Introduction aux architectures web de Single Sign-on

Olivier Salaün
Comité Réseau des Universités
Campus de Beaulieu - Rennes
Olivier.salaun@cru.fr
15 Octobre 2003

Résumé

L'article aborde la problématique d'un SSO web en décrivant les apports de ce type d'architecture du point de vue de l'utilisateur et du point de vue du système d'information. Nous détaillerons le fonctionnement d'un SSO et le cas particulier des architectures multi tiers. Nous décrirons brièvement les spécificités de quelques projets de SSO développés par des universités. Enfin nous aborderons plus généralement la problématique de gestion des identités numériques inter-établissement.

Mots clefs

SSO, web, authentification, gestion des droits

1 Introduction

Compte tenu de la multiplication des applications web dans les établissements, les utilisateurs sont amenés à s'authentifier de nombreuse fois auprès de chacune de ces applications, en multipliant les couples identifiant/mot de passe à retenir. Le déploiement des annuaires LDAP, outre leur apport fonctionnel pour la gestion des groupes, a permis de simplifier la situation en utilisant un référentiel d'authentification commun à la majorité des applications. Ainsi pour les applications utilisant ce référentiel d'authentification, l'utilisateur peut utiliser un mot de passe unique. La mise en place d'un système de Single Sign-On (SSO) doit permettre à l'utilisateur de saisir ce mot de passe une seule fois pour accéder à toutes les applications web de l'établissement, améliorant ainsi à la fois l'ergonomie d'accès aux applications et la sécurité du système d'information en limitant la circulation des mots de passe.

En préalable à la description des architectures d'authentification il importe de différencier deux fonctions souvent confondues : l'authentification et l'autorisation. L'authentification consiste à déterminer l'identité de l'utilisateur, alors que la fonction d'autorisation, pour une opération donnée, détermine les privilèges de l'utilisateur en fonction de ses attributs. Si certains de ces attributs sont gérés par les applications concernées, d'autres (comme les fonctions ou l'appartenance à des groupes) ont leur place dans l'annuaire LDAP de l'établissement.

A l'heure où les relations entre établissements se développent (campus numériques, accès à des ressources bibliothécaires, étudiants inscrits dans plusieurs établissements, groupes de travail transversaux,…) l'authentification et la gestion de droits d'accès pour les personnes extérieures à l'établissement sont au mieux gérées de façon artisanale ou pas du tout (URLs cachées, mots de passe partagés). L'utilisation du SSO comme service commun d'authentification au sein de l'établissement peut permettre de développer la gestion des identités entre établissements.

2 Objectifs d'un système de Single Sign-On

2.1 Apport ergonomique pour l'utilisateur

L'atout évident d'un service de SSO est la simplification des procédures d'authentification pour l'utilisateur. Alors qu'il devait s'identifier successivement auprès de chaque application web lors d'une session de travail, le SSO lui permet de ne s'authentifier qu'une seule fois. L'identifiant de l'utilisateur et ses attributs sont ensuite propagés vers les applications. Certains logiciels de SSO assurent également la fermeture de toutes les sessions applicatives de l'utilisateur lorsqu'il se déconnecte.

Ce besoin de cohérence des systèmes d'authentification dans les applications web de l'établissement est renforcé dans le cadre du déploiement de portails web d'établissement. En effet ceux-ci visent à présenter tous les outils mis à disposition de l'utilisateur de façon homogène et cohérente, alors que les applications sont très hétérogènes.

2.2 Amélioration de la sécurité

L'accès nomade par le web à des applications nécessite de sécuriser la phase d'authentification sans faire d'hypothèse sur la sécurité du réseau qui relie clients et serveurs. L'architecture de type SSO, en concentrant cet effort de sécurisation sur le (ou les) serveur(s) d'authentification, permet de mettre en œuvre sur ce point une politique de sécurité cohérente tout en allégeant l'effort d'écriture des applications. L'utilisation d'un service commun d'authentification devrait également faciliter l'évolution des méthodes d'authentification (certificats X509, Kerberos,…) ou la prise en compte de plusieurs niveaux d'authentification en fonction de la sensibilité des applications accédées.

La mise en place d'un SSO est en outre l'occasion de donner de bonnes habitudes aux utilisateurs : il ne doivent délivrer leur mot de passe qu'au seul serveur d'authentification dont la bannière de login et l'URL (utilisation d'un certificat serveur recommandée) sont bien identifiés.

