their IDENTITY

FOREIGN := The server where the LOGIN form lives (cookie root)

Steps:

1. User logs in as IDENTITY on HOME (mechanism of this login is up to the implementation of HOME as it has no bearing on the system).

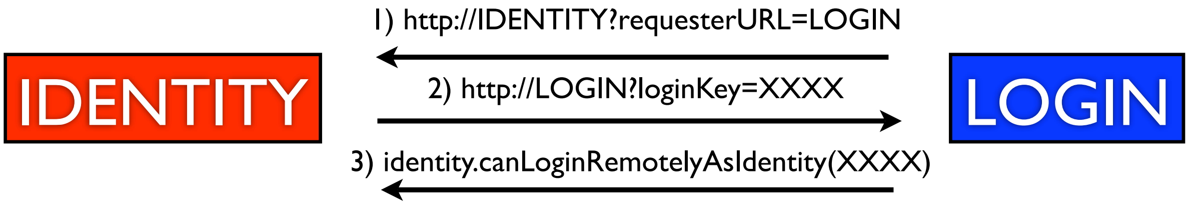
2. User visits LOGIN and pastes in their IDENTITY then clicks submit.

3. FOREIGN receives request with IDENTITY, from USER (submitted in step 2), stores it into server-side session, appends a GET variable of “requesterUrl”, set to LOGIN, and redirects the user to this amended IDENTITY URL.

4. HOME processes the incoming request and sees requesterUrl in the GET variables so it sends the user a confirmation request in the IDENTITY page (LOGIN can actually be shown in the confirmation as the requester). The user clicks accept.

5. HOME receives acknowledgement and forwards the user back to the given “requesterURL” but appends a GET variable of “loginKey” set to a string generated by IDENTITY.

6. FOREIGN processes the incoming request and reads loginKey. It contacts IDENTITY via the identity.canLoginRemotelyAsIdentity RPC call, sending loginKey. If this call succeeds, FOREIGN sets the login state for the user to IDENTITY.


1) http://IDENTITY?requesterURL=LOGIN
IDENTITY
2) http://LOGIN?loginKey=XXXX
LOGIN
3) identity.canLoginRemotelyAsIdentity(XXXX)

Notes on skipping steps:

The only parts of this process which can reasonably be skipped are those relating to **step 4**: the UI for the USER’s confirmation at HOME.

It is recommended that implementations either remember sites which have previously been granted the identity or provide an option to the user to do so. In these cases, the USER doesn’t need to acknowledge the operation, at all, and **step 4** can be reduced to simply verifying that the USER is logged in as IDENTITY and then redirecting with the login key without ever serving a page to the user.

Conventions for future possibilities:

In order to open the possibility for browsers to autofill the identity URL, in the future, it is recommended that the **input field** for the user’s remote identity URL have its **name** attribute set to “**oa:identity**”.