Open Identity Management Framework for SaaS Ecosystem

Wang Bin, Huang He Yuan, Liu Xiao Xi, Xu Jing Min
China Research Laboratory
IBM
{wangbcrl,huanghey,liuxx,xujingm}@cn.ibm.com

***Abstract*—As Software-as-a-Service (SaaS) becomes more and more popular, the identity management and federation among SaaS applications also become an important factor impacting the growth of SaaS ecosystem. Typically, there are three major functions to be enabled in identity federation: 1) Single Sign-On across different services. 2) Account provisioning to different services. 3) Secure backend service call between services. Current SaaS delivery platforms provide these functions in an ad-hoc way, which might limit the growth of SaaS ecosystem. To overcome the limitations, this paper proposes an open identity framework, which leverages open identity protocol such as OpenID and OAuth. Moreover, an OAuth broker is proposed to mediate backend service calls among SaaS applications. The framework can bring benefits to all the roles involved in the ecosystem 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 SaaS 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.**

Keywords-SaaS; Identity Management; Identity Federation; OpenID; OAuth

I. INTRODUCTION

One of the hot topics in computer technology is software as a service, in which vendors host applications on the Internet and provide them via a browser to users, who perform and store their work online [1]. Typical SaaS ecosystem, as depicted in Figure1, is mainly consisting of a service delivery platform, many cataloged services and customers. The service delivery platform provides a hosting environment to attract both customers and service providers; therefore the ecosystem is in a virtuous cycle since more customers join in to search and consume plenty services.

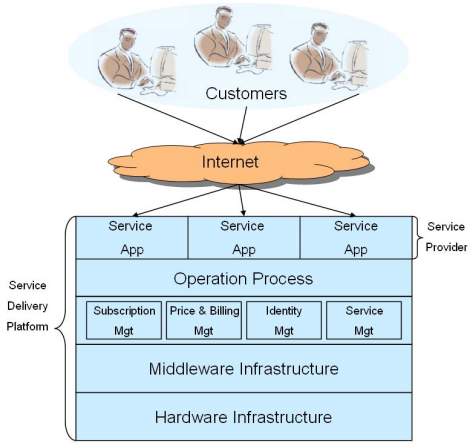

Figure 1. SaaS Ecosystem

So the service delivery platform take responsibly of managing the operational work such as service on-boarding, subscription, pricing, billing and identity management. To provide an easy consuming environment to customer and service provider, the platform usually act as an identity provider thus the service provider can be fast integrated into the platform because the platform help him reduce identity authentication and account management effort. The customer also wishes to single sign-on across the services he subscribed. From service provider's perspective, he wishes his applications could be easy and fast integrated into platform and can consume others' services to enhance the features of his applications. As a result, more customers might be attracted. So it's perfect to position the platform as role of identity provider

In identity management area, a federated single sign-on system is an identity management system (IMS) that allows the use of the same user's Personal Identification Information (PII) across multiple organizations within a federation [2]. In SaaS ecosystem, the platform and the services had established a service federation that allow customer to use the services he subscribed in the federation. From the business requirements and prior art, the platform would provide three major functions if it plays identity provider role in the federation:

1) Single Sing-On across the service federation that customer only need sign in once to consume different services.

978-0-7695-3842-6/09 $26.00 © 2009 IEEE
DOI 10.1109/ICEBE.2009.82

2) Account provisioning to services that meet business requirements from some service provider that want to control the customer's account.

3) Secure delegation mechanism that enables one service on behalf of current customer to call another service. For example, accounting service need to retrieve partner information of current user which provides by CRM service.

Current service delivery platform usually publish a user validation API for service provider to invoke that avoid re-authentication [3][4], these APIs usually employ different security mechanisms to protect user privacy. This requires service provider to follow the platform's specific API or apply for a developer account. If the service provider decides to quit from original platform and join another platform, he needs to re-implement the authentication and secure service call framework. This will add efforts for service provider. For platform vendor, they need to decide whether to employ a standard based identity management protocol or invent new.

For safe and effective digital ID management, many studies are underway including OASIS's SAML, Microsoft's WS-*, and URL-based OpenID. Among these, OpenID, which is a new user-centric digital ID management system, supports decentralized architectures, while the existing .Net Passport system is centralized. Without incurring any further cost, anyone can use the OpenID system and operate websites which provide authentication based on OpenID identifiers [5]. Because OpenID is an open protocol that do not require user create and manage a new account in order to sign in to any OpenID-enabled websites, so if the platform employs OpenID to solve single sign-on problem, it will bring benefits both to customer and service provider.

The OAuth[6] protocol enables websites or applications (Consumers) to access protected resources from a web service (Service Provider) via an API, without requiring users to disclose their service provider credentials to the consumers. WS-Trust[7]/WS-Federation[8] provides standards which can be used as part of the solution. However, OAuth has been published as an open protocol so it's a good choice for service delivery platform to employ to implement secure delegation access.

However, there are some shortcomings for using OpenID and OAtuh in a service ecosystem. For example, some studies revealed these two protocols are vulnerable to phishing [9][10]. In this paper, we propose an identity management framework to well employ the two open protocols to solve single sign-on, account provisioning and secure delegation mechanism for a service delivery platform and service provider. In the following, some related works will be introduced. As underlying technologies, OpenID and OAuth will be introduced briefly. After that, the identity management framework is illustrated. The last section concludes this paper and points out some future work directions.

II. RELATED WORK

IBM Tivoli Federated Identity Manager(TFIM)[11] is a powerful federated identity management product, this product implements solutions for federated single sign-on (F-SSO), web services security management, and provisioning that are based on open standards, such as SAML 1.x/2.0, Liberty WS-Federation and WS-Provisioning. In version 6.2, OpenID and Information Card protocol also be joint in TFIM family, so the customer can choose which protocol to use based on their business scenario. But TFIM depends on many other IBM products, such as WebSphere Application Server, Tivoli Directory Integrator to implement identity lifecycle management. What's more, TFIM using direct trust model, that means the partners in a federation must set up a peer to peer trust relationship. In SaaS ecosystem, the service provider may be small company, so it's a high threshold for him to take part in the platform. In further step, service providers may do not know each other, so ask them to set up a peer to peer trust relationship might be unpractical.

Salesforce has its own system of user authentication, but some companies prefer to use an existing single sign-on capability to simplify and standardize their user authentication. So Salesforce provides two options to implement single sign-on-delegated authentication or federated authentication using SAML [12]. These two options assume the customer will login Salesforce within a corporation. All other scenarios require the customer login from salesforce.com at first. Salesforce provides some user authentication API for application provider; the application provider must follow platform specific requirements to on board their application.

Google provides a hybrid approach that combine OpenID and OAuth together- OpenID OAuth Extension [13][16]. 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 ask for the user's approval for relying party(RP)'s 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 service provider are the same services. So this is some different in this paper's work where service provider will consumer another service provider's services that is not an identity provider.

It will cost much money to invest a federated identity management using mature business product such as TFIM, this might be unpractical for most service providers. Also, some existing platform using their ad-hoc method to implement single sign-on function will bring hindrance for

service providers. Though the combination protocol of OpenID and OAuth reduce the authentication and access effort, but it is not suit for a service delivery platform.

III. OPENID AND OAUTH CONCEPT

OpenID is an open, decentralized, free framework for user-centric digital identity. 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.

A. OpenID Stack

As described in Figure2, the goal of OpenID is to create a framework which balances the need to be flexible and adaptable with the need to simple and pragmatic to enable broad adoption.

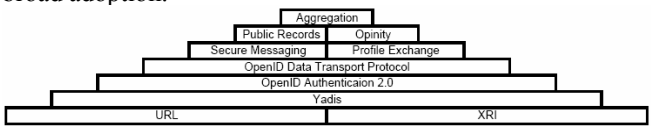

Figure 2. OpenID Stack

The URLs and XRIs are the base as the end user's identifier, the upper layer-Yadis, providing simple service discovery using the XRDS document format from OASIS. OpenID authentication providing the basic single sign-on layer, the OpenID data transport protocol providing a level of abstraction for higher level services that depend on trusted data exchanged. On top of this base, other identity-based services such as secure messaging or profile exchange can be layered depending on the needs of a specific implementation [14].

B. OpenID Authentication

In OpenID, RP uses the user's input URL to discovery the user's OP and requests the user's authentication. Figure3 describes how OpenID authentication works.

1) User sends his/her OpenID to an RP to request a service or services.

2) The RP discovers the user's OP information such as address using the Yadis protocol. The RP associates with OP and shares session key etc.

3) RP redirects the user to the OP.

4) If the user is not authenticated, the OP authenticates him/her and notifies the RP of the authentication via the user browser.

5) The RP checks the user's authentication and decides whether or not to provide its service to the user.

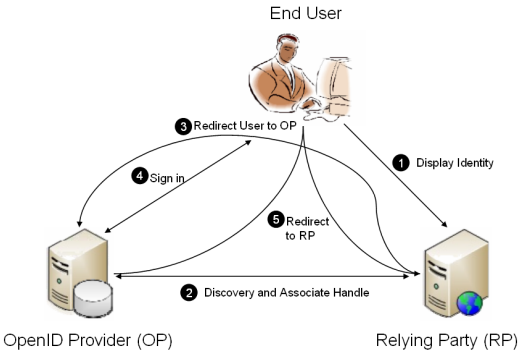

Figure 3. OpenID Authentication Process

C. OAuth Authentication Flow

The 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. So it's a good candidate for secure web service call between service providers.

OAuth authentication 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. As described in Figure4, OAuth authentication is done in three main steps:

1) Contains 1 and 2, the consumer obtains an unauthorized request token.

2) Contains 3 and 4, the user authorizes the request token.

3) Contains 5 and 6, the consumer exchange the request token for an access token.

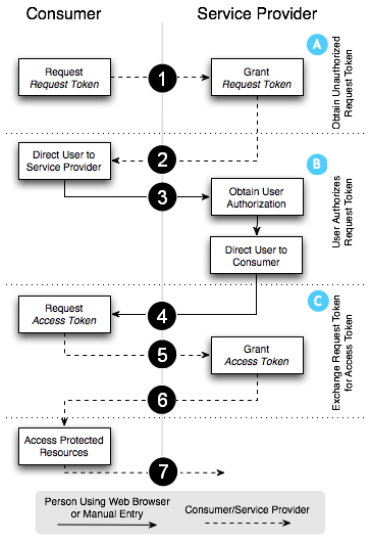

Figure 4. OAuth Authentication Flow

Once the consumer obtains an access token, the consumer can access the protected resources on service

provider on behalf of current user, such as the data stored on the service provider.

IV. PROPOSED FRAMEWORK

A. Motivation

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 identity data [15]. To better meet the 3 requirements described in section 1, we decide to employ the 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 identity data, they will definitely realize in which transaction be involved, then they can receive a clear bill and think it's a good way to protect their privacy.

The popular anti-phishing method for OpenID is asking user to provide additional information when they sign up. When RP redirects the user to the OP, 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.

The on board process for service provide could speed up by using public library based on open protocol. Since there are no integration efforts such as implement the platform's API or using the library provided by the platform. What's more, for secure service call, we need to bridge the gap between different service providers since they even did not know each other. So it's difficult for them to publish/consume services directly. Each of them only needs to communicate with the platform, then using a broker to mediate the request across the ecosystem.

B. Design Goal

The goal of this work is to deliver a lightweight and user-centric identity management framework for SaaS ecosystem.

1) lightweight: to speed up the integration process including service application on board, single sign-on, secure service call between two or more service applications even their provider do not know each other.

2) user centric: give full control of identity lifecycle to the customer, they decide when and how to use their identity information.

C. Architecture Description

The framework can be divided into two parts-the service delivery platform and the service application.

In Figure5, the identity provider is the main part of the service delivery platform. Also is the most important component of this framework. Service applications, which in the upper layer, they will be integrated into the platform and also contains some components defined in the framework.

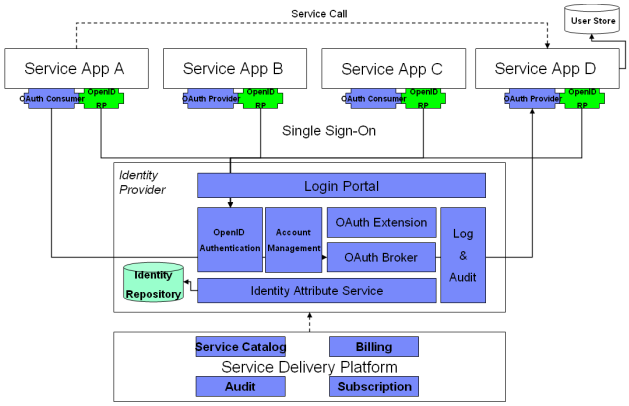

Figure 5. Framework Architecture

As there are two important components in the service application, one is the OpenID RP component, 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.

