Single Sign-On (SSO) – First part

1) What is SSO ?

The users do their best to manage the soaring number of logons and passwords they have. Unfortunately this often includes the use of unsafe practices such as: writing passwords on one's diary or on post-it stickers, using the same password for most of their access, or keeping connections opened when they leave their workstation.
A good solution for this problem is to rely on a solution for Single Sign-On. Single Sign-On (SSO) is a process that allows a user to authenticate once, and then to access multiple applications or resources without any new authentication.
But as practical as they are, the SSO are rarely made simply to make life easier for users. They are generally integrated in a wider safety project in which they are often considered as a secondary element.

The types of authentication SSO

There are two main types of SSO solutions: the first one is called "Web SSO", and the second is called "Enterprise SSO" (eSSO).

Web SSO addresses all applications that use a web browser to logon to the applications.

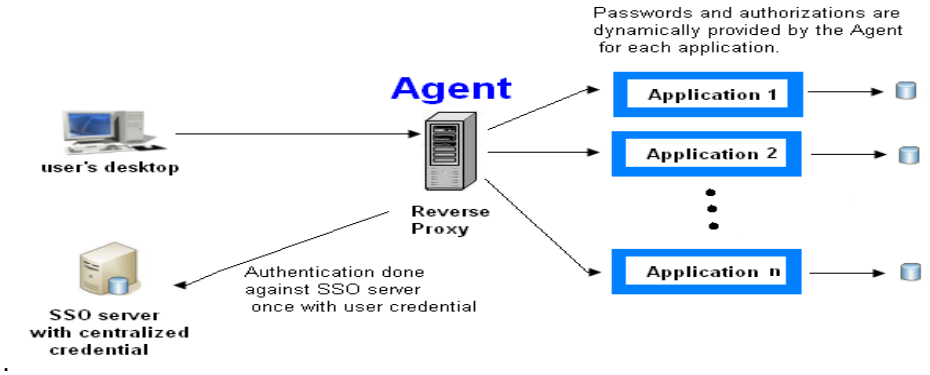

Fig 1: Web SSO architecture

On the other hand, **eSSO** systems are not limited to web applications. They are designed to minimize the number of times a user must type their login and password to sign-in into multiple enterprise applications.

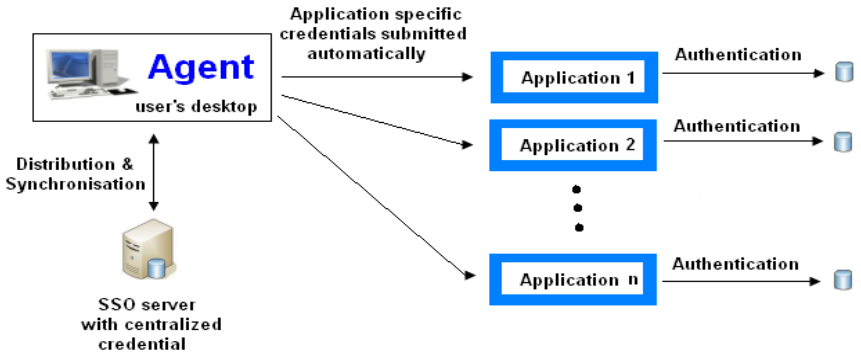

Fig 2: Enterprise SSO architecture

SSO components

In SSO system, we find generally the following components:

- The client which requests access to the application. Typically, this is a web browser. But any other client/server applications could also be involved, and the client could be as well a telnet client for example.
- The Authentication Server is the database that contains all the credentials. It is actually the core element of the SSO system.
- The application server which delivers the resources according to the result of authentication process.
- The agent which verifies if the user is authenticated. The agent can be located on different place according to the architecture and can be hardware or software.

2) The different SSO approches

2.1 The Centralized approach

The principle here is to have a centralized database which contains all the users credentials. This model is very suitable when you want to centralize the management of user accesses. It could be implemented using LDAP.

