République Algérienne démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme Master Recherche en Informatique

Option: Réseaux et systèmes distribués

Thème

Proposition d'un système d'authentification unique pour l'université de Bejaïa

Réalisé par :

M^{elle}. ADRAR Amal

Devant le jury composé de :

Président :	Dr. M. MOHAMMEDI	M.C.A	U.A/Mira Bejaia
Examinateur :	Dr. K. OUAZINE	M.C.B	ESTIN Amizour Bejaia
Promotrice :	Dr.L. HAMZA	M.C.A	U.A/Mira Bejaia

Promotion 2021-2022

Remerciements

Je tiens à remercier avant tout le bon **Dieu** de m'avoir donné l'opportunité de suivre ce master et de m'avoir donné la force, la volonté et la patience pour réaliser ce travail.

Je remercie infiniment ma très chère promotrice Mme HAMZA Lamia pour son soutient, ses encouragements et ses précieux conseils. Je tiens à lui exprimer ma profonde gratitude pour son sérieux, sa gentillesse et pour m'avoir toujours orienté de manière très efficace.

Je remercie également les membres du jury d'avoir accepté d'examiner ce mémoire.

Je remercie vivement tous ceux et celles qui m'ont encouragé pour suivre ce master et soutenu tout au long cette période. Parmi eux mes collègues du centre de calcul en général et ceux de la section E-learning en particulier.

Je ne remercierai jamais assez mon adorable famille pour leur présence, leur soutient, leur encouragement, …

Tables des matières

Liste des abréviations

API: Application Programming Interface

AS: Authentication Server

CSP: Cloud Service Provider

IaaS : Infrastructure as a Service

IAM : Identity and Access Management

IAMaaS : IAM-as-a-service

IDaaS: Identity-as-a-service

IdP: Identity Provider

IdSP: Identity Service Provider

JWT: JSON Web Token

KDC: Key Distribution Center

LDAP: Lightweight Directory Access Protocol

Moodle: Modular Object Oriented Dynamic Learning Environment

NIST: National Institute of Standards and Technology

OASIS: Organization for the Advancement of Structured Information Standards

OAuth: Open Authorization

OP: OpenID Provider

PaaS: Platform as a Service

RP: Relying Party

SaaS: Software as a Service

SAML: Security Assertion Markup Language

SP: Service Provider

SPML: Service Provisioning Markup Language

SSO: Single Sign-On

TGS: Ticket Granting Server

XACML: eXtensible Access Control Markup Language

Table des figures

Liste des tableaux

Introduction générale

Aujourd'hui, de nombreuses entreprises comptent des centaines voire des milliers d'employés. Chaque utilisateur dispose de plusieurs comptes pour accéder à de nombreuses applications accessibles à distance via le réseau [1]. Ces applications peuvent être internes à l'entreprise ou externes telles que les applications SaaS (Software as a Service) hébergées par des fournisseurs de services Cloud. Les entreprises se tournent de plus en plus vers les fournisseurs de services externes pour offrir des applications spécifiques et ciblées à leurs utilisateurs et éliminer la tâche complexe de prise en charge du système en interne [1]. À mesure que le nombre d'applications par utilisateur augmente, il devient plus difficile de gérer les informations d'authentification. Dans ces environnements, les systèmes de gestion des identités sont d'une importance capitale. En effet, l'utilisateur doit mémoriser un mot de passe distinct pour chacune des applications, cette situation a plusieurs conséquences. Tout d'abord, plus les utilisateurs doivent mémoriser de mots de passe, plus la probabilité qu'ils utilisent des mots de passe non sécurisés (faciles à deviner pour les attaquants) est grande ce qui est une menace pour la sécurité globale du système. Aussi, plus l'utilisateur doit mémoriser de mots de passe, plus il est probable qu'il en oublie certains ce qui conduit à l'augmentation du nombre d'appels au service d'assistance. Pour les grandes entreprises, les dépenses liées au service d'assistance peuvent atteindre un montant non négligeable. En outre, en s'authentifiant séparément à chaque application, les utilisateurs perdent du temps, les tentatives d'authentification infructueuses en raison d'informations d'identification mal saisies peuvent prolonger cette perte de temps et de productivité. Tous ces problèmes peuvent être résolus par la mise en place d'un système d'authentification unique (ou Single Sign-On (SSO) en anglais) qui est une fonctionnalité importante d'un système de gestion des identités. L'authentification unique permet aux utilisateurs de s'authentifier une seule fois et d'utiliser ensuite librement tous les services et ressources mis à leur disposition [2]. L'université de Béjaia comme toute organisation ou toute université met à disposition de ses étudiants et son personnel enseignants et ATS un ensemble de services nécessitant une authentification. La mise en œuvre d'un système d'authentification unique au sein des universités est importante pour éviter ou remédier à tous les problèmes cités auparavant. À l'université de Bejaia, ce besoin se fait de plus en plus ressentir.

L'objectif de ce mémoire est de mettre l'accent sur l'importance de la mise œuvre d'un système d'authentification unique pour palier au problème lié à la multiplicité des systèmes d'authentification au sein d'une organisation et proposer une solution de mise en œuvre de ce système pour l'université de Bejaia.

Nous avons organisé ce mémoire de la manière suivante.

Dans le chapitre 1, nous avons présenté des notions de base sur l'identité, l'authentification, le Cloud computing et la gestion des identités et des accès. Ces notions sont fondamentales pour la compréhension des autres chapitres de ce mémoire.

Dans le chapitre 2, nous avons décrit le fonctionnement de quelques protocoles d'implémentation du SSO qui sont SAML, Kerberos et OpenID Connect et passer en revue quelques travaux qui ont utilisé ces protocoles pour l'authentification et le SSO. Nous avons terminé ce chapitre en citant les avantages et inconvénients du SSO.

Dans le chapitre 3, nous avons parlé du besoin de la mise en place d'une solution d'authentification unique dans le cas de l'université de Béjaia et nous avons proposé une solution permettant de répondre à ce besoin.

Enfin, nous avons conclu par un résumé de ce qui a été présenté dans ce mémoire et quelques perspectives.

Chapitre 1 Généralités sur l'authentification, le Cloud computing et la gestion des identités et des accès

1.1 Introduction

Tout système informatique disposant de ressources internes ou externes à l'organisation nécessite un système de contrôle d'accès permettant de restreindre l'accès à ces ressources aux seuls utilisateurs autorisés. Le mécanisme de contrôle d'accès passe par plusieurs processus qui sont l'identification, l'authentification, l'autorisation et la traçabilité [3]. Ce chapitre est consacré à l'introduction de plusieurs concepts qui sont l'identité, l'authentification, le Cloud computing et la gestion des identités et des accès.

1.2 Définition de l'identité

L'ISO/IEC 24760-14 basé sur la recommandation UIT-T Y.2720 donne la définition suivante de l'identité « information utilisée pour représenter une entité dans un système d'information et de communication ». Sachant qu'une entité peut représenter une personne physique ou morale (organisation, entreprise, etc.), une ressource (tel qu'un matériel informatique, etc.) ou un groupe d'entités individuelles. Une entité peut posséder plusieurs identités numériques en fonction de l'environnement. Par exemple, un individu peut posséder une identité professionnelle représentée par des informations concernant son activité professionnelle et une identité familiale décrite par d'autres informations plus personnelles [4].

1.3 Définition de l'authentification

Lorsque dans un système seuls les individus ou objets sélectionnés sont autorisés à entrer et à utiliser les ressources du système, un mécanisme de contrôle d'accès, notamment d'authentification, devient nécessaire. En effet, le fait que l'utilisateur prétende être le propriétaire de l'identité ne signifie pas nécessairement que c'est vrai. Par conséquent, pour que l'utilisateur obtienne la permission ou les droits d'accéder et d'utiliser les ressources, il doit prouver son identité au système. C'est ce qu'on appelle l'authentification. En d'autres termes, l'authentification est un processus de confirmation ou de validation de l'identité d'un individu. L'information ou les informations supplémentaires (les preuves) fournies par l'utilisateur pour prouver qu'il est vraiment celui qu'il prétend être est appelé un facteur ou des facteurs d'authentification. Par exemple lorsqu'un utilisateur souhaite consulter son courrier électronique par exemple à partir de Gmail, le système lui demande de saisir son adresse mail pour s'identifier, si l'utilisateur saisit mal son adresse mail, un message d'erreur s'affiche lui indiquant que cette adresse mail n'existe pas, par contre, s'il saisit correctement son adresse mail, le système lui demandera de prouver qu'il est bien le propriétaire de cette adresse mail en lui demandant quelque chose qui ne peut être fourni que par le propriétaire réel de cette adresse mail. Dans ce cas, il s'agit du mot de passe de cette adresse électronique qui est un exemple de facteur d'authentification [3].

1.4 Les facteurs d'authentification

Les facteurs d'authentification couramment utilisés sont : something you know (quelque chose que connais l'utilisateur), something you have (quelque chose que l'utilisateur possède), et something you are (quelque chose qui caractérise l'utilisateur).

1.4.1 Le facteur « something you know »

Dans ce facteur d'authentification l'utilisateur va utiliser ce qui est stocké dans sa mémoire pour prouver qu'il est vraiment celui qu'il prétend être. Un bon exemple de cette méthode d'authentification est l'utilisation d'un mot de passe ou d'un code PIN dans un distributeur automatique de billets. Le processus d'authentification utilisant cette méthode se déroule en deux étapes, la première est la phase d'enregistrement ou l'utilisateur choisit un mot de passe de son choix. Le mot de passe est ensuite stocké dans une base de données. La deuxième étape est l'étape d'authentification durant laquelle l'utilisateur saisit son mot de passe. Le mot de passe saisi est alors comparé avec le mot de passe enregistré précédemment. S'ils correspondent, un accès au système ou aux ressources est accordé. Sinon, l'accès est refusé.

1.4.2 Le facteur « something you have »

Ce facteur d'authentification exige qu'un utilisateur possède un objet physique. Cela inclut les cartes à puce et les jetons d'authentification. Un jeton d'authentification est un petit appareil, généralement sous la forme d'un porte-clés, avec un écran qui affiche un ensemble de chiffres qui sont utilisés comme mot de passe durant le processus d'authentification. Un jeton d'authentification peut être synchrone ou asynchrone. Un jeton synchrone est synchronisé avec un serveur d'authentification. Ce jeton utilise une fonction f() qui prend l'heure actuelle t comme entrée pour calculer le mot de passe PWD_t correspondant à ce moment particulier. C'est-à-dire $PWD_t = f(t)$. Lorsqu'un utilisateur souhaite se connecter à un système, il entre un nom d'utilisateur ainsi que le numéro affiché sur le jeton. Concernant le processus d'authentification, le serveur d'authentification effectue le même processus. C'est-à-dire que le serveur d'authentification utilise également la fonction f(), qui prend l'heure actuelle t comme entrée, pour calculer le mot de passe PWD_t. Les deux mots de passe calculés sont comparés. S'ils correspondent alors, l'utilisateur est authentifié avec succès et est autorisé à entrer dans le système. Le mot de passe calculé par le jeton et le serveur d'authentification change en fonction de l'heure actuelle. Pour un jeton RSA SecurID (Figure 1), qui est un exemple de jeton d'authentification synchrone, le numéro sur le jeton change toutes les minutes.

Le jeton d'authentification asynchrone fonctionne d'une manière légèrement différente du jeton synchrone. Avec ce type de jeton, le serveur d'authentification et le jeton n'ont pas besoin d'être synchronisés. Lorsqu'un utilisateur souhaite se connecter à un système, un défi x est généré par le serveur d'authentification et est utilisé par le jeton comme entrée dans une fonction f(), généralement une fonction de hachage. Un mot de passe $PWD_x = f(x)$ est alors calculé et transmis au serveur d'authentification par le jeton. Du côté du serveur d'authentification, le processus d'authentification est effectué en calculant également le $PWD_x = f(x)$, qui est ensuite comparé au mot de passe reçu. S'ils correspondent, l'authentification est un succès, sinon le processus échoue. Le principal avantage de l'utilisation d'un jeton d'authentification, qu'il soit synchrone ou asynchrone, est que le mot de passe de l'utilisateur change à chaque fois qu'il se connecte ce qui réduit le risque que les mots de passe soient devinés par des attaquants. C'est pour cela d'ailleurs

qu'ils sont appelés « time-based one-time password ou TOTP » pour les jetons synchrones et « hash-based one-time password ou HOTP »pour les jetons asynchrones [3].

Figure 1: Jeton RSA SecurID[5]

1.4.3 Le facteur « something you are »

Le troisième facteur d'authentification consiste à utiliser des informations biométriques pour authentifier un utilisateur. La biométrie couramment utilisée à des fins d'authentification comprend les empreintes digitales, l'empreinte palmaire, la géométrie de la main, le visage, l'iris, la rétine, etc. Cependant, seules les empreintes digitales, la rétine et l'iris sont considérées comme véritablement uniques. Le processus d'authentification biométrique fonctionne en deux parties principales. La première est l'enregistrement ou l'inscription. La seconde est l'authentification (Figure 2). Lors de la phase d'enregistrement, les données biométriques sont capturées par un dispositif de détection biométrique tel qu'un lecteur d'empreintes digitales pour les données d'empreintes digitales ou une caméra pour les données faciales. Les données capturées sont ensuite traitées afin de les transformer en un modèle biométrique. Le modèle biométrique obtenu est stocké dans une base de données où un processus d'authentification peut être effectué ultérieurement. Le processus d'authentification consiste à capturer les données biométriques de l'utilisateur et les traiter puis, les comparer au modèle biométrique qui est stocké dans la base de données. S'ils correspondent, le processus d'authentification est un succès. Sinon, le processus d'authentification échoue. L'authentification biométrique pose cependant un problème, à savoir qu'il est possible que les caractéristiques humaines changent avec le temps. Cela est dû au vieillissement, à une maladie ou à une blessure. Par conséquent, des mécanismes d'authentification de secours ou de sécurité doivent être créés pour y remédier [3].

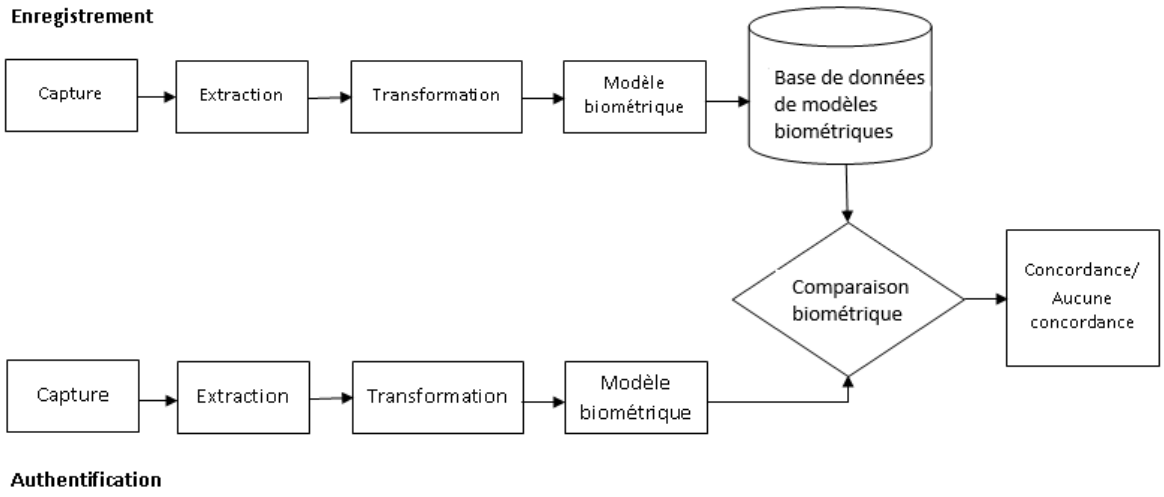


Figure 2: Processus d'authentification biométrique [3]

1.5 L'authentification multi-facteurs

Le système traditionnel d'authentification par mot de passe est vulnérable aux attaques en raison de l'utilisation de mots de passe faibles et de la réutilisation des mots de passes. L'attention s'est donc tournée vers l'authentification à deux facteurs pour fournir un niveau de sécurité supplémentaire au système traditionnel. L'authentification à deux facteurs nécessite que les deux facteurs d'authentification proviennent de deux catégories différentes. Par exemple, si un mot de passe et un code PIN sont utilisés lors du processus d'authentification, cela n'est pas considéré comme une authentification à deux facteurs car le mot de passe et le code PIN appartiennent à la même catégorie de facteurs d'authentification, à savoir la catégorie « something you know ». Si dans une autre situation, un mot de passe doit être saisi et une empreinte digitale doit également être scannée. Cela est considéré comme une authentification à deux facteurs, car le mot de passe est dans la catégorie « something you know » et l'empreinte digitale est dans la catégorie « something you are ».Un autre exemple de processus d'authentification à deux facteurs est illustré par la Figure 3. Cette figure montre que pour se connecter à un système, une personne doit entrer son nom d'utilisateur et son mot de passe comme premier facteur d'authentification. Elle reçoit alors un code secret sur son appareil mobile. Le code secret est alors entré dans le système en tant que deuxième facteur d'authentification. Si le mot de passe de l'utilisateur (qui appartient à la catégorie « something you know ») et le code secret de l'appareil mobile (qui appartient à la catégorie « something you have ») sont tous les deux corrects, l'utilisateur se verra accorder un accès au système [3].

