Logout in Single Sign-on Systems

Sanna Suoranta, Asko Tontti, Joonas Ruuskanen, Tuomas Aura

Logout in Single Sign-on Systems

Sanna Suoranta[1], Asko Tontti[2], Joonas Ruuskanen[1], and Tuomas Aura[1]

[1] Aalto University, Department of Computer Science and Engineering
[2] CSC - IT Center for Science
Espoo, Finland

Abstract. Single sign-on (SSO) helps users to cope with many online services that require authentication. Systems such as OpenID and SAML-based Shibboleth offer federated identity management where an Identity Provider authenticates the user on behalf of the services. Much research concentrates on making authentication stronger, preventing phishing and making the systems more user friendly but less attention has been paid to the termination of the authentication sessions i.e. logout. It is, however, equally important that the sessions do not remain open when, for example, a student using shared computers in a university library leaves the workstation. In this article, we describe challenges related to logout in federated identity management on web based services and give guidelines for implementing reliable logout from services that use single sign-on.
Keywords: Single Logout, Logout in Single Sign-On Systems, Shibboleth

1 Introduction

Nowadays both at work and in leisure time people are using many services that have a web browser as the user interface and platform instead of stand-alone applications. Often the services require the user to authenticate in order to provide personalized service, to access to the user's private information or to verify the user's real-world identity. Usually user authentication is based on passwords, but many users choose poor passwords [10] or share the same password between unrelated services [11]. These issues weaken the authentication.

The single sign-on (SSO) systems help users to access several services using a single password; thus, the user does not need to remember service specific passwords. In a federated SSO, the user authentication is separated from the service to be handled on a separate Identity Provider (IdP). Two common systems are OpenID [17] and Shibboleth [21], which is based on SAML [20]. These are mainly used locally or for specific online services. No dominant world-wide SSO system exists as the service providers do not want to outsource their user management [8]. The SSO technologies are an active field of researc and development. Impovements have been proposed to the architecture [4, 5, 23, 27], authentication strength [16, 26] , usability [14, 24, 25], and privacy [1]. On the other hand, less attention has been paid to the termination of the authentication sessions, even though it is a critical part of the authentication process.

SSO systems are common in universities where, in addition to the university itself, also external organizations can offer web-based services for students and staff. For example in Finland, the universities have a common library service that provides access

to online electronic publications using Shibboleth for authentication. The university students often use shared computers; thus the proper logout from the services is very important so that the next person using the computer cannot gain access to the services using the first one's privileges.

In this article, we have investigated the logout and its implementation in services that use the Shibboleth SSO. We found many problems in the logout procedures: none of the services implements single logout, and usually at least some server side sessions are still active after logout. Based on our findings, we give a list of actions that help in implementing more secure and usable logout for Shibboleth. Some of the solutions are general and could be used to improve also other SSO systems.

The rest of the paper is organized as follows. Next, we discuss single sign-on systems in general, and OpenID and Shibboleth in detail. In Section 3, we go through how logout works in SSO systems in theory, and in Section 4 how it works in practice in the web services of Aalto University. Then we discuss the solutions for logging out in the SSO systems. In Section 6, we describe what the other researchers have done. Then we list the improvements that should be implemented and, finally, conclude the article in Section 7.

2 Single Sign-on

Single sign-on systems allow users to log in to several services with one authentication. The SSO systems can be divided into four categories: local and proxy-based, pseudo and true SSO systems [18]. In the local pseudo-SSO system, a user's device has a password manager that automatically works on behalf of the user when she is logging into a service. The proxy-based pseudo-SSO system is similar, but the password manager is located on a network proxy. The local true SSO system rely on trusted component on the client device, but otherwise is quite similar to the local pseudo-SSO. The proxy-based true SSO system has an external server for authenticating the user for services. In this article, we concentrate on proxy-based true SSO systems, often called federated SSO. For example, OpenIDand Shibbolethare proxy-based true SSO systems. In addition, many service providers allow other web service developers to use their user authentication with their proprietary protocols. For example, Facebook Login [9] and Microsoft's Windows Live ID [15] provide centralized authentication for third party web applications where users can log in using their existing identities at Facebook and Microsoft's services.

Both OpenID and Shibboleth separate the user authentication from the actual service. Figure 1 depicts the general architecture of a federated SSO system. When a user wants to use a service, she first starts her web browser and connects to the service that is running on a server of a Service Provider (SP). A SP can have several services on the server, and some of them can take care of their own session management but often the SP saves the session information on behalf of the service. The SP responds to the unauthenticated user by redirecting the browser to an Identity Provider (IdP), or if there are several IdPs, the user can choose to which one the connection is redirected. The IdP authenticates the user, creates a SSO session for her, and redirects the user's web browser back to the SP with authentication assertions. Then, the SP saves the session

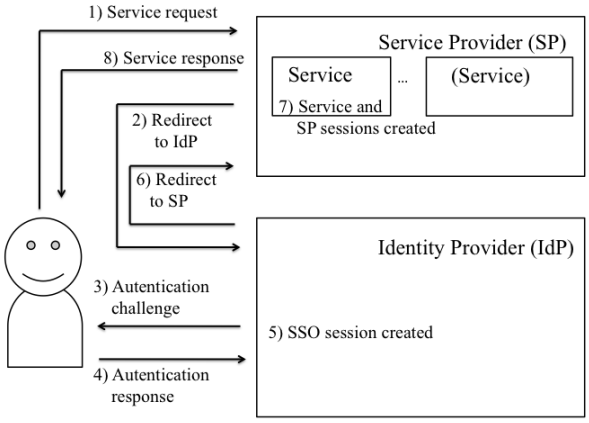

Fig. 1: Architecture of a federated SSO system

information and gives the originally requested access to the user. If the user wants to use another service with the same authenticated identity, the IdP does not need to re-authenticate her. It only checks that the valid SSO session exists, and then it gives the authentication assertions directly to the second SP. However, the IdP does not keep track to which SPs the user has been authenticated. Next, we describe shortly the differences of OpenID and Shibboleth.

2.1 OpenID

OpenID allows user to choose any identity provider she wants. There are plenty of OpenID identity providers [28] but only a few services that accept other than their own IdP.OpenID does not require pre-established trust between SP and IdP [22], and that might be one reason why it has not gained worldwide acceptance as a service authentication solution even though many services use it for access control with their own IdP. Moreover, popular OpenID identity providers such as Google do not verify the user's identity in the registration phase, only that the user has a valid email address [5]. However, strong user authentication is possible, e.g. in Estonia, a mobile phone operator acts as an IdP that provides strong verified authentication [12].

2.2 Shibboleth

Whereas OpenID requires no prior trust relationship between IdP and SP, Shibboleth participants need to trust each other. However, the IdPs and SPs do not need to belong to a same organization, they just need to share common metadata declaring the trusted IdPs and what attributes each service needs. IdP and SP communicate using SAMLmessages through user's browser using HTTP redirect and post command. Authentication and service sessions are stored in cookies on the cliend side.

In Finland, all the universities have joined to a federation called HAKA [6] that handles federated identity management. The universities have their own IdPs and a separate organization, namely CSC that provides high performance computing facilities and Internet access to universities, manages the federation and its metadata on behalf of the universities. The HAKA federation and its services have worked well for reducing the number of necessary passwords. Furthermore, the federation has joined together with the other Nordic federations providing access to research resources over the borders in the Nordic countries [13]. We have chosen to investigate how the services using the Shibboleth SSO manages the logout.

2.3 Implementing the Shibboleth SSO

Shibboleth Consortium [21] offers open source implementations for selected SAML2 profiles as Shibboleth IdP and SP. Their Shibboleth IdP is a Java based server that works on top of Tomcat [2] Java Servlet platform. It offers several ways to authenticate the users: e.g. username and password query, remote user, and previous session. The first and second methods are for initial phase user authentication and the third allows the actual single sign-on after the user's identity has once been verified. The first asks username and password from the user and checks that they are correct using Java Authentication and Authorization Service (JAAS) interface to connect to a LDAP directory (this method is used in Aalto University's IdP) or Kerberos server. The second method uses a web server, such as Apache, to authenticate the user instead of the IdP program. The third method uses authentication cookies that are stored in the web browser after previously and successfully done user authentication. This allows user to log in to several services with single authentication.