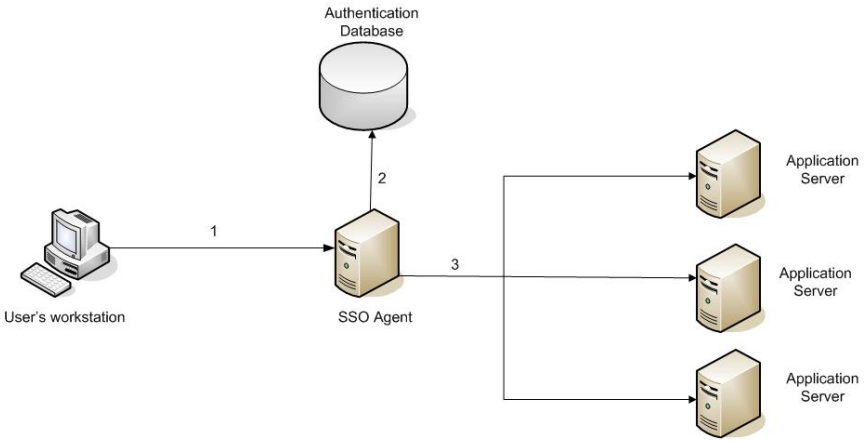

Fig 3: Centralized SSO

1. The client wants to access an application. In this case, the Agent runs on a Reverse Proxy and it intercepts the request.
2. The agent authenticates the user thanks to the authentication database which can be a LDAP directory.
3. Once the user authenticated, he can access to the application.

2.2 The Federal approach

A big challenge in today's authentication infrastructures is to extend the SSO scope to cover many "different" authentication authorities (implemented on different platforms or governed by different organizations). Federation allows extending access control and SSO across organizational boundaries. Indeed, the implementation of federated identity and the extension of SSO across enterprises permit to distribute control and maintenance activities, and thus having even more convenience and time savings for both organizations and users alike.
Federated approach is that they allow a user to seamlessly traverse different sites, services within a given federation. Every service manages a part of the data of a user but share the user information with the partner services. This approach was developed to answer a need of decentralized

management of the users where every partner service wishes to preserve the control of its own safety policy.

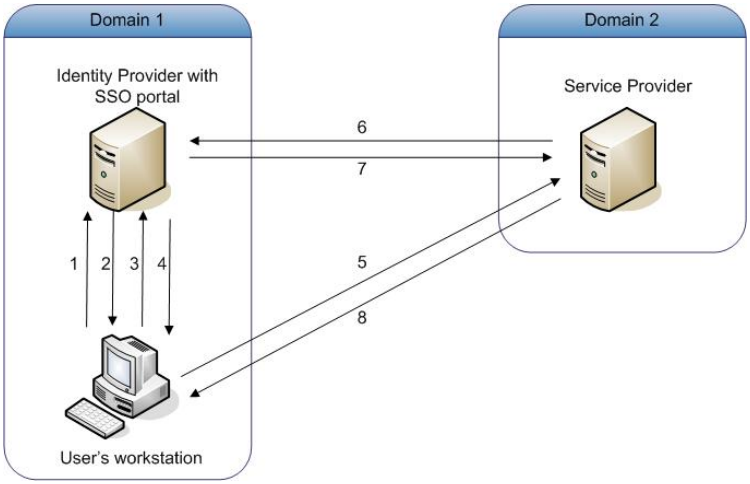

Fig 4: Federated SSO

1. The user logs in to the Identity Provider (IdP).
2. After successful authentication, the IdP sends to the user informations about applications that he can access.
3. The user clicks on the Service Provider link (SP) in the portal. This is a special link, which doesn't connect to the SP directly.
4. The IdP gets the request and creates an Identity Assertion. The IdP stores this Identity Assertion with an "Artifact" pointer in the cache pool. Then, IdP returns a re-direct response to the client browser.
5. The browser is redirected to the SP with the "Artifact".
6. The SP gets this request and contacts the IdP with the "artifact" to request the real identity assertion
7. The IdP gets the request, and looks up the entry in the cache table of identity assertions using the "artifact" as an index. It creates an actual identity assertion in SAML token format, which it sends back to the SP as a SOAP and HTTP response.
8. The SP extracts the user information from the received identity assertion. Finally, after the local authentication succeeds, the user is allowed to access the service.

