Extensions à OpenSSO : compatibilité et gestion des autorisations

Philippe Beutin
DSI Grenoble Universités
351, Avenue de la bibliothèque 38041 Grenoble Cedex 9
Philippe.Beutin@grenet.fr

Thierry Agueda
Université Pierre Mendès-France
38041 Grenoble Cedex 9
Thierry.Agueda@upmf-grenoble.fr

Gérard Forestier
Université Joseph Fourier
38041 Grenoble Cedex 9
Gerard.Forestier@ujf-grenoble.fr

Le Quyen La
Université Joseph Fourier
38041 Grenoble Cedex 9
Le-Quyen.La@ujf-grenoble.fr

Thi Hue Nguyen
Université Joseph Fourier
38041 Grenoble Cedex 9
Thi-Hue.Nguyen@ujf-grenoble.fr

Sylvain Pasutto
DSI Grenoble Universités
351, Avenue de la bibliothèque 38041 Grenoble Cedex 9
Sylvain.Pasutto@grenet.fr

Résumé

L'authentification unique (SSO) est une fonctionnalité majeure pour les accès aux applications accessibles par le Web. Si dans les propriétés de cette fonction les premières versions se sont attachées à l'aspect authentification, le SDET 2.0 met l'accent sur la partie gestion des autorisations. D'une manière générale, c'est la gestion de l'identité qui est au cœur du problème. Après étude, notre choix s'est porté sur le système Access Manager de SUN. Cette solution offre un sur ensemble fonctionnel de CAS et nous permet de gérer ces aspects naturellement. Access Manager est appelé OpenSSO depuis son passage dans le monde des « logiciels libres ». Pour ne pas être en marge de la communauté CAS et nous permettre de récupérer / contribuer à l'intégration de services métiers communs à différents établissements, nous montrons l'architecture de la solution qui rend compatible OpenSSO avec CAS 2.0. Cet article présente la démarche suivie et explique les mécanismes mis en œuvre dans la version opérationnelle de l'Espace Numérique de Travail (ENT) actuel des universités de Grenoble. Il présente les différents avantages que nous avons trouvés à la solution déployée et montre l'ouverture offerte vers les grandes fédérations d'identité existantes, suite logique du SSO.

Mots clefs

Accès Unique, SSO, OpenSSO, CAS, AAS, Authentification, Autorisation.

Introduction

Lors de la définition des ENT, l'accent a été mis sur l'aspect authentification unique (SSO). Cette brique du socle est primordiale. Si dans les propriétés de cette fonction les premières versions se sont attachées à l'aspect authentification, le SDET 2.0 met l'accent sur la partie gestion des autorisations.

Dans un cadre d'administration de services mutualisés, il est intéressant de disposer de mécanismes système permettant à différents acteurs humains de collaborer sur la gestion des droits, ce qu'un simple SSO ne sait offrir.

Lors de l'analyse des besoins, la dimension autorisation est apparue clairement. De même, un autre point qui nous a semblé vital pour rester cohérent avec une architecture haute disponibilité est que la fonction d'authentification doit pouvoir être redondée.

Cette redondance ne doit pas s'appliquer seulement à la tolérance aux pannes d'un composant, mais aussi à la qualité du service perçue par l'utilisateur.

Après étude, notre choix s'est porté sur le système Access Manager de SUN. Cette solution libre de droit d'exploitation nous permet de gérer ces aspects naturellement. Access Manager est maintenant dénommé OpenSSO depuis qu'il a été placé dans l'Open Source.

La publication du schéma de données et des protocoles utilisés nous assurent la pérennité de cette brique logicielle, d'une importance capitale.

Ce choix, différent de celui effectué par les autres universités, notamment celles qui ont choisi E-SUP, nous aurait placé, si nous ne faisions rien, en marge de la communauté. Il ne nous permettrait pas de récupérer / contribuer à l'intégration de services métiers communs à différents établissements. Il ne nous permettrait pas non plus de récupérer localement le travail fait sur des outils soutenus par le Conseil Régional Rhône-Alpes, comme par exemple le Bureau Virtuel. Il fallait ainsi pouvoir mettre en œuvre des mécanismes rendant notre SSO compatible avec CAS, ce qui a été possible car OpenSSO est un sur ensemble fonctionnel de CAS 2.0. OpenSSO présente également une API bien documentée qui permet d'étendre ses fonctions.

