Collaboration and the Knowledge Economy: Issues, Applications, Case Studies
Paul Cunningham and Miriam Cunningham (Eds)
IOS Press, 2008 Amsterdam
ISBN 978–1–58603–924-0

Design and Reference Implementation of a Federated Single Logout

Luis MELÉNDEZ[1], Diego LÓPEZ[2] , Victoriano GIRALT[3], Sergio GÓMEZ[4]
[1]*University of Córdoba, IT Services, Córdoba, 14071, Spain*
Tel: +34 957 211022, Fax: +34 957 218116, Email: luism@uco.es
[2]*RedIRIS, Avda. Reina Mercedes s/n, Sevilla, 41012, Spain*
Tel: +34 95 5056621, Fax: +34 95 5056651, Email: diego.lopez@rediris.es
[3]*University of Málaga, Central Computing Facility, Málaga, 29071, Spain*
Tel: +34 95 2132366, Fax: +34 95 2131492, Email: victoriano@uma.es
[4]*University of Córdoba, IT Services, Córdoba, 14071, Spain*
Tel: +34 957 212132, Fax: +34 957 218116, Email: sergio@uco.es

Abstract: Web Single Sign On (SSO) deployments and Identity Federations, though able to improve user experience and security (no more many passwords to remember, just one place to provide credentials, ...) have also some risks, and maybe the most important of them is associated to situations when the user does not completely close the session when leaving the computer. Both the sessions that are started with applications and the one with the Identity Provider or Authentication Server must be closed to avoid someone else being able to open applications taking over the user's identity. The Single Log Out (SLO) function is at least as important as the SSO one, but there has been much less effort and research on it. We present one way to implement this functionality that is easy, flexible and compatible with existing AAI and SSO systems.

1. Introduction

When an institution participates in an identity federation [1] its Identity Provider (IdP) normally uses the services of a local WebSSO system (such as CAS [2], PAPI [3], etc.), which in turn will have its own Authentication Server (AS). So, the user may have sessions opened with different Service Providers (SPs) of one or more institutions as well as with other local services. After ending a working session, any user will like to close all of these sessions, as well as the session with the IdP and the AS, so that no other person can use the browser to open new sessions without being required to present any credential.

Given the fact that Single Sign On (SSO) systems automatically provide the authentication information that an application needs in order to grant access, maybe the user ends up not knowing which of the services contacted along a working session have required authentication and which of them have not, so the user is not aware of what applications must be closed. It is usual in corporate environments that many applications share the same look and feel, so the user is not aware of when a 'restricted' zone requiring authentication has been entered, because the SSO system has provided it automatically. We consider this fact a specifially dangerous effect of SSO systems when there is no SLO facility implemented.

An important measure against this danger is to include a clear visual clue indicating the user that the browser is in an authenticated state and should log out before leaving, but this is not always implemented.

The main result of [4] is that when a multiservice environment uses SSO for user authentication, a single logout (SLO) should also be used instead of expecting users to

separately log out from each service. We think the same holds true for interinstitutional services such as federations.

We face several problems here:

- How to keep track of the applications opened by the user.
- How to close all of them automatically (or almost).
- How to close every particular session.
- How to close the session with the IdP or AS.

We present an approach to face all of these issues, although the more important and innovative solution lies in the way the sessions opened by the user are tracked and the way to close them. The description provided is concise.

2. Context

There exist several implementations of SLO, but no one is perfect. SAML 2.0 [5] introduces a profile for it, but it will be available only in concrete implementations of its specifications. In this moment there are many deployments of AAI and SSO technologies that lack a SLO facility. Our proposal is aimed to integrate many of these, not only as an interim solution but also as a test bed for some of the concepts and functionality that a SLO system should have.

3. Objectives

We wanted to design a single logout system that could be easily deployed and as little intrusive as possible in pre-existing application or federation code and configurations. We also wanted to explore new ways to expose the closing of open sessions to the user.

