Open Identity Management Framework for Mashup

Ding Chu, Qing Liao, Jingling Zhao
Beijing University of Posts and Telecommunications, Beijing, P.R.China
yeucd@163.com

Abstract

Mashups have emerged as a Web 2.0 phenomenon, connecting disjoint applications together to provide unified services. However, scalable access control for mashups is difficult. To enable a mashup to gather data from legacy applications and services, users must give the mashup their login names and passwords for those services. This is not user-centric and the all-or-nothing approach violates the principle of least privilege and leaves users vulnerable to misuse of their credentials by malicious mashups.

To overcome the limitations, this paper proposes an open identity framework, which leverages open identity protocol such as OpenID and OAuth. The framework can bring benefits to all the roles involved in the system in a non-intrusive and user-centric way. Open is a good design principle, and it is also the attitude and sprit of collaboration. We think that a mashup system based on open technologies could make the composition of services easier and accelerate the on-boarding of service providers. Moreover, more customers might also be attracted by the openness of the system.

Keywords: *Mashup, Identity Management, OpenID, OAuth*

I. Introduction

Mashups create new services and provide more convience to people by integrating different web services together[1].

Traditional mashup model requries users to provide their identity credentials (usernames and passwords) for back-end services that mashup will access. After the user logins mashup, mashup impersonates the user to get his protected resources at back-end services, then implements the integration and finally presents result to the user.

However, a mashup built in such manner introduces many security and privacy risks. For example, if the mashup server is compromised, the attacker can take over all the accounts in the back-end services by capturing the user authentication credentials stored in the mashup. Also, mashups often get more privileges than they really need. For example, even if the mashup just needs to read a user's calendar, the mashup will receive all of the user's calendar privileges.

Also, in a user-centric system, users wish to single sign-on across the services he subscribed instead of remembering all the accounts which might be different from each other. From service providers' (including mashups) perspective, he wishes his applications could be easy and fast integrated into system and can consume others' services to enhance the features of his applications. As a result, more customers might be attracted.

Based on the discussion above, we can list the common issues and requirements in an open identity management framework for mashup.

Requirement 1: Delegation and trust. Users should not have to provide their authentication information to the mashup when mashup tries to access back-end services. Delegation is the secure way to retrive users' protected resources without sharing identity credentials with the third party.

Requirement 2: Fine-grained authorization. Users should be able to provide fine-grained delegation. In other words, mashups should receive exactly the permissions they require to function. For example, if all a mashup needs is to read user data from a back-end service, then the mashup should only receive read permission instead of read-write permission.

Requirement 3: Single sign-on[2]. In the user-centric service system, it's the ultimate goal to maximize the convenience for users, and it is also important for identity management.

Requirement 4: Open. With open protocols and service APIs, new service providers are easier to join in the framework, and they can consume other services easier. This will attract more consumers.

In this paper, we made the following contributions: we proposed an open identity framework which leverages open identity protocols such as OpenID and OAuth to solve single sign-on, secure delegation mechanism and fine-grained controll, it releases mashups and service providers from identity authentication and account management. In the following, some related works will be introduced. After that, the identity management framework is illustrated. Then we give the implemention to our conception. The last section concludes this paper and points out some future work directions.

978-1-4244-6359-6/10/$26.00 ©2010 IEEE

II. Releated works

A. OpenID

OpenID is an open, decentralized, free framework for user-centric digital identity[3]. It is a single sign-on protocol that solves the problem of having an individual login and password for any web site which supports OpenID. Without incurring any further cost, anyone can use the OpenID system and operate websites which provide authentication based on OpenID identifiers. Because OpenID is an open protocol that do not require user to create and manage a new account in order to sign in to any OpenID-enabled websites, so if the framework employs OpenID to solve single sign-on problem, it will bring benefits to both mashups and service providers.

B. OAuth

OAuth protocol enables websites or applications to access protected resources from a web service via an API, without requiring users to disclose their service provider credentials to the consumers[4]. So it's a good candidate for secure web service call between service providers.