2.3 Rationalisation des applications et de la gestion des comptes

D'une manière générale, les applications web souffrent de leur manque d'intégration avec le système d'information. Les services sont rendus par une multitude de CGI sans relation les uns avec les autres et pour lesquels on a chaque fois redéveloppé la gestion de l'authentification. L'utilisation d'un service commun d'authentification doit permettre le développement plus rapide d'applications offrant toutes les mêmes garanties de sécurité.

Avec son système d'authentification propre, chaque application assure sa propre gestion des comptes utilisateurs. Cela complique la mise en place d'une politique cohérente de gestion des comptes (allocation, expiration, mise à jour, …) et impose aux administrateurs la tâche ingrate de synchronisation de ces différentes sources de données avec les bases de référence (Exemple : Harpège, Apogée). La mise en place d'annuaires LDAP reflète ce besoin d'administration centralisée des informations utilisateurs. Le serveur d'authentification doit pouvoir se baser sur ce référentiel commun pour authentifier l'utilisateur ou propager certains de ses attributs auprès des applications.

3 Architectures classique d'un SSO web

L'architecture de la plupart des produits de SSO est inspirée de Kerberos [1] ; ils utilisent largement sa terminologie et partagent ses concepts de base qui sont les suivants :

- Les applications sont déchargées du travail d'authentification des utilisateurs. Cette tâche est assurée par un serveur d'authentification dédié.
- Le serveur d'authentification délivre des tickets au client (maintient de la session d'authentification) et aux applications (transmission de l'identité de l'utilisateur). Ce second ticket transite également par le client.
- L'application ne recueille jamais les éléments d'authentification de l'utilisateur (couple identifiant + mot de passe par exemple).
- Il existe une relation de confiance entre les applications et le serveur d'authentification. A noter que Kerberos n'utilise que des techniques de cryptographie symétriques ; l'utilisation de certificats X509 (utilisant des algorithmes asymétriques) peut permettre de simplifier l'architecture du système.

Les principes de base d'un système de Single Sign-On web on été par ailleurs définis dans des documents de référence. Compte tenu du développement par plusieurs universités américaines de systèmes de SSO « maison », Internet 2 a initié le groupe de travail WebISO [2] chargé de définir les caractéristiques d'un système de Single Sign-On web. Le projet Liberty Alliance [3], quant à lui, est un regroupement d'entreprise (AOL, Cisco, France Telecom, HP, Novell, Sun, Verisign,…) qui, en réaction face au projet « *Microsoft Passport »*, publie des recommandations autour de la notion de « *Federated Network Identity»*, une authentification unique avec partage des données de l'utilisateur avec des partenaires de confiance.

3.1 Le serveur d'authentification

Le serveur d'authentification est l'élément central du système de SSO puisqu'il assure l'authentification de l'utilisateur, la persistance de sa connexion et la propagation de l'identité de l'utilisateur auprès des applications.

L'utilisateur fournit ses éléments d'authentification au serveur d'authentification. Si le mode d'authentification est le mot de passe, la phase d'authentification implique la vérification du mot passe de l'utilisateur auprès d'une base de référence. La plupart des systèmes de SSO implémentent plusieurs *backend* d'authentification (/etc/password, NIS, LDAP). Si le serveur implémente l'authentification par certificats sa tâche consistera à vérifier la validité du certificat, la chaîne de certification et les listes de révocation. En concentrant la logique d'authentification sur un serveur d'authentification, on peut plus

facilement faire évoluer les méthodes d'authentification (certificats, OTP,…) ainsi que les bases d'authentification reconnues (PAM, Radius,…).

Lorsque l'utilisateur a été authentifié, le serveur d'authentification maintiendra la session de l'utilisateur en positionnant un cookie HTTP sur le poste de l'utilisateur. Les données stockées dans le cookie sont protégées (cryptage ou utilisation d'un ticket interprété par le serveur) et sa portée est idéalement limitée au serveur d'authentification. Le cookie HTTP est le seul moyen technique fiable à notre disposition pour que l'utilisateur soit reconnu comme authentifié lors de son prochain accès au serveur.