That is, the two main concerns for us were that the system should be:

- Simple to implement.
- Applicable to most of the federated identity software in use today.

This is with no doubt an initial step towards a complete solution, since we admit that it has shortcomings that need to be addressed. Anyway it is an already usable option for sites not able to implement another approach or even that prefer the benefits of this one.

4. Description

Our proposal uses two simple Web services that are contacted by the user agent and by other elements of the AAI. Each has, of course, its own URL. This approach splits the complexity and allows for better flexibility, as we will show later.

The first service is the so-called BAR (Browser Activity Registry). Its role is clearly denoted by its name: it associates an opaque identifier with the federated applications the user visits during the working session. It can be contacted by several elements in the AAI, depending on the complexity of the modifications necessary in every element, the pros and cons of every option, etc.

The second service is the SLO, and it is where the user is redirected when log out is requested. It will guide the user through the closing of all the applications that have been opened.

Those services (we have developed reference implementations of both of them) can be deployed in any AAI, since they are independent of any other element. The BAR has not any interface visible to the user, and it is only contacted by other services and by the user agent just to load some element that can set a cookie. The SLO has a user interface, so its appearance can be customized.

4.1 Tracking access to the resources

The sequence of events in a typical Shibboleth [6] deployment can be:

1. The user tries to access a federated application (SP)
2. This redirects the user to the WAYF service to select its local IdP

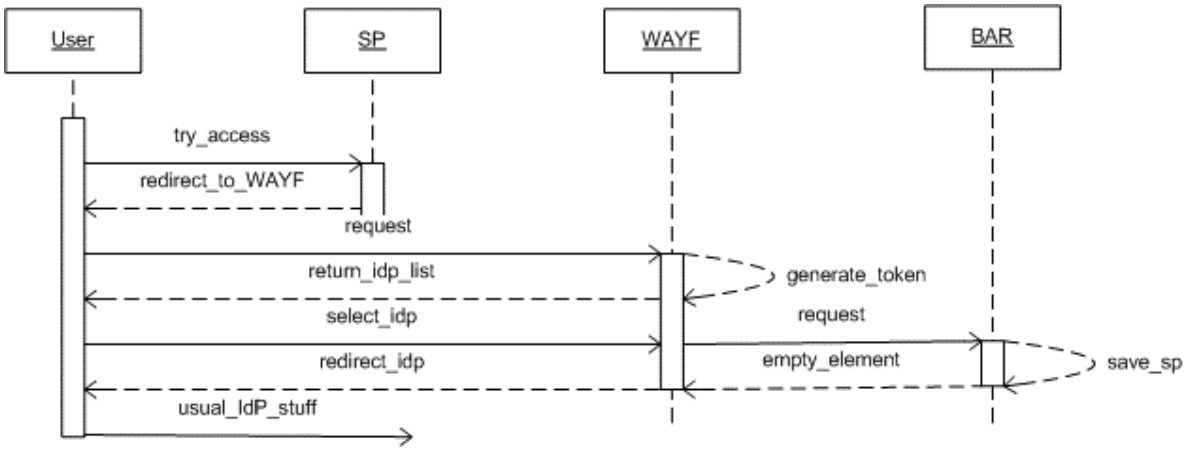
The WAYF:

3. Determines that the user has not previously selected an IdP (a cookie is absent, for example)
4. Generates an opaque identifier (token) that may be just a random number.
5. In the user interface where the IdP is selected, it includes an HTML element whose SRC attribute is a URL of the BAR with both that token and the URL of the SP as query parameters.

Upon receipt of the query generated by the user agent for that element, the BAR:

6. If the browser does not present the cookie called BARtoken, it sets it with the value of the token query parameter. If the cookie is present (subsequent accesses) the token query parameter is ignored.
7. Inserts the URL of the SP in its database, the token and the time stamp.
8. Returns an empty image, an empty script.
9. The WAYF presents the list of IdPs for the user to select the appropriate home institution and does the usual job of this service.