The Shibboleth SP is usually connected to web server as additional module or filter. The web server can be e.g. Apache or Microsoft ISS. The web server mediates the authentication information to the web service application that offers the actual service for the user. The web server and SP pair can host several service applications. There are two basic ways for the service to use Shibboleth for authentication. The service taking care of its own user information management during the session can use Shibboleth for user authentication. The other way is that the service outsources all user session management to the SP.

3 Logout in SSO Systems

Logging out means clearing the sessions that are created when the user is signed in. In a traditional web service, the user session is usually terminated either when the user explicitly presses logout button or when the service timeout is reached. In the SSO systems, logging out is not so straightforward. The user might want to log out either from a single service or from the whole SSO session. From the SSO point of view, there are even more forms of logout, e.g. Shibboleth and SAML have the following:

– *Logout in the service application*. If the application takes care of its own user session management, it terminates the user sessions but the SP session needs to be

terminated separately. If the SP takes care of the user session management on behalf of the service, the service need to register a handler in SP for logout.
- *Logout in SP*. If SP takes care of all user session management, it offers an API to the services to start the logout in SP. In logout, the SP removes the sessions. In an IdP initiated logout process, the SP removes its sessions and asks the application to end its sessions, too, if necessary.
- *Logout in IdP*. The IdP takes care of the SSO session during which the user can log in to services without re-authentication. If logging into other services is not preferred, the IdP session can be removed. Removing the IdP session does not affect the session any way on the SP and the services.
- *Local logout*. If the user wants to log out from a single service, she uses the local logout that closes both the service and SP sessions related to that service but not the IdP session, nor sessions of other services and SPs.
- *Single logout (SLO) or Global Logout*. If the user wants to log out from all the services, she chooses the single logout. In addition to the currently used service and SP, the IdP goes through all SPs and asks them to log out the user, and ends its own session (IdP initiated logout).
- *Partial logout*. When the service does not work correctly in logout, the result is a often partial logout. Then some of the user's sessions, typically either in the service or in the SP, remains active. The only way to recover from this situation is to close the web browser.

In the SSO systems, the session should end either to local logout or single logout. In practice, the latter is often local logout combined to the IdP logout, not a real SLO. Usability studies [14, 25] has shown that when using a SSO system, the users expect also single logout. Correctly implemented SLO is extremely important when shared computers are used e.g. by students in a university library. However, many users have personal computers, thus a local logout from a single service might often be more practical, e.g. when the user will log in to another service with the same IdP after logging out from the first service. If also the IdP session has been terminated, the benefits of using the SSO system are overturn since the user has to sign in again every time.

Next we go through how SLO works in different SSO systems and federations.

3.1 SAML2 Logout

OASIS, the standardization organization behind SAML, took into account the lessons of another federated identity management system, namely Liberty Identity Federation Framework (ID-FF), while developing SAML2. Liberty ID-FF and Web Service Framework (WSF) allow service providers to communicate directly with identity provider through back channel, which allows SLO in Liberty [1].

SAML2 only describes the logout in general and different profiles complete it with conflicting suggestions. For example, the order of removing the sessions on SP and IdP is not clear. Following describes the SAML2 logout with front channel binding that uses web browser to communicate with different actors.

1. The user chooses logout in a web service.

2. The service removes the user's session.
3. The service redirects the web browser to the SP's logout handler. It can set a return parameter if the browser should be redirected to an URL after successful logout.
4. The SP removes the user's SP session.
5. The SP searches the IdP's logout service address from the metadata and redirects the web browser to it with the SAML2 LogoutRequest message.
6. The IdP removes the user's SSO session (PreviousSession)
7. If necessary, the IdP asks an external authentication service to remove its session.
8. The IdP sends the SAML2 LogoutRequest messages through the web browser to all the SPs except that which started the logout. Each SP removes its sessions and asks the services to remove the sessions, too. All the SPs send messages about success of the logout to the IdP through the web browser.
9. The IdP sends information about the success of the logout through the web browser using the SAML2 LogoutResponce message to the SP that started the process.
10. Finally, the SP can either tell the user that logout has been proceeded successfully or even redirect the user to the URL given in step 3.

The SAML2 logout provides SLO but requires the IdP to keep track of all the SPs that the user has logged in, which requires more resources. Moreover, if the user has logged in using different IdPs, only the services using the same IdP will be logged out.

3.2 HAKA Federation Logout

Since the first version of Shibboleth does not provide any logout, the HAKA federation developed its own logout for Shibboleth SSO used in Finnish universities. The result is usually only partial logout where the service and SP sessions remains open in the other SPs than the one starting the process. The beginning of the HAKA logout works as the SAML2 logout in the steps 1-7, and the rest as follows [7].

8. The web browser returns to the URL if SP askedit while it called the IdP.
9. The browser tells the user that she is logged out from the service and the IdP.
10. User is encouraged to close the web browser to be sure that logout has been completed. The purpose is to prevent using the session from other SPs than that started the logout.

However, even though the user would close the web browser, other SPs and their service sessions remain open until timeout. If an attacker finds out the session identifier, he can use them for a while. Shibboleth SP 2 supports SAML2 logout but the IdP 2 does not. Future IdP 3 will probably have a back-channel-based SLO where the messages are passed using SOAP.

3.3 OpenID Logout

OpenID does not provide single logout although IdPs can provide a logout URL for the service providers. For example, Google staff in the developer discussion forum suggests that the service developer should either provide logout only from the service or redirect

the browser to Google's logout URL [3]. However, the service should know which IdP the user has used to log in to the service in order to provide the correct logout URL. This is hard because there is no pre-established connection between the SP and IdP. Furthermore, OpenID IdP does not keep track of the services the user has logged in.

4 Logout in Practice

We went through commonly used services of the HAKA federation to find out why they behave differently when a user logs out from the service. Almost all possible logout forms listed in Section 3 were found. In this section, we present the results of our tests.

4.1 Only Service Session Removed

Half of the Finnish universities use Oodi[3] to record all course grades, and for students, to sign in for courses and exams. The front-page of the service reminds the users to exit from all browser windows when logging out to prevent the next user to gain access to the first user's information. The warning is needed since when the user logs out, only service's local session is terminated and both the IdP and SP sessions remain. The service returns to the service front-page and gives no information about logout status.

In the Oodi login process, two sessions are created: a SP session with eight hours timeout and a local Tomcat session with 30 minutes timeout. The Oodi service copies all information from the SP's session to a session presented by JSESSIONID cookie. When the user logs out from the service, only the Oodi service's own session cookie is removed but the SP session still remains, let alone the IdP session. Same way, when the local session reaches timeout, the service does not inform the SP about the logout. This means that the user can get directly back in to Oodi without re-authentication because the SP session is still valid.

In this case, the biggest security problem is that the user cannot get out from the service at all if she does not close the web browser. The service does not tell to the user about still valid sessions. Even if she closes the web browser, both the SP and IdP SSO sessions remain open until timeout. The problem is Shibboleth specific since the session management is divided between the SP and the service application.

4.2 SP Session Removed but Service Session Exists

The libraries of Finnish universities have a joint service called the Nelli[4] portal for scientific articles and databases. When the user logs out from the service, the connection is redirected to IdP. The IdP tells the user has logged out from the SSO session and that there might still be active sessions, and only closing the browser will end them. Indeed, there still is an active session, because if the user immediately returns to the Nelli portal and press the login button, she gets in without re-authentication. However, she does not get in to new services since the IdP session has actually been removed.

[3] E.g. https://oodi.aalto.fi
[4] http://www.nelliportaali.fi

The Nelli portal is implemented using MetaLib program that uses Patron Directory Services (PDS) for authentication [19]. When the user logs in to the Nelli portal, the service checks first a PDS_HANDLE cookie. If it exists, its information is copied to the main session. Otherwise the service asks the user information from the SP. Nelli creates also a local session key that is stored in a ML_SESSION_ID (MetaLib Session ID) cookie and included in all page URLs. If this cookie does not exist, it is always created, thus removing it does not affect anything. Removing the key from the URL does not affect either, if the ML_SESSION_ID cookie exists. The Nelli service session does not end until the user navigates to other pages for more than 10 minutes or press the logout button. In addition to service specific cookies, SP removes its session but because the IdP session is still valid, the next user can get in without authentication. Removing first the SP session eliminates the possibility to logout from the IdP because the necessary information was stored in the SP cookie.

When the user logs out from the Nelli portal, the service does not remove the PDS_HANDLE cookie but the SP session is already ended. The problem in this case is that the service session remains open. Thus the user can get in without the service checking from either IdP or SP if their sessions are still valid.

In a way, the logout from the Nelli portal does not work at all. Different sessions are active, even though the user think that they have been ended. The problem is Shibboleth specific but session timeout can be problem in other SSO systems, too.

4.3 Local Logout

The Aalto University Wiki[5] tries to execute local logout. When the user logs out from the wiki, the connection is returned to login page of the wiki service. Only one line of text "You have been successfully logged out. Return to wiki.aalto.fi or login again" separates the page from the original login page that offers several possibilities to log in. The user may not notice the text at all.

Actually the Wiki service tries to execute SAML2 logout instead of HAKA logout since it asks the SP to redirect the connection to its own logout pages and not to IdP. Because the current IdP does not handle SAML2 logout, the SP removes only its session and rest of the session removals fail.

The problem is that the user is not informed from which session she has logged out, and that this was due to compatibility problems. If the user closes the web browser, the IdP SSO session and possible other SP sessions remain open until timeout. If the user does not close the web browser, the next user can get in to all services using the same IdP SSO session, even to the wiki. The problem is generic in the SSO systems.

4.4 Local logout together with IdP Logout

All the course webpages and announcements are in the Noppa[6] portal that was developed at Aalto University and later brought into use in two other universities. When the user signs in the portal, both the SP and Noppa's local Tomcat sessions are created.

[5] https://wiki.aalto.fi/dashboard.action
[6] https://noppa.aalto.fi/noppa/app

All user information is copied to session identified by either JSESSIONID cookie or org.apache.tapestry.locale cookie. The SP session is valid for eight hours, but if it is unused, it expires in one hour. The Noppa portal offers only the local logout together with the IdP logout for the user.

The user can continue using other services even though the IdP session has ended because the IdP does not have means to inform the other services to end their sessions. Thus, the real SLO is not executed. If the user closes the web browser while other services are still active, their SP and service sessions remain active until timeout. The problem is generic in the SSO systems and may confuse users that cannot separate the real SLO from logging out from the IdP.

4.5 Choosing between Local and IdP Logout

One obsolete service offered to a user a possibility to choose between the local and IdP logout, which was implemented as the HAKA logout from both the SP and IdP. When the user logged into the service, the SP copied the essential user information into long hash and gave it to the service as part of the URL. The service removed the string from the URL and created a cookie with the same name. When the user logged out, the session information was removed from the server memory.

It seems that the logout implementation has been left for the service developers to choose. Moreover, many of the services do not actually provide correct logout, they just end some of the sessions. Furthermore, it is up to the user's enlightment, can she distinguish the local logout from the IdP logout when getting out from different services. The problem is generic since the users do not always know where they want to log out and offering choices may be too complicate for them.

5 Solutions for Logout in the Federated SSO Systems

In centralized SSO systems, an IdP keeps track to which SPs the user has been authenticated, and communicates to all SPs about logout through a backend channel, thus the single logout is possible. Federated SSO systems do not usually use backend channel for communication and an IdP does not know about active sessions on SPs.

There are several steps that make the logout in the federated SSO systems more secure. At least, the user should be able to logout from the service she is using but also ending the SSO session in the IdP is necessary, if chosen. To provide true single logout, the whole federated SSO needs modifications.

5.1 Modifications in Services

When a service is added to the federated identity management, it has to be modified. The developers should decide if the service keeps a session itself or does it use the session provided by SP. If the SP takes care of the sessions, the service just need buttons for the local and IdP logout. Pressing these buttons can start the logout procedures with proper attributes in the SP, and the SP takes care of everything.

If the service takes care of the logout, the procedure is quite the same as above but changes are needed. The service has to remove its own session before calling the logout method of the SP. Equally important is to have a handler to the IdP initiated logout. For this, the service should register a single logout handler for the SP. When a SLO request arrives, the service has to remove all sessions. The request can come through the web browser or using a SOAP interface. In the latter, it is not possible to remove the cookies from the browser, but the service has to remove the session from its own memory. Finally, the lifetime of service session should be shorter than the lifetime of SP session or otherwise the incoming logout requests do not work correctly.

5.2 Cookie Management in Web Browsers

Handling of the cookies has not changed much during the recent years. To mitigate problems caused by attacks, there are rules how cookies should be handled. These rules prevent the IdP and SP from seeing each other's cookies.

For making single logout possible, the rules should be changed. For example, the IdP could create a collection of cookies connected to the SSO session. The SPs could add their own cookies to the collection. Then as a creator of this collection, the IdP could remove cookies and set the deletion rights to all servers of the federation. In the single logout, the IdP could remove all the SP cookies that are left over. However, this change may cause unexpected security weaknesses if not done with extra care.

5.3 Connections between the IdPs and SPs

Single logout changes the actions of the IdP and SP. For example in the original Shibboleth, the IdP takes only care about the validity of SSO sessions and the SP has the necessary user attributes in its memory. The IdPs and SPs could have been rebooted since all necessary state information has been stored in the cookies on the user's web browser and the other information could have been fetched again. For single logout, the IdP has to store information about all SPs where the user is logged in. This would change a lot of the use and importance of IdP's SSO session.

Another way to do automatic logout is based on AJAX. The web browser could send *keep alive* messages to the SPs and IdP when the user has an open window for a service. When the user navigates to some other web page than that of the service, the browser simply stop sending these messages to the SP that can then end the user's session. This way the sessions will not be "left behind". Moreover, when the user has moved out from all the services of the federation, the web browser stops sending the keep alive messages to the IdP, too. Then the IdP can remove the SSO session and, to make single logout absolutely sure, start the IdP initiated logout fom all SPs. However, there are two problems: This approach burdens the IdP and SP with additional messages, and it does not work if JavaScript is turned off or the web browser does not support it.

On the other hand, single logout is possible to implement using a *polling mechanism*. When a user chooses single logout in a service, the service would execute both local and IdP logout, thus ending both its local session on the SP and service, and the SSO session on the IdP. The other SPs would notice that the SSO session is no longer valid because they ask periodically its existence using e.g. a SOAP interface of the IdP.

Then, they too could execute the local logout, and all the sessions would end through true single logout. Of course, the interval of the queries should not burden the IdP. The downside is that with this addition Shibboleth would not be SAML2 compliant software.

5.4 Clear User Interface

It is hard for a user to separate a local and single logout and understand their differences when some services of the federation provide the first and some the latter. It would be nice if the user could see on a page the services where the authentication sessions that are still valid. Technically and from the user interface point of view this is hard.

Especially important for the user is to know that global single logout has really removed all the session information from all the services. Closing the web browser ends the sessions in the client side and solves the problem almost satisfactorily. However, the sessions on the server side remains until timeout but they cannot be used without knowing the session key. Another problem rises if the user has also other web browser windows and tabs open, and does not want to close them, just log out from one of the services. At least the web browsers should be improved to really allow to end the sessions without closing the whole browser.

For example, the WebApp of Microsoft Outlook asks from the user before login if the user has a private computer or if she is using a public computer. This affects how long timeout is used for the session when the user is inactive. In the same way, the login of a service could give two possibilities that finally lead to either local or single logout.

6 Related work

Logout in SSO systems is not widely researched topic. Often it is only mentioned when some other aspects of the SSO systems are improved. For example, Sun et al. [25] investigated usability of OpenID system because a single sign-on system may confuse users when it redirects the user from a service to an identity provider for logging in. They found out that despite of many passwords, users are still reluctant to use SSO, e.g. for critical services because they do not understand how it works. They implemented a system [24, 25] that integrates OpenID login into a web browser and they tested the usability of their system with a small group of users. They list single logout as one important requirement in their system but do not describe its features nor its usability, they only state that it must automatically end all authentication sessions. However, based on the screenshots of their solution, it seems to offer users two possibilities: either single logout or logout only from the current service. In their other article [23], the login process of their system is elaborated in more detail but handling the logout process is not described at all.

Cahill et al. [4] have created a client-based authentication system that has a trusted hardware-based secure container for user credentials in the user's device. The system allows both active and passive authentication. The latter means e.g. facial recognition using the device's camera or RFID badge that is near the device. Thus, service providers do not need to implement timeout for their service sessions, but also SAML-based SLO

is implemented. The system requires much from the hardware and the user is bound to use one device. Mustafic et al. [16] describe similar system that combines SSO with continuous behavioral biometric-based authentication. Their system uses keystroke dynamics as proof that the user is still the same as that authenticated in the beginning of the session. If the system notices that the user has changed, it starts the SAML SLO. However, this only works for services that require typing.

