Practical

David Recordon
drecordon@verisign.com

Brian Ellin
brian@janrain.com

What was OpenID 1.1?

- An identity authentication system
- A protocol
  - gratis, libre
- Not a service or company
  - not Passport
  - not TypeKey
- Survives if companies turn evil or go out of business

Why Was It Developed?

lame

- No authentication was way too common
- Comment spam
- Auth interop
  - LiveJournal
  - TypePad
  - Movable Type
  - WordPress
  - DeadJournal

Design Goals For Auth

- Low barrier to entry
  - Works with static HTML pages
  - Decentralized
  - Understandable identity (a URL)
    - No new namespace
    - No public keys (key revocation, etc...)
  - No SSL required
  - No browser plugins
- Most simple protocol possible
  - Other needs layered atop

What is OpenID 2.0?

- An identity system framework
- Multiple protocols
  - Discovery (Yadis)
  - Authentication
    - URLs
    - i-names
  - Messaging (DTP)
  - Profile Exchange (Many layered atop DTP)
- Still not a service or company
- Open community development within the Apache Heraldry Podling

Design Goals For OpenID 2.0

- Identity 2.0
  - User Centric
  - Internet Scale
  - Privacy Protecting
  - Community Driven
- Framework of interoperable specifications
  - Handful of twenty page specs versus one one-hundred-fifty page spec
- Extensible
- Interoperable

How's Auth Work?

- Proves “who” you are
  - You own a URL or an i-name
  - One-time assertions w/ digital signature
  - See **openid.net** for specs, libraries, etc
- Not a trust system…yet
  - Spammers can/will/have setup OpenID Authentication servers
  - Better than the state of email today
  - Trust/reputation providers can easily build atop the OpenID framework

Role of the OpenID Server

- Provide a URL/i-name which the user “owns”
- Provide a way to authenticate users of the server
  - Auth mechanism not in OpenID spec
  - Password over SSL
  - Fingerprint, secure token, etc
    - By using strong auth in one place, all relying parties benefit
- Asserts to a relying party that the person using the browser owns the given identifier

Why URLs as identifier?

- Already the convention
  - Comment by Matt at 7:23pm
  - Mouseover to see which Matt
- Users don't understand public keys
- Users don't understand namespaces
- Users associate email addresses with spam
- Users do understand URLs
  - 10+ years of billboards and TV commercials
- You can click them
  - Tangible

Why XRIs as identifier?

- Cool technology
- Simple
  - http://davidrecordon.com
  - =david.recordon
- Can be treated as a URL
  - http://xri.net/=david.recordon
- Transportable
- Complimentary
- Convergence

Deployment

- Relying Parties
  - Six Apart's blogging properties
  - Zooomr
  - ClaimID
  - Opinity
- Patches / Active Development
  - WordPress
  - MoinMoin
  - Drupal
  - phpBB
  - MediaWiki
- Identity Providers
  - *MyOpenID.com*
  - *VeriSign Lab's Personal Identity Provider*
  - *GetOpenID.net*

Bounty Program

- Implement OpenID 2.0 support as a relying party or identity provider and pass the OpenID compliance testing tool
- Are distributed as part of a project's core under an OSI-approved license
- Serve at least 200,000 internet users or 5,000 downloads per month
- Require no more than one configuration setting for an administrator to enable support
- Include the OpenID logo in the signon form
- **http://IWantMyOpenID.org**

What can you do with OpenID today?

- Ignore building your own authentication system when writing your web app
- Open your existing web app to the millions of already OpenID enabled users
- Provide an OpenID server for your users
- Build cool services
  - Anti-spam tools
  - Trust networks
  - Reputation
- Get involved!

Code!

- Free libraries on openid.net
  - PHP
  - Perl
  - Python
  - C#
  - Ruby
  - Java
  - C++
- Similar API across languages
- Hides low level details of the protocol

Delegation

- You may use a URL that you own without running your own OpenID server
- Have account http://brian.myopenid.com/
- Can verify http://brianellin.com by embedding:

```
<head>
  <link rel="openid.delegate" href="http://brian.myopenid.com/" />
  <link rel="openid.server" href="http://www.myopenid.com/server" />
</head>

OR using Yadis

<xrds:XRDS>
  <XRD>
    <Service>
      <Type>http://openid.net/signon/1.1</Type>
      <URI>http://http://www.myopenid.com/server</URI>
      <OpenID:Delegate>http://brian.myopenid.com</OpenID:Delegate>
    </Service>
  </XRD>
</xrds:XRDS>
```

OpenID Consumer Demo

1. Enter URL on OpenID consumer
2. Consumer redirects browser to server
3. Enter credentials on server, and approve transaction
4. Redirect back to consumer with server signed identity assertion

Verifying an OpenID URL

- Example in Ruby, but applies to all JanRain OpenID libraries
- Start with the OpenID::Consumer and OpenID::Store objects

```
store = OpenID::FilsystemStore.new('/path/')

consumer = OpenID::Consumer.new(session,store)
```

Begin the OpenID Transaction

```
request = consumer.begin(params[:openid_url])
```

- Finds the OpenID server for the URL
- Associates with server (shared secret exchange using Diffie-Hellman)

Redirect to OpenID Server

```
if request.status == OpenID::SUCCESS
  return_to = 'http://example.com/openid-response'
  trust_root = 'http://example.com/'
  redirect_to request.redirect_url(trust_root, return_to)
end
```

- Using the response object, we generate a server URL to which we redirect the user
- return_to is the URL on our site which will handle the server response
- trust_root is a string "descending" from the return_to which is used to identify the consumer site on the server

User authenticates with server

- At this point, the user's browser has been redirected to their OpenID server
- User provides authentication credentials to server
- Server redirects back to consumer with assertion of ownership

Handle OpenID Server Response

```
response = consumer.complete(params)
if response.status == OpenID::SUCCESS
  openid_url = response.identity_url
end
```

- params is a dictionary like object of HTTP query parameters
- The complete method checks the identity assertion and response signature
- response.identity_url is the user's verified identifier

- OpenID::SUCCESS!

Questions?

www.OpenID.net
www.OpenIDEnabled.com
www.IWantMyOpenID.org
yadis@lists.danga.com

David Recordon (drecordon@verisign.com)
Brian Ellin (brian@janrain.com)