Figure 3: Un exemple de processus d'authentification à deux facteurs [3]

L'authentification multi-facteurs ou Multi-Factor Authentication (MFA) comme son nom l'indique est un processus d'authentification qui utilise plusieurs facteurs d'authentification [3].

1.6 L'authentification unique (Single Sign-On ou SSO)

L'authentification unique est une technique d'authentification permettant à un utilisateur d'utiliser un seul ensemble d'informations d'authentification pour accéder à plusieurs services [6]. Avec le SSO, l'utilisateur n'aura pas à se connecter séparément à chaque service.

1.7 La gestion des identités et des accès (IAM : Identity and Access Management)

La gestion des identités et des accès(IAM) est l'ensemble des processus, technologies et politiques qui gèrent les identités et l'accès des identités aux ressources numériques et déterminent les autorisations des identités sur ces ressources [7]. Une solution IAM permet aux administrateurs informatiques de gérer de manière sûre et efficace les identités numériques des utilisateurs et les privilèges d'accès associés [8].

1.7.1 Composantes d'un système IAM

La Figure 4 présente les principaux services qui composent un système IAM permettant de gérer les identités et l'accès aux services informatiques au sein d'une organisation :

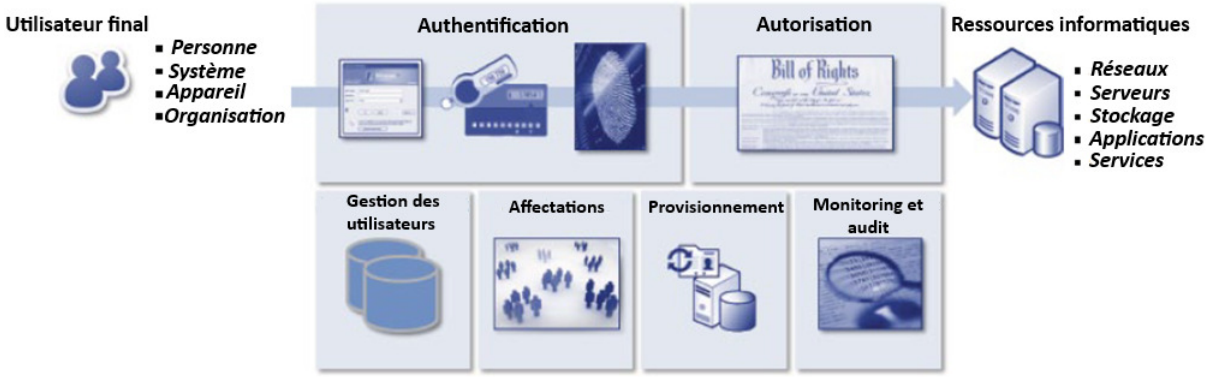


Figure 4: Architecture IAM d'une entreprise [7]

- Authentification : ce service couvre les processus et la technologie permettant de déterminer que les utilisateurs sont quoi ou ce qu'ils prétendent être.
- Autorisation : ce service couvre les processus et la technologie permettant de déterminer qu'un utilisateur dispose des autorisations appropriées pour accéder aux ressources informatiques.
- Gestion des utilisateurs : ce service couvre les activités qui gèrent efficacement le cycle de vie des identités (création, modification, désactivation).
- Affectations : ce service couvre le mappage des autorisations aux identités et aux attributs associés.
- Provisionnement: ce service consiste en la propagation des données d'identité et d'autorisation aux ressources informatiques via des processus automatisés ou manuels.
- Monitoring et audit : ce service couvre le monitoring, l'audit et le reporting de la conformité de l'accès des utilisateurs aux ressources informatiques sur la base des politiques définies.

1.8 La gestion des identités et des accès et le Cloud computing

Avant de parler de la gestion des identités et des accès dans le Cloud, il est important de commencer par donner quelques notions sur le Cloud computing.

1.8.1 Définition du Cloud computing

Le NIST (National Institute of Standards and Technology) définit le Cloud computing comme un modèle permettant un accès réseau omniprésent, pratique et à la demande à un pool partagé de ressources informatiques (par exemple, réseaux, serveurs, stockage, applications et services) configurables qui peuvent être rapidement provisionnés et libérés avec un effort minimal de gestion ou d'interactions avec le fournisseur de services. Ce modèle de Cloud est composé de cinq caractéristiques essentielles, de trois modèles de service et de quatre modèles de déploiement [9]. Amazon Web Services[1]et Microsoft Azure[2] sont des exemples de fournisseurs de services Cloud.

1.8.2 Caractéristiques essentielles du Cloud computing

Les cinq caractéristiques essentielles du Cloud computing sont le libre-service à la demande, large accès au réseau, mise en commun des ressources, élasticité rapide et le service mesuré.

1.8.2.1 Libre-service à la demande

Un consommateur peut allouer des ressources Cloud telles que le stockage, etc, selon ses besoins automatiquement sans nécessiter une interaction humaine avec chaque fournisseur de services [9].

[1]https://aws.amazon.com/fr/
[2]https://azure.microsoft.com/fr-fr/

1.8.2.2 Large accès au réseau

Les ressources du Cloud sont disponibles sur le réseau et sont accessibles via des mécanismes standards permettant leur utilisation par des plates-formes clientes hétérogènes par exemple : les téléphones mobiles, les tablettes, les ordinateurs portables et les stations de travail [9].

1.8.2.3 Mise en commun des ressources

Les ressources informatiques du fournisseur sont regroupées pour servir plusieurs consommateurs à l'aide d'un modèle multi-tenant, avec différentes ressources physiques et virtuelles affectées et réaffectées dynamiquement en fonction de la demande des consommateurs. Il existe une indépendance géographique dans la mesure où le client n'a généralement aucun contrôle ou connaissance de l'emplacement exact des ressources fournies, mais peut être en mesure de spécifier l'emplacement à un niveau d'abstraction plus élevé (par exemple, pays, état ou centre de données). Des exemples de ressources incluent le stockage, le traitement, la mémoire et la bande passante du réseau [9].

1.8.2.4 Élasticité rapide

Les ressources du Cloud peuvent être allouées et libérées de manière élastique, dans certains cas automatiquement, pour s'adapter rapidement à la demande croissante ou décroissante du consommateur. Pour ce dernier, les ressources disponibles semblent souvent illimitées et peuvent être allouées en n'importe quelle quantité et à tout moment [9] [10].

1.8.2.5 Service mesuré

Les systèmes Cloud contrôlent et optimisent automatiquement l'utilisation des ressources en tirant parti d'une capacité de mesure. Généralement, cela se fait sur une base de paiement ou de facturation à l'utilisation selon le niveau d'abstraction adapté au type de service (par exemple, stockage, traitement, bande passante et comptes d'utilisateurs actifs). Et cela d'une manière transparence à la fois au fournisseur et au consommateur du service utilisé [9].

1.8.3 Les types de services du Cloud Computing

Les trois types de services offerts par le Cloud computing selon le NIST sont : SaaS (Software as a Service), PaaS (Platform as a Service) et IaaS (Infrastructure as a Service) :

1.8.3.1 SaaS (Software as a Service)

Dans ce type de service, les capacités fournies au consommateur consistent à utiliser les applications du fournisseur de services qui s'exécutent sur une infrastructure[3] Cloud. Le consommateur ne gère pas et ne contrôle pas l'infrastructure Cloud sous-jacente incluant le réseau,

[3]L'infrastructure Cloud peut être vue comme contenant à la fois une couche physique et une couche logique. La couche physique comprend les ressources matérielles nécessaires pour prendre en charge les services Cloud fournis, il s'agit des serveurs, du stockage et des équipements réseaux. La couche logique se compose des logiciels déployés sur la couche physique. Le matériel et les logiciels sont mis en place de telle sorte à satisfaire les cinq caractéristiques essentielles du Cloud computing.

les serveurs, les systèmes d'exploitation, le stockage et les applications, à l'exception possible de la configuration limitée des paramètres des applications spécifiques à l'utilisateur [9]. Des exemples d'applications SaaS : le courrier électronique Gmail, l'application de visioconférence Zoom, etc.

1.8.3.2 PaaS (Platform as a Service)

Dans ce type de service, le consommateur a la possibilité de déployer sur l'infrastructure Cloud des applications créées ou acquises par le consommateur et développées à l'aide de langages de programmation, de bibliothèques, de services et d'outils pris en charge par le fournisseur de services Cloud. Le consommateur ne gère pas et ne contrôle pas l'infrastructure Cloud sous-jacente incluant le réseau, les serveurs, les systèmes d'exploitation ou le stockage, mais contrôle les applications déployées et éventuellement les paramètres de configuration de l'environnement d'hébergement de ces applications [9].

1.8.3.3 IaaS (Infrastructure as a Service)

La capacité fournie au consommateur consiste à allouer les capacités de traitement, le stockage, les réseaux et d'autres ressources informatiques fondamentales sur lesquelles le consommateur va déployer et exécuter des logiciels qui peuvent inclure des systèmes d'exploitation et des applications. Le consommateur ne gère pas et ne contrôle pas l'infrastructure Cloud sous-jacente, mais contrôle les systèmes d'exploitation, le stockage et les applications déployées et possède éventuellement un contrôle limité sur certains composants réseau (par exemple, les pare-feu hôtes) [9].

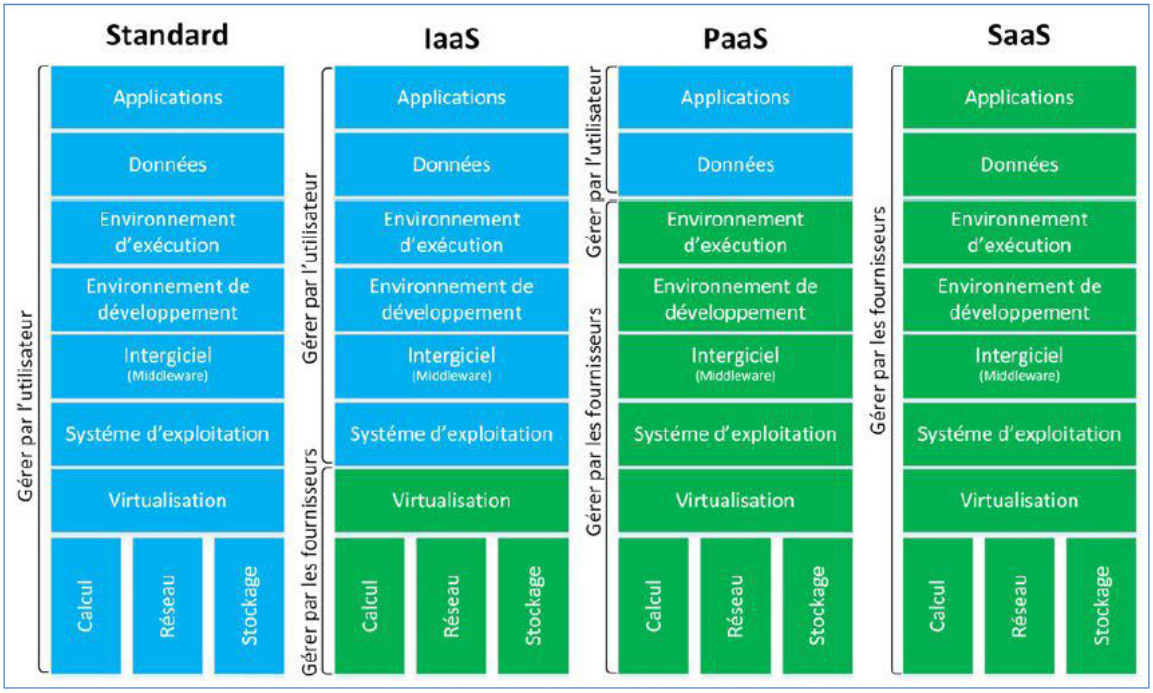


Figure 5: Types de services du Cloud computing [10]

.

1.8.4 Modèles de déploiement du Cloud computing

Les quatre modèles de déploiement du Cloud computing selon le NIST sont : le Cloud privé, Cloud communautaire, le Cloud public et le Cloud hybride :

1.8.4.1 Cloud privé

Le Cloud est utilisé exclusivement par une seule organisation comprenant de multiples consommateurs. Il peut appartenir et être géré par l'organisation, un tiers ou une combinaison des deux. Il peut être mis en place à l'intérieur ou à l'extérieur de l'organisation.

1.8.4.2 Cloud communautaire

L'infrastructure Cloud est utilisée exclusivement par une communauté spécifique de consommateurs appartenant à des organisations qui ont des choses en commun (par exemple, la mission, les exigences de sécurité, etc). Il peut être détenu et géré par une ou plusieurs organisations de la communauté, un tiers ou une combinaison de ces derniers. Il peut être mis en place sur site (en interne) ou hors site.

1.8.4.3 Cloud public

Le Cloud est utilisé par le grand public. Il peut être détenu et géré par une entreprise, une université, une organisation gouvernementale ou une combinaison de ces dernières. Il est situé dans les locaux du fournisseur du Cloud.

1.8.4.4 Cloud hybride

L'infrastructure Cloud est une composition de deux ou plusieurs infrastructures Cloud distinctes (privées, communautaires ou publiques) liées par une technologie standardisée ou propriétaire qui permet la portabilité des données et des applications [9].

1.8.5 Architectures IAM pour le Cloud

Les entreprises disposent aujourd'hui d'applications et de données sur site ainsi que dans des Cloud. Le défi consiste à gérer l'accès des utilisateurs aux ressources où qu'elles se trouvent, de la manière la plus transparente possible [11].

Dans un environnement informatique traditionnel, les utilisateurs doivent être ajoutés, modifiés ou supprimés d'un système et également se voir attribuer certaines autorisations afin d'accéder aux ressources numériques. Les processus généraux de gestion des identités ou des accès restent les mêmes dans un environnement de Cloud computing. Trois modèles peuvent être distingués pour gérer les identités et les accès dans un environnement de Cloud computing : la gestion locale des services IAM, l'utilisation d'un fournisseur de services d'identité (identity service provider / IdSP) et l'utilisation d'IAM-as-a-service (IAMaaS) [7].

1.8.5.1 Gestion IAM locale

Dans le modèle de gestion IAM local illustré par la Figure 5, les utilisateurs peuvent accéder aux ressources internes de leur organisation, telles que les données et les applications en s'authentifiant

localement. Grace à des outils adaptés, des comptes utilisateurs sont créés automatiquement aux utilisateurs autorisés au niveau du CSP pour leur permettre d'accéder aux services Cloud. En outre, les services d'authentification unique (SSO) peuvent être utilisés pour réduire le nombre d'identités qu'un utilisateur doit posséder pour accéder aux ressources sur site et aux services Cloud. L'authentification unique est basée sur la mise en œuvre d'une identité fédérée entre plusieurs domaines. L'avantage de ce modèle est qu'il offre le meilleur contrôle sur les services IAM. Cependant, ce modèle nécessite une maintenance locale coûteuse pour se connecter aux CSPs. Ce qui peut rendre ce modèle inadapté pour les organisations qui font appel à plusieurs fournisseurs de services Cloud [7].

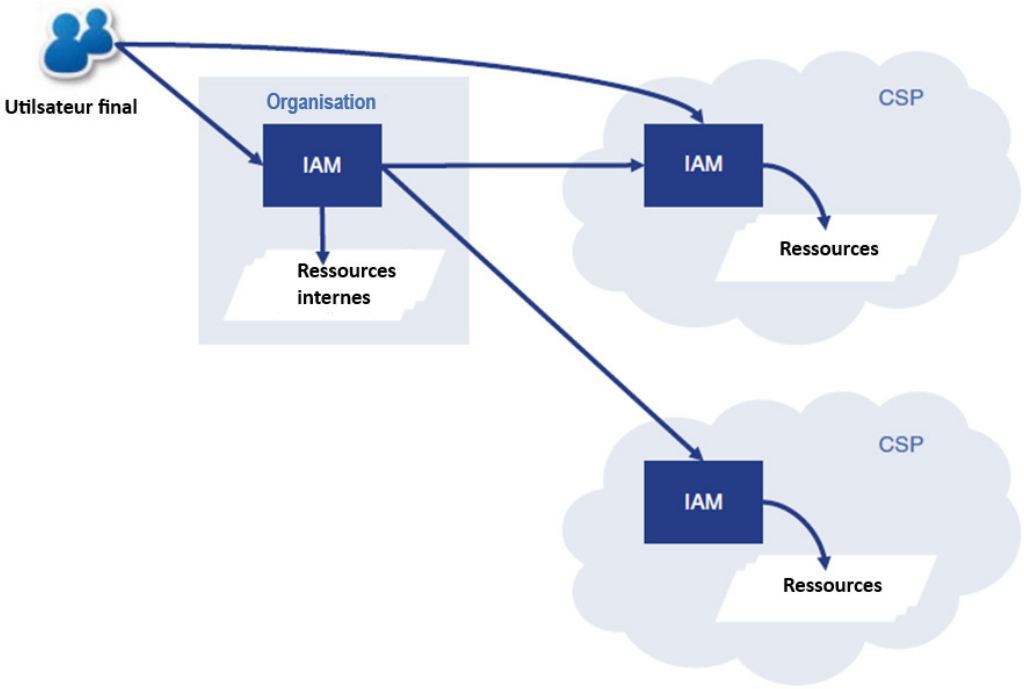


Figure 6: Modèle de gestion IAM local [7]

1.8.5.2 Utilisation d'un fournisseur de services d'identité

Le modèle IdSP, illustré par la Figure 6 permet l'utilisation d'un fournisseur d'identité tiers comme Google, DigiD, Facebook, etc, pour fournir aux utilisateurs des moyens de s'authentifier (de prouver leur identité) afin d'accéder aux ressources locales ou Cloud. L'authentification devient la responsabilité de l'IdSP ce qui est un avantage. Un autre avantage est que les utilisateurs n'ont pas à maintenir plusieurs identités pour accéder à leurs ressources autorisées. Les solutions IAM existantes des organisations peuvent interagir avec l'IdSP en utilisant des normes de fédération telles que SAML et OAuth. L'une des principales inconvénient préoccupations de ce modèle est que le fait que l'authentification soit contrôlée par un tiers, cela réduit le contrôle de l'organisation ou de l'utilisateur sur la force et les moyens d'authentification[7].

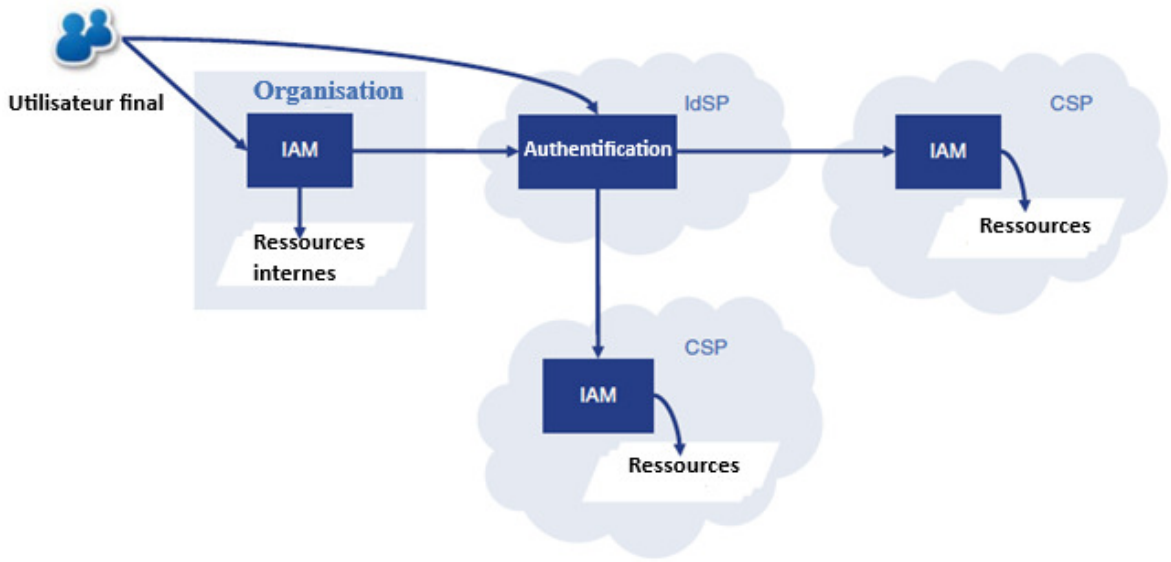


Figure 7: Modèle IAM avec fournisseur de services d'identité [7]

1.8.5.3 IAM en tant que service (IAMaaS) ou Identity as a Service (IDaaS)

Le dernier modèle (voir Figure 7) consiste à déléguer les processus IAM à un CSP et à l'utiliser en tant que service (IAMaaS ou IDaaS). L'accès à tous les autres fournisseurs de services Cloud sera géré et contrôlé via cet IAMaaS. L'accès aux ressources internes se fera également via ce service IAM. Le CSP peut être le point unique pour s'authentifier, cela donne aux utilisateurs la possibilité d'utiliser le SSO. Cette approche entraînerait le moins de responsabilités pour les organisations et les utilisateurs individuels en effet, en cédant la tâche de la mise en place d'une solution de gestion des identités et des accès à un tiers spécialisé [7]. L'organisation n'aura plus à se soucier de gérer l'infrastructure, d'assurer la sécurité, d'installer et mettre à niveau les logiciels, sauvegarder les données, etc, c'est l'IDaaS qui se chargera de toutes ces tâches [12].Cependant, cela signifie également la plus grande perte de contrôle sur les services IAM, car la visibilité diminue considérablement avec l'externalisation vers le Cloud. Il existe différents fournisseurs sur le marché proposant des solutions IAMaaS, par exemple SailPoint[4], Ping Idenity[5], etc.

[4]https://www.sailpoint.com/fr/
[5]https://www.pingidentity.com/en.html

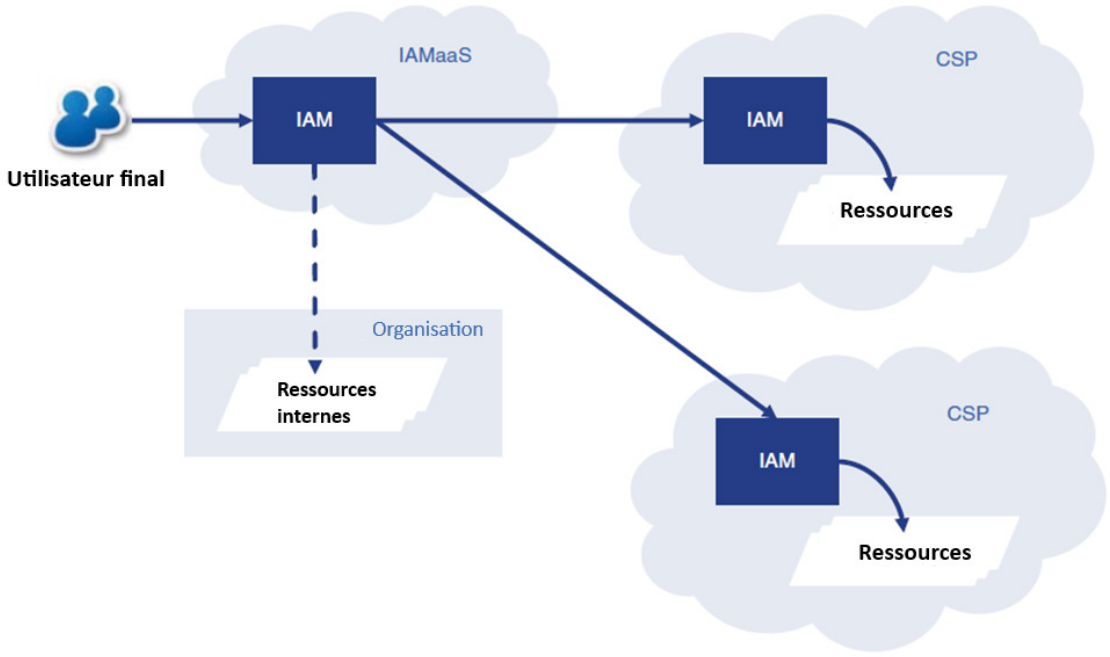


Figure 8: Modèle IAM en tant que service (IAMaaS) [7]

1.8.6 Standards liés à l'IAM

Un ou plusieurs des standards et protocoles suivants sont souvent utilisés dans la gestion des identités et des accès. Une description détaillée de certains de ces protocoles sera donnée au deuxième chapitre :

- **OAuth** : OAuth est un standard ouvert qui permet aux utilisateurs d'autoriser des applications ou des sites web à accéder à leurs données sans avoir à leurs fournir leurs informations d'identification (par exemple, nom d'utilisateur et mot de passe).
- **OpenID Connect :** Open ID Connect est un standard ouvert basé sur le protocole OAuth 2.0 qui permet aux utilisateurs de s'authentifier à un IdSP pour accéder aux sites prenant en charge cet IdSP. Cela permet aux utilisateurs de s'authentifier sur un site différent de celui à partir duquel ils demandent l'accès à une ressource numérique. Cela élimine le besoin pour les sites de créer et de maintenir leurs propres mécanismes d'authentification.
- **SAML :** Security Assertion Markup Language (SAML) est un protocole basé sur XML qui spécifie les règles d'échange de données d'authentification et d'autorisation entre les parties. Les données sont échangées sous forme d'assertions qui contiennent des déclarations que les fournisseurs de services utilisent pour prendre des décisions de contrôle d'accès.
- **SPML :** Service Provisioning Markup Language est un framework basé sur XML qui permet aux organisations de provisionner et de gérer des comptes d'utilisateurs [7].
- **XACML** : XACML (eXtensible Access Control Markup Language) est un langage de politique de contrôle d'accès à usage général. Cela signifie qu'il fournit une syntaxe (définie en XML) pour gérer l'accès aux ressources [13].

1.9 Conclusion

Dans ce chapitre nous avons donné quelques définitions de certains concepts comme l'identité, l'authentification, le Cloud computing et la gestion des identités et des accès. Ces notions sont importantes pour la compréhension des chapitres qui vont suivre. Dans le chapitre suivant, nous allons étudier un peu plus en détail l'authentification unique (le Single Sign-on), plus particulièrement certains protocoles d'implémentation du SSO à savoir Kerberos, SAML et OpenID Connect. Nous allons également présenter un ensemble de travaux de recherche sur l'implémentation du SSO.

Chapitre 2 L'authentification unique (Single Sign-On ou SSO)

2.1 Introduction

De nos jours, de nombreux sites et services web nécessitent de s'authentifier pour accéder à leurs fonctionnalités et à leur contenu. Chaque système dispose de son propre mécanisme d'authentification ce qui conduit à une redondance d'information. De plus, l'utilisateur doit se connecter à chaque application séparément et avec un nouveau compte pour chaque nouvelle application. Ce qui constitue une contrainte pour l'utilisateur. Ce dernier doit également mémoriser de nombreux mots de passe, ce qui peut causer le problème de fatigue de mot de passe qui à son tour peut conduire les utilisateurs à opter pour des mots de passe simples et presque identiques. Cette approche est simple mais présente une menace potentielle. En effet, un attaquant peut deviner le mot de passe et accéder à toutes les informations confidentielles. Tous ces problèmes ont conduit au développement d'un système plus efficace assurant la sécurité et l'accessibilité. Ce système est l'authentification unique (Le Single Sign-On ou SSO). Avec l'introduction du SSO, les utilisateurs n'ont qu'à s'authentifier une seule fois et peuvent ensuite accéder facilement et en toute sécurité aux multiples applications exécutées sur différents domaines [14][15]. Dans ce chapitre, nous allons aborder le mécanisme du SSO, son principe de fonctionnement, ces avantages et inconvénients. Nous allons également présenter différents protocoles d'implémentation du SSO qui sont Kerberos, SAML et OpenID Connect. Nous terminerons par la présentation d'un ensemble de travaux sur l'implémentation du SSO qui sera suivi par une discussion et une conclusion.

2.2 Objectifs et principe de fonctionnement de l'authentification unique

Le système d'authentification unique ou bien Single Sign-On (SSO) en anglais est développé dans le but de réduire la contrainte des utilisateurs à se souvenir de différents identifiants et mots de passe pour accéder à différents sites/applications web et mobiles. Il réduit également la contrainte de se connecter séparément à chaque application [15].

Dans un système SSO (Figure 9), un utilisateur s'authentifie une seule fois et peut ensuite accéder à différentes applications ou services. Ces applications peuvent être au sein d'une seule organisation (du même domaine) ou de différentes organisations (réparties dans plusieurs domaines différents). On parle de fédération lorsque le SSO permet l'accès à des applications dans différentes organisations [14].

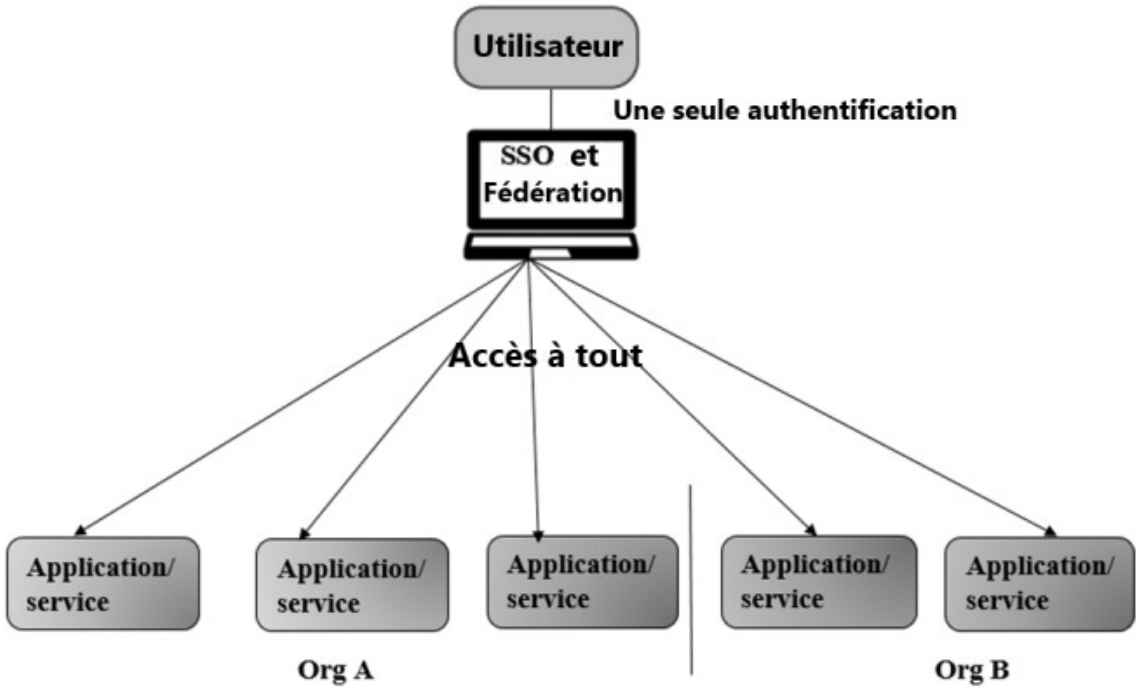


Figure 9: Présentation du SSO [14]

Un exemple bien connu du SSO est une fois connecté à son compte Gmail sur un appareil, l'utilisateur pourra accéder à différents services de Google (youtube, meet, etc) sans nécessiter de s'authentifier à nouveau. Un autre exemple est celui des boutons **« se connecter avec Facebook », « se connecter avec Twitter »**, etc, (Figure 10) qui se trouve sur de nombreux sites et applications permettant aux utilisateurs une fois connecté par exemple à leur compte Facebook ou Twitter d'accéder à tous ces sites [16]. Ainsi l'utilisateur n'aura plus besoin de créer un compte sur chacun de ces sites et de se connecter à chaque site séparément. Cela est possible lorsque ces sites ou applications utilisent Google, Facebook ou autre comme fournisseur d'identité et l'implémente à l'aide de protocoles tel que SAML ou OpenID Connect.

Figure 10: SSO avec Google, Facebook et Twitter comme fournisseur d'identité [17]

2.3 Classification des systèmes SSO

La Figure 11 illustre une classification de l'authentification unique selon les auteurs de [18], ces derniers ont classé les systèmes SSO en quatre catégories, selon l'endroit où ils sont déployés (Intranet, Extranet, Internet) ; comment ils sont déployés (architecture simple, architecture complexe) ; les informations d'identification qu'ils utilisent (token, certificat) et les protocoles qu'ils utilisent (Kerberos, SAML, OpenID, etc).

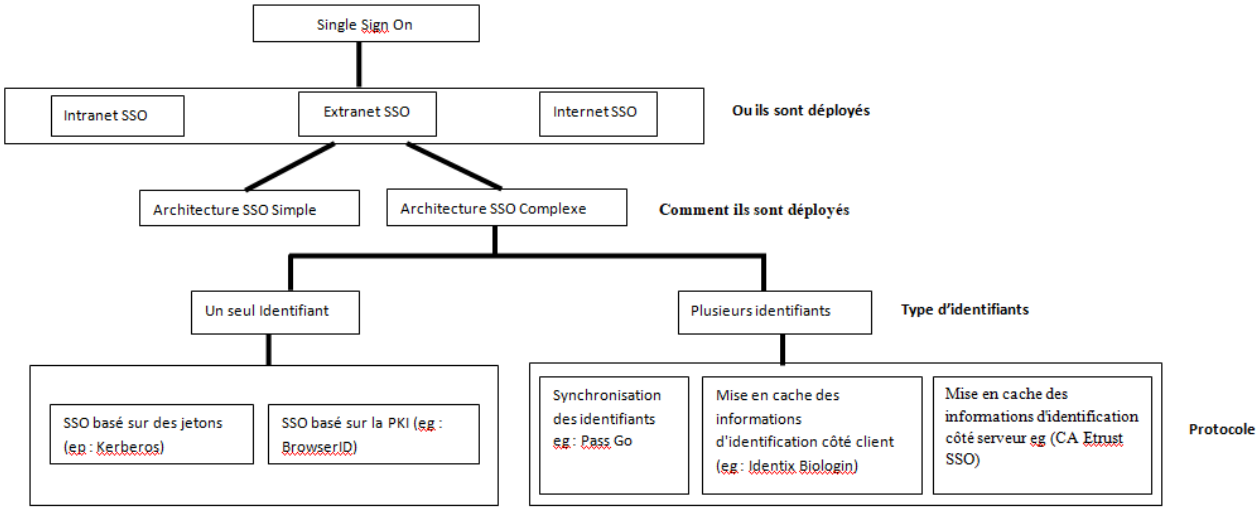


Figure 11: classification de l'authentification unique [18]

2.3.1 Architecture SSO simple

Ce type d'architecture utilise une seule autorité d'authentification[6] et est mis en œuvre en intranet. Les systèmes SSO mis en œuvre en intranet appelées aussi SSO d'entreprise (Entreprise SSO ou ESSO en anglais) permettent de se connecter une seule fois pour accéder à plusieurs systèmes au sein de la même entreprise [19][18].

2.3.2 Architecture SSO complexe

Ce type d'architecture utilise plusieurs autorités d'authentification implémentées sur différentes plates-formes et régies par différentes organisations. Cette architecture peut être mise en œuvre sur Internet ou Extranet. Les systèmes SSO mis en œuvre en extranet aussi appelées SSO multi-domaines permettent de se connecter une seule fois avec les mêmes identifiants à plusieurs systèmes au sein de la même entreprise (ou la même organisation) et à toutes les applications des partenaires commerciaux (ou d'autres organisations). Le web SSO ou bien les systèmes SSO mis en

[6] Les autorités d'authentification sont des entités auxquelles les utilisateurs font confiance pour s'authentifier afin d'accéder aux ressources faisant partie de son domaine. Lorsqu'un utilisateur se connecte avec succès au domaine de l'autorité d'authentification, il peut accéder de manière transparente à toutes les ressources du domaine sans s'authentifier à nouveau auprès de chaque ressource séparément.

place à travers l'internet permettent d'accéder avec une connexion unique aux applications déployées sur internet [19][18].

2.4 Protocoles d'implémentation du SSO

Dans cette section, nous aborderons différents protocoles utilisés sur des architectures SSO simples et complexes. Ces protocoles sont des standards permettant de les implémenter sur différentes plateformes sans causer de problèmes d'interopérabilité et il est également possible de les utiliser pour la mise en œuvre du SSO inter domaines.

2.4.1 Kerberos

Kerberos est un protocole d'authentification réseau, conçu pour fournir une authentification forte pour les applications client/serveur en utilisant la cryptographie à clé secrète [20] qui permet à l'utilisateur et au service de s'authentifier l'un à l'autre et de chiffrer leurs communications. Cette authentification mutuelle et le cryptage des communications permettent de maintenir la confidentialité et l'intégrité des données pour l'utilisateur et le service [21].

Ci-dessous, les définitions de certains termes utilisés dans ce protocole.

- **Principal :** un principal est associé à un utilisateur ou un service dans la base de données et permet de les identifier d'une façon unique.
- **Key Distribution Center (KDC) :** c'est le cœur de ce protocole, il est constitué de trois parties : le serveur d'authentification (Authentication Server ou AS), le serveur d'octroi de tickets (Ticket Granting Server ou TGS) et la base de données.
- **Authentication Server (AS) :** Le serveur d'authentification est la partie du KDC qui répond à la demande d'authentification initiale du client. En réponse à sa demande, l'AS émet un ticket spécial appelé Ticket Granting Ticket, ou plus brièvement TGT. Si l'utilisateur est bien celui qu'il prétend être, il peut utiliser le TGT pour obtenir un ticket de service et le réutiliser pour obtenir d'autres tickets de service, sans avoir à ressaisir son mot de passe (c'est le principe du SSO).
- **Ticket Granting Server (TGS) :** Le serveur d'octroi de tickets (TGS) est le composant KDC qui distribue aux clients présentant un TGT valide les tickets de service leur permettant d'accéder aux serveurs d'application.
- **La base de données :** contient les entrées associées aux utilisateurs et aux services, chaque entrée contient plusieurs informations dont le principal et la clé secrète associée à ce dernier.

La figure ci-dessous illustre l'ensemble des messages (les requêtes et les réponses aux requêtes) échangés entre le client et le KDC et entre le client et le serveur d'applications pour assurer l'authentification du client et du service et l'accès du client aux services demandés.

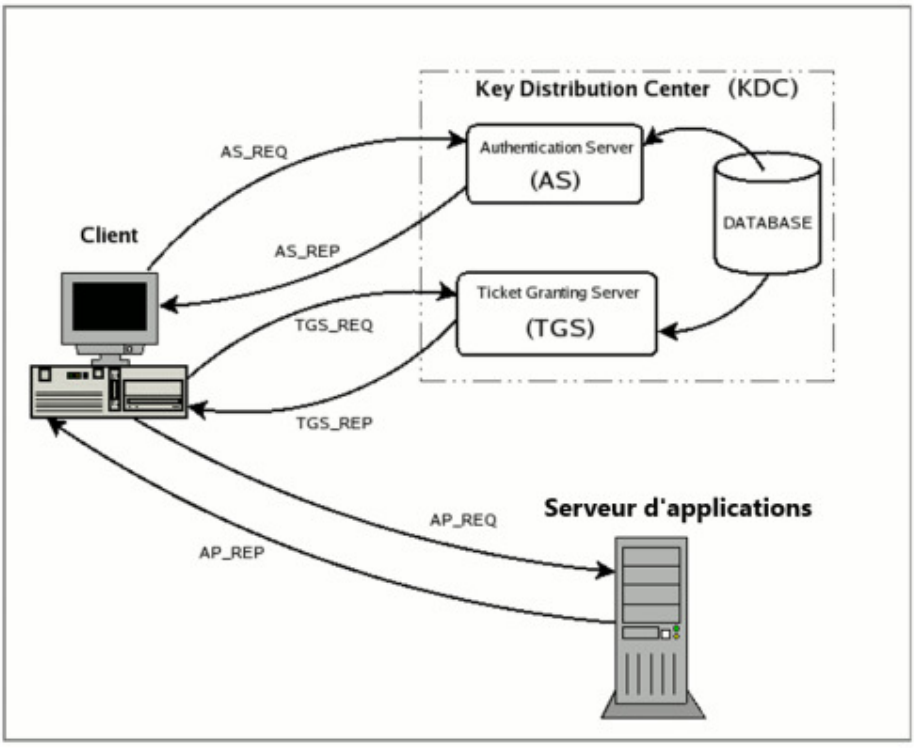


Figure 12: messages échangés entre le client, le KDC et le serveur d'applications lors de l'authentification

Ci-dessous, l'explication de ces messages sachant qu'une explication plus détaillée est donnée en Annexe 1) :

- **AS_REQ** est la requête d'authentification initiale de l'utilisateur auprès du serveur d'authentification (AS) ;
- **AS_REP** est la réponse du serveur d'authentification à la requête précédente. Elle contient en particulier le TGT (chiffré à l'aide de la clé secrète du TGS) et la clé de session (chiffrée à l'aide de la clé secrète de l'utilisateur) ;
- **TGS_REQ** est la requête du client auprès du TGS (Ticket Granting Server) pour obtenir un ticket de service. Ce message comprend le TGT obtenu à l'étape précédente et un authentifiant généré par le client et chiffré avec la clé de session ;
- **TGS_REP** est la réponse du TGS (serveur d'octroi de tickets) à la requête précédente. A l'intérieur se trouvent le ticket de service demandé (chiffré avec la clé secrète du service) et une clé de session de service générée par le TGS et chiffrée à l'aide de la clé de session précédente générée par l'AS ;
- **AP_REQ** est la requête que le client envoie au serveur d'application pour accéder à un service. Elle contient le ticket de service généré par TGS lors de la réponse précédente et un authentificateur à nouveau généré par le client, mais cette fois chiffré à l'aide de la clé de session du service (générée par le TGS) ;

- **AP_REP** est la réponse que le serveur d'application donne au client pour prouver qu'il est bien le serveur attendu par le client. Ce paquet n'est pas toujours demandé. Le client n'en fait la demande au serveur que lorsqu'une authentification mutuelle est nécessaire [22].

2.4.2 SAML (Security Assertion Markup Language)

Le protocole SAML (Security Assertion Markup Language) développé par l'organisme de normalisation OASIS (the Organization for the Advancement of Structured Information Standards) est un standard ouvert basé sur XML pour l'échange d'informations d'authentification et d'autorisation entre des domaines de sécurité, c'est-à-dire entre un fournisseur d'identité et un fournisseur de services [18]. Les informations échangées sont exprimées sous la forme d'assertions XML. Une assertion est généralement transportée entre les entités dans un message, qui lui-même est transmis à l'aide de protocoles de bas niveau (exp : HTTP ou SOAP). La norme OASIS SAML a défini une syntaxe et des règles précises pour demander, créer, transporter et utiliser ces assertions [23].

2.4.2.1 Les acteurs impliqués dans le protocole SAML

Le protocole SAML permet de satisfaire plusieurs cas d'utilisation appelés « Profils » le plus important est le profil « web browser SSO » permettant la mise en place de l'authentification unique Web multi-domaines ou souvent simplement SSO. Les acteurs impliqués dans ce profil sont :

Le fournisseur de service (service provider (SP)) : est l'entité fournissant le service, généralement sous la forme d'une application.

Le fournisseur d'identité (identity provider (IdP)) : Un fournisseur d'identité (IdP) est l'entité capable d'authentifier un utilisateur. Il contient généralement des informations supplémentaires sur l'utilisateur telles que le prénom, le nom, le numéro de téléphone, l'adresse, etc. Selon l'application, certains fournisseurs de services peuvent exiger un profil très simple (nom d'utilisateur, e-mail), tandis que d'autres peuvent exiger un ensemble plus riche de données utilisateur (adresse, emplacement, etc.) [24].

Un principal (un sujet) : une entité qui peut être authentifiée par le fournisseur d'identité dans le contexte d'un domaine de sécurité particulier afin d'accéder un service.

2.4.2.2 Concepts de base du protocole SAML

Les assertions : le protocole SAML permet à une entité (généralement le fournisseur d'identité) d'affirmer des informations de sécurité sur un sujet (principal) sous la forme de déclarations (statements). Ces déclarations peuvent être de trois types : déclarations d'authentification, d'attribut ou d'autorisation. Une assertion peut contenir les trois types de déclarations. La structure et le contenu valides d'une assertion sont définis par son schéma XML décrit par le protocole SAML.

- Les déclarations d'authentification (Authentication statements) : elles sont créées par l'entité qui a authentifié le sujet, elles décrivent (au minimum) la méthode par laquelle l'utilisateur a été authentifié et la date et l'heure ou l'authentification a eu lieu.

- Les déclarations d'attribut (Attribute statements) : Ceux-ci contiennent des attributs spécifiques au sujet.
- Les déclarations d'autorisation (Authorization decision statements) : elles définissent les autorisations accordées au sujet.

La figure (Figure 13) montre un exemple d'une assertion contenant une seule déclaration qui est une déclaration d'authentification. (Le texte XML dans la Figure 13 a été formaté à des fins de présentation).

```
 1: <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 2:   Version="2.0"
 3:   IssueInstant="2005-01-31T12:00:00Z">
 4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
 5:     http://idp.example.org
 6:   </saml:Issuer>
 7:   <saml:Subject>
 8:     <saml:NameID
 9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:         j.doe@example.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>
```

Figure 13: exemple d'une assertion SAML

Une assertion est définie par l'élément <saml:Assertion> à l'intérieur duquel plusieurs informations sont fournies à savoir la version du protocole SAML utilisée, la date et l'heure de sa création ainsi que l'entité qui l'a émise (http://idp.example.org dans cet exemple). Une assertion est associée à un sujet définit par l'élément <saml:Subject>.

L'élément <saml :NameID> à l'intérieur de l'élément <saml:Subject>définit le format et la valeur de l'identifiant associé à ce sujet. Dans cet exemple, le sujet est un utilisateur, ayant comme identifiant une adresse mail (j.doe@example.com). Des conditions d'utilisation de l'assertion (exp : la période de validité de l'assertion) sont définies à l'aide de l'élément <saml :Conditions >.Des conditions supplémentaires sur l'utilisation de l'assertion peuvent être ajoutées à l'intérieur de cet élément. Il existe des conditions prédéfinies dans SAML mais il est également possible de définir ses propres conditions. La déclaration d'authentification définie par l'élément <saml :AuthnStatement> montre que le sujet a été authentifié à l'aide du mécanisme « PasswordProtectedTransport » (par exemple en fournissant un nom d'utilisateur et un mot de passe à travers une connexion chiffrée (SSL/TLS)) à l'heure et à la date indiquées.

Les messages du protocole : les messages du protocole SAML sont utilisés pour créer des requêtes et les réponses associées. La structure et le contenu de ces messages sont définis par les schémas

XML décrits par ce protocole. Le profil « Web Browser SSO » utilise principalement deux messages SAML ; à savoir un message de demande d'authentification (Authentication Request message) envoyé d'un SP à un IdP, et un message de réponse (Response message) contenant une assertion SAML envoyée par l'IdP au SP.

Les bindings : les bindings définissent comment les protocoles de niveau inférieur (tels que HTTP ou SOAP) sont utilisés pour transporter les messages du protocole SAML entre les entités ayant établi entre elles une relation de confiance. Les bindings qui peuvent être utilisé par le profil « The Web Browser SSO » sont HTTP Redirect, HTTP POST et HTTP Artifact.

Les profils : les profils définissent la façon dont les messages et les bindings sont combinés pour satisfaire des cas d'utilisation particuliers. Le protocole SAML a défini plusieurs profils dont le profil « Web Browser SSO » permettant la mise en place de l'authentification unique multi domaines ou plus souvent juste l'authentification unique.

2.4.2.3 Fonctionnement du profil « Web Browser SSO »

Le profil « Web Browser SSO » définit comment utiliser les messages du protocole SAML et les bindings pour mettre en place l'authentification unique (web SSO). Plusieurs variantes de ce profil existent et sont liées à deux dimensions de choix, premièrement selon que l'authentification unique est initiée par le fournisseur de service (SP) ou par le fournisseur d'identité (IdP) et deuxièmement, selon les bindings utilisés pour transporter les messages entre le SP et le IdP.

- Choix de l'initiateur de l'authentification unique : Il y a deux approches par lesquelles l'utilisateur peut initier l'authentification unique. La première est appelée SP-Initiated, dans cette approche l'utilisateur commence par visiter le fournisseur de service (SP) en cliquant par exemple sur un lien qui l'amène directement à celui-ci, l'utilisateur n'étant pas authentifié auprès du SP, ce dernier dirige l'utilisateur vers un IdP pour s'authentifier. L'IdP construit une assertion représentant l'authentification de l'utilisateur auprès de l'IdP, puis renvoie l'utilisateur vers le SP avec l'assertion. Le SP traite l'assertion et détermine s'il convient d'accorder à l'utilisateur l'accès à la ressource. La deuxième approche est appelée IdP-Initiated, dans cette approche, l'utilisateur commence par s'authentifier auprès d'un IdP puis, clique sur un lien le dirigeant vers un SP partenaire. L'IdP construit une assertion représentant l'état d'authentification de l'utilisateur au niveau de l'IdP et dirige le navigateur de l'utilisateur vers le service consommateur d'assertions du SP qui traite l'assertion et crée un contexte de sécurité local (une session de connexion) pour l'utilisateur au niveau du SP.

- Choix des bindings à utiliser pour transporter les messages entre le SP et le IdP : le profil Web SSO utilise principalement deux messages SAML ; à savoir un message de demande d'authentification (Authentication Request message) envoyé d'un SP à un IdP, et un message de réponse (Response message) contenant une assertion SAML qui est envoyée de l'IdP au SP (et secondairement des messages liés à la résolution d'artefact si ce binding est choisi). Le protocole SAML a défini les bindings qui peuvent être utilisés avec ces deux messages, plus précisément, un message de demande d'authentification peut être envoyé d'un SP à un IdP à l'aide des bindings HTTP Redirect, HTTP POST et HTTP Artifact. Le message de réponse peut être envoyé d'un IdP à un SP à l'aide des bindings HTTP POST ou HTTP Artifact. Le message de

demande d'authentification et le message de réponse peuvent utiliser des bindings différents d'où plusieurs combinaisons peuvent être construites dont les trois scénarios ci-dessous :

- Authentification unique SP-initiated utilisant le Binding HTTP Redirect pour l'envoi du message de demande d'authentification <AuthnRequest> du SP au IdP et le Binding HTTP POST pour l'envoi du message de réponse <Response> de l'IdP au SP.
- Authentification unique SP-initiated utilisant le Binding HTTP POST pour l'envoi du message de demande d'authentification <AuthnRequest> du SP au IdP et le Binding HTTP Artifact pour l'envoi du message de réponse <Response> de l'IdP au SP
- Authentification unique IDP-initiated utilisant le Binding HTTP POST pour l'envoi du message de réponse <Response> de l'IdP au SP. Aucun message de demande d'authentification du SP au IdP n'est requis.

Dans ce qui suit, est expliqué le fonctionnement du premier scenario. Dans ce scénario, l'utilisateur tente d'accéder à une ressource au niveau du fournisseur de service (sp.example.com). Cependant, l'utilisateur n'étant pas authentifié, le SP dirige le navigateur de l'utilisateur vers le fournisseur d'identité (idp.example.org) pour que l'utilisateur s'authentifie. L'IdP fournit au SP une assertion concernant l'utilisateur authentifié. Pour ce cas d'utilisation spécifique, le binding HTTP Redirect est utilisée pour transmettre le message SAML <AuthnRequest> à l'IdP et le binding HTTP POST est utilisée pour renvoyer le message de réponse <Response> contenant l'assertion au SP. La figure (Figure 14) illustre le flux des messages échangés dans ce scénario.

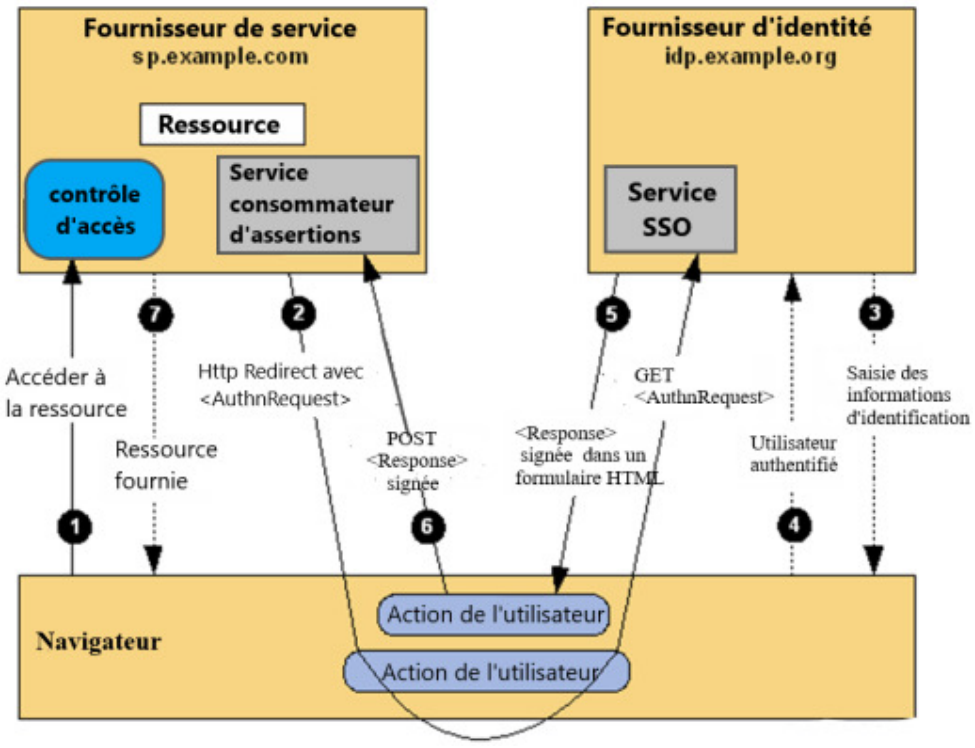


Figure 14: SSO SP-initiated/ HTTP Redirect et HTTP POST

Ci-dessous, l'explication des différentes étapes de ce schéma :

1. L'utilisateur tente à travers le navigateur d'accéder à une ressource au niveau du fournisseur de service (sp.example.com) mais n'étant pas authentifié sur ce site, le SP enregistre l'URL de la ressource demandée dans les informations d'état locales, elle sera utilisée dans les échanges SSO ultérieures.

2. Le SP envoie une réponse de redirection HTTP au navigateur contenant l'URI du service d'authentification unique au niveau de l'IdP ainsi que le message de demande d'authentification <AuthnRequest> et les informations d'état local. Ces deux derniers sont codés sous forme de variables d'URL nommées SAMLRequest et RelayState :

```
<samlp:AuthnRequest
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_1"
Version="2.0"
IssueInstant="2004-12-05T09:21:59Z"
AssertionConsumerServiceIndex="1">
<saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
<samlp:NameIDPolicy
AllowCreate="true"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

Figure 15: message de demande d'authentification <AuthnRequest>

Le navigateur traite la réponse de redirection et envoie une requête HTTP GET au service d'authentification unique de l'IdP avec les paramètres d'URL SAMLRequest et RelayState.

```
https://idp.example.org/SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token
```

3. Le service d'authentification unique au niveau de l'IdP vérifie si l'utilisateur dispose d'une session de connexion valide qui répond aux exigences de la requête d'authentification<AuthnRequest>. Si ce n'est pas le cas, l'IdP demande à l'utilisateur de fournir des informations d'identification valides à travers le navigateur.
4. L'utilisateur fournit des informations d'identification valides et une session de connexion est créé pour l'utilisateur au niveau de l'IdP
5. Le service SSO de l'IdP crée une assertion SAML représentant le contexte de sécurité de connexion de l'utilisateur. L'assertion est signée puis placée dans un message SAML <Response>:

```
<samlp:Response
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_2"
InResponseTo="identifier_1"
Version="2.0"
IssueInstant="2004-12-05T09:22:05Z"
Destination="https://sp.example.com/SAML2/SSO/POST">
<saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
<samlp:Status>
<samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_3"
Version="2.0"
IssueInstant="2004-12-05T09:22:05Z">
<saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
<!-- a POSTed assertion MUST be signed -->
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
<saml:Subject>
<saml:NameID
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
3f7b3dcf-1674-4ecd-92c8-1544f346baf8
</saml:NameID>
<saml:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData
InResponseTo="identifier_1"
Recipient="https://sp.example.com/SAML2/SSO/POST"
NotOnOrAfter="2004-12-05T09:27:05Z"/>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Conditions
NotBefore="2004-12-05T09:17:05Z"
NotOnOrAfter="2004-12-05T09:27:05Z">
<saml:AudienceRestriction>
<saml:Audience>https://sp.example.com/SAML2</saml:Audience>
</saml:AudienceRestriction>
</saml:Conditions>
<saml:AuthnStatement
AuthnInstant="2004-12-05T09:22:00Z"
SessionIndex="identifier_3">
<saml:AuthnContext>
<saml:AuthnContextClassRef>
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
</saml:AuthnContextClassRef>
</saml:AuthnContext>
</saml:AuthnStatement>
</saml:Assertion>
</samlp:Response>
```

Figure 16: message de réponse <Response> à la demande d'authentification

Le message <Response> est ensuite codé avec un codage base64 puis placé dans un formulaire HTML en tant qu'élément caché du formulaire nommé SAMLResponse. Si l'IdP a reçu une valeur RelayState du SP, il doit la renvoyer non modifiée au SP dans un autre élément caché du formulaire nommé RelayState. Le service d'authentification unique renvoie le formulaire HTML au navigateur dans la réponse HTTP.

```
<form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>
<input type="hidden" name="SAMLResponse" value="response" />
<input type="hidden" name="RelayState" value="token" />
...
<input type="submit" value="Submit" />
</form>
```

Figure 17: formulaire HTML contenant le message <Response> codé

6. Le navigateur émet une requête HTTP POST pour envoyer le formulaire au service consommateur d'assertions au niveau du SP où les valeurs des paramètres SAMLResponse et RelayState sont extraites du formulaire HTML.

```
POST /SAML2/SSO/POST HTTP/1.1
Host: sp.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnn
SAMLResponse=response&RelayState=token
```

Figure 18: requête HTTP POST pour l'envoi du formulaire HTML contenant la réponse à la demande d'authentification

Le service consommateur d'assertions du fournisseur de services extrait le message <Response> du formulaire HTML, puis, vérifie la validité de la signature numérique de l'assertion, si valide, il traite le contenu de l'assertion afin de créer une session de connexion pour l'utilisateur au niveau du SP. Le SP récupère ensuite la valeur du paramètre RelayState pour se rappeler de l'URL de la ressource demandée par l'utilisateur. Il envoie ensuite une réponse de redirection HTTP au navigateur lui demandant d'accéder à la ressource demandée (non illustrée sur le schéma).

7. Un contrôle d'accès est effectué pour déterminer si l'utilisateur dispose de l'autorisation appropriée pour accéder à la ressource. La ressource est alors renvoyée au navigateur si l'utilisateur est autorisé à l'avoir [23].

2.4.3 OpenID Connect

OpenID Connect, publié en 2014 est une évolution du standard OpenID et s'appuie sur le protocole OAuth 2.0 en y ajoutant une couche d'authentification [25]. Il permet aux clients (site/application web, application mobile, etc) de vérifier l'identité de l'utilisateur final sur la base de l'authentification effectuée par un serveur d'autorisation et d'obtenir des informations de profil de base sur l'utilisateur final et cela, d'une manière interopérable et de type REST [26].

Pour comprendre le fonctionnement d'OpenID Connect, il est impératif de comprendre d'abord comment fonctionne le protocole OAuth2.0. Au moment de la rédaction de ce mémoire, OpenID Connect est à sa version 1.0 et OAuth à sa version 2.0.

2.4.3.1 Le protocole OAuth2.0 :

OAuth 2.0 est un protocole de délégation d'autorisation c'est-à-dire, il permet à un utilisateur de donner l'autorisation à une application d'accéder à ses données disponibles sur une autre application sans lui fournir ses identifiants.

2.4.3.2 Les acteurs définis par le protocole Oauth2.0

Il y a quatre acteurs dans le protocole OAuth2.0 qui sont définis dans la RFC 6749 [27].

- Resource owner (utilisateur final): c'est l'utilisateur final propriétaire d'une ressource protégée et capable d'accorder l'accès à cette ressource à une application tierce (client).

- Client : application souhaitant accéder aux ressources protégées.
- Resource server (serveur de ressources) : c'est le serveur qui héberge les ressources protégées et est capable d'accorder l'accès à ces ressources au client grâce aux jetons d'accès (access token).
- Authorizationserver (serveur d'autorisation) : délivre au client des jetons d'accès (access tokens) lui permettant d'accéder aux ressources protégées après avoir authentifié l'utilisateur final (resource owner) et obtenu son autorisation. Le serveur d'autorisation et le serveur de ressources peuvent être le même serveur ou des serveurs distincts.

2.4.3.3 Principe de fonctionnement du protocole OAuth2.0

Le protocole OAuth définit quatre types d'attribution d'autorisation qui sont définis dans la RFC 6749 [27] le plus utilisé est le type « Authorization code grant» que nous allons présenter dans ce mémoire. La figure ci-dessous illustre l'interaction entre les différents acteurs (rôles) de ce protocole suivant le type « Authorization code grant »:

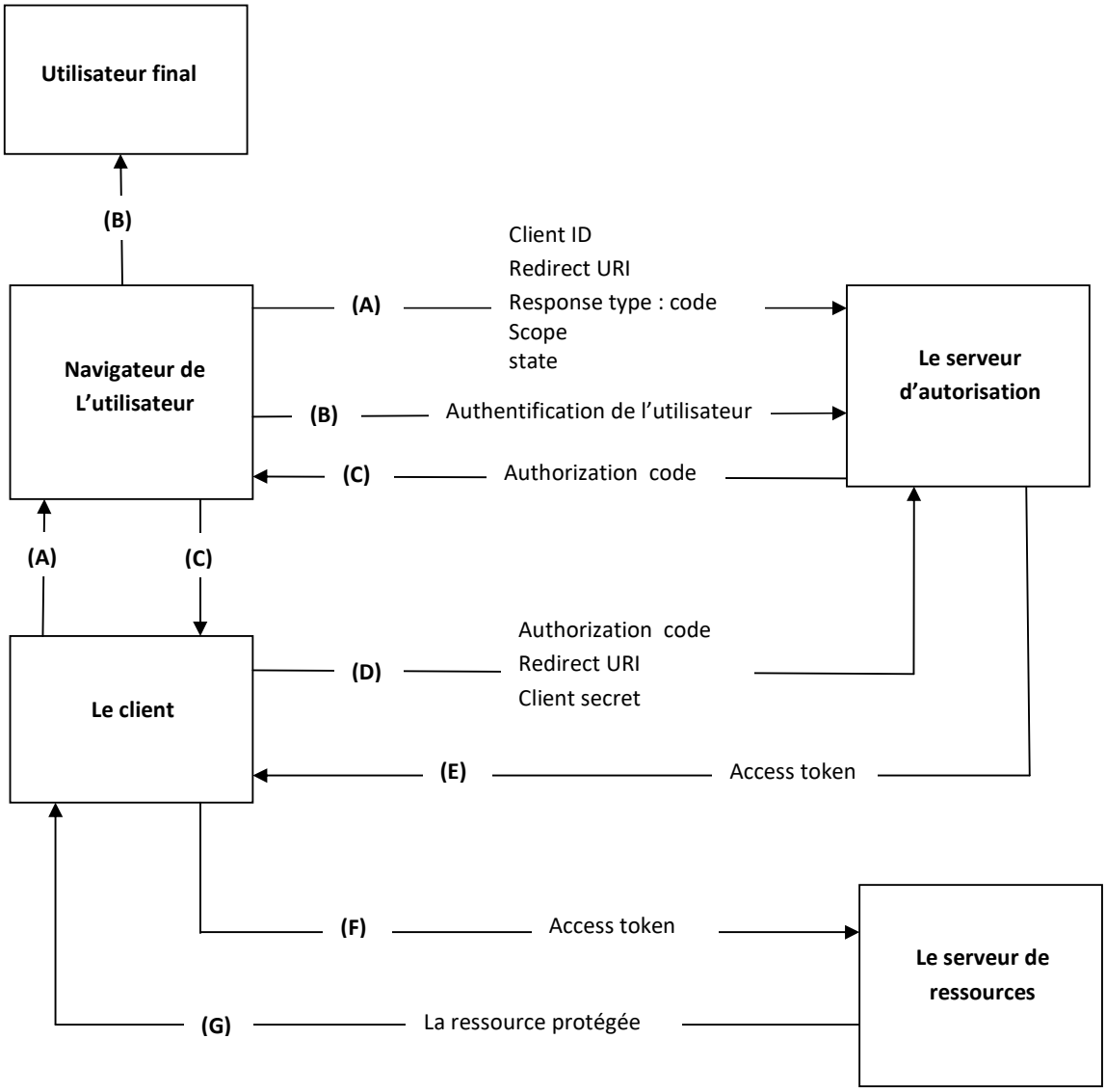


Figure 19: Fonctionnement du protocole OAuth

Remarque :

- Les lignes illustrant les étapes (A), (B) et (C) sont divisées en deux parties car les échanges se font à travers le navigateur (front channel).
- Les échanges dans les autres étapes se font directement entre le client et le serveur d'autorisation (back channel) qui est le canal le plus sécurisé.

Avant d'expliquer les étapes décrites dans la figure ci-dessus, il est important de connaitre les définitions de certains paramètres utilisés par ce protocole :

- Redirect URI (redirect_uri) : il s'agit de l'URL vers laquelle le serveur d'autorisation dirige le navigateur de l'utilisateur après que ce dernier a accordé son autorisation.

- Response type (response_type) : la valeur « code » de ce paramètre est associé au type « Authorization code grant». Elle permet au client d'indiquer au serveur d'autorisation qu'il souhaite recevoir un code d'autorisation (autorisation code).
- Authorization code : c'est un code (chaine de caractères) envoyé par le serveur d'autorisation au client à travers le navigateur après avoir authentifié l'utilisateur et obtenu son autorisation. Il matérialise l'autorisation qui a été accordée par l'utilisateur [28]. Il est utilisé par le client pour obtenir un jeton d'accès (access token) lui permettant d'accéder à la ressource protégée.
- Access token : il s'agit d'un jeton délivré au client par le serveur d'autorisation et représentant une autorisation pour accéder à la ressource protégée.
- Scope : permet au client de spécifier les autorisations qu'il souhaite obtenir. « profile » et « contacts » sont des exemples de valeurs que ce paramètre peut avoir.
- Client ID (client_id) et Client secret (client_secret) : permettent d'authentifier le client auprès du serveur d'autorisation. Ces paramètres sont obtenus après enregistrement du client auprès du serveur d'autorisation.

Description par étape du fonctionnement du protocole OAuth2.0 :

A. Le client demande l'autorisation d'accès à la ressource protégée auprès de l'utilisateur finale par l'intermédiaire du serveur d'autorisation en dirigeant le navigateur de l'utilisateur vers le serveur d'autorisation. Le client inclut dans sa requête son identifiant (client_id), le/les scope(s) demandés, le paramètre state, le paramètre response_type qui doit avoir la valeur « code » et l'URL de redirection (redirect_uri) auquel le serveur d'autorisation renverra le navigateur de l'utilisateur une fois l'accès a été accordé (ou refusé).
B. Le serveur d'autorisation authentifie l'utilisateur et lui demande d'accepter ou non la demande d'accès du client.
C. En supposant que l'utilisateur accorde l'accès, le serveur d'autorisation crée un code d'autorisation et redirige le navigateur de l'utilisateur vers le client à l'aide de l'URI de redirection fournie précédemment. L'URI de redirection inclut le code d'autorisation créé pour le client.
D. Le client demande un jeton d'accès (access token) au serveur d'autorisation en s'authentifiant auprès de ce dernier et en lui présentant le code d'autorisation obtenu dans la précédente étape.
E. Le serveur d'autorisation authentifie le client et délivre un jeton d'accès (access token) après avoir validé le code d'autorisation.
F. Le client demande l'accès à la ressource protégée auprès du serveur de ressources en lui présentant le jeton d'accès (access token) obtenu dans la précédente étape.

G. Le serveur de ressources accepte la demande du client après avoir validé le jeton d'accès (access token).

Ajout d'une couche d'authentification au protocole OAuth2.0 grâce à OpenID Connect

OAuth 2.0 tel que mentionné précédemment est un protocole de délégation d'autorisation c'est-à-dire qu'il permet à un utilisateur de donner l'autorisation à une application d'accéder à ses données disponibles sur une autre application sans lui fournir ses identifiants. OAuth2.0 ne donne aucun

moyen au client d'authentifier l'utilisateur car l'utilisateur s'authentifie auprès du serveur d'autorisation.

C'est là qu'entre en jeu OpenID Connect. Ce dernier permet d'assurer cette authentification grâce à un autre type de jeton appelé ID Token qui est un jeton sécurité envoyé au client et qui contient des informations concernant l'authentification d'un utilisateur final par un serveur d'autorisation et éventuellement d'autres informations demandées [29]. Pour authentifier l'utilisateur final, les mêmes étapes décrites précédemment sont répétées sauf que le client précise dans sa requête qu'il souhaite authentifier l'utilisateur en spécifiant la valeur openid pour le paramètre scope. Le client recevra alors deux jetons : le jeton d'accès (access token) et l'ID token qui est au format JWT (JSON Web Token).

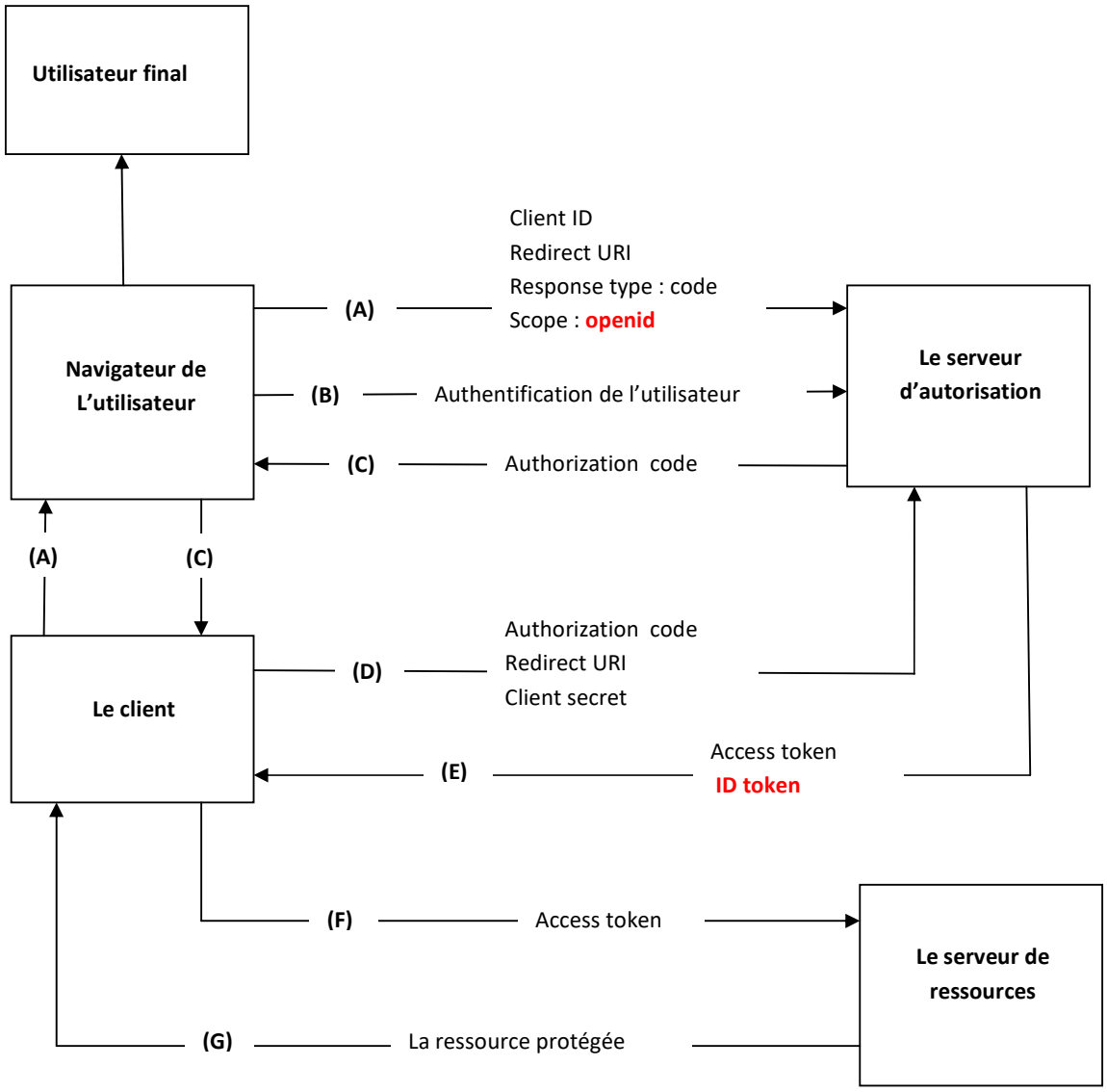


Figure 20: fonctionnement du protocole OpenID Connect

Dans le jargon d'openID Connect, il est courant d'utiliser les appellations Relying Party (RP) et OpenID Provoder (OP) pour représenter les acteurs impliqués dans ce protocole :

Relying Party (RP) : est l'application cliente dans le protocole OAuth2.0 nécessitant l'authentification de l'utilisateur final et d'autres informations auprès du fournisseur d'identité (OpenID Provider).

OpenID Provider (OP) : est le serveur d'autorisation dans le protocole OAuth2.0, qui est capable d'authentifier l'utilisateur finale et de fournir des informations au client (Relying party) concernant l'utilisateur et l'authentification qui s'est déroulée.

En résumé, le protocole OpenID Connect suit les étapes suivantes :

1. Le RP (Client) envoie une requête à l'OP (OpenID Provider)
2. L'OP authentifie l'utilisateur final et obtient l'autorisation
3. L'OP répond à la requête du RP avec un jeton d'identification (ID token) et généralement un jeton d'accès (access token).
4. Le RP peut utiliser le jeton d'accès (access token) pour obtenir d'autres informations concernant l'utilisateur final [30].

2.5 Comparaison entre les protocoles Kerberos, SAML et OpenID Connect

Le tableau 1, ci-dessous, présente une comparaison des protocoles Kerberos, SAML et OpenID Connect :

	Kerberos	SAML	OpenID Connect
Authentification	✓	✓	✓
Authentification mutuelle	✓		
Autorisation		✓	✓
Prise en charge des applications natives et mobiles			✓
Ne nécessite pas une infrastructure supplémentaire coté client		✓	✓
Fédération d'identité	✓	✓	✓

Tableau 1: comparaison des protocoles Kerberos, SAML et OpenID Connect

2.6 Avantages et inconvénients du SSO

2.6.1 Avantages

Le SSO a de nombreux avantages pour l'utilisateur et pour l'organisation en général:

- Grâce à cette technologie, il est possible de se connecter avec un seul identifiant à de nombreux services et applications. L'utilisateur n'a donc plus besoin de retenir (ou de noter) tous ses mots de passe**.** Aussi, avec le SSO, il n'y a plus besoin de saisir ses identifiants à plusieurs reprises, tout cela permet de gagner du temps et donc améliorer la productivité des utilisateurs [16] [14].

- L'authentification unique permet aussi d'augmenter la productivité des développeurs, en effet, le mécanisme SSO étant indépendant et opérationnel, les développeurs n'auront plus à se soucier de l'authentification mais se concentrer sur le service ou l'application à développer [14].
- La gestion des comptes utilisateurs est simplifiée lorsque les applications utilisent le SSO [14]. En effet, les administrateurs informatiques n'auront plus à gérer de multiples systèmes d'authentification qui généralement contiennent les mêmes types de données (cela évite la redondance de données).
- Les utilisateurs disposant d'un seul mot de passe pour accéder à toutes leurs applications n'auront pas besoin d'assistance aussi souvent [31].
- Comme le principe du SSO permet aux utilisateurs de se souvenir d'un seul mot de passe pour accéder à plusieurs applications, ils sont plus susceptibles de créer un mot de passe plus difficile à deviner et moins susceptibles de l'écrire sur des post-its. Cela améliore la sécurité du système en réduit le risque de vol de mot de passe[31][14].

2.6.2 Inconvénients

Le principal inconvénient de cette technologie est que si l'accès à l'authentification unique est compromis par un cybercriminel, ce dernier peut avoir accès à tous les services et applications utilisées par la victime. Pour éviter que cela n'arrive, il est important de renforcer la sécurité du SSO en privilégiant l'authentification unique à deux ou multi facteurs. Il peut être difficile, long et couteux de mettre en place un système d'authentification unique et l'intégrer aux applications existantes [16][14].

2.7 Travaux antérieurs

Après avoir vu le principe de fonctionnement des protocoles d'implémentation du SSO Kerberos, SAML et OpenID Connect. Dans cette partie nous allons présenter quelques travaux de recherche sur l'implémentation du SSO en utilisant ces protocoles.

2.7.1 Implémentation du SSO avec le protocole Kerberos

Dans [2] une proposition d'une architecture d'un système d'authentification unique (SSO) basée sur le protocole Kerberos a été donnée. Dans ce travail les auteurs ont mentionné que les systèmes de gestion des identités sont largement utilisés dans différents types d'organisations et que le SSO est une fonctionnalité importante de ces systèmes.

L'architecture proposée par les auteurs est une architecture SSO centralisée avec des fournisseurs d'identité indépendants utilisant Kerberos comme fournisseur d'identité centralisé (chargé de fournir le SSO). Les fournisseurs d'identité indépendants sont quant à eux chargés de l'autorisation. Bien que l'architecture proposée soit une architecture centralisée, les auteurs ont cependant précisé qu'il est possible d'utiliser Kerberos dans les architectures SSO fédérée.

Les auteurs ont choisi Kerberos en raison des différents avantages qu'offre ce protocole.

Le principal avantage de l'approche proposée est qu'elle fournit l'authentification mutuelle qui aide à se protéger des attaques de phishing et de man in the middle, elle permet d'intégrer la PKI, les

communications à travers le réseau sont sécurisées vu que les secrets ne sont jamais envoyés en clair, etc.

Kerberos ne fournit cependant pas de fonctions d'autorisation ni de fonctions d'audit. Par conséquent, il est nécessaire d'utiliser d'autres outils et applications pour assurer ces tâches.