Si l'utilisateur a été orienté vers le serveur d'authentification par une application cible, le serveur doit, en retour, fournir l'identité de l'utilisateur à l'application. Par identité on entend soit uniquement un identifiant, un email et/ou un ensemble d'attributs de l'utilisateur. La transmission de l'identité de l'utilisateur à l'application transitera forcément par le poste de l'utilisateur. Soit le serveur d'authentification utilise une redirection HTTP, soit il renvoie à l'utilisateur un document HTML incluant un programme *Javascript* de redirection. Dans tous les cas le navigateur de l'utilisateur est redirigé vers l'application, muni des éléments d'identification. L'application ne fait appel qu'une fois au serveur d'authentification et gère ensuite une session applicative classique avec l'utilisateur.

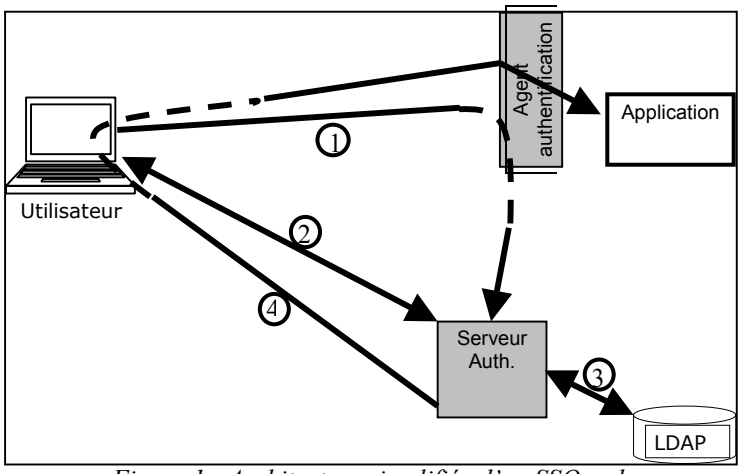

Figure 1 : Architecture simplifiée d'un SSO web

3.2 L'agent d'authentification

L'agent d'authentification est la brique du SSO intégrée à l'application cible par exemple sous forme d'une librairie applicative ou d'un module apache. L'agent vérifie que l'utilisateur est authentifié ; s'il ne l'est pas, il le redirige vers le serveur d'authentification. Si le client s'est déjà authentifié auprès du serveur d'authentification (attesté par le présence d'un cookie) le serveur le redirige directement vers l'agent d'authentification demandeur, de façon non bloquante. Lorsque l'utilisateur revient du serveur d'authentification, authentifié, l'agent vérifie l'origine des données (les données peuvent être signées) et les transmet à l'application.

Certains systèmes de SSO ne véhiculent pas directement les attributs de l'utilisateur entre le serveur d'authentification et l'agent mais un jeton d'authentification ; l'agent d'authentification contacte directement le serveur d'authentification (accès HTTPS ou web services) et lui présente le jeton pour validation ; en retour il reçoit les données sur l'utilisateur. Ce type de SSO est particulièrement adapté aux architectures multi tiers.

Il peut se révéler coûteux ou difficile de modifier les applications pour y intégrer un agent d'authentification. On peut envisager une solution de type « reverse proxy » où on installe un agent d'authentification en frontal d'un ensemble d'applications. L'agent reverse proxy intercepte les accès utilisateurs ; une fois l'utilisateur identifié il transmet les données d'identification à l'application via des variables d'environnement (dans le cas d'un module du serveur web) ou dans des champs d'entêtes HTTP. La plupart des SSO est fournie avec un module de ce type pour Apache.

