

A "Push" Identity-Data Relationship System: idAuth

Created: 5/15/2008
Revised: 5/26/2008

Kyle Brady
brady.k@gmail.com

Goal:

- to create a "push" system for data pertaining to specific identities
- based on a unique identifier
- can be easily transitioned into further OpenID support
- machine validation possible

Use-Case Scenario:

- aggregating a user's posted comments on blogs that support idAuth to an aggregator of choice, while retaining validity of ownership

Technology:

- microformats
- XML
- email / OpenID
- browser cookies
- key verification

Flow:

Prior to Use

- the user creates a relationship with a lifestream aggregator
- identifies the ID they will use for idAuth (email, OpenID)
- creates keys for use
 - may be generic, with a broad span such as "blog comments"
 - may be specific, such as "kyle-brady.com blog comments"
- sets a cookie containing all the user's idAuth information, in a standardized format

-Cookie Details:

Name = "idAuth"

Content Example:

```
idAuth Browser Cookie
----
id = brady.k@gmail.com
type = email
aggregator = OneSwirl
----
blog-comment,d432aLk0
kyle-brady.com,Xy6eE43
```

note: line breaks should be marked by "\n" instead

On Blog

- the user fills out the comments form with appropriate information
- the comment system looks for an idAuth cookie

- if one is found, it retrieves the id, type, and aggregator fields
- looks for an appropriate key, by domain first

- Example:

- search for "kyle-brady.com" key - fails

- search for "blog-comment" key - success

- if there is no appropriate key found, displays box for manual entry

- Example (Full Data Necessary for Submission):

- id = "brady.k@gmail.com"

- type = "email"

- aggregator = "OneSwirl"

- key = "17ab436x"

- comment = "Hello World"

- form submitted

- a hash ID is created: md5(identifier + uniqueURL + key)

- Example:

- identifier = "brady.k@gmail.com"

- uniqueURL = "http://www.kyle-brady.com/2008/05/15/fake-post#comment429"

- key = "17ab436x"

- hash ID

- md5("brady.k@gmail.comhttp://www.kyle-brady.com/2008/05/15/fake-post#comment42917ab436x")

- => "2f61567c613c721a4c973e1a30482bdc"

- aggregator is sent the data in XML format

- Example:

- <?xml version="1.0" encoding="UTF-8"?>

- <idAuth>

- <guid>2f61567c613c721a4c973e1a30482bdc</guid>

- <type>blog-comment</type>

- <title></title>

- <description>Hello World</description>

- <link>http://www.kyle-brady.com/2008/05/15/fake-post#comment429</link>

- <key>17ab436x</key>

- </idAuth>

- comment is posted, with microformats

- Example:

- <span class="idAuth" rel="blog-comment"

- id="2f61567c613c721a4c973e1a30482bdc"

- name="http://www.kyle-brady.com/2008/05/15/fake-post#comment429">

- ...normal HTML markup...

-

On Aggregator

- receives the XML data

- creates hash ID out of incoming data: md5(identifier + uniqueURL + key)

- if matches the guid of the XML data

- compares the given key to the user's key list

- valid key?

- saves data
- displays, when rendered, using microformats

-Example:

```
<span class="idAuth" rel="blog-comment-auth"
id="2f61567c613c721a4c973e1a30482bdc"
name="http://www.kyle-brady.com/2008/05/15/
fake-post#comment429">
    ...normal HTML markup...
</span>
```

- invalid/expired/blacklisted key?
- ignores data

Benefits:

- allows low level identity-data relationship authentication
- not dependent on OpenID
- allows easy conversion to full support of OpenID authentication
- machine validation possible
 - here is rel="blog-comment", is there a matching rel="blog-comment-auth"?*
- easy data aggregation
- first step towards a *real* token-based authentication-for-data system
 - makes an easier transition for the less tech-savvy or habit-ingrained

Initial Implementations:

- gain support of some current lifestream aggregators
- create libraries for common languages (PHP, Ruby, Perl)
- create plugins for common blog software (Wordpress, MoveableType, etc.)

User Flow Example:

Step 1:

The image shows a web form titled "Post a Comment" with a dark grey header. In the top right corner of the header, there is a link "[use idAuth]" with a mouse cursor pointing to it. The form itself has a light blue background and contains four input fields: "Name", "Email", and "Website" (each with a text input box), and "Comment" (with a larger text area). At the bottom right of the form is a "post" button.

Step 2:

Post a Comment

[use normal]

ID *brady.k@[...].com* Type

Aggregator

Key

Comment

I think you are wrong because I am a terrible troll!

Step 3:

