An Architecture for User-Centric Identity, Profiling and Reputation Services

Gennaro Costagliola, Rosario Esposito, Vittorio Fuccella, Francesco Gioviale
Department of Mathematics and Informatics
University of Salerno
{gencos,vfuccella,cescogio}@unisa.it, scalax.uep@virgilio.it

Abstract

This paper presents a work in progress whose objective is the definition of a novel architecture for solving several challenges related to Web navigation, such as accessing to multiple Web sites through a single identity and verifying the identity and the reputation of a peer involved in a transaction. The proposed model tries to solve the above challenges in an integrated way through the introduction of a specialized Web Mediator acting on behalf of the user during usage of the Net, identity providers for identity data centralization, and a two way negotiation system among parties for mutual trust.

1. Introduction

The need for introducing new functionalities to improve the user Web experience is more and more widely felt. Lately, researchers are closely taking into account the following important issues:

1. Registering and accessing to multiple services using a single identity for all services (single sign-on systems);

2. Verifying the identity and the reputation of a peer (user or organization) involved in a transaction;

3. Keeping the property and control of personal information such as: user profile, reputation, etc;

In this paper we propose an architectural model aimed at pursuing the above objectives trough the introduction of a *Web Mediator* (*WM*) acting on behalf of the user during Web navigation and an *Identity Provider* for the identity data centralization. The former is responsible for maintaining user personal data and profile to use in content personalization (as similarly done in [1]). The latter is responsible for keeping user identity and reputation data, and to vouch for the user in registration and authentication procedures.

Our model enables a two way negotiation system among parties for mutual trust: in a transaction both parties can mutually authenticate and verify reputation and profile. This sort of handshake, will allow them to decide whether the transaction can go on or should stop. It is worth noting that despite adding new functionalities to the actual Web application interactions, the architecture works with the actual Web protocols.

The advantages deriving from the availability of a solution to the three issues mentioned before are evident in several scenarios occurring daily during Web navigation. For instance, mutual trust is useful in the detection of phishing: let us suppose a user receives an e-mail containing a link to an important document about his/her bank account stored on the bank Web site. By connecting to the link with our framework enabled, the user can both check whether the remote Web server supports the architecture and verify its credentials. The phishing attempt can be immediately detected in the former case and after a reputation check in the latter case. The availability of user profile and reputation is useful in many cases: i.e., profile is used for offering personalized services, reputation in on-line auction services. Their availability to the user is advantageous since: data are already available when a user starts requesting a service at a new provider (it is not necessary to wait for a new profile or reputation to be built); the user is owner of his/her personal data which can be used with different sites offering the same services.

The above mentioned issues have been faced separately so far, that is, to our knowledge, there are no proposals of a generic architecture offering a solution for them all in literature. I.e., platforms for single sign-on [6] trust and reputation management [3] are available, as well as methods for preventing phishing [5]. In order to propose a unique solution to the above challenges, we have decided to extend a well established *SSO* platform, *OpenID* [6], with the support of a mutual trust establishment procedure. In particular, we have extended the *OpenID* Authentication procedure. The interaction among user's and peer's modules involved in the procedure are described through the paper.

In our prototype, the Web browser can communicate with user's *WM* through a special plug-in.

The rest of the paper is organized as follows: in section 2, we introduce the *OpenID* platform; the architectural model, including a detailed description of the involved entities and their interaction model, are presented in section 3. In section 4, we will describe the implemented prototype and its instantiation in a real-life application scenario. Final remarks and a discussion on future work conclude the paper.

2. The OpenID Platform

OpenID was firstly developed in 2005 as a user-centric and URI-based identity system. Its main objective was to support the *SSO* functionality. The initial project has grown and has evolved in a framework enabling the support of several functionalities which can be added to the basic platform.

The *OpenID* architecture components are: the user, the remote Web-server (also know as *Relying Party*) where the user wants to authenticate and the *Identity Provider* (*IdP*) that provides vouch for user identity certification. *OpenID* has a layered architecture. The lower layer is the *Identifier* layer. This layer provides an unique identifier for address based identity system. The address identifier (*OpenID* URL) is used by the *Relying Party* (*RP*) to contact the user's *Identity Provider* and retrieve identities data. Both URL and XRI [7] address formats are supported as identifiers.

The above layer is the service discovery layer. It is implemented trough the *Yadis* protocol [4]. The purpose of this layer is to discover various type of services reachable through an identifier. In the case of *OpenID* it is used to discover the *Identity Provider* location.

The third layer is the *OpenID Authentication*. The main purpose of this layer is to prove that an user is the owner of an *OpenID* URL and, consequently, of the connected user data.

The fourth layer is the *Data Transfer Protocol*. This protocol is used to transmit user related data from the *IdP* to the *RP*. In *OpenID Authentication 1.1* this layer is implemented through the *SREG* protocol (*Simple Registration Protocol*), which allows the transmission of simple account related data [2]. Currently, the *OpenID* research community is defining a new version of the protocol capable to transmit various type of data other than identities related one.

3. The architecture

In this section we give a description of the proposed architectural model, including the involved entities and their interactions in a trusted negotiation, which is a typical interaction where two parties gradually establish trust [8]. It is based on the previously described *OpenID* platform, and extends it to support the features outlined in the introduction.

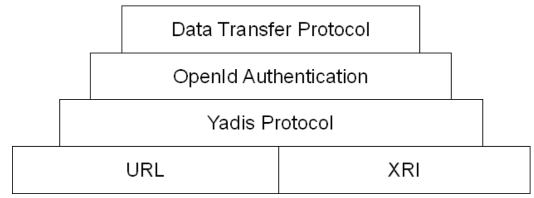

Figure 1. The *OpenID* layered architecture.

Our model extends the *OpenID* platform by enabling the establishing of mutual trust and the exchange of reputation and profile data between two parties. In particular, it adds *Profile* and *Reputation* layers upon the uppermost *OpenID* layers and a *Mutual Trust* layer above them (Fig 2).

Reputation management service is provided as an extension of the *DTP* layer. In particular, the data model supported in the information exchange occurring at this layer is extended with reputation data. The discussion on how to represent, create and manage these data are out of the socpe of this paper and will not be treated here.

User profile data are managed by the *WM*, which also works as a profile provider, and can be accessed only after the *OpenID* Authentication procedure is successfully completed.

The *Mutual Trust* layer implements the handshake procedure that will authorize the user application to proceed with an interaction after identity, reputation and profile of remote peer are checked.

In a typical scenario, our architecture is composed of the following components:

A) The Web Browser equipped with a specific plug-in (i.e. a Firefox add-on) to communicate with the *WM*;

B) A *Web Mediator* (*WM*): the software module responsible to communicate with other remote peer *WMs*, in order to perform a trusted negotiation. The *WM* can perform two functions: issue a transaction request to remote peers *WMs* or receive incoming transaction requests from remote peer *WMs*. In the case it is the first to send a request will refer to the *WM* as *User Web Mediator* (*UWM*); otherwise we will refer to it as *Remote WEB Mediator* (*RWM*). More in details, a *WM*, by referring to a preference table set by the user, verifies the identity, reputation and profile of remote peers and, after that all checks are passed, it authorizes the application to proceed with the transaction. Furthermore, in scenarios that needs this feature, it also checks that the resource retrieved as a transaction result fits user's preferences (i.e. content filters).

C) An *Identity Provider* (*IdP*), deployed on a third party server, that is responsible for guaranteeing the veracity of the credentials issued by the *WMs*; it is also responsible to

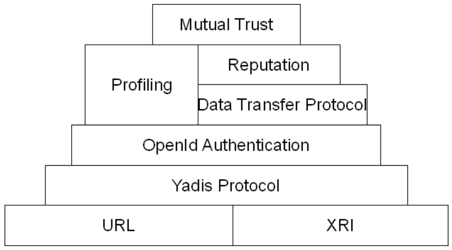

Figure 2. The proposed architecture.

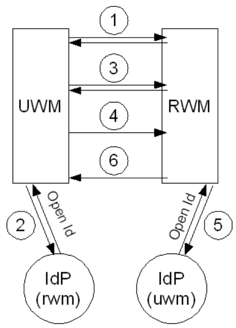

Figure 3. The WM Handshake

provide, by extending the common data already passed during an *OpenID* authentication, the reputation data.

D) The remote application that provides the requested resource after being authorized to do so from the *RWM*.

Before we start to discuss the fundamental phases that occurs in a transaction we will describe the *WM Handshake* procedure between *WMs* in which *UWM* and *RWM* proceed to establish a mutual trust with the help of one or more *IdPs*. During this phase the *WMs* exchange profile and reputation data and verify that the user parameters are satisfied. More in details, as shown in figure 3:

1. *UWM* requests the *OpenID* URL to the RWM and receives it;

2. *UWM* starts the authentication procedure by contacting *RWM*'s IdP which authenticates *RWM* and replies with the *RWM*'s reputation data;

3. *UWM* recovers *RWM*'s profile data trough a GET request to the *RWM* using a standard URL;

4. *UWM* checks the received profile and the reputation data and, if all checks are passed, sends its *OpenID* URL to *RWM*;

5. *RWM* starts the authentication procedure by contacting *UWM*'s IdP which authenticates *UWM* and replies with *UWM*'s reputation data;

6. *RWM* recovers *UWM*'s profile data trough a GET request to the *RWM* using a standard URL;

7. *RWM* checks the received profile and the reputation data and, if all checks are passed, sends an OK message to *UWM*.

The authentications in step 2 and 5 follow the *OpenID* protocol and consist of sending username and password to the *IdP* (through a POST request) to prove to be the owner of the identity related to the previously sent *OpenID* URL.

For sake of clarity no exceptions are shown in the procedure. In the case something goes wrong, the *UWM* is the one in charge of notifying the user application that the handshake did not succeed.