OAuth authorization is the process in which users grant access to their protected resources without sharing their credentials with the consumer. OAuth uses tokens generated by the service provider instead of the user's credentials in protected resources requests. OAuth has been published as an open protocol so it's a good choice for our system to employ to implement secure delegation access.

C. Previous works

Hasan and his team proposed an delegation solution based on Delegation Permit[5] in 2008. They defined their solution as ''a scalable, stateless delegated authorization protocol'', however this view hasn't been widely accepted[6]. Delegation Permit adopts lifespans to permit of authorization, which can provide protection to users' resources in some degree when the permit is illegal used. One of the main purposes of it is to solve the problem of authorization granularity. In its framework, PGS (Permit Grant Server) asks user about authorization and sends the permit to mashup. Despite the fact that Delegation Permits was originally thought to grant authorization to mashup, its operation can be generalized to other kinds of systems, but it has no consideration on SSO, and we have no way to identify mashup to anti-phishing, besides, the five modules and complex interaction process also block it's spreading.

Google provides a hybrid approach that combine OpenID and OAuth together-OpenID OAuth Extension[7]. It describes a mechanism to combine an OpenID authentication request with the approval of an OAuth request token. So after the user authenticated on the OpenID provider, the provider asks for the user's approval for relying party's (RP) unauthorized token for this user, after the user is redirected back to RP, the RP can exchange the approved request token to an access token that can further consume the OP's services on behalf of the current user. The protocol requires the OP and OAuth service provider are the same service and the RP and OAuth consumer are the same service. So this is some different in this paper's work where mashup (or service provider) will consume other service provider's services that is not an identity provider.

Wang Bin and his team from IBM proposed their framework for SaaS Service Platform[8] in 2009. It sets the SDP (Service Delievery Platform) as a center for identity management. There are two important components in the service application, one is the OpenID RP component, and this component implements the standard OpenID protocol to act as the RP role in the protocol. Another component is the OAuth provider or consumer. This component also is the standard implementation of OAuth protocol, if the service application wants to expose protected service related to the customer, they need to implement the consumer part; if they want to consume the services on the platform. they need to implement the service provider part. This makes it possible for service providers to call each other, but they all have to communicate with SDP, which is the biggest weakness, because each time a request is maken, OAuth sequence has to be excuted twice, even worse the server browser in SDP present as both service provider and consumer which is a big burden for the system. And this is what we will avoid in our framework.

III. Proposed framework

A. General description

From the evolution process of identity management, a currently emerging paradigm is that of user centricity, that is, the idea of giving the user full control of transactions involving her privacy data. To better meet the 4 requirements described in section 1, we decide to employ OpenID and OAuth protocol as the base of the framework. If the framework is open enough, service provider could be easily integrated into the platform, vice versa. Since the customers have more control ability on their privacy data, they will definitely decide what mashup can retrive and realize it's a good way to protect their privacy.

The popular anti-phishing method for OpenID is requiring users to provide additional information when they sign up. When OpenID RP redirects the user to OpenID Provider, the information provided by the user appears firstly, because the user can verify this private information, so he can identify whether this is a phishing site, if not, he can provide his sign in password safely.

B. Architecture Description

The structure of our system can be depicted in Figure 1.

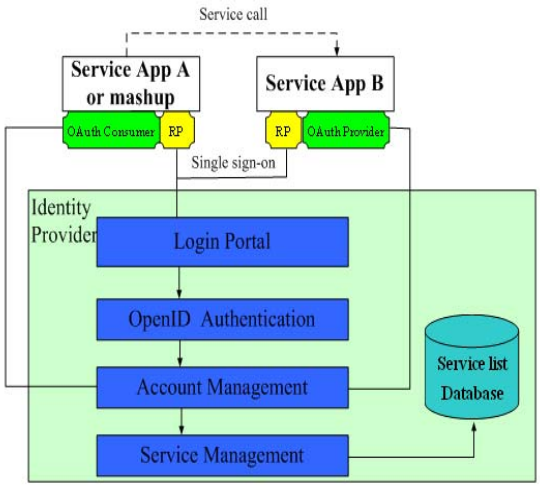

Figure 1. Framework architecture.