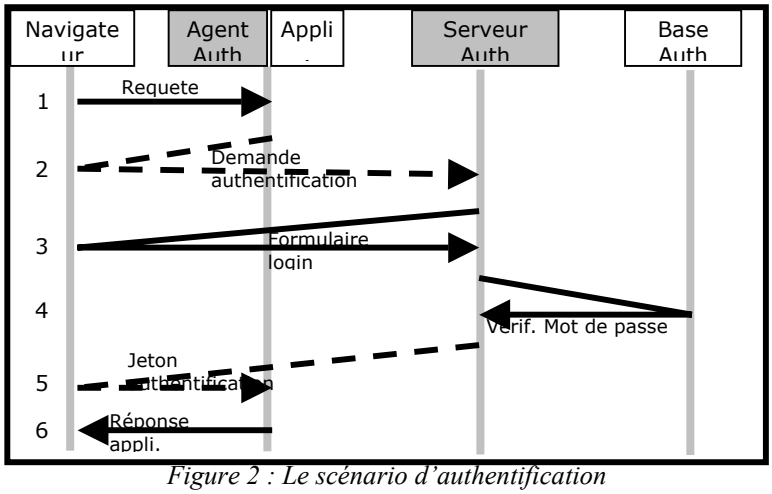

Figure 2 : Le scénario d'authentification

Sur la figure 2, les flèches en pointillé représentent des redirections HTTP ; les flèches en trait plein, des requêtes HTTP et les réponses du serveur. Lors du premier accès au serveur d'authentification, les phases (3) et (4) d'authentification sont requises. Pour les accès suivants la phase d'authentification sera transparente pour l'utilisateur.

4 Le cas des architectures multi tiers

L'authentification dans une architecture multi tiers, comportant au moins un intermédiaire entre l'utilisateur et l'application cible, pose le problème de la délégation des privilèges. Deux exemples de ce type : un portail web agissant comme intermédiaire vis-à-vis d'une application ; un webmail, intermédiaire vis-à-vis du serveur IMAP. Ce type de relation est donc courante et doit supporter la migration vers un système de SSO web. L'adaptabilité d'un SSO à ce type d'architecture est un critère de choix important. L'article *Trusted Delegation of Privileges in an N-Tier Environment [4]* décrit bien les 4 mécanismes permettant de transmettre une authentification dans une architecture multi tiers :

1. *« Usurpation d'identité »* (masquerade) : l'intermédiaire transmet les éléments d'authentification (identifiant+mot de passe) à l'application comme s'il était l'utilisateur. Ce mécanisme est adapté à des applications offrant une interface d'authentification simple (Ex: soumission d'un formulaire auprès d'un CGI).

2. *« Relation de confiance »* : l'intermédiaire, une fois authentifié auprès de l'application cible, peut demander un service à l'application pour n'importe quel utilisateur. Inconvénient du système : si l'application intermédiaire est compromise, l'attaquant a accès à toutes les applications du réseau de confiance.

3. *« Accès direct aux données »* : l'intermédiaire court-circuite l'application et accède directement à ses données. Cette solution requiert une bonne connaissance de l'application finale.

4. *« Procuration »*(proxied credentials) : le serveur d'authentification attribue à l'intermédiaire une procuration sous forme d'un ticket à présenter à l'application finale. L'application vérifiera le ticket auprès du serveur d'application.

La solution (1) est utilisée dans certains produits de SSO décrits plus loin. (2) correspondrait à une relation de confiance entre un portail et ses canaux, le portail fournissant au canal l'identité de l'utilisateur.

La solution (4) est celle qui offre les meilleurs gages de sécurité puisque l'application fournissant l'identité de l'utilisateur est dédiée. En revanche ce type de produits impose de modifier toutes les applications concernées (webmail + serveur IMAP dans le cas d'un webmail). La version 2.0 de CAS, le SSO développé par l'Université de Yale, gère la délégation de cette manière.

5 D'autres architectures de SSO

Si l'architecture de type Kerberos est la plus séduisante en terme de sécurité et la plus structurante pour le système d'information, certains produits de SSO adoptent d'autres organisations.

Une solution radicalement différente consiste à déporter la fonction de Single Sign-On sur le poste client. Le logiciel mémorise les couples identifiant/mot de passe saisis pour chaque application (stockage sur support sécurisé : carte à puce, clé USB) et prend la main lorsqu'une bannière de login se présente. Ce type de logiciels ne fait qu'étendre le concept du gestionnaire de formulaire intégré dans les principaux navigateurs. L'avantage incontestable de cette solution : aucune modification des applications web n'est nécessaire pour y intégrer la logique d'un SSO. Par ailleurs le Single Sign-On peut être étendu à des applications non web. Inconvénients : le seul apport est d'ordre ergonomique ; ni la sécurité ni la cohérence de gestion des comptes ne sont améliorées. Le choix de ce type de solutions induit par ailleurs un gros travail de déploiement sur les postes client. Exemple de produits : Cryptogram SSO, SSOX.