Linden et al. [14] present a usability study on logout in a SSO system among students and personnel of a university. They found out that especially students expect also single logout when using single sign-on. On the other hand, the personnel stated that their sessions in the university services have no clear beginning or end. More important for the users was that they knew if they were authenticated or if they were anonymous, all the time while they used the services. When Linden et al. did their study, the SSO systems were new and many universities implemented the student services as integrated web portals. This may explain some of the misunderstandings that single sign-on means also single logout.

7 Conclusions

In order to create reliable logout for the federated SSO systems, standardization organizations, web browser vendors, and service developers should consider also the termination of the sessions. For example, thorough testing of logout behaviour could improve the current situation. Moreover, the following changes would make logout better:

- unified and standardized process for ending the sessions,
- improving the cookie management in browsers,
- creating a mechanism to check the existence of an IdP session, and
- unified user interface for logout.

The *unified process* for removing the session cookies and managing their timeout periods would prevent the current problems of logout. The SAML single logout is too complex and contradictory to work properly. For example, the order of the session removal is not clear. The standardization should redefine the single logout process because the currently used federation specific definitions are not sufficient, as is shown in Section 4. Examples could also be provided for service developers on how the session management should be handled.

The *cookie management improvements* consist of two things: ending the sessions without closing the web browser by removing the cookies, and allowing the IdP to remove the SP cookies. The former depends on features in web browsers and requires support from browser vendors. The latter could be taken into account in standardization of HTML 5, for example. However, the current model of session management with cookies is not likely to change. Moreover, if the IdP keeps track of all SPs to which the user has logged in, there are implications to user privacy.

Another possibility for checking the existence of the IdP session is to create a *polling mechanism* for SPs. If the SP polls the IdP, the IdP does not need to keep track of the SPs to which the user has logged in. Since the user is not bound to use only a

single IdP, the SP may need to poll all the IdPs that the users have chosen. Of course, the polling frequence must be reasonable and not burden the IdP and SPs.

The *unified user interface* means that a federation or the SSO system standardization should define from the user's point of view how the services should end the sessions. Because the user can work either with public or private computers, two options should be provided by all services: single logout that ends both the IdP and all service and SP sessions, and local logout that leaves the IdP session active. Also, all services should inform the user from where she has logged out.

The process of ending the authenticated sessions has not gained the attention it deserves. We tested many services that use Shibboleth as a SSO system and almost all had problems with logout. Because some of the session information remains when the service ends in partial logout instead of local logout, the next user of a shared computer can access the service with the previous user's account. Where the single logout is suggested in the user interface, it often does not work as expected. Even though the IdP removes the SSO session, the service and SP sessions remain active. Often services rely on closing the web browser but even then the server side sessions remain until timeout. Small changes such as unified logout process and user interface would allow the services to provide at least local logout instead of partial logout. For true single logout, the federated SSO systems require modifications to the standards for cookie handling and federation practices such as a polling mechanism for checking the IdP sessions.

8 Acknowledgments

We whish to thank Aapo Kalliola and Jaakko Kotimäki for valuable comments.

References

1. M. Alsaleh and C. Adams. Enhancing consumer privacy in the liberty alliance identity federation and web servies frameworks. In *PET'06: Proceedings of the 6th international conference on Privacy Enhancing Technologies*. Springer-Verlag, June 2006.
2. Apache Software Foundation. Apache Tomcat. http://tomcat.apache.org, 2012.
3. D. Balfanz. Sign-out from Google federated login api. Google API discussion forum, URL: https://groups.google.com/forum/?fromgroups=#!searchin/google-federated-login-api/sign-out/google-federated-login-api/dBpKzRh1Amc/dEJhGRDwTE0J, Feb 13 2009.
4. C. P. Cahill, J. Martin, V. Phegade, A. Rajan, and M. W. Pagano. Client-based authentication technology: User-centric authentication using secure containers. In *DIM'11: Proceedings of the 7th ACM workshop on Digital Identity Management*, pages 83–92. ACM, 2011.
5. D. W. Chadwick, G. L. Inman, K. W. Siu, and M. S. Ferdous. Leveraging social networks to gain access to organisational resources. In *DIM'11 Proceedings of the 7th ACM workshop on Digital Identity Management*, pages 43–52. ACM, 2011.
6. CSC - IT Center for Science. Haka federation. URL: http://www.csc.fi/english/institutions/haka, June 2006. (Referred 8.11.2012).
7. CSC - IT Center for Science. *Haka Logout*. http://www.csc.fi/hallinto/haka/ohjeet/ohjeet-yllapitajille/haka-logout, 2012. (Referred 30.10.2012).
8. R. Dhamija and L. Dusseault. The seven flaws of identity management, usability and security challenges. *IEEE Security and Privacy*, 6(6):24–29, 2008.