Cet article présente la démarche qui a été suivie et explique les mécanismes mis en œuvre dans la version opérationnelle de l'ENT actuel. Nous montrons l'architecture de la solution qui rend compatible OpenSSO avec CAS 2.0. Nous présentons ici les différents avantages que nous avons trouvés dans le système déployé et l'ouverture offerte vers les fédérations d'identités, suite logique des systèmes d'authentification unique. Dans ce domaine, il faut constater que les établissements d'enseignement supérieur et de recherche sont ouverts sur les autres structures du même type, qu'elles soient en France ou à l'étranger, mais également sur des partenaires industriels. La dimension cercles de confiance présente alors un grand intérêt pour ouvrir l'accès à des tiers connus à des sous-ensembles du Système d'Information. Se pose alors le choix de la fédération. Certains établissements ont fait le choix naturel de Shibboleth, brique logicielle compatible CAS/LDAP, certains autres ont choisi Liberty Alliance avec qui OpenSSO sait dialoguer. Le choix de notre solution avec ses extensions, nous permet d'opérer avec l'ensemble des infrastructures d'authentification déployées par les universités, mais également par des partenaires institutionnels publics ou privés qui ont fait le choix de l'autre fédération.

1 Le SSO CAS

CAS (Central Authentication Service) est un système de SSO développé par l'Université de Yale.

1.1 Rappels sur CAS

Il est basé sur un échange de tickets entre le client (navigateur), le serveur d'authentification et l'application protégée.

Le TGC[1] est le cookie de session, stocké par le navigateur client, authentifiant la session utilisateur. Le ST[2] est généré par le serveur CAS et transmis via le navigateur client à l'application sécurisée dans l'entête de la requête (paramètre du GET HTTP).

1.2 Fonctionnement de CAS

La Figure 1 schématise le fonctionnement de CAS.

1 : Le client adresse une requête d'accès à l'application.

2 : Il est redirigé vers le serveur CAS.

3 : Après authentification, CAS délivre un *Service Ticket* et redirige l'utilisateur vers l'application.

4 : L'application vérifie la validité du *Service Ticket* auprès de CAS.

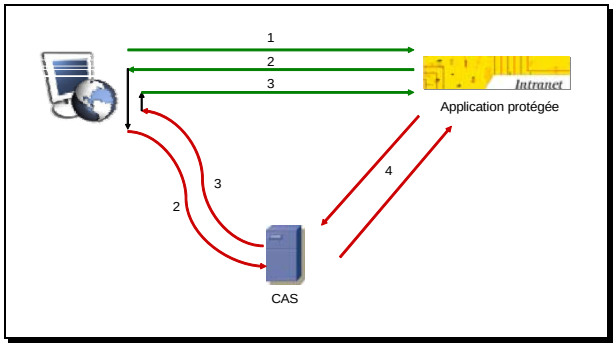

Figure 1 : *fonctionnement de CAS*

1.3 Avantages et limites de CAS

Le mécanisme de SSO CAS est une solution très répandue dans les établissements universitaires français pour laquelle de nombreuses applications sont déjà intégrées. Il nous paraît primordial de ne pas perdre cette compatibilité tant pour bénéficier de l'existant que pour participer à son enrichissement.

Toutefois, l'esprit dans lequel a été fait CAS n'ouvre pas le système à la gestion des autorisations. Il ne traite ni des droits des utilisateurs, ni de la validité du compte. Il est apparut que des utilisateurs dont le compte, par nécessité de gestion, était présent mais bloqué pouvaient toujours accéder aux applications protégées[3].

Une autre limite de ce SSO dans notre contexte mutualisé, est qu'il n'est pas possible de le faire fonctionner sur plusieurs établissements. Il faut un serveur par établissement ou par annuaire d'authentification.

Sur le plan de la haute disponibilité, bien que très fiable, le système dans sa version 2.0 ne peut pas être redondé (fiabilisé par j'adjonction de matériels spécialisés…). Des travaux en cours devraient permettre cette redondance.

Nous nous limitons à ces quelques éléments descriptifs sur CAS pour la compréhension de l'article. Pour plus d'informations, nous invitons le lecteur à consulter les

[1] TGC : Ticket-Granting cookie

[2] ST : Service Ticket

[3] Les universités de Grenoble utilisent l'annuaire triode Process® (Agalan), surensemble de SUPANN. Il gère les utilisateurs selon 4 statuts : PROVisoire, OFFIciel, SUSPendu ou OBSOlète.

nombreuses documentations disponibles, en particulier, celles données sur le site ESup-Portail[4].

2 OpenSSO

OpenSSO est le terme maintenant employé par Sun pour proposer son gestionnaire d'accès *Access Manager* dans le monde du logiciel libre, de la même façon qu'il propose OpenSolaris, OpenPortal… [1] *Access Manager* est un sur ensemble d'OpenSSO car il intègre d'autres outils propres à la suite Sun Java ES décrite dans le chapitre 4.