Une autre architecture hybride garde le principe d'une « boîte à mots de passe » mais cette fois-ci déportée côté serveur. Un « reverse proxy » remplie le double rôle d'authentification de l'utilisateur et de login de l'utilisateur auprès de chaque application. Comme la solution précédente cette architecture épargne le travail de modification des applications web existantes tout en évitant le travail déploiement sur les postes client. On peut également maintenir un bon niveau de sécurité en sécurisant les échanges entre le « reverse proxy » et les serveurs applicatifs. En revanche ce type de solutions nécessite une bonne connaissance de l'API de login des applications web.

La généralisation des portails en frontal des applications de l'établissement incite à localiser le service d'authentification au sein du portail. On considère donc que le portail est le seul point d'accès aux applications. Que les applications soient des « web services » ou des applications autonomes (de type CGI), elles délèguent au portail le travail d'authentification des utilisateurs. Au lancement de l'application, le portail fournit à l'application l'identité (et certains attributs) de l'utilisateur. Si cette solution a le mérite de proposer une architecture simple, elle n'offre pas toutes les garanties de sécurité. En effet, le portail n'est pas un modèle d'application dédiée ce qui augmente les risques de compromission du service d'authentification.

6 Quelques projets universitaires

Répondant à la problématique de l'authentification dans les applications web, de nombreuses universités en Europe et aux Etats-Unis, ont développé des solutions locales de SSO qui sont devenues des produits. Quelques uns de ces produits sont décrits ci-dessous ; pour la plupart distribués sous un régime de licence libre. Les architectes de ces différents projets participent par ailleurs au groupe de travail TF-AACE au sein de TERENA. Ce groupe a notamment produit un rapport sur les architectures d'Authentification et d'Autorisation [5].

6.1 A-Select (Surfnet – Pays Bas) [6]

Développé à l'initiative de SurfNet et utilisé dans le cadre de projets pilotes, A-Select gère l'authentification intra et inter établissement. L'architecture permet d'utiliser plusieurs «connecteurs» d'authentification locaux ou distants (Radius, LDAP,...) ainsi que plusieurs méthodes d'authentification (mot de passe, OTP, certificats,…) en fonction du niveau d'authentification requis par l'application.

6.2 CAS (université de Yale) [7]

Ce produit est le plus inspiré de Kerberos et bénéficie d'un bon niveau de sécurité. Il est le seul à prendre en charge la délégation de privilèges dans les architectures multi tiers (*proxied credential*) grâce à l'utilisation de tickets de délégation. Sa gestion de tickets « non rejouables » (invalidés après la première vérification) offre un très bon niveau de sécurité. CAS souffre initialement d'un manque de documentation, mais l'équipe du projet Esup-Portail [8] (projet de campus numérique basé sur des logiciels libres) a largement contribué à combler ce manque.

6.3 Moria (FEIDE - Norvège) [9]

FEIDE est un projet de système d'administration des utilisateurs, commun aux établissements d'enseignement supérieur Norvégiens. Moria, le serveur d'authentification de FEIDE permet le Single Sign-On web ; il assure également la diffusion des attributs utilisateurs. Le système propose a 2 interfaces :

- web pour interagir avec l'utilisateur (seul des tickets sont échangés)
- SOAP pour délivrer les attributs utilisateur aux applications (sur présentation du ticket).

Le système semble adopter une architecture centralisée (1 serveur Moria / N serveurs LDAP).

6.4 PAPI (RedIris – Espagne) [10]

PAPI gère l'authentification et l'autorisation pour l'accès à des ressources. Le système est utilisé pour contrôler l'accès à des bibliothèques universitaires espagnoles. Le système impose que l'utilisateur s'authentifie avant de contacter le gestionnaire de ressources. Autre inconvénient : une liste exhaustive des ressources doit être maintenue au niveau du serveur d'authentification.

6.5 Pubcookie (université de Washington) [11]

Un produit très bien packagé et documenté. Les échanges entre le serveur d'authentification (*login server*) et les agents d'authentification (*Apache module*) sont cryptés avec des clés symétriques ; un serveur de clés assure la distribution de ces clés sur les différents serveurs. Le module Apache permet de définir précisément l'ensemble des ressources à contrôler.