An approximate sequence diagram is:



When the user goes to another SP that uses the same WAYF service, the sequence is the same, except that the WAYF service usually does not display again its interface and assumes the same IdP as before (this is indicated by the presence of some cookie). So to force the access to the BAR the WAYF, instead of redirecting immediately to the IdP, returns a simple HTML page that includes a link to an element with SRC attribute pointing to the BAR (like before) and a little javascript that forces the redirection to the IdP as soon as this little page is loaded.

If the user accesses later a different WAYF service or an application that uses an internal WAYF (but has been modified to integrate within the environment we describe in this paper), it generates a BARtoken, but it is not actually used:, When the browser contacts the BAR, sends a previous BARtoken cookie and uses it in its record and not the one that presents the browser in the query parameter.

Since the key to tracking the applications contacted by the user is an opaque identifier, there is no way to link those activities to a person, thus privacy is actively preserved. In the simplest case, such opaque identifier is generated each time by the WAYF, although only the first is used, because the BAR sets its cookie with it.

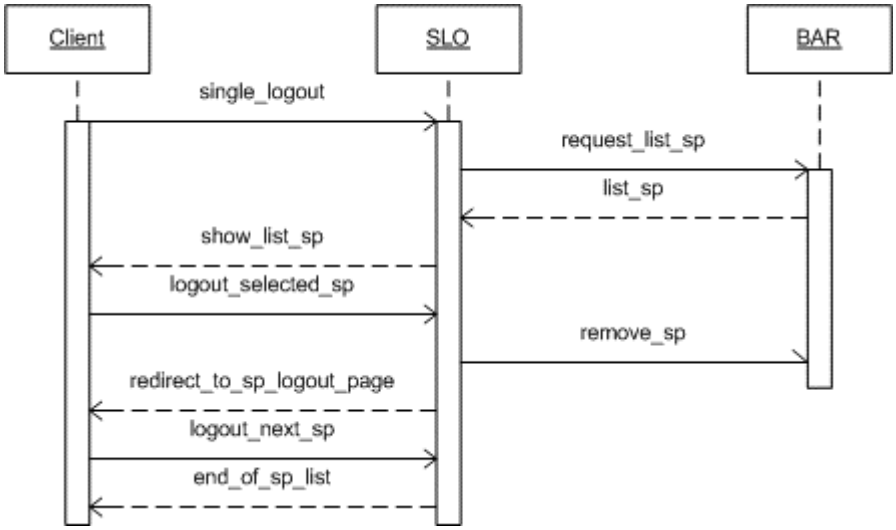
4.2 The Logout Phase

When an user starts to logout, it is very probable that it is intended to do so from all other applications and from the federation itself, and it is good to let the user decide, but it is not good to provide two links, one for exiting the application and another for exiting all of them.

Our proposal is: the logout link of the applications should be changed so that it redirects to the BAR service (with a URL indicating a Logout), which will redirect to the SLO passing it the value of the BARtoken. This first redirection to the BAR is necessary because the BARtoken cookie was set by it. Another option would be to include the SLO functionality in the BAR service. The SLO service presents the user the choice of closing only this application or all of then opened plus the federation association. The user has also the option to select which applications must be closed. The SLO service asks the BAR for the list of SPs visited (using the BARtoken cookie).

If it is not possible or convenient to change the logout links of the applications, the user can be instructed to contact the SLO service directly, as we will show later.

The approximate sequence diagram:



4.3 Sequentially Closing the Sessions

Our proposal for this is simple: not to do it fully automatically. Some applications, like the Moodle e-learning suite for example, asks the user for confirmation after clicking the Logout link. The logout can also be performed using the Logout service of the SP software that protects the application, but this does not allow it to perform the administrative tasks that it should. The logout of a Web application running in a federated environment is a field in which some research has to be done.