Mashup can be regarded as a service provider, but the reason we have been emphsis the role of mashup is that we don't just want the framework to simply support a serivce provider to consume service from another service provider, we also want mashup can integrate different services together, it can even present as an identity aggregator which integrates the user's identity attributes in different service providers so that it can provide authentication service to the user. Obviously mashup is a scalable component.

As the most important part, Idp is responsible for providing identity authentication and service management. For example each user has their own username and password on Idp, and can apply for an OpenID account; each Service App (including mashup) has to apply for their own ID and secret from Idp. This ID and secret is used to authenticate to Idp. Every time the user requests service from a service provider or mashup, Idp will check if the requested service is in the service list by Service Management component.

C. Scenarios

1) Scenario 1： Customer sign up

A user accesses Idp and provides his credential so that Idp creates an OpenID account for this user. As described in last section, to avoid phising attack, the sign up information must contain a specify message or pic which is provided by the user.

2) Scenario 2: Service application *on board*

A service application (including mashup) needs to do 2 things when it on board:

a) Apply for an application ID and secret which is private between service applications and Idp. So Idp will add the service applications into it's service list database.

b) Configure Idp as one OpenID provider.

3) Scenario 3: Customer single sign-on to Mashup

The login portal to the OpenID account is an special link that after the user clicks, the user will be asked to provide his OpenID identifier; mashup contacts default Idp configuated in last scenario, and gets authentication token by his ID and secret; User agent will be redirected to Idp, the customer sign in to the login portal by his OpenID account; After authenticating the user's identity, Idp redirects user agent to mashup with anthentication token to the user.

4) Scenario 4: Mashup requests user's protected resource

The process that mashup requests the protected resources is still based on token scheme:

Mashup redirects user to Sp. Sp authenticates to Idp and checks the authentication tokens for the user and mashup. After authenticated, Sp will ask the user whether he is going to authorize mashup and he will be prompt to submit options about authorization granularity and expiration date. After the user submits his choices, user's browser will be redirected to mashup and mashup will get the user authorized token. Then Mashup exchanges it for an access token. Sp checks the token's expiration date, if the token is not expired, Sp will send an access token to mashup. Mashup takes the access token to get the user's protected resources from Sp. Figure2 shows the interactive process, and Figure 3 is the request sequence.

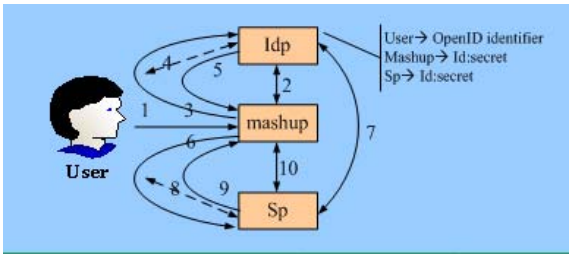

Figure 2. **Interactive process of service in our framework.** 1.User requests resources from mashup. 2.Mashup authenticates to Idp with its ID and secret, and gets authentication token. 3.Mashup redirects user to Idp. 4.User authenticates to Idp. 5.Idp redirects user to mashup with user's authentication token. 6.Mashup redirects user to Sp with all the tokens gets in previous steps. 7.Sp authenticates to Idp with tokens from mashup. 8.User authorizes to mashup on Sp, and submit choices on authorization granuarity and expiration date. 9.Sp redirects user to mashup with user authorized token. 9.Mashup exchanges for access token with Sp. 10.Sp sends access token to mashup. Then mashup can get users' protected resources on Sp.