OpenSSO, contrairement à CAS, fonctionne sur un **échange** de cookie entre l'application et le serveur d'authentification. Ce cookie de session va permettre de récupérer auprès du serveur d'authentification le jeton SSO (*SSOToken*).

OpenSSO bénéficie aujourd'hui d'une communauté en forte expansion (voir [2]).

2.1 Pourquoi OpenSSO ?

OpenSSO est conçu pour fournir des services d'authentification, d'autorisation et de gestion de sessions pour les applications Java, les applications Web et les applications orientées services. Il fournit aussi des services de fédération d'identité.

Access Manager permet de couvrir de nombreux besoins de notre environnement et offre de multiples possibilités d'évolution et d'intégration. Parmi les fonctionnalités les plus marquantes, citons :

2.1.1 Authentification, autorisations et journalisation

A ce niveau, nous avons la possibilité d'associer à un utilisateur un ensemble de services par rapport à ses fonctions (rôles) ou à son appartenance à une entité (organisation), ainsi qu'un mode d'authentification. Il est fourni avec un grand nombre de modules d'authentification déjà intégrés : LDAP bien sûr, mais aussi Active Directory, Certificats, JDBC, MSISDN[5], Radius, SAML (2.0), SecurId[6], etc. et la possibilité de les chaîner.

Le système SSO permet de gérer les accès à des applications autant qu'à des pages Web avec une grande finesse d'administration et un héritage fort. Les systèmes de SSO sont basés sur un modèle client/serveur dont la partie cliente est généralement incluse dans l'application (mode intrusif). Il est possible d'éviter cette intrusion dans les applications de type Web par l'utilisation d'un module complémentaire au serveur qui héberge l'application.

La suite JES dans laquelle se trouve OpenSSO offre un ensemble d'outils de statistiques et de suivi d'utilisation des services. Il est possible de suivre les taux de fréquentation, le nombre de sessions actives, les temps de réponses moyens du portail, etc. mais aussi, les créations de sessions, les appels SSO, les politiques d'accès…

2.1.2 Administration fine et dynamique

Une gestion hiérarchique des entités et des utilisateurs est possible grace au mécanisme de délégation. La notion de rôle avec priorités permet de proposer très finement des services aux utilisateurs selon leur appartenance à une entité, à une fonction, à un groupe…

Côté gestion du système une console d'administration permet de gérer les différents paramètres, les identités, et d'administrer les différentes briques de la suite logicielle comme le portail.

2.1.3 Haute disponibilité

La possibilité de rendre l'ensemble des briques hautement disponibles avec tolérance de panne. La récupération de session est assurée par *Message Queue* sui n'a pas été retenue pour l'instant.

2.2 Fonctionnement d'OpenSSO

Nous allons nous intéresser à la gestion de l'authentification unique et des autorisations. Comme nous l'avons vu précédemment, *Access Manager* manipule un arbre d'organisations (Universités et composantes) avec, pour chacune, des services, des utilisateurs, des rôles et des politiques.

Pour authentifier l'utilisateur et pouvoir proposer ses propriétés à une application cliente, Access Manager utilise un cookie dénommé *iPlanetdirectoryPro*. Ce cookie va permettre de récupérer le *SSOToken*, un objet Java contenant des attributs de l'utilisateur (identifiant, rôles, organisation…).

2.2.1 Authentification unique

Le principe de l'authentification unique est le suivant :

- Un « **agent** » est installé sur le serveur d'application à protéger. Son but est d'intercepter la requête du client et de dialoguer avec *Access Manager* pour vérifier si l'utilisateur a le droit d'accéder à la ressource qu'il demande et de répondre en conséquence au client.

- Sur Access Manager est définie une liste de « **politiques** ». Ces politiques décrivent les droits d'accès aux ressources en fonction du statut de l'utilisateur, de ses rôles, de son rattachement...

2.2.2 Policy Agent

Sun Microsystems propose un ensemble d'agents, librement téléchargeables, avec interface d'installation graphique qui simplifie grandement le travail. Le lecteur peut consulter sur internet les sites :

[4] http://www.esup-portail.org/consortium/espace/SSO_1B/cas

[5] MSISDN : Mobile Station ISDN, WAP…

[6] RSA SecurID® L'authentification à deux facteurs est fondée sur la combinaison d'un identifiant immuable connu (mot de passe ou PIN) et d'un dispositif physique (ou authenticateur). Ce système garantit un niveau d'authentification utilisateur bien plus fiable que les mots de passe réutilisables. Le mot de passe est différent toutes les 60 secondes, aucune violation de sécurité répertoriée depuis 15 ans…[3]

http://www.sun.com/download/index.jsp?cat=Identity Management&tab=3&subcat=Policy Agents

https://opensso.dev.java.net/public/agents.html

afin d'obtenir les agents disponibles.

À titre d'exemple, l'agent pour Apache 1.3.33 est en fait un module d'Apache qui intercepte la requête puis renvoi à l'utilisateur une page d'erreur 403 si celui-ci n'est pas habilité à voir la ressource demandée.

En dehors des serveurs d'applications les plus courants (Apache, Tomcat, IIS, JBoss, Oracle, SAP…) pour lesquels Sun propose des agents, il est toujours possible d'en développer de nouveaux ou d'utiliser les APIs fournies (java ou C). Étant donné la richesse des fonctionnalités d'OpenSSO, ce développement est plus complexe qu'avec CAS.

2.2.3 Politique

Une politique est composée de **règles** (une url à protéger, un service, un profil d'utilisateur, un Web Service…), de **sujets** (des utilisateurs, les membres d'un rôle, un groupe LDAP, les membres d'une organisation…) et de **conditions** (la méthode d'authentification utilisée, l'adresse IP du client, l'heure…).

Exemple : la politique *gu_appli_DSI* protège l'url http://intranet.grenoble-universites.fr:8080/adminDSI/* (règle) en accordant l'accès aux utilisateurs qui ont le rôle *gu_admin_DSI* (sujet) si leur adresse IP est dans la plage 130.190.217.000 à 040 (condition). Aucune action n'est nécessaire sur l'application protégée, le pilotage de la politique de sécurité se fait au niveau d'*Access Manager*.

Access Manager décide si, selon l'url demandée, il autorise ou non l'utilisateur connecté à y accéder. Ni l'agent, ni l'application ne gèrent les droits de l'utilisateur. Ce mode est **non intrusif**.

2.2.4 Schéma de fonctionnement

La Figure 2 présente le fonctionnement de OpenSSO.

1 : Le client adresse une requête d'accès à l'application qui est interceptée par l'agent SSO.

2 : Il est redirigé vers le serveur d'authentification Access Manager.

3 : Après authentification, *Access Manager* délivre un *SSOToken* et redirige l'utilisateur vers l'agent.

4 : L'agent vérifie auprès d'Access Manager la validité de la session.

5 : Puis, il lui demande si le client a l'autorisation ou le refus d'accéder à l'application.

6 : L'agent renseigne enfin Access Manager pour la journalisation des actions effectuées.

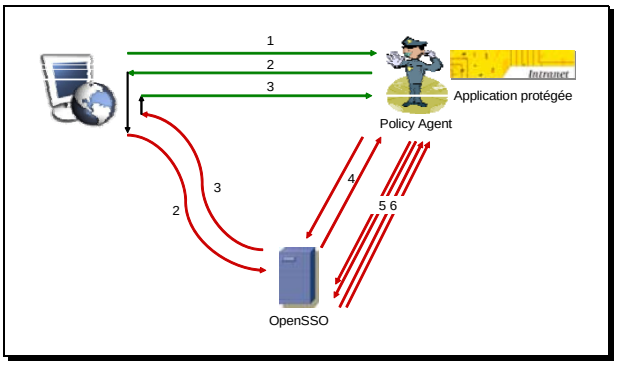

Figure 2 : *fonctionnement d'OpenSSO*

2.3 Limites et points forts

Access Manager intègre un système de partage de sessions entre un ensemble de serveurs d'authentification faisant partie d'un cercle de confiance. Ceci lui permet, contrairement à une solution basée sur CAS, d'avoir un procédé réparti d'authentification et d'habilitation des accès à l'ENT et à ses applications externes. La brique Message Queue (non implémentée dans notre architecture) permet le stockage partagé des sessions et donc leur récupération même après un crash. Il supporte donc la redondance, la montée en charge et les tolérances aux pannes. Il est tout à fait à sa place dans une solution de haute disponibilité.

Cependant, OpenSSO souffre pour le moment d'une limitation dans le fait que les agents, briques nécessaires à la mise en place du SSO, ne sont pas disponibles pour toutes les plates-formes de serveurs d'applications. Exemple : le serveur WebObject sur Mac OS.

D'un autre côté, OpenSSO étant maintenant dans le monde du logiciel libre, on ne peut qu'espérer que les solutions et remèdes à ce problème arriveront sans tarder, par l'effort de la communauté, non seulement universitaire, mais aussi par celle du logiciel libre.

Pour l'aspect sécurité, OpenSSO fonctionne sur un cookie lié au domaine, dans le sens DNS. Celui-ci ne peut être propagé lorsque le service que l'on désire fournir est dans un autre domaine que celui du gestionnaire d'accès. Le *Cross Domain SSO* permet de prendre en compte de façon maîtrisée l'aspect multi-domaine.

3 Extension d'OpenSSO : émulation du protocole CAS

3.1 Pourquoi étendre OpenSSO ?

Il nous est apparu primordial de conserver, d'un côté, la compatibilité avec les applications CAS et sa communauté et, d'un autre côté, de profiter pleinement des fonctionnalités offertes par la suite.

Les pré-requis étaient de conserver une transparence de fonctionnement vis-à-vis des applications cassifiées, souvent fournies par des prestataires extérieurs donc non maîtrisées. Exemples : Encyclopædia Universalis, Bureau Virtuel Rhône-Alpes, KSup…

L'extension d'OpenSSO est une émulation du protocole CAS au sein d'Access Manager. Il doit permettre à un utilisateur authentifié sur l'ENT d'accéder aux applications cassifiées sans se ré authentifier et inversement.

3.2 Fonctionnement de l'extension d'OpenSSO

Comme nous l'avons vu dans le chapitre 1.2, CAS utilise un cookie (TGC) pour assurer la persistance de la session utilisateur et un Service ticket (ST) pour dialoguer avec l'application.. Dans l'extension d'OpenSSO que nous avons réalisée, le TGC n'est plus nécessaire. Le cookie *iPlanetDirectoryPro* est un sur ensemble du TGC dont il couvre toutes les fonctionnalités. Seul le ST doit être généré. Access Manager va assurer la persistance de la session et donc assurer la haute disponibilité de cette session. Dans le paragraphe 3.3.4, les traces illustrent les différences de fonctionnement, côté navigateur client, entre les deux systèmes.

La Figure 3 schématise le fonctionnement de l'extension à OpenSSO dans le cas d'un utilisateur non authentifié.

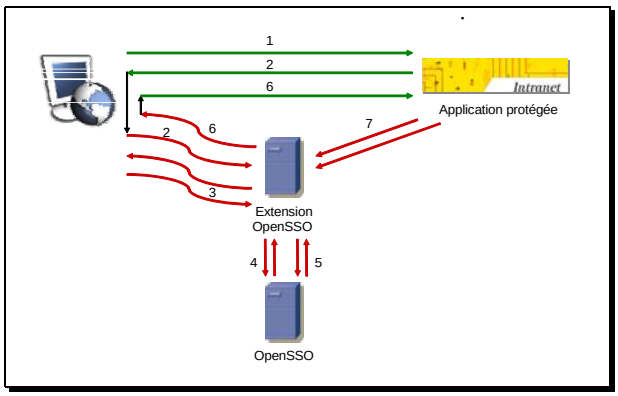

Figure 3 : *fonctionnement de l'extension d'OpenSSO*

1 : Le navigateur client adresse une requête d'accès à l'application protégée (cassifiée).

2 : L'application le redirige vers l'extension d'OpenSSO.

3 : L'extension présente à l'utilisateur le formulaire d'authentification. Celui-ci donne son login/mot de passe et valide le formulaire.

4 : L'extension créé la session du client auprès d'OpenSSO et vérifie sa validité.

5 : L'extension vérifie auprès d'OpenSSO si l'utilisateur est autorisé à accéder à l'application protégée.

6 : À partir du *SSOToken* validé par OpenSSO, l'extension fournit au navigateur de l'utilisateur un cookie *iPlanetDirectoryPro* et fabrique un *Service Ticket* à destination de l'application. Il redirige l'utilisateur vers l'application.

7 : L'application vérifie la validité du *Service Ticket* auprès de l'extension.

L'extension d'OpenSSO se comporte comme un frontal CAS à OpenSSO.

Dans un autre scénario, si l'utilisateur est déjà authentifié sur l'ENT ou sur CAS, il a déjà son cookie *iPlanetDirectoryPro* et l'étape 3 n'est pas exécutée. La validité de la session est juste vérifiée auprès d'OpenSSO. L'extension informe également OpenSSO que l'utilisateur a rafraîchit sa session.

3.3 Réalisation et mise en œuvre

L'extension d'OpenSSO joue le rôle d'un serveur CAS mais il est aussi un relais d'OpenSSO. À sa date de développement, OpenSSO fournissait deux APIs JAVA et C pour réaliser des clients OpenSSO. Notre choix s'est porté sur JAVA. Nous avons réécrit tous les servlets normalement fournis par le serveur CAS au client CAS : un servlet pour l'authentification, un servlet pour la validation de l'authentification (*Service Ticket)* et un servlet pour terminer la session.

3.3.1 Servlet pour l'authentification

Il réalise les actions suivantes :

- récupération des identifiants de l'utilisateur ;
- authentification auprès d'OpenSSO ;
- création du *Service Ticket* et mise en cache ;
- envoi du *Service Ticket* et du *SSOtoken* au navigateur ;
- redirection vers l'application.

3.3.2 Servlet de validation de l'authentification

Il assure la validation du *Service Ticket* et retourne l'identité de la personne, dans un format XML.

3.3.3 Servlet de déconnexion

Il est chargé de libérer la session en détruisant le *SSOToken*.

Pour la mise en œuvre de l'extension à OpenSSO, nous avons déployé les servlets sur un serveur *Tomcat* couplé avec *Apache* et *ssl*. Pour rester transparent auprès des clients CAS, nous avons mis en place une configuration de telle manière que les urls implémentées dans le client CAS restent identiques.

Pour que l'extension puisse utiliser openSSO pour créer la session ou la valider, il est nécessaire d'installer et de configurer le client amSdk fourni par OpenSSO.

3.3.4 Traces d'utilisation

Si, avec un navigateur (Firefox avec TamperData), nous regardons les traces d'exécution de CAS puis de l'extension d'OpenSSO, on observe les caractéristiques suivantes.

Pour CAS :

- Le cookie CAS : CASTGC=TGC-320-ZBFJ5rJx8Tb7QFFWPTLS8aaomRqXVq14xJp9wkwy85ZMdW0QBN ; Path=/UDS ; Secure
- Une url d'appel : https://cas.grenet.fr/UDS/login?service=http://www.universalis-edu.com/?sso_id=966
- La redirection par CAS : http://www.universalis-edu.com/?sso_id=966&ticket=ST-523-2NPVL1VhbqirKl2lV38L

Et pour l'extension d'OpenSSO :

- Le cookie OpenSSO : iPlanetDirectoryPro=AQIC5wM2LY4SfcxbtS22m4KtloR9dRVN2FjitOl9JI8Vud0E%3D%40AAJTSQACMDEAAlMxAAIxMQ%3D%3D%23;
- L'url d'appel : https://cas.grenet.fr/UJF/login?service=http://www.universalis-edu.com/?sso_id=947
- La redirection par l'extension : http://www.universalis-edu.com/?sso_id=947&ticket=ST-23518261517786872212

4 L'extension d'OpenSSO dans notre architecture ENT

Nous pouvons maintenant regarder la place de cette extension faite sur OpenSSO dans notre architecture ENT.

L'ENT est commun aux 5 établissements de l'académie de Grenoble :

- Université Joseph-Fourier – Grenoble 1
- Université Pierre-Mendès-France – Grenoble 2
- Université Stendhal – Grenoble 3
- Institut National Polytechnique de Grenoble
- Université de Savoie

4.1 Architecture logicielle

Les différentes briques composant la suite Sun Java Enterprise System sont les suivantes :

- *Directory Server* : serveur d'annuaire LDAP qui fournit la persistance des données propres à l'ENT. C'est l'annuaire applicatif qui stocke les comptes utilisateurs du portail et leurs personnalisations, droits, configurations… Il gère plus de 90 000 utilisateurs et une centaine de rôles. [5]
- *Access Manager* : assure les services d'authentification, d'accréditation et de journalisation. Il propose une console d'administration très puissante permettant des modifications à chaud, une finesse de détails profonde, basé sur l'héritage, et une gestion d'identité complète. [6]
- *Message Queue* : permet la récupération de session même après un arrêt du service. Non implémenté sur Grenoble pour l'instant.
- *Portal Server* : est un moteur de génération de portail. Il permet de créer un portail « simplement » avec une interface graphique. Chaque utilisateur se voit attribuer un portail logique (description XML) et un ensemble de fichiers pour le générer selon son appartenance à une entité, ses rôles et ses personnalisations. Il n'y a pas de limites, il est possible de créer des portails complètement différents, autant dans la forme que dans le fond, simplement par l'appartenance à un rôle ou non, à une organisation... Sur Grenoble, nous avons fait le choix de conserver une description logique du portail identique à tous les établissements afin de pouvoir proposer des services d'une autre université aux utilisateurs. [7]
- *Portal Server Mobile Access* : propose un ensemble de protocoles supplémentaires pour afficher le portail (HDML, cHTML, iHTML, JHTML, XHTML, VoiceXML, WML) via le langage aml ainsi qu'un mécanisme de détection de client, une interface de personnalisation de l'affichage selon le type de client…
- *Search Engine* : Portal Server intègre un moteur de recherche pouvant indexer tout site, à la manière d'un crawler. Les informations (Resources Descriptions) sont stockées dans plusieurs bases de données afin de fournir à l'utilisateur une modularité de la recherche.
- *Portal Server Secure Remote Access* : passerelle fournissant un accès sécurisé à l'intranet.