So the SLO service presents the user two frames: a bigger one where the closing of the application can be seen and it is possible to interact with it if necessary, for example to confirm the logout. This also serves the purpose of making the user well aware of the closing. Another smaller frame is used for navigation: one button takes the user to the next application to close.

If it is not possible or convenient to change the logout links of the applications, the user can be instructed to contact the SLO service directly, where the applications to close can be selected.

The SLO will contact the BAR to delete the records for the applications the user decides to close, so that a later visit to the same service will show only those that remain open.

For the SLO service to know the target URLs where every application should be contacted to perform the real logout, there are several options. There can be a configuration file, or a SAML 2.0 Metadata file which includes SingleLogoutService element can be used, substitute the /SAML/POST part of the SP with /Logout, etc.

This is also a field of experimentation and improvements.

5. Implementation and deployment

Our initial implementation, just enough for testing the concept, is composed of the following elements:

- Slight modifications to the WAYF service (in our case, a homemade one), no more than 20 lines of PHP code.
- Two simple web applications, the BAR and the SLO. In the very simple initial version, they consist of no more than 60 and 190 lines of PHP code.
- Modification of the logout link of the applications protected by an SP so that they point to the SLO application.

Even in the case of a more elaborated implementation, and without considering other options (see the next section) a basic deployment could simply be done following those same steps. Of course, we will release the source code and detailed instructions as soon as it becomes mature enough.

6. Future Work

In our initial implementation, is the WAYF the one contacting the BAR service, because we considered it was the easiest way to implement the architecture concepts. The connection to the BAR can also be made by the IdP after a successful authentication, but even so the user may be not authorized to access de application.

The best place could be at the SP, after it has authorized the access. But this can be easy to implement in some cases and very difficult in others. If we cannot adapt all SPs to contact the BAR, we must still let the WAYF service (or equivalent) to do it. If we have an SP adapted to know about the BAR, it should contact the BAR to delete the record placed by the WAYF in the case of an access that is not authorized.

What we have to do then is to further develop these concepts, explore the possible usage scenarios, etc.

7. Conclusions

The system presented here is not perfect, but in some small to medium environments it allows for the relatively easy implementation of a fully functional Single Log Out facility.

Quoting one of the SAML standards: "All that said, it is not intended as a panacea, but simply an alternative to fill another deployment niche"

Talking about the user experience, we propose a way for the user to be visually aware of the closing of every service, and since this process is not totally automatic, she has to click on a button to go through all of them. It can be regarded as uncomfortable, but it is not really so much, and the user is more aware that all gets closed.

We think this is matter that is more subjective than technical, and would like that our proposal promotes some discussion about these usability-related items.

We plan also to do some more research with real users to get feedback about the way they prefer: more automatic or more awareness.

Our system has been implemented and tested in the federated identity infrastructure for the Andalusian Universities, and the user feedback received so far has been positive. As the infrastructure is deployed, we will be able to expose a wider user base to the system and get a better assessment about its usage in the real world.

References

[1]. S. S. Y. Shim, G. Bhalla, V. Pendyala - Federated Identity Management. In: Computer, Vol. 38, No. 12- IEEE Computer Society, December 2005, pp. 120-122

[2] S. Bramhall - Understanding uPortal Security and the Yale Central Authentication Server. 8th JA-SIG Conference,Westminster, Colorado, June 2003

[3] R. Castro-Rojo, D. R. Lopez - The PAPI system: point of access to providers of information. Computer Networks, Volume 37, Issue 6, December 2001, pp 703-710

[4] M.Linden and I.Vilpola, - An Empirical Study on the Usability of Logout in a Single Sign-on System. In: Information Security Practice and Experience. Springer Berlin / Heidelberg, 2005, pp 243-254

[5] Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML). OASIS Standard, September 2003. Available at: http://www.oasis-open.org/committees/download.php/3405/oasis-sstc-saml-bindings-profiles-1.1.pdf

[6] S. Cantor (editor) - Shibboleth Architecture. Protocols and Profiles.10 September 2005. Available at: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-200509.pdf