The main example of the federated approach is Liberty Alliance.

It is mainly based on the SAML standard, as well as http and SSL.
Another protocol to establish identity trust between disparate systems exists. This is a relatively new protocol which is called Web Service Federation (WS-Federation).

2.3 The cooperative approach

This approach is similar to the Federal approach. It meets the needs of institutional structures, for example, research laboratories or governments. In the cooperative approach, each user depends on partner entities. When he tries to reach a network service, the user is authenticated by the partner from whom it depends. As in the Federal approach, all network services independently manage their own security policy. With this approach, identification of user security is not traded. The main representatives of this approach are Shibboleth and Central Authentication Service (CAS).

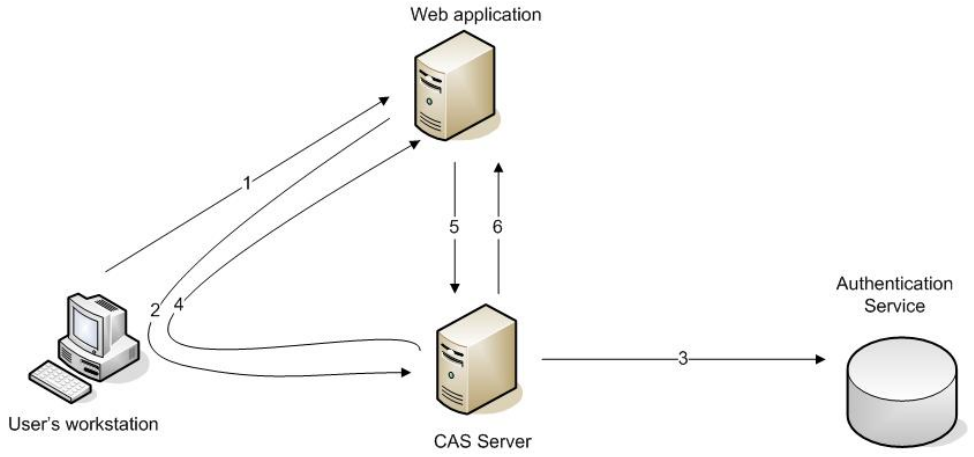

Fig 5: functional scheme of CAS

1. The client visits an application requiring authentication.
2. The application redirects it to CAS.
3. CAS validates the client's authenticity, usually by checking a username and password.
4. If the authentication succeeds, CAS returns the client to the application, passing along a service ticket (ST).
5. The application then validates the ticket by contacting CAS over a secure connection.
6. CAS then gives the application trusted information about whether a particular user has successfully authenticated.

3) SSO Architectures

The major SSO architectural models are:

- client-side SSO
- server-side SSO
- and hybrid systems.

Most of enterprises SSO (**eSSO)** are based on the client-side SSO architecture whereas Web SSO uses a server-side.

3.1 Client-side SSO

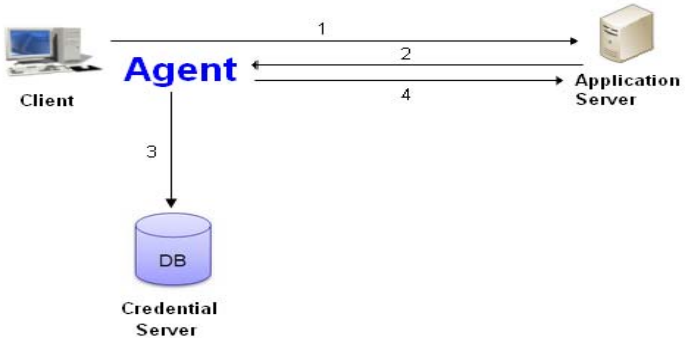

Fig 6: Client side SSO with a central credential database

1. The client wants to access an application.
2. The Application requests for user credential. The Agent located on the client's workstation intercepts the request.
3. The agent pulls credentials from the centralized credential Server.