– *Web Server* : serveur http pouvant exécuter du code Java (sauf EJB). Il est possible d'utiliser d'autres serveurs certifiés J2EE mais l'avantage de celui-ci est d'être intégré dans l'installation de la suite, d'avoir une interface d'administration web et d'être très performant.

– *Calendar Server, Messaging Server, Instant Messaging Server* : gestionnaire d'agenda, de mail, de chat. Ces outils n'ont pas encore fait l'objet de tests poussés.

Nous avons fait le choix d'installer ces éléments sur des plates-formes Solaris 10 plutôt que Linux au vu des performances et de la stabilité pour ce produit particulier. Concernant Windows, cela n'était pas adapté à notre culture, aussi, cette plateforme n'a pas été testée.

Dans la Figure 4, nous présentons les briques utilisées dans notre architecture d'ENT.

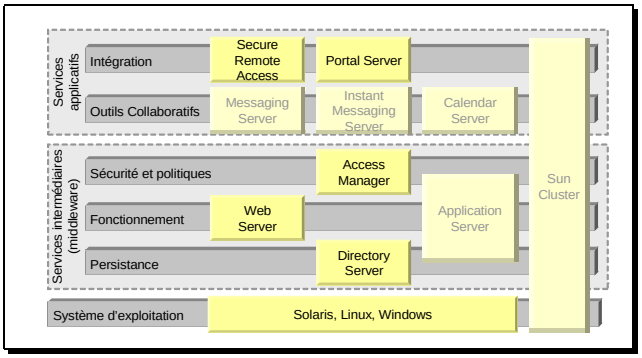

Figure 4 : *briques de la suite Sun Java ES*

Dans le Schéma Directeur des Espaces numériques de Travail (SDET), ces briques, et les applications externes au socle de l'ENT, trouvent leur place comme le montre la Figure 5 :

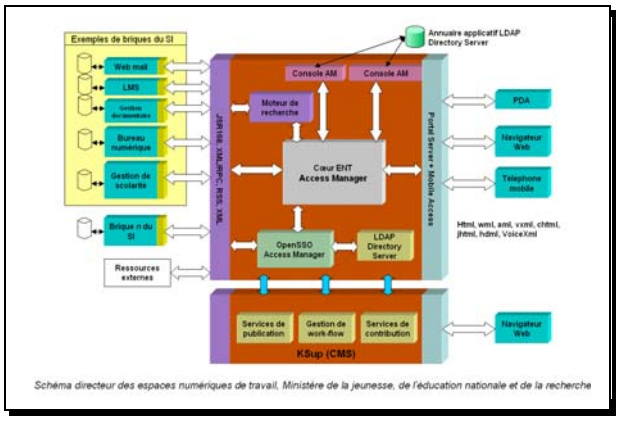

Figure 5 : *instanciation du SDET par le socle JES*

4.2 Architecture matérielle

Deux serveurs se partagent le service d'authentification. Ils ont chacun un annuaire applicatif propre à l'ENT (*Directory Server*) en *Multi-Mastering* (MMR) et un serveur d'authentification, d'autorisation et de journalisation (*Access Manager*) en équilibrage de charge. Deux autres serveurs assurent la génération du portail utilisateur (*Portal Server*) en équilibrage de charge également. L'équilibrage de charge est assuré par le routeur (*Cisco 6500*) qui effectue également la détection de perte de service.

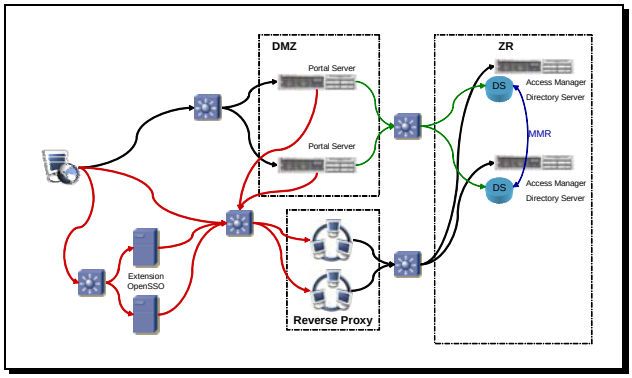

Figure 6 : *Architecture matérielle de l'ENT*

4.2.1 Quelques chiffres

Le nombre total d'utilisateurs potentiels est de 80 000. Le 10 septembre 2007 (après 6 mois de fonctionnement, un jour moyen de fonctionnement), 2554 utilisateurs se sont connectés avec succès sur le premier serveur d'authentification, 2767 sur le second serveur et 2309 authentification ont été générées par l'intermédiaire de l'extension d'OpenSSO, soit un total de 7630 sessions. Des pointes de trafic atteignent les 2500 sessions simultanées.