Dans [32], les auteurs suggèrent l'utilisation du protocole Kerberos comme système d'authentification dans le Cloud computing. Pour les auteurs, le Cloud computing est la technologie la plus utilisée de nos jours et est également l'un des domaines le plus connu dans le monde de la recherche en informatique. Sa facilité d'accès aux services est la principale raison de sa popularité. Grâce au Cloud computing, il n'est plus besoin d'acheter, d'installer et de maintenir des logiciels coûteux car ceux-ci sont accessibles via le Cloud avec un payement par utilisation. Malheureusement, le Cloud computing attire aussi les personnes malintentionnées cherchant à exploiter des failles pour en tirer des avantages illégaux. La sécurité est donc un facteur très important. Différentes politiques de cryptage et de contrôle d'accès sont développées pour fournir la meilleure sécurité possible. Mais, il n'est pas possible de toutes les appliquer car cela augmentera le coût de calcul et les besoins en mémoire pouvant causer un ralentissement ou un disfonctionnement du système. La sécurité doit fournir des fonctionnalités telles que la confidentialité, l'intégrité, la disponibilité et l'authentification. Une communicationsécurisée entre l'utilisateur du Cloud et le serveur est indispensable. Les auteurs dans cet articleont proposé une solution qui consiste à utiliser Kerberos comme système d'authentification dans le Cloud Computing car selon eux, Kerberos fournit la sécurité requise par le système et qu'il est meilleur que le système d'authentification basé sur le nom d'utilisateur et le mot de passe. Les auteurs ne font pas que proposer Kerberos mais demandent carrément de l'inclure dans le Cloud Computing.

2.7.2 Implémentation du SSO avec le protocole SAML

Dans [1], les auteurs décrivent l'implémentation du protocole SAML pour fournir des solutions d'authentification unique (SSO) sécurisées pour les applications hébergées en externe.Le but étant de permettre aux utilisateurs de l'organisation d'accéder aux ressources de l'organisation et aux services externes en s'authentifiant à partir du fournisseur d'identité de l'organisation. Le protocole SAML va permettre l'échange sécurisé d'informations de sécurité de l'utilisateur entre le fournisseur d'identité (appartenant à l'organisation) et les fournisseurs de services externes (ASP ou SaaS) via le processus de fédération d'identité. Il est nécessaire de configurer proprement toutes les entités membre de la fédération à savoir l'entité local (fournisseur d'identité de l'organisation) et les entités partenaires (les fournisseurs de services externes). Les auteurs ont cité plusieurs produit propriétaires ou open source prenant en charge la fédération via SAML versions 1.1 et 2.0 sans préciser de préférence à un produit particulier. SAML étant un standard, tous les produits peuvent être utilisés de manière interchangeable sans problème de compatibilité tant qu'ils sont conformes aux normes SAML. L'entité locale va être configurée manuellement dans le logiciel de fédération mais avec SAML 2.0, le processus de configuration de l'entité partenaire a été simplifié grâce à l'introduction des métadonnées (metadata). Étant donné que la norme SAML est flexible et peut permettre un certain nombre de configurations personnalisées, certains accords et informations de configuration doivent être initialement mis en place entre deux partenaires. Cela se fait grâce à l'échange des métadonnées contenant ces informations spécifiques. Une fois le fichier metadata qui

est un fichier XML est reçu par l'entité partenaire, il est chargé dans le logiciel de fédération sans qu'aucune configuration supplémentaire ne soit nécessaire pour l'entité partenaire. Par rapport à la configuration manuelle, les métadonnées permettent de gagner du temps et réduisent les risques d'erreur. Le fichier metadata contient des éléments et des attributs dont les éléments EntityDescriptor et EntityID qui spécifient à quelle entité les informations de configuration font référence. La liste complète des éléments que peut contenir un fichier metadata est disponible sur le site du consortium OASIS [33]. Lors de la configuration manuelle de l'entité locale, il est nécessaire de déterminer les paramètres à transmettre dans l'assertion notamment l'identifiant (le nom d'utilisateur) unique pour chaque utilisateur par exemple une adresse e-mail ou le numéro d'employé. Il est important aussi d'indiquer comment les paramètres seront transmis (dans les attributs ou dans le nameID) ainsi que les certificats de sécurité de l'association et la stratégie de signature requises. Les auteurs ont donné un exemple d'un fichier metadata qui sera envoyé de l'entité locale (dans ce cas c'est le fournisseur d'identité) à l'entité partenaire (fournisseur de services) ou il sera chargé dans le logiciel de fédération. Certains éléments du fichier metadata sont obligatoires, tels que entityID, et d'autres sont facultatifs, tels que ID et OrganizationName. Les auteurs ont précisé les éléments à prendre en considération qui sont : « binding, location, protocolSupportEnumeration, KeyDescriptor () et KeyInfo () ». Les auteurs ont donné également un autre exemple d'un fichier metadata à envoyer d'un fournisseur de services à un fournisseur d'identité pour être chargé dans le logiciel de fédération. Une fois les métadonnées échangées et toutes les entités configurées, l'assertion peut être testée et vérifiée à l'aide d'outils de navigateurs et des décodeurs. Un exemple d'un test a été donné dans cet article. Avec SAML les organisations peuvent facilement et en toute sécurité partager des informations d'identité. Les utilisateurs n'ont plus besoin de mémoriser plusieurs noms d'utilisateur et mots de passe, ce qui permet également de réduire les appels au service d'assistance. Les auteurs ont conclu par des recommandations à prendre en considération pour une meilleure implémentation de ce protocole :

- Les entreprises doivent disposer de la documentation à échanger avec les partenaires lors de la mise en place de SAML, car chaque cas d'utilisation de SAML peut être personnalisé en fonction des besoins de l'entreprise. Cette documentation peut permettre de gagner du temps pendant les phases de mise en œuvre et de test du projet ou lors de la résolution d'un problème.
- Durant les phases de test, il est utile de tester avec un site exemple pour le fournisseur de services et également de faire des tests avec des assertions SAML signées uniquement (non cryptées). Le site exemple va servir à tester la connexion SAML entre les deux partenaires avant de tester l'application elle-même. Le test avec des assertions signées uniquement permet de valider les données transmises au fournisseur de services. En effet, l'assertion n'étant pas cryptée, permet après décodage de voir et de valider les données transmises au fournisseur de services.
- Le fournisseur d'identité et le fournisseur de services doivent utiliser les fichiers metadata pour accélérer le travail de configuration et réduire les erreurs.
- Les fournisseurs d'identité et les fournisseurs de services doivent être informés de toute modification du standard SAML approuvé par le consortium OASIS car rester à jour et ne pas s'écarter des normes permet d'assurer la compatibilitéentre les organisations.

2.7.3 Implémentation du SSO avec le protocole OAuth et OpenID Connect

Dans [34] les auteurs proposent une implémentation d'un système d'authentification unique pour les établissements d'enseignement en utilisant OpenId connect et OAuth 2.0. Le système d'authentification proposé par les auteurs est composé de deux parties, un fournisseur Open Id Connect (OpenID Provider/OP). Ce serveur est chargé de stocker toutes les informations d'identification de l'utilisateur (c'est à dire tous les noms d'utilisateur utilisés par les différentes applications telles que l'id et l'e-mail) et un mot de passe commun à tous ces noms d'utilisateur. L'autre partie est le fournisseur de service (Relying party/RP) sur lequel sont déployées les applications. L'idée de base est qu'à chaque fois que l'utilisateur veut accéder à une application au lieu qu'il s'authentifie sur chaque site séparément (comme c'est le cas dans le système existant), l'application envoie une requête au fournisseur Open Id Connect. Ce dernier vérifie si l'application demandant l'authentification unique est une application (RP) valide. Si c'est le cas, le fournisseur authentifie l'utilisateuren lui proposant une seule page de connexion ou il doit saisir ses informations d'identification pour vérification.

Les avantages attendus du système proposé sont :

- Meilleure gestion des données de l'utilisateur.
- Facilité d'utilisation pour le l'utilisateur final.
- Aucun problème de sécurité.
- Eviter le partage de données avec d'autres fournisseurs de SSO tels que Facebook, Google, etc.

Le système proposé est mis en œuvre à l'aide de Liberty Websphere d'IBM et est déployé sur deux systèmes différents, à savoir le système A et le système B. Les deux systèmes ont été configurés de manière à est-ce que le système A fonctionne en tant que fournisseur Open Id (OP) et le système B en tant que fournisseur de service (RP) sur lequel sont déployées les applications.

Le nouveau système a été testé sur cinq utilisateurs novices n'ayant pas l'habitude d'utiliser des sites web/applications nécessitant une authentification. Il leur a été demandé de se connecter à Gmail de l'université et à l'espace étudiant. Puis, il leur a été demandé de se connecter aux mêmes sites mais en utilisant la page de connexion du nouveau système. D'après cette expérience, les auteurs ont démontré que le système existant présente des lacunes, notamment pour les nouveaux utilisateurs. En effet, lorsque l'utilisateur n'est pas habitué à l'environnement, il lui faut du temps pour se rappeler des informations nécessaires pour se connecter. Si l'utilisateur est un habitué du système existant, il ne sera pas confronté à ce problème. Mais il n'est pas possible d'affirmer que même l'utilisateur habitué au système se souviendra de toutes les informations d'identification sans prendre un temps de réflexion. Le système proposé fonctionne donc beaucoup mieux pour les nouveaux utilisateurs car il leur permet d'accéder au système sans avoir à se rappeler de plusieurs combinaisons de nom d'utilisateur et de mot de passe. En effet, ils pourront s'authentifier avec n'importe quelle combinaison de nom d'utilisateur et mot de passe dont ils se souviennent et cela peu importe à quel site il souhaite accéder. Par rapport au système existant où chaque site ou application dispose de sa propre base de données d'utilisateurs, page de connexion et mécanisme d'authentification. Le nouveau système dispose d'un seul endroit pour stocker les informations

d'identification (et autres informations relatives aux étudiants et aux facultés), pour s'authentifier et pour changer son mot de passe. Le système proposé permet aux utilisateurs de changer leur mot de passe. Le mot de passe est alors mis à jour sur toutes les applications ce qui n'est pas le cas du système existant. Les auteurs ont ensuite mesuré les performances du nouveau système par rapport au système existant selon des paramètres bien définis qui sont (DNS Lookup , TCP/IP Connect et HTTPS Handshake). Pour cela, ils ont sélectionné un site du système existant et mapper ses données vers le système proposé. L'analyse a révélé que les deux systèmes ont a peu près les mêmes performances mais selon les auteurs, le nouveau système est plus sécurisé étant donné qu'il offre aux utilisateurs la possibilité de changer leurs informations d'identification qui n'est pas le cas pour le système existant. Pour conclure, selon les auteurs, la mise en place d'un système d'authentification unique utilisant OpenID Connect et OAuth ne facilite pas seulement la tâche à l'utilisateur, mais l'organisation en bénéficiera dans l'ensemble.

Les auteurs de [35] décrivent l'implémentation del'authentification unique pour un système de gestion d'une bibliothèque en ligne de l'université indonésienne « Universitas Negeri Semarang »en utilisant Google comme fournisseur d'identité. De nos jours, les réseaux sociaux tels que Facebook ou LinkedIn ou bien même Google peuvent fonctionner comme des IdPs. Ils utilisent pour cela des API (Application Programming Interface ou API) qu'ils mettent à disposition des développeurs d'applications tiers afind'intégrer à leurs applications un système de connexion via ces réseaux sociaux. Ce qui permet de mettre en place facilement un système d'authentification unique pour plusieurs systèmes. C'est justement cette solution que les auteurs ont appliqué afin de mettre en place un système d'authentification unique pour l'accès aux services d'une bibliothèqueen ligne de l'université « Universitas Negeri Semarang » en utilisant Google comme fournisseur d'identité.

2.7.4 Discussion

Pratiquement tous ces travaux parlent de l'implémentation du SSO pour pallier aux problèmes des systèmes d'authentification traditionnels à savoir le problème de redondance des données qui complique la gestion des utilisateurs aux équipes IT des différents types d'organisations et le problème de mémorisation de plusieurs identifiants et mots de passes par les utilisateurs ce qui réduit leur productivité et peut aussi causer des problèmes de sécurité.

Le travail de [34] est intéressant dans le sens ou avec un seul protocole qui est OpenID Connect (sachant que celui-ci se base sur le protocole OAuth2.0) ils ont pu mettre en place un système d'authentification unique permettant aux utilisateurs d'accéder aux applications on-premises (espace étudiant) et aux applications externes (Gmail de l'université). Les auteurs ont préféré implémenter eux même ce système que de louer le service IDaaS chez un fournisseur de services Cloud car pour eux cela évite de partager des données de l'institution avec des applications tierces. Un autre avantage c'est de pouvoir configurer et gérer ce système selon ses préférences et ses besoins. L'inconvénient est que la maintenance de ce système est une tâche qu'ils doivent aussi assurer. En lisant cet article, on sent que les auteurs ont mis l'accent sur l'avantage du SSO qui consiste à n'utiliser qu'une seule paire (identifiant et mot de passe) pour accéder aux différentes applications/sites et l'avantage que procure ce système pour les nouveaux utilisateurs, cependant, les auteurs n'ont pas mis l'accent ou bien très peu sur l'autre avantage du SSO qui consiste à se connecter qu'une seule fois pour accéder à toutes les applications/sites web c'est-à-dire qu'il n'est