4. The agent simulates real user in sending credential to application. In fact, the agent gives the credentials to the web browser. So, the identification is transparent for the user.

3.2 Server-side SSO

There are two types of Server-Side SSO: those based on a reverse Proxy and those based on server agents. With this architecture, it is not necessary to install any agent on every user's PCs.

Server Agent

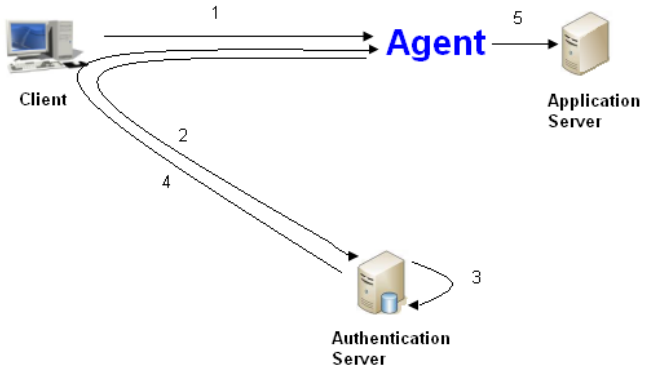

Fig 7: Server Side SSO with Agent on Application server

1. The client wants to access an application. The Agent located on the application server intercepts the request.
2. The agent verifies that the user has already been authenticated; if he is not, the agent redirects the user towards the authentication server. This authentication could then be implemented through a web portal or a pop up window. The user supplies his credentials to the authentication server.
3. The authentication server verifies user identity in a reference database.
4. Once the user authenticated, the authentication server sends back an http cookie on the user workstation which will allow maintaining the session of the user.
5. The agent passes them on the application server.

Reverse Proxy

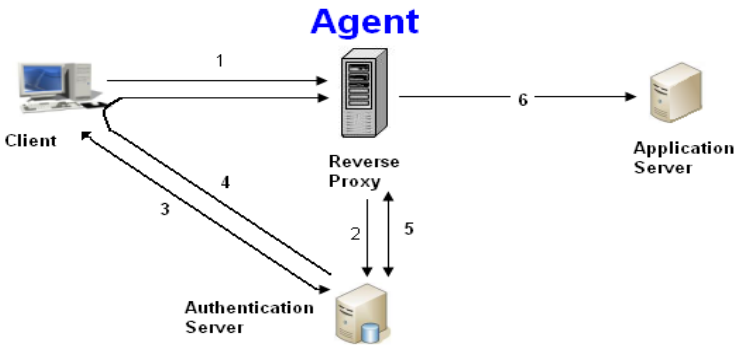

Fig 8: Server Side SSO with Agent on Reverse Proxy

1. The Web browser attempts to connect to the web Application. This connection attempt is automatically redirected towards the Reverse Proxy.
2. The Agent on the reverse proxy intercepts the request and verifies that he has already be authenticated via the authentication server.

3. If the user has not be authenticated yet, the authentication server asks his credentials to the user. The user supplies his credentials to the authentication server
4. The authentication server sends a token as a cookie and redirects the browser towards the reverse proxy.
5. The Agent on the reverse proxy intercepts the request again and verifies the authentication of the user via the authentication server with the token. The authentication server sends the login and authorization information that is associated with the token.
6. The agent allows access to the application.

3.3 Hybride SSO

There are many hybrid approaches. They combine Client-Side SSO and Server-side SSO and aim at reducing some Client-Side SSO problems. Intrusive constraints on both client and server sides are present.

For more information

http://www.01net.com/article/256916.html
http://fr.wikipedia.org/wiki/Authentification_unique
http://www.cesnet.cz/doc/techzpravy/2006/web-sso/
http://www.cru.fr/documentation/federation/index
http://www.ufinity.com/media/pdf/whitepapers/SSO%20Architecture%20Comparisons.pdf
http://www.projectliberty.org/

Note: Next month, the second part of this article will describe the features of SSO and some tools.

End of document