7 Gestion des identités entre établissements

Les ressources web d'un établissement sont principalement destinées à ses personnels, enseignants, étudiants mais pas uniquement. Des relations entre établissements se tissent pour le partage de ressources documentaires, la mise en place de groupes de travail, la formation à la carte inter universitaire. Des ressources locales doivent, dès lors, être accessibles à des personnes extérieures, ce qui requiert un contrôle d'accès donc une authentification ; or ces personnes ne sont pas connues du système d'authentification local. Ces exceptions sont aujourd'hui gérées de façon artisanale (gestion de « comptes invité », mots de passe partagé par un groupe, URLs cachées) donc à la fois peu sure et peu conviviale, alors que ces situations tendent à se généraliser. Shibboleth et Liberty Alliance sont 2 architectures possibles pour répondre à ce besoin de partage des identités numériques.

Le language SAML [15], récemment normalisé par le groupe *Oasis*, est utilisé dans la majorité des produits de gestion d'identités pour assurer les échanges entre les différentes briques. SAML définit des assertions d'authentification, d'autorisation ou des attributs ; les développeurs de Shibboleth ont développé la librairie OpenSAML, distribuée sous un régime de licence libre de droits. L'autre travail de normalisation en cours concerne le langage XACML [16] de définition de politiques de sécurité. L'objectif est de définir les autorisations d'accès aux ressources de l'établissement en dehors des applications, de façon à pouvoir maintenir une politique de sécurité cohérente. XACML offre des perspectives intéressantes mais il est encore peu implémenté.

7.1 Shibboleth [12]

Shibboleth est un projet universitaire de Internet2 visant à mettre au point un modèle permettant le contrôle d'accès à des ressources web, au-delà des frontières d'un établissement. Le projet envisage les relations entre établissements universitaires mais également les relations avec d'autres partenaires (revues en ligne par exemple). Le concept de base de Shibboleth est que le site gérant la ressource web accédée (*target site*) est responsable du contrôle d'accès à cette ressource ; l'authentification de l'utilisateur est déléguée à son établissement d'origine (*origin site*) qui devra également fournir certains attributs de l'utilisateur au processus de contrôle d'accès. Shibboleth permet au site d'origine d'utiliser le SSO de son choix. En délégant ainsi l'authentification on résout les problèmes de sécurité lié à la circulation des mots de passe et on évite la multiplication des comptes d'un même utilisateur dans plusieurs établissements. Le système permet par ailleurs de définir précisément la liste des attributs dévoilables à un site, en fonction de la relation de confiance avec ce dernier. On peut notamment ne fournir qu'un attribut générique de l'utilisateur, sans dévoiler aucune information nominative (Exemple : « étudiant en 2ème année de biologie »). Cette préoccupation de protection des données nominatives est très importante pour les architectes du projet Shibboleth.

7.2 Liberty Alliance [3]

Le projet regroupe de nombreux industriels (HP, Sun, Novell, France Telecom, American Express, AOL,…) et vise à définir des standards pour la gestion et l'échange des identités (alternative à Microsoft Passport). Contrairement à Shibboleth qui considère qu'un utilisateur dépend d'un organisme (*origin site*), Liberty Alliance considère un ensemble de comptes d'un même utilisateur et propose le recoupement (*federated identity*) de ces identités entre les partenaires. Un des objectifs du système est le Single Sign-On. Ce modèle semble plus adapté à des relations entre partenaires commerciaux qu'à des échanges entre universités.

7.3 Microsoft Passport [13]

Avec *Passport*, Microsoft propose un service de SSO web utilisable par tout un chacun et permettant même à l'utilisateur de contrôler les informations personnelles transmises à chaque site. Les technologies utilisées sont les mêmes que les autres

solutions de SSO web : redirections HTTP, cookies, tickets d'authentification. Seule la vision « MS-centrique » diffère puisque toutes les informations personnelles sur les utilisateurs sont centralisées sur le site *passport.com,* passage obligé pour s'identifier. *Passport* n'est donc pas un produit mais un service.

8 Conclusion

Les établissements de notre communauté ont encore rarement mis en œuvre un service commun de Single Sign-on web. C'est pourtant un élément fédérateur pour les applications web avant même le portail. C'est pourquoi les quatre projets de campus numériques retenus par le ministère intègrent tous une solution de SSO.