pas nécessaire de se connecter à chaque application séparément. Aussi, dans cet article, les auteurs font référence au protocole OAuth2.0 comme étant un protocole d'authentification. Cependant OAuth2.0 n'est pas un protocole d'authentification mais de délégation d'autorisation comme on l'a déjà vu précédemment dans ce mémoire. D'ailleurs, Nat Sakimura, le président de la Fondation OpenID au moment de la rédaction de ce mémoire rejette l'idée de considérer OAuth comme étant un protocole d'authentification et précise que dans le cas de Facebook,OAuth n'est pas utilisé pour l'authentification mais ils ont développé une extension appelée « Signed request » qui est la même chose que le jeton d'identification (ID token) d'OpenID Connect sauf que « Signed request » ne fonctionne qu'avec Facebook comme fournisseur d'identité [36]. Pourquoi alors développer OpenID Connect qui ajoute une couche d'authentification en dessus du protocole OAuth2.0. Cet article n'est d'ailleurs pas le seul à appeler ce protocole un protocole d'authentification, par exemple les auteurs de [37] considèrent OAuth comme une méthode alternative du mécanisme d'authentification. OAuth2.0 permet d'authentifier le client, lors de l'enregistrement du client auprès de l'IdP, ce dernier génère un ID client et un secret client permettant de l'authentifier mais il ne faut pas le confondre avec d'autres types d'authentification telle que l'authentification des utilisateurs [36].

Dans [2], une proposition d'une architecture d'un système d'authentification unique (SSO) basée sur le protocole Kerberos est donnée en citant les avantages que peut apporter cette solution.

L'architecture proposée est une architecture SSO centralisée, mais de nos jours, de plus en plus d'organisations font appel à des services Cloud comme les applications SaaS, d'où la nécessité de recourir à des architectures fédérées. Les auteurs ont cependant précisé que Kerberos ne se limite pas à la gestion centralisée des identités mais peut aussi être utilisé dans les architectures SSO fédérée sans pour autant proposer une architecture.

Les auteurs de [32] évoquent le besoin très fort de sécurité dans le Cloud computing tant pour l'utilisateur que pour le fournisseur de services Cloud. Ils ont proposé de ce fait un système d'authentification en utilisant Kerberos qui selon eux, est plus sécurisé. L'implémentation de la solution et son évaluation n'ont cependant pas été réalisés, les auteurs mentionnent que cela pourrait être fait dans des travaux futurs.

Les auteurs de [1] expriment dans ce travail le besoin dans une entreprise d'un système d'authentification unique prenant en charge à la fois les applications on-premises et les applications externes et proposent l'implémentation du protocole SAML pour fournir cette solution.

Dans [35], les auteurs ont implémenté une solution SSO pour un système de gestion d'une bibliothèque en ligne d'une université indonésienne en utilisant Google comme fournisseur d'identité. La solution est intéressante car elle utilise un système d'authentification existant qui est Google. L'équipe IT n'a pas à se soucier de la maintenance, de la sécurité, de la mise à jour, etc, de ce système, Google s'en chargera. La question que l'on peut se poser est pourquoi les auteurs n'ont pas généralisé cette solution sur l'ensemble des services de l'université, pourquoi uniquement la bibliothèque. Peut-on implémenter cette solution pour prendre en charge l'ensemble des services qu'offre une université ?

2.8 Conclusion

Les auteurs dans [34] voient que peu de choses sont faites pour améliorer l'authentification des utilisateurs dans les établissements d'enseignement mis à part des solutions SSO basées sur le Cloud comme EduTone et OneLogin.Il est en effet intéressant de proposer des solutions d'implémentation du SSO selon le type d'organisation (université, entreprise, etc) car même si l'authentification unique règle les mêmes problèmes quel que soit le type d'organisation et des protocoles d'implémentation du SSO sont également disponibles, il reste que chaque type d'organisation dispose de son propre type d'utilisateur, ses propres objectifs, ses propres moyens, types d'applications utilisées, etc, qui peuvent nécessiter des configurations ou des moyens matériels ou logiciels particuliers . Dans le chapitre suivant, nous allons parler du besoin de la mise en œuvre d'une solution d'authentification unique dans le cas de l'université de Béjaia et nous essayerons de proposer une solution qui répondra à ces besoins.

Chapitre 3 Solutions pour l'implémentation du SSO à l'université de Béjaia

3.1 Introduction

L'université de Béjaia comme toute organisation dispose d'un ensemble de services mis en place pour l'ensemble de la communauté universitaire. On cite comme exemple, la messagerie électronique et une plateforme Moodle. Chaque service dispose de son propre système d'authentification qui veut dire sa propre base de données d'utilisateurs. Les utilisateurs disposent ainsi de plusieurs identifiants et mots de passe à mémoriser sans compter ceux des autres sites ou services n'appartenant pas à l'université. Ce qui conduit les utilisateurs à souvent oublier leur mot de passe et parfois même leur nom d'utilisateur car ce dernier est aussi différent pour chaque service. Les demandes de changement de mot de passe sont alors très fréquentes. De plus, l'utilisateur doit s'authentifier séparément à chaque service ce qui est une perte de temps en plus d'ennuyer l'utilisateur. Concernant les administrateurs de ces sites, ils doivent faire face à ces nombreuses demandes de modification de mot de passe sachant que l'université compte plus de 40000 étudiants et plus de 1700 enseignants sans compter le personnel ATS. Sans oublier la redondance de données, en effet, les administrateurs devront gérer pratiquement les mêmes données pour chaque service. À chaque développement d'une nouvelle application, un nouveau système d'authentification (une nouvelle base de données/ nouveaux identifiants) est mis en place. Les administrateurs auront donc à gérer le service et la base de données des utilisateurs associée. La multiplicité des systèmes d'authentification est une contrainte pour les utilisateurs et pour les administrateurs informatiques. Il est donc nécessaire de mettre en place une solution pour y remédier. La solution consiste à la mise en place d'un système d'authentification unique (Simgle sign-on ou SSO).

Dans ce chapitre, nous allons proposer deux solutions de mise en place du SSO. Dans la première solution, nous allons utiliser Google comme fournisseur d'identité. Quant à la deuxième solution, nous allons mettre en place un système d'authentification unique en local. Ainsi, nous allons commencer par une étude de l'existant puis, pour chaque solution proposée, nous allons indiquer le protocole utilisé, les étapes de mise en œuvre de la solution et on terminera par une discussion et une conclusion.

3.2 Étude de l'existant

À l'université de Béjaia, parmi les services auxquels accèdent les utilisateurs et qui nécessitent une authentification on cite, la messagerie électronique qui est Gmail et d'une manière générale tous les services de Google Workspace (google drive, google meet, etc). Il s'agit donc des applications SaaS fournies par le fournisseur de services Google Cloud. Pour l'accès à ces services, les utilisateurs disposent d'un compte qui est une adresse mail sous la forme : prénom.nom@univ-bejaia.dz pour le personnel enseignants et ATS et de la forme prénom.nom@faculté.univ-bejaia.dz pour les étudiants, sachant que « faculté » correspond aux initiales ou au diminutif du nom de la faculté à laquelle appartient l'étudiant. Par exemple, un étudiant de la faculté de technologie et un étudiant de la faculté des sciences humaines et sociales auront respectivement des adresses mail de la forme : prénom.nom@tech.univ-bejaia.dz et prénom.nom@shs.univ-bejaia.dz.

Un autre service est la plateforme Moodle de l'université mise en place en interne (en local ou sur site). Le nom d'utilisateur du personnel enseignant et ATS est sous la forme nom.prénom quant aux étudiants, il s'agit de leur matricule construit à partir du numéro d'inscription du bac. Il existe d'autres services que nous ne citerons pas dans ce mémoire qui sont spécifiques à un ensemble particulier d'utilisateurs. Nous avons donc à l'université de Béjaia des applications SaaS (Google Workspace) et des applications locales pour lesquelles il faut créer un système d'authentification unique.

3.3 Proposition 1 : utiliser Google comme fournisseur d'identité

Dans cette solution, nous allons utiliser Google comme point central pour l'authentification c'est-à-dire utiliser les identifiants Google qui sont l'adresse mail mentionnée auparavant et son mot de passe associé pour accéder à toutes les applications Cloud et locales. Pour les applications Cloud, le système est déjà fonctionnel étant donné qu'une fois connecté à son compte Gmail, l'utilisateur se retrouve automatiquement connecté aux autres applications de Google Workspace. Pour les applications internes, il faut les configurer pour utiliser Google comme fournisseur d'identité c'est à dire que les utilisateurs devront s'authentifier auprès de Google (avec leur compte Google) pour pouvoir accéder à ces applications. Nous devons donc utiliser un protocole SSO inter domaines. Nous allons utiliser pour nos tests, une plateforme Moodle de test installée en locale. Avec Moodle, nous avons le choix d'utiliser le protocole OpenID Connect ou bien SAML, d'ailleurs la solution d'utiliser Google comme fournisseur d'identité pour s'authentifier à Moodle avec OpenID Connect nous a été proposée et nous l'avons mis en place sur une plateforme Moodle de test l'année dernière c'est à dire avant même notre inscription à ce master. Nous avons cependant testé cette solution avec le protocole SAML sur une autre plateforme Moodle de test et cela, durant la réalisation de ce mémoire.

3.3.1 En utilisant le protocole OpenID Connect

Dans ce cas, Moodle est le client et Google est le fournisseur d'identité. Les étapes à suivre pour la mise en place du SSO sont décrites dans [38]. On peut les résumer comme suit :

Étape 1 : Pour commencer, il faut activer dans Moodle le plugin qui permet d'utiliser Google comme fournisseur d'identité, il s'agit du plugin OAuth 2 (Figure 21) :

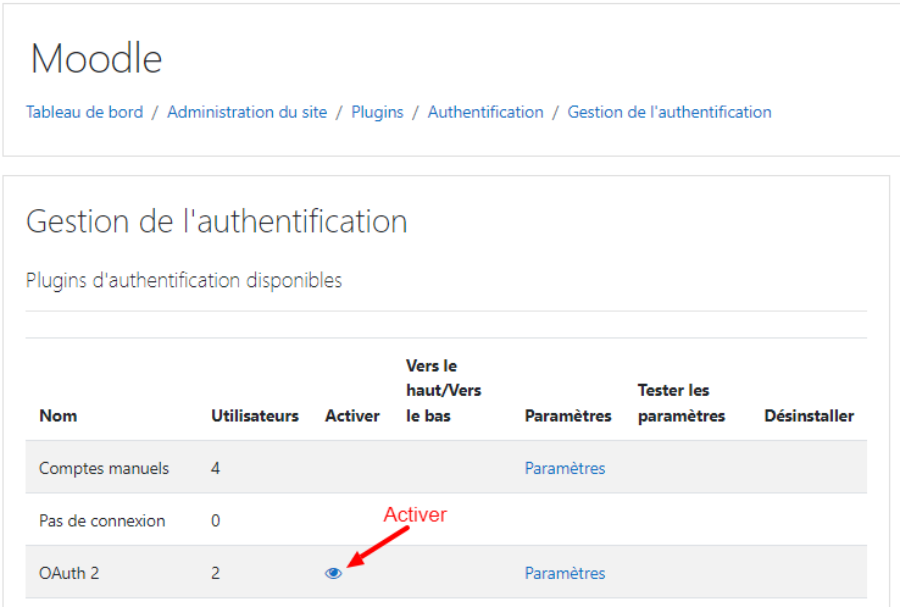


Figure 21: Activation du plugin OAuth 2 dans Moodle

Étape 2 : la deuxième étape consiste à enregistrer Moodle (le client) auprès du fournisseur d'identité (Google) afin d'obtenir l'identifiant Client (Client ID) et son mot de passe (Client secret). C'est une façon de construire la relation de confiance entre le client et le fournisseur d'identité. Pour obtenir l'ID client et le mot de passe client, il faut suivre les instructions indiquées dans [39].

Le fournisseur d'identité a besoin dans cette étape de connaitre l'URL de redirection correspondante à Moodle qui est de la forme : https://moodle.example.com/admin/oauth2callback.php il s'agit de l'URL ou l'utilisateur sera redirigé une fois authentifié auprès du fournisseur d'identité.

Étape 3 : dans cette étape, il est question de configurer Moodle pour utiliser Google comme fournisseur d'identité, cela se fait sur la page de configuration du fournisseur qui s'affiche en cliquant sur le bouton « Google » (Figure 22) :

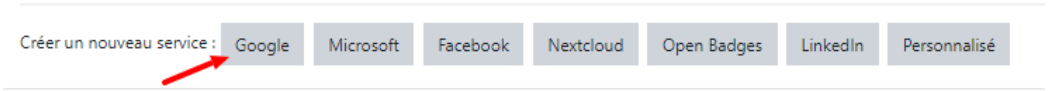


Figure 22: Accès à la page de configuration de l'IdP Google dans Moodle

Il est à noter que Moodle peut prendre en charge différents fournisseurs d'identités comme Microsoft, Facebok, etc.

L'ID client et le secret obtenus dans la précédente étape doivent être insérés dans les champs adéquats (Figure 23). Les scopes requis par Moodle sont : openid, profile et email :

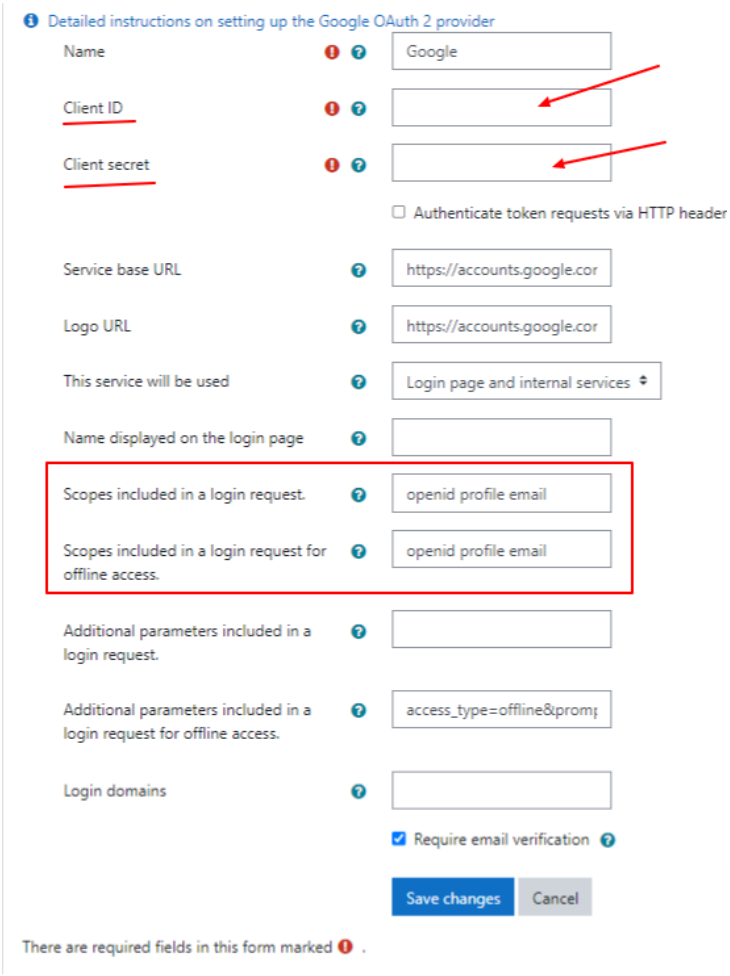


Figure 23: configuration de l'IdP dans Moodle

Étape 4 : après que le fournisseur d'identité a bien été ajouté et configuré dans Moodle (Figure 24), il est important de vérifier ou de configurer les mappages des champs utilisateur c'est-à-dire la correspondance des attributs du fournisseur d'identité avec ceux de Moodle (Figure 25).

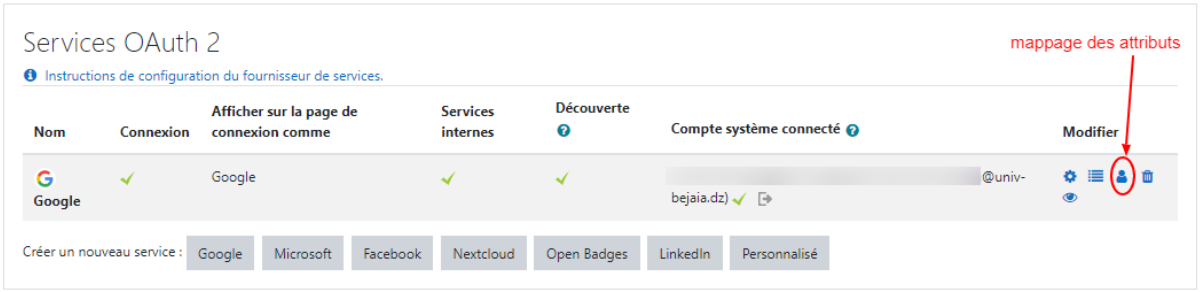


Figure 24: aperçu de l'IdP une fois configuré

Par défaut, le nom d'utilisateur dans Moodle lui correspond l'adresse mail de Google. Il est également possible de créer d'autres mappages :

Figure 25: Mappage des attributs utilisateur

Fonctionnement : lorsqu'un utilisateur souhaite se connecter à Moodle, il verra sur la page de connexion un bouton l'invitant à se connecter à partir de Google :

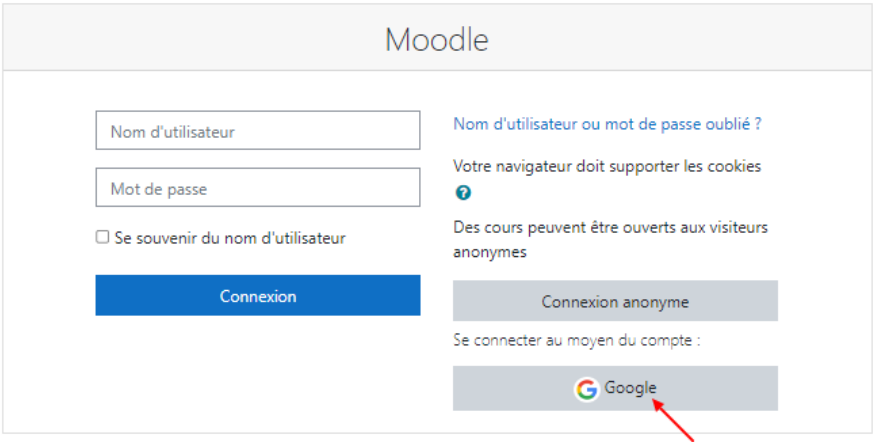


Figure 26: Bouton de connexion à partir de Google

Lorsque l'utilisateur clique dessus, son navigateur se redirige vers le fournisseur d'identité Google pour s'authentifier :

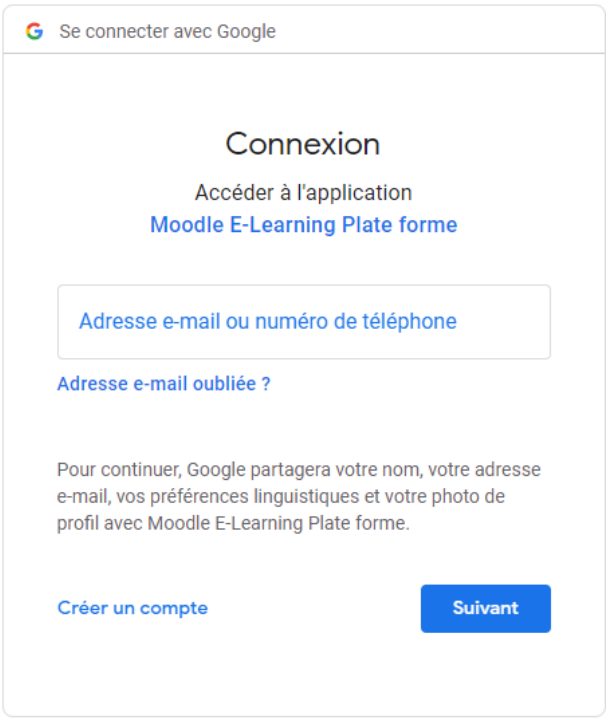


Figure 27: Authentification à partir de Google

L'utilisateur saisi alors son adresse mail et son mot de passe. Une fois connecté, l'utilisateur est redirigé à nouveau vers Moodle. Si l'utilisateur ne dispose pas de compte dans Moodle alors, un compte lui sera créé dont les valeurs des champs du profil seront ceux obtenus à partir du fournisseur d'identité grâce au mappage d'attributs (voir l'étape 4). Si un compte existe déjà pour cet utilisateur (par exemple créé manuellement) alors il sera demandé à cet utilisateur de lier son compte Google avec le compte existant. Sachant que la vérification de l'existence d'un compte se fait sur la base de l'adresse mail.

L'utilisateur une fois connecté au fournisseur d'identité Google il sera automatiquement connecté à tous les services qui auront intégré ce fournisseur d'identité comme système d'authentification.

3.3.2 En utilisant le protocole SAML

Il est également possible dans Moodle d'utiliser le protocole SAML pour la mise en place du SSO avec Google comme fournisseur d'identité. Pour faire le test, nous avons installé une nouvelle plateforme Moodle de test sur laquelle nous avons installé et activer un plugin permettant la prise en charge du protocole SAML car il n'est pas disponible par défaut dans Moodle, il s'agit du plugin « SAML2 Single sign on » téléchargé à partir de [40]. Les étapes à suivre pour configurer Moodle (le fournisseur de service) et le fournisseur d'identité (Google) sont indiquées dans [41]. Ci-dessous, un résumé des étapes importantes:

Étape 1 : lors de la création d'une application SAML personnalisée au niveau du fournisseur d'identité (il faut avoir pour cela le privilège d'administrateur dans Google Workspace), il est

nécessaire de télécharger les métadonnées du fournisseur d'identité Google (Figure 28) et les insérer dans la page de configuration du plugin SAML dans Moodle (Figure 29) :

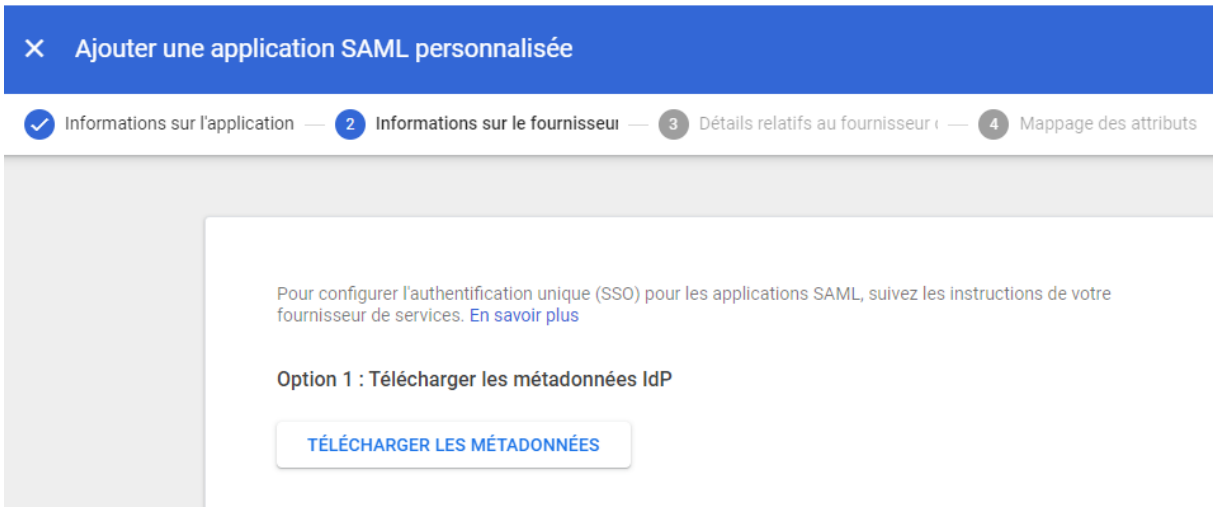


Figure 28: télécharger les métadonnées de l'IdP Google