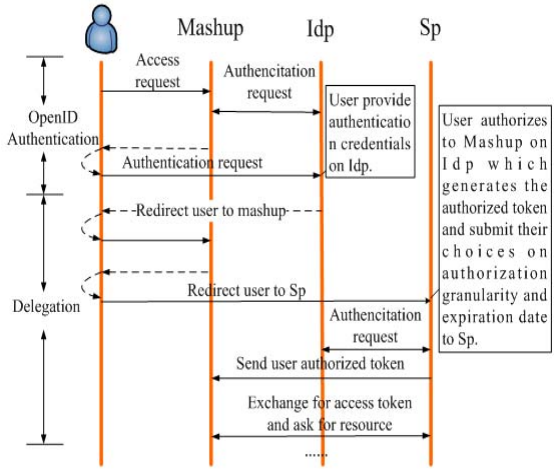

Figure 3. Request sequence.

IV. Implementation

To demonstrate the feasibility of our approach we implemented a prototype using Java 1.5, and did a test deployment in our corporative network.

For simplicity, we just require mashup to integrate two services. One of them is a calendar service which can provide schedule records service to users, we set it in a separate server; the other is an ads provider which can provide related ads business according to the matching fields, and this service is created as servlet running under the mashup server. The Idp is under another server, we choosed to use our own OpenID identifier, so we simplified the normalization and discovery process.

Besides, we based all the authentication interactive process on SAML, which helps to protect the privacy of the roles in our framework.

So when the user logins mashup and asks for his schedule from calendar server, mashup will firstly get the schedule information from calendar server, and then provide related ads to user on his schedule records. Figure 4 shows a screenshot of our mashup page, an ebay advertisement is provided to the user upon his "EventTitle" attribute retrived from Calendar Server.

The Returned Attributes:

Attributes found.

Attribute Name	Attribute Value
EventTitle	Shopping on the net
Service Provider	Calendar Server
Start Time	2010-5-11T08:30:00.000+08:00
End Time	2010-5-11T09:30:00.000+08:00
User ID	buptchuding@gmail.com

Figure 4. Mashup page from our implementation.

V. Conclusions

As mashups become more widespread in the Internet, ensuring a proper identity management and delegation model for mashups becomes essential. In this paper, we presented a scalable, open identity management framework and a practical implementation to it. The framework leverages open identity protocol OpenID and OAuth to solve single sign-on, secure delegation mechanism and fine-grained control, it releases service providers (including mashups) from identity authentication and account management. Our practical implementation of a proof of concept proves the usability of our approach.

Open is a good design principle, and it is also the attitude and sprit of collaboration. We think that a mashup ecosystem based on open technologies could make the composition of services easier and accelerate the on-boarding of service providers. Moreover, more customers might also be attracted by the openness of the ecosystem. In future, mashups as an identity aggregator might be considered.

References

[1] N.Kulathuramaiyer. Mashups: Emerging application development paradigm for a digital journal. Journal of Universal Computer Science, 13(4):531-542, April 2007.

[2] OpenSSO Web site. Last accessed in December, 2009 at: https://opensso. dev. java. net.

[3] OpenID Web site. Last accessed in December, 2009 at http://openid.net.

[4] OAuth Web site. Last accessed in December, 2009 at http://oauth.net/core/1.0a.

[5] Hasan R, Conlan R, Slesinsky B, Ramani N, Winslett M. Please permit me: stateless delegated authorization in mashups. In: Annual computer security applications

conference (ACSAC); 2008.
[6] Jorge Fontenla Gonza´ lez et al., Reverse OAuth: A solution to achieve delegated authorizations in single sign-on e-learning systems, Comput. Secur.(2009), doi:10.1016/j. cose.2009. 06.002
[7] OpenID OAuth Extension Web site. Last accessed in December, 2009 at: http:// step2.googlecode.com/svn/spec/openid_oauth _extension/latest/openid_oauth_extension.ht ml.
[8] Wang Bin, Huang Heyuan, Liu Xiaoxi, Xu Jingmin. Open Identity Management Framework for SaaS Ecosystem. In: IEEE International Conference on e-Business Engineering; 2009.