Le service de SSO se situe à la croisée des chemins entre la technologie LDAP et les services de gestion des identités à venir. Bien entendu les établissements ayant investi dans la mise en place d'annuaires LDAP d'établissement ont un effort moindre à fournir pour déployer un service de SSO. En outre, si l'annuaire LDAP d'établissement permet des gains d'efficacité importants sur les tâches d'administration, le service de SSO lui améliore directement la qualité du service rendu aux personnes. La transmission d'identités et d'attributs entre établissements, comme elle est envisagée dans Shibboleth, offre de nouvelles perspectives de coopération entre Universités ; elle semble la suite logique des travaux de normalisation du groupe SupAnn [17] dans le domaine des annuaires. Parallèlement, le groupe AAS [14], créé pour les besoins du Schéma Directeur des Espaces Numériques de Travail, a émis des recommandations concernant la gestion des identités. Ce document met en garde contre la propagation des mots de passe jusqu'aux applications.

La galaxie des produits de SSO web offre une diversité inattendue ; plus d'une dizaine de solutions universitaires ainsi que de nombreuses offres commerciales (Netegrity SiteMinder, SunOne Identity server,…). La comparaison est d'autant plus difficile que les architectures et surtout les terminologies diffèrent. Outre la solidité de l'architecture il importe de baser le choix d'un SSO sur ses capacités d'intégration dans le système d'information. Alors que SAML s'impose comme une norme pour les échanges entre services d'authentification, la normalisation des API permettant de communiquer une identité du serveur de SSO vers l'application se fait cruellement sentir. Dans ces circonstances, la disponibilité de plusieurs librairies clientes (Java, PHP, Perl) permet l'intégration légère avec la plupart des applications web. Cette démarche entamée, la possibilité d'une bonne intégration du service SSO devient un critère important dans le choix des solutions. Il importe également que le SSO s'appuie sur les services d'authentification mis en place dans l'établissement (NIS, LDAP, Kerberos,…). Les produits de SSO web s'inspirant de Kerberos sont ceux qui offrent les meilleurs gages de sécurité. CAS [7] notamment est un des seuls produits à pouvoir s'adapter aux architectures applicatives multi tiers sans concession sur son modèle de sécurité. Ces architectures multi tiers se sont pourtant généralisées avec le développement de SOAP et des services web.

Références

[1] Jennifer G. Steiner, Clifford Neuman et Jeffrey I. Schiller, *Kerberos : An Authentication Service for Open Network Systems*. Mars 1988. ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS

[2] R. Morgan, *WebISO: Service model and component capabilities. Octobre 2002.* http://middleware.internet2.edu/webiso/docs/draft-internet2-webiso-model-01.html

[3] Thomas Wason, Liberty Alliance, *Liberty ID-FF Architecture Overview*. 2003. http://www.projectliberty.org/specs/draft-lib-arch-overview-v1.2-04.pdf

[4] Chad La Joie , *Trusted Delegation of Privileges in an N-Tier Environment*. Juillet 2002. http://middleware.internet2.edu/webiso/docs/draft-lajoie-trust_and_delegation-02.html

[5] TERENA TF-AACE, *Deliverable B1 : Investigate the different approaches to inter- and extra-institutional A&A, analyzing the alternatives in architecture and protocols*. Juillet 2003. http://www.terena.nl/tech/task-forces/tf-aace/Del/B.1/TF-AACE-B1-1.0.pdf

[6] A-Select : http://a-select.surfnet.nl/

[7] CAS : http://www.yale.edu/tp/auth/

[8] Esup-Portail - *Groupe 1B - SSO et gestion des autorisations*.http://www.esup-portail.org/consortium/espace/SSO_1B/

[9] Moria : http://www.feide.no/programvare/authentication_model.html

[10] PAPI : http://www.rediris.es/app/papi/index.en.html

[11] PubCookie : http://www.pubcookie.org/

[12] Shibboleth : http://shibboleth.internet2.edu/

[13] Microsoft Passport Review Guide. 2003 http://www.microsoft.com/net/services/passport/review_guide.asp

[14] *Schéma directeur des environnements de travail – groupe AAS*. Juillet 2003. http://www.educnet.education.fr/equip/sdet.htm#a3

[15] *OASIS Security Services TC.* http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[16] *OASIS eXtensible Access Control Markup Language TC* . http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
[17] *SupAnn, groupe de travail sur les annuaires dans l'enseignement supérieur*. http://www.cru.fr/ldap/supann/