There are several important components in the identity provider. The bottom is the identity attribute service, which is in charge of access the identity information in the identity repository. Account management provides customer registration, profile management function. OpenID authentication component is in to handle authentication request from OpenID RP, which is service application in Figure5. The OAuth Broker component will bridge the invocation gap between service providers. It means service provider and consumer only need to communicate with identity provider, this component will first act as a service provider when it receives the consumer from one service application and then act as a consumer to request the real service which provided by another service application. Because of this component, the service providers only need to know they want to expose or call services on the platform instead of to care about where and how the service could be published or invoked. To reduce the consent burden for each OAuth service call for the customer, OAuth extension component will help to manage the authorization relationship between customer and services. So if the customer decides to authorize service application A can retrieve his email, friend's list on service application D for one month. Then there is no need to get the customer's consent in following one month at every OAuth service call. The extension component mainly in charge of authorization rule establishment and revocation that describe which service application can get what information in a valid time period. The Log and Audit component will trace each identity transaction so can be integrated to the

service delivery platform to meet business requirement. The upper layer is a unified login portal so that user interface of registration, login, authorization and other transaction can be composed.

For user provision requirement issued by some service providers, they can implement specific application programming interface so that after the user consent that the platform can provision an account to the service provider, the platform could invoke these specific APIs to send identity information to them. So this function also can be implemented by OAuth protocol and the identity attribute service component on the platform.

D. Scenarios

1) Scenario 1: Customer sign up

A user accesses the service delivery platform and provides his identity information so that platform creates an OpenID account for this user. After the user becomes a customer of the platform, he can subscribe and consume different services as he wish. As described in last section, to avoid phising attack, the sign up information must contain a specify message which provided by the user, such as "I like Open Framework".

Since the platform is an OpenID provider, so the user can sign in many OpenID RP site beyond the service ecosystem. This can introduce further new service delivery platform and then enhance the whole ecosystem.

2) Scenario 2: Service application on board

An application ID and secret will be shared between service provider and platform when they on board. The service application will use this ID and secret to finish OAuth service invocation. What's more, this credential information only shared between service provider and the platform. The service application also configures the platform identity provider as their one option for OpenID provider to reduce the discovery time cost which is in the first step of OpenID protocol. For some service applications that want to expose services through API, they need to register this information so that identity provider can publish and bridge this service information. In summary, a service application needs to do 3 things when it on board:

a) Apply for an application ID and secret.

b) Configure the service delivery platform as one OpenID provider.

c) Register the API for services he wants to expose.

It will reduce the integration effort by introduce a central service catalog that contains all the public services API. Then the service provide has the chance to discovery and consume smoothly.

3) Scenario 3: Customer single sign-on to the service applications

a) The customer first sign in to the login portal by his OpenID account.

b) The identity provider establishes a security context for this customer.

c) The customer enters to his service catalog page and decides to access accounting service.

d) The URL to the accounting service is an special link that after the user clicked, the user agent will be redirected to accounting service protected resource at first, and because the customer has not sign in this service application before, so accounting service application will redirect the user agent to default OpenID provider which configured in last scenario.

e) Because login portal has established a security context for this customer. So the identity provider would no logger to challenge the customer to enter id and password. Instead, the user agent will be redirected back to accounting service with authenticated flag.

4) Scenario 4: Accounting Service call CRM Service to retrieve customer list

As described in Figure6, the Accounting Service acts as a standard OAuth Consumer role as well as the CRM Service acts as a standard OAuth Service Provider role. The identity provider mediates the request from the Accounting Service to the CRM Service.

a) The customer clicks a link on Accounting Service so trigger the service call to get his customer list.

b) OAuth authentication flow is started, as described in step 1 and 2, the consumer request a request token from the broker. This request token actually is a proxy request token.

c) At step 3, the consumer redirects the customer to Broker, the OAuth extension starts to look up related authorization rule, if no rule found, the broker will ask for the user's approve on this request token. If the rule is exist and valid, then the customer will be directed back.

d) What's more, in step 4 and 5, there is a server side browser to act as the consumer role in OAuth to play with the real service provider. This step will get an authorized request token from the service provider.

e) At step 7 and 8, the consumer requests the access token by the authorized request token, the proxy still works to return a proxy access token.

f) Once the consumer obtains the access token, Accounting Service will initiate the service call to the broker, as described before, there is a fine-grained access control for this invocation, if everything goes well, then the proxy will request the data on the service provider.