Note that, by following the previous steps, *UWM* is the first to see the other's reputation and profile data. Furthermore, the *RWM* will be able to access to the *UWM* data only if it is considered worth to receive it. This is the *UWM-first* version of our architecture. The *RWM-first* version is easily obtained by letting the *UWM* start sending its own *OpenID* URL and modifying the next steps accordingly.

In the following, we describe the complete transaction between two Web applications (user and remote applications) by following the *UWM-first* approach (the other case can be easily derived). More in detail, as shown in figure 4:

1. the user makes a request to the application to execute a transaction with a remote application;

2. the user application contacts its *UWM* to obtain an authorization for the transaction;

3. the *WM* Handshake between the corresponding *UWM*, *RWM* and *IdPs* occurs as described above;

4. if the handshake succeeds, the *UWM* sends the shared *RWM OpenID* authorization token to the user application;

5. the user application sends its original request together with the authorization token to the remote application;

6. the remote application uses the token to query its RWM for the identification and profile of the requester (as built with the *UWM*);

7. the *RWM* returns the required resource; from now on the transaction between the two applications does not involve the underlying levels.

In the case the *WM* Handshake does not succeed, the user application, based on its configuration, may decide whether to start or not a traditional transaction with the remote application. In fact one of the advantages of this approach is that it does not alter the current Web model.

In our lab, we have built a basic prototype implementing the procedures above in the context of OpenID and applied it to the case of browsing a simple web application.

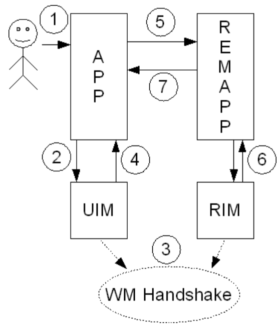

Figure 4. The general architecture.

4. The Online Auction Websites case study

In this section we will show how our architecture can be easily instantiated to a real-life application.

4.1. The case

Alice is an Ebaia power seller with a positive feedback rate of 99%. Thanks to her excellent reputation, Alice reaches big sales volumes. During the Web surfing, Alice finds a new online auction system, called Xbid that offers more convenient commissions on sales. Alice, interested by the offer decides to test the new system but then she finds a serious obstacle: there are no ways to migrate her excellent reputation data (that builds up in a long time span) from the current system to the new one. Discouraged, she decides not to try Xbid.

The adoption of our model, thanks to the relocation of the reputation data on an Identity Provider, allows the user to access to more online auction systems, even at the same time, increasing the seller presence on the market. Also, due to the centralized reputation data, users can compare sellers on different auction platforms allowing a deeper level of filtering. Last but not least, due to the buyers' certified identity, the seller is able to exclude malicious users that can alter the auctions.

4.2. The implementation

The user application is the Web browser (the buyer's one, in this case) and the remote application is the auction system Web server that will request to the seller *RWM* the autorization to proceed with the transaction. The seller *RWM* will be identified by the *UWM* due to a metatag link present in the product page as usually done with *OpenID* delegation. The transaction steps are then so instantiated:

1. the user selects the 'buy now' option;
2. the browser contacts the user *UWM*, through a plugin, to obtain an authorization for the transaction;
3. the *WM* Handshake occurs;
4. if the handshake succeeds, the *UWM* sends the shared *RWM OpenID* authorization token to the browser;
5. the browser sends the 'buy' request together with the authorization token to the auction web system;
6. the auction system uses the token to query its *RWM* to receive the authorization for the incoming request;
7. the auction system shows the payment procedure to the user.

5. Conclusions

In this paper we have presented an architecture for improving some aspects related to Web navigation. The work is still in progress and, due to the complexity of the different addressed issues, many aspects are still to be investigated: some scenarios have been outlined and the architectural model has been presented and tested in one of them. As future work, we plan to test the architectural model in many other scenarios and contexts.

References

[1] A. Ankolekar and D. Vrandečić. Kalpana - enabling client-side web personalization. In *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 21–26, New York, NY, USA, 2008. ACM.

[2] J. Hoyt, J. Daugherty, and D. Recordon. Openid simple registration extension 1.0. June 2006. http://openid.net/specs/openid-simple-registration-extension-1_0.txt.

[3] A. J, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[4] J. Miller. Yadis 1.0. March 2006. http://yadis.org/papers/yadis-v1.0.pdf.

[5] Y. Oiwa, H. Takagi, H. Watanabe, and H. Suzuki. Pake-based mutual http authentication for preventing phishing attacks. In *18th International World Wide Web Conference (WWW2009)*, April 2009.

[6] D. Recordon and D. Reed. Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA, 2006. ACM Press.

[7] D. Reed and D. McAlpin. Extensible resource identifier syntax 2.0 (oasis xri committee specification). November 2005. http://www.oasis-open.org/committees/download.php/15377.

[8] A. C. Squicciarini, A. Trombetta, E. Bertino, and S. Braghin. Identity-based long running negotiations. In *DIM '08: Proceedings of the 4th ACM workshop on Digital identity management*, pages 97–106, New York, NY, USA, 2008. ACM.