9. Facebook. Getting started with Facebook login. URL: https://developers.facebook.com/docs/technical-guides/login/, 2012. (Referred 7.11.2012).
10. D. Florêncio and C. Herley. A largescale study of web password habits. In *Proceeding WWW '07 Proceedings of the 16th international conference on World Wide Web*, 2007.
11. S. Gaw and E. W. Felten. Password management strategies for online accounts. In *Symposium on usable privacy and security (SOUPS) 2006*, pages 44–55, July 2006.
12. Ideelabor. Openid in Estonia. URL: http://openiddirectory.com/openid-providers-c-1.html, 2008. Referred 27.2.2009.
13. Kalmar2. Kalmar2 - access to nordic higher education with single login. https://www.kalmar2.org/kalmar2web/front_page.html. (Referred 20.12.2012).
14. M. Linden and I. Vilpola. An empirical study on the usability of logout in a single sign-on system. In *Proceedings of the 1st International Conference in Information Security Practice and Experience*, volume 3439 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
15. Microsoft. Windows live id web authentication sdk. URL: http://msdn.microsoft.com/en-us/library/bb676633.aspx, 2012. (Referred 7.11.2012).
16. T. Mustafic, A. Messerman, S. A. Camtepe, A.-D. Schmidt, and S. Albayrak. Behavioral biometrics for persistent single sign-on. In *DIM'11 Proceedings of the 7th ACM workshop on Digital Identity Management*, pages 73–82. ACM, 2011.
17. OpenID Community. Openid authentication 2.0 - final. URL: http://openid.net/specs/openid-authentication-2_0.html, December 5 2007. (Referred 8.11.2012).
18. A. Pashalidis and C. J. Mitchell. A taxonomy of single sign-on systems. In *Information Security and Privacy - 8th Australasian Conference, ACISP 2003, Wollongong, Australia*, number 2727 in LNCS, pages 249–264. Springer-Verlag, July 2003.
19. J. Pennanen. Wiki nelli metalib. URL: https://wiki.helsinki.fi/display/Nelli/MetaLib, August 3 2009. (Referred 7.11.2012).
20. N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, and T. Scavo. Security assertion markup language (saml) v2.0 technical overview. Technical report, OASIS, February 2007.
21. Shibboleth Consortium. Shibboleth. URL: http://shibboleth.net/, 2012.
22. S.-T. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov. A billion keys, but few locks: the crisis of web single sign-on. In *NSPW '10: Proceedings of the 2010 workshop on New security paradigms*, september 2010.
23. S.-T. Sun, K. Hawkey, and K. Beznosov. Openidemail enabled browser: Towards fixing the broken web single sign-on triangle. In *DIM'10*. ACM, October 8 2010.
24. S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov. Openid-enabled browser: Towards usable and secure web single sign-on. In *CHI EA '11: Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*. ACM, May 2011.
25. S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov. What makes users refuce web signle sign-on? an empirical investigation of openid. In *SOUPS'11: Proceedings of the Seventh Symposium on Usable Privacy and Security*. ACM, July 2011.
26. S. Suoranta, A. Andrade, and T. Aura. Strong authentication with mobile phone. In *Information Security, 15th Internation Conference, ISC 2012*, number 7483 in LNCS, pages 70–85. Springer, 2012.
27. Y. Takeda, S. Kondo, Y. Kitayama, M. Torato, and T. Motegi. Avoidance of performance bottlenecks caused by http redirect in identity management protocols. In *DIM'06: Proceedings of the second ACM workshop on Digital identity management*. ACM, November 2006.
28. The OpenIDDirectory. Openid providers. URL: http://openiddirectory.com/openid-providers-c-1.html, February 2009. Referred 27.2.2009.