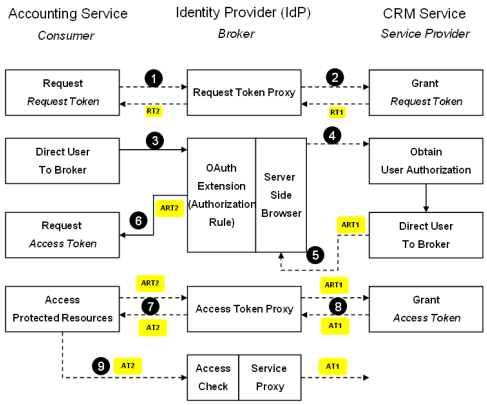

Figure 6. Secure Service Call Flow

V. CONCLUSION

As user-centric digital ID, OpenID is a simple, light-weight identity management system. With some anti-phishing method, this protocol can be introduced to the SaaS ecosystem. For service delivery platform, the OpenID provider is a lightweight and standard-based identity provider so he does not need to provide extra development packages and guides for service providers. For service provider, it's a good choice for using an open and standard protocol to reduce the identity management work. For end user, he has more control on his identity usage.

To provide an easy-to-compose service platform, each service provider should publish and consume the services on this platform directly instead of looking up and contacting each service owner. So the platform needs to check, log and mediate each service call, the central management solution could provide a clear business bill. OAuth protocol is an open and standard protocol to enable one service application to access the protected resources on another service application without disclosing the user's credential. So once the service provider and consumer implement the OAuth protocol, they can easy integrate together. What's more, the platform adopts a mediation method to help the consumer integrate with the service provider rapidly.

Open is a good design principle, and it is also the attitude and sprit of collaboration. We think that a SaaS 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, fine-grained authorization based on open protocol will be considered. What's more, we are still working on the research and implementation on user-centric federated identity management system for SaaS ecosystem.

REFERENCES

[1] George Lawton, Developing Software Online with Platform-as-a-Service Technology.

[2] Suriadi Suriadi, Ernest Foo, Audun Jøsang, A User-centric Federated Single Sign-on System.

[3] Alisoft, http://wiki.isv.alisoft.com/index.php?tracelog=doc_from_home.

[4] Salesforce.com, http://www.salesforce.com/us/developer/docs/api/index.htm.

[5] HwanJin Lee, InKyung Jeun, Kilsoo Chun, Junghwan Song, A New Anti-Phishing Method in OpenID.

[6] OAuth,http://oauth.net/.

[7] WS-Trust, http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html.

[8] WS-Federation, http://msdn.microsoft.com/en-us/library/bb498017.aspx

[9] P.Austel S. Bhola, S. Chari, L. Koved, M. McIntosh, M. Steiner, S. Weber, Secure Delegation for Web 2.0 and Mashups.

[10] OpenID Phishing Brainstorm.http://wiki.openid.net/OpenID_Phishing_Brainstorm.

[11] Tivoli Federated Identity Manager, http://www-01.ibm.com/software/tivoli/products/federated-identity-mgr/.

[12] Salesforce.com,Security Implementation Guide, https://na1.salesforce.com/help/doc/en/salesforce_security_impl_guide.pdf

[13] OpenID OAuth Extension http://step2.googlecode.com/svn/spec/openid_oauth_extension/latest/openid_oauth_extension.html

[14] David Recordon , Drummond Reed,OpenID 2.0: A Platform for User-Centric Identity Management

[15] Abhilasha BhargavSpantzel, Jan Camenisch , Thomas Gross, Dieter Sommer ,User Centricity: A Taxonomy and Open Issues.

[16] Bringing OpenID and OAuth Together, http://googledataapis.blogspot.com/2009/01/bringing-openid-and-oauth-together.html.Bringing OpenID and OAuth Together, http://googledataapis.blogspot.com/2009/01/bringing-openid-and-oauth-together.html