4.3 Comparatif des fonctionnalités

Si nous étudions les fonctionnalités de CAS, d'OpenSSO et de notre extension d'OpenSSO, nous remarquons que nous tirons parti des avantages de chacun et fournissons l'ensemble des fonctionnalités disponibles.

Le Tableau 1 propose une comparaison des fonctionnalités :

Fonction	CAS	OpenSSO	Extension OpenSSO
Authentification	☑	☑	☑
Autorisation	☐	☑	☑
Journalisation	☑	☑	☑
Cross-Domain SSO	Non concerné	☑	Non concerné
Fédération	Shibboleth	Liberty Alliance	Liberty Alliance et Shibboleth
Interface d'administration	☐	☑	☑
Haute disponibilité	☐	☑	☑
Configuration dynamique	☐	☑	☑
Multi-site multi configurations	☐	☑	☑[7]
Gestion centralisée des politiques d'accès	☐	☑	☑[8]

Tableau 1 : *fonctionnalités comparées de CAS, OpenSSO et l'extension d'OpenSSO*

Conclusion et perspectives

Le système mis en œuvre est opérationnel et fonctionne soit en autonome (l'utilisateur se connecte directement à des applications cassifiées), soit à travers l'ENT. Dans ce dernier cas, il s'authentifie à partir de la mire de l'ENT et peut ensuite accéder aux différentes applications qu'il utilisait précédemment, comme l'application « bureau virtuel ».

Nous apprécions particulièrement la console d'administration qui nous permet de suivre l'utilisation de notre ENT, des sessions en cours et de leur positionnement sur les serveurs en équilibrage de charge.

Lors de la bascule SSO de CAS à notre système, nous avons pu constater que les applications existantes déjà intégrées ont migré de façon transparente pour l'utilisateur, mais aussi pour les administrateurs de ces applications.

La gestion des autorisations par *Access Manager* nous permet d'adopter une autre démarche d'intégration d'applications qui évite, pour certaines d'entre elles, des imports d'utilisateurs dans la base de l'application. Les droits sont gérés hors application à partir de scripts ou de la console d'administration. Ce qui nous permet de définir une politique globale de sécurité et de contrôle d'accès.

Nous allons maintenant pouvoir avancer sur différents axes :

- D'une part, s'assurer de la réutilisation de portlets JSR168 déjà cassifiées.

- D'autre part, améliorer l'infrastructure pour la rendre encore plus transparentes aux pertes de services en mettant en place la brique *Message Queue* qui permet d'assurer la persistance de services sans perte de sessions.

Nous souhaitons également aller plus loin dans la gestion des autorisations et exploiter pleinement les fonctionnalités d'Access Manager pour appliquer intégralement notre politique de sécurité et d'accès aux applications. À titre d'exemple, la brique *Portal Server Secure Remote Access* [8] nous permet d'assurer le SSO dès l'accès aux réseaux non sécurisés et de résoudre les problèmes de sécurité du nomadisme.

Notre architecture nous permet d'intégrer dans une politique de fédérations d'identités des services respectant les spécifications de Liberty Alliance comme de Shibboleth.

Bibliographie

[1] Christophe Bardy, Après Novell, Sun publie officiellement son projet open source de gestion d'identités, OpenSSO. *Le Monde Informatique*, 30 août 2006.

[2] OpenSSO : http://opensso.dev.java.net/

[3] RSA Security : http://france.rsa.com/

[4] Dico du Net : http://www.dicodunet.com/

[5] Documentation Sun Java ES Directory Server : http://www.sun.com/software/products/directory_srvr_ee/

[6] Documentation Sun Java ES Access Manager : http://www.sun.com/software/products/access_mgr/

[7] Documentation Sun Java ES Portal Server : http://developers.sun.com/portalserver/reference/docs/

[8] Patrick Petit, Philippe Beutin, Jean-François Scariot et Christian Lenne. Et si l'infrastructure ENT servait à gérer le nomadisme. *JRES 2007*. Strasbourg.

[7] Limité par les applications intégrées (cassifiées) qui ne peuvent traiter le multi-site.

[8] Limitées à un ensemble d'utilisateurs a droit ou non à l'accès à la ressource selon certaines conditions.