

Identity Management in PRPL using OpenID

Debangsu Sengupta (debangsu@cs), Aditya Rajgarhia (adityar@cs)

1. Problem Statement

The goal is to explore different means of performing identity management in PrPI. The system today employs the most popular means of authentication: username/passwords. This model needs improvement because studies have shown that users generally choose low entropy passwords in spite of education. Further, with the proliferation of web applications, users need to create "yet another" set of credentials to access PrPI services. Users should be able to re-use their existing accounts across the web. The authentication should be delegated outside of the PrPI system.

Further, mobile platforms are becoming increasingly important. Smartphones have different types of keyboard inputs from hard keyboards to touch or stylus based soft keyboards. In both forms, typing lengthy, strong passwords is a tedious experience. We need to enable a scheme that is open to experimentation different sign-on schemes.

The final system goal is that existing single sign-on mechanisms involve significant information leaks to the single sign-on provider. An PrPI identity management system should be decentralized to minimize this.

2. Related Work

Several efforts have been made to provide single sign-on services. Some notable examples include domain-specific sign-on services such as Windows LiveID, Microsoft Passport, Google SSO and Yahoo! BBauth, site-specific ones such as Stanford WebAuth, and internet-wide ones such as Web SSO and SlashID. None of these have been adopted widely across the internet, though, as they suffer from a variety of problems. Firstly, they are centralized in the sense that the user cannot choose where the credentials are stored -- it is chosen by the service provider, and typically resides in servers owned by the provider. Moreover, many of these services are implemented as closed standards, hindering adoptability outside their specific domains.

In recent years, two separate projects have aimed to solve some of these problems. OpenID was motivated by the need for an open, decentralized and lightweight solution for single sign-on. The primary goal of OpenID is to enable a single login credential while using just the web browser. A second project started out as Microsoft's Cardspace, but the protocol is now referred to as Information Cards. Information cards were designed to provide a secure mechanism for single sign-on. In combination with OpenID, information cards can provide a user-friendly, secure experience.

3. Key Ideas

In order to address some of the challenges facing identity management in PrPI, we have implemented OpenID as a single sign-on mechanism on PrPI. First, we will briefly discuss how OpenID works. Some common terminology used in this report are:

- a) *End-user* -The person who wants to assert his or her identity to a site.
- b) *Identifier* - The URL chosen by the end-user as their OpenID identifier.
- c) *OpenID provider (OP)* - A service provider offering services of registering OpenID URLs and providing OpenID authentication.
- d) *Relying party (RP)* - The site that wants to verify the end-user's identifier.
- e) *User-agent* - Typically, the browser. It could also be a client application.

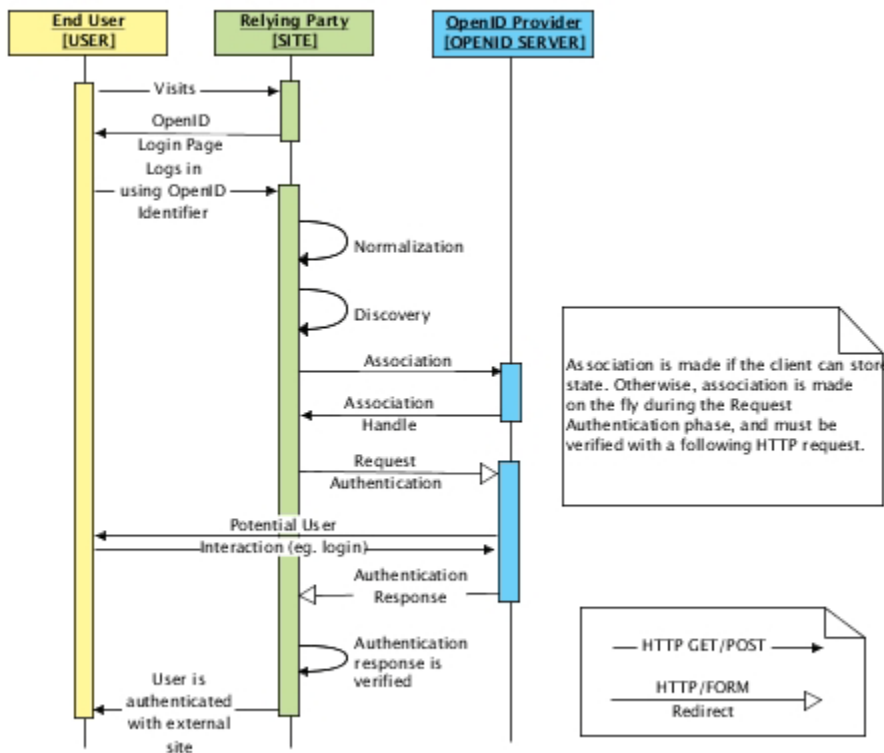


Figure Source: *TheServerSide*

The following are the key steps of logging a user into an external web site using OpenID:

1. The user is presented with an OpenID login form by the RP.
2. The user responds with their OpenID identifier, which takes the form of a URL.
3. The RP normalizes the OpenID URL and requests an XRDS document from the normalized version. The XRDS lists the user's OPs, from which the RP selects one. This is known as the discovery phase.

4. Next, the RP associates itself with the OP, exchanging a shared secret.
5. The user's web browser is redirected to the OP for authentication.
6. The user sends a login ID and password to the OP (in fact, any authentication mechanism may be used as desired by the OP).
7. The user is redirected to either the success URL (typically the RP) or the failure URL depending on whether the authentication was successful.

4. Design and Rationale

In this section, we discuss some of the design requirements and our solution. Our first major requirement is to have the PrPI system support OpenID. Specifically, this means both the PCB and the PrPI applications (e.g. web applications).

In order to have PrPI PCB's support OpenID's, we propose that new PCB's should directly use a user's OpenID identifier. Existing/legacy PCB's may continue to use e-mail addresses as user keys. They can still leverage OpenID as the sign-on process, and then fetch the key (e.g. e-mail address) from a user's OP provider via OpenID's Attribute Exchange.

In addition to the PCB's, PrPI applications should use OpenID's to communicate with PCB's via the mechanism described above. In PrPI, there are two classes of applications - web applications (e.g. webUI, Photo browser, RSS Feeds, TimeLine, Map etc). In addition, there are non-web applications e.g. harvesters. Our implementation targets web applications by hosting a standalone Relying Party on the web server. We have designed solutions for non-web applications, and have partial implementations ready.

The final design requirement is to assert one's own identity, to minimize the problem of information leaks to the single sign-on service provider. In order to do so, we propose running the OP on computer under the user's control (e.g. PCB, PC, or phone (future)). Users may use OpenID delegation to make stronger assertions using a more trusted OP.

5. Implementation details

The OpenID Relying Party is implemented in the PrPI Web Applications code, allowing the login application to use it. It has two main uses:

1. Directly use for signing on to PrPI PCB's that use OpenID keys
2. Use OpenID SReg extension/Attribute exchange for legacy PCB's using e-mail address keys.

The RP code is implemented as a Java Servlet, and listens to HTTP POST requests with OpenID passed in as argument. The key steps are:

1. DoAuthRequest - create the authentication request to be sent to the OP.

2. Perform Discovery - go to OpenID endpoint, (get redirected) and fetch user's XRDS document.
3. Associate with one of the shared endpoints and establish pre-shared secret using D-H key exchange (optional). Add to session.
4. Attribute Exchange/SReg - fetch additional information requests in the authentication request e.g. user's e-mail, name etc.
5. Provide returnTo URL to the request
6. Forward - send request to user via HTML form redirection (OpenID 2.0)
7. ProcessReturn (Validation on return from OP)
 1. VerifyResponse
 2. Retrieve discovery information from session if exists.
 3. Verify signature or with the OpenID provider
 4. Do housekeeping (expire handles)
 5. Extract OpenID and any attributes
8. Add own processing e.g. PrPl login specific information
9. Forwards to intermediate page to merge into PrPl login flow

OpenID Provider runs on the PCB, and client on the PC as standalone server. Both the RP and OP use *stateless mode* to support OP's on private machines. The OP's main functions are:

1. Set OP's listening endpoint
2. DoProcessRequest
 1. Associate mode:
 - Processes associate request from OP.
 2. Authentication request: CheckID_setup or Check_immediate
 - Do user interaction to gather data on user's selected ID, claimed ID, whether user authorizes
 - Process authentication request with gathered data.
 - Forward user to "returnTo" Url contained in requests.
3. Check_authentication mode: Verifies request from user.
4. Forwards result back to user.

The OpenID authentication is integrated into the existing PrPl login mechanism. OpenID authentication occurs in parallel using o*.jsp. Once the authentication has been completed, the flow merges with the normal sign-on process.

Example: OpenID YADIS XRDS document hosted at <http://pip.verisignlabs.com>

```
<?xml version="1.0" encoding="UTF-8"?>
<xrds:XRDS
  xmlns:xrds="xri://$xrds"
  xmlns:openid="http://openid.net/xmlns/1.0"
  xmlns="xri://$xrd*(($v*2.0))">
```

<XRD>

```
<Service priority="0">  
  <Type>http://specs.openid.net/auth/2.0/signon</Type>  
  <Type>http://openid.net/sreg/1.0</Type>  
  <Type>http://openid.net/extensions/sreg/1.1</Type>  
  <Type>http://schemas.openid.net/pape/policies/2007/06/phishing-resistant</Type>  
  <Type>http://schemas.openid.net/pape/policies/2007/06/multi-factor</Type>  
  <Type>http://schemas.openid.net/pape/policies/2007/06/multi-factor-physical</Type>  
  <URI>http://pip.verisignlabs.com/server</URI>  
  <LocalID>http://debangsu.pip.verisignlabs.com/</LocalID>  
</Service>
```

```
<Service priority="1">  
  <Type>http://openid.net/signon/1.1</Type>  
  <Type>http://openid.net/sreg/1.0</Type>  
  <Type>http://openid.net/extensions/sreg/1.1</Type>  
  <URI>http://pip.verisignlabs.com/server</URI>  
  <openid:Delegate>http://debangsu.pip.verisignlabs.com/</openid:Delegate>  
</Service>
```

</XRD>

</xrd:XRDS>

6. Lessons Learned:

During the research, design and implementation phases we discovered several interesting things about OpenID and information cards.

6.1 Security problems with OpenID

OpenID is currently very vulnerable to simple phishing. Unfortunately, there is no obvious, generic solution. An interesting approach would be to use a web browser plugin to detect phishing attempts, as described in the future work section. OpenID is also susceptible to DNS cache poisoning attacks. Such attacks can be avoided by using SSL at every stage. i.e. both, the URL for the XRDS document as well as the URL for the OP would need to use SSL in order to maintain integrity. Without SSL at the OP end, the Diffie-Hellman key exchange is also susceptible to a MITM attack. As discussed previously, after signing in at the OP, the user is redirected to the RP. The problem is that anyone who can obtain this URL (by sniffing, for example) can replay it and log into the RP as the victim user. Some OP's provide nonces which make sure that only one person can log in and subsequent attempts fail. However, a fast attacker who is sniffing the wire could obtain the URL and immediately reset the user's TCP connection, and then execute the replay attack. Cross-site request forgery (CSRF) attacks could also succeed

because the OP, and not the RP, dictates the user login security policy.

6.2 Privacy issues with OpenID

Firstly, in order to free up desirable userids, many large OPs may recycle OpenID identifiers belonging to inactive accounts. If an OpenID is recycled, the new owner will be able to access the previous owner's data if the RP is not aware that the OpenID has changed ownership. This is a very problematic issue for mainstream OPs, unlike say, the recycling of email addresses or cell phone numbers. For example, if someone (unknowingly) uses a recycled OpenID to sign into Flickr, the user may see the previous owner's private photos. Secondly, since every OpenID authentication request goes through the user's OP, the OP is able to track the user's online activities. This is a major privacy concern which could severely hinder the adoption of OpenID. Finally, because OpenID is a single sign-on mechanism, an OP being compromised by an attacker could be a potentially huge problem, on a much larger scale than any single website being compromised. An attacker who has a user's OP credentials could pose as the user on any website that accepts OpenID. All of these privacy concerns motivate the need for the users to own their OPs, such as on their mobile phone or PCB.

6.3 Information cards are more secure by design

Each information card utilizes a distinct pair-wise digital key for every realm where a key is requested. A realm may be a single site or a set of related sites all sharing the same target scope information when requesting an information card. The use of distinct pair-wise keys per realm means that even if a person is tricked into logging into an imposter site with an information card, a different key would be used at that site than the site that the imposter was trying to impersonate. Hence, simple phishing attacks would not be successful. It may be noted that information cards are susceptible to walk-up attacks (caused by users not logging out of websites and someone else using the same machine) unless the information card is protected by a local PIN. This can be enforced by the card provider if the card is a managed one. Self-signed cards would also suffice even for high-risk transactions if local PINs could be enforced for particular self-issued cards, and such cards would mitigate privacy concerns.

6.3.1 Disadvantage of information cards

One disadvantage of managed information cards is that similar to the privacy concern with OpenID OPs. Here, the card-issuer is able to track the user's activities. Information cards also require client card selector software. This is in particular a concern for mobile phones, which presently do not have standard development environments and code is hard to port between different mobile platforms, in particularly any graphical user interface code.

7. Future Work

a) Implementing an OpenID provider on mobile phones. As mentioned above, this would be a big step towards solving privacy issues facing OpenID. It would also support one of the fundamental ideas of POMI, that users should be in control of their data, not external parties. The

key steps to solving the scenario for mobile devices on private networks are proxy and/or modifying the OpenID protocol to support digital signatures.

b) A web browser plug-in could be used to detect OpenID phishing attempts. When a user navigates to a page containing an OpenID log-in form, the plug-in can detect this and then make sure that the URL that the browser is redirected to is contained in the user's XRDS document (whose location could be stored in the plugin). If none of the OPs listed in the XRDS match with the page to which the user was redirected, an alert would be raised to notify the user of a possible phishing attempt.

c) Enabling non web-based PrPI apps to use OpenID, preferably seamlessly. In the case of web-based applications, such as the photo browser or RSS feeds, the web browser's built-in session management can be used to stay authenticated with the PCB. However, non web-based applications, such as the harvesters, would need to manage the authentication within the application. A possible approach is to create an authentication token on the PCB, which is displayed on the web browser when the user logs in using OpenID. This token can then be manually entered into the PrPI configuration and would subsequently be used by any applications. The PCM would only grant access to the application if this authentication token matched the one stored on the PCB. Ideally, though, one would like to design a mechanism so that the authentication does not have to be manually copied by the user into the configuration file. We have built a design around this where an application registers with the PCB and gets granted long and short term tokens.

d) Implementing an information card selector on mobile phones. As mentioned in the previous section, portability is a major issue. However, a feasible solution is to implement a web-based card selector that runs on the cloud. In fact, if it runs on a user's PCB, then concerns about privacy are mitigated too.

e) Modifying the OpenID protocol to use digital signatures as opposed to shared secret.

f) Making OpenID/Information cards work with user and OP in different private networks. This would require a proxy-based solution.

8. References:

1. OpenID 2.0: a platform for user-centric identity management: <http://portal.acm.org/citation.cfm?id=1179532>
2. OpenID Discovery Using XRI and XRDS: <http://middleware.internet2.edu/idtrust/2008/papers/01-reed-openid-xri-xrds.pdf>
3. Identity federation and privacy: one step beyond: <http://www2.pflab.ecl.ntt.co.jp/dim2008/DIM2008Jacques.pdf>
4. User-centric PKI: <http://portal.acm.org/citation.cfm?id=1373290.1373300&coll=Portal&dl=ACM&CFID=14905102&CFTOKEN=53038432>
5. A delegation framework for federated identity management: <http://portal.acm.org/citation.cfm?id=1102486.1102502&coll=Portal&dl=ACM&CFID=14905102&CFTOKEN=53038432>

6. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps: <http://www.ai-lab.it/armando/GoogleSSOVulnerability.html>
7. A single sign-on protocol for distributed web applications based on standard internet mechanisms: <http://www.springerlink.com/content/q1315024071154g7/>
8. Single Sign-On for the Internet: A Security Story: <https://www.blackhat.com/presentations/bh-usa-07/Tsyrklevich/Whitepaper/bh-usa-07-tsyrklevich-WP.pdf>
9. OpenID: <http://openid.net/>
10. OpenID providers: <http://www.openid.org/>, <https://pip.verisignlabs.com/>, <https://www.myopenid.com/> etc.
11. OpenID tools and libraries: <http://openidenabled.com/>,
12. Libraries: <http://wiki.openid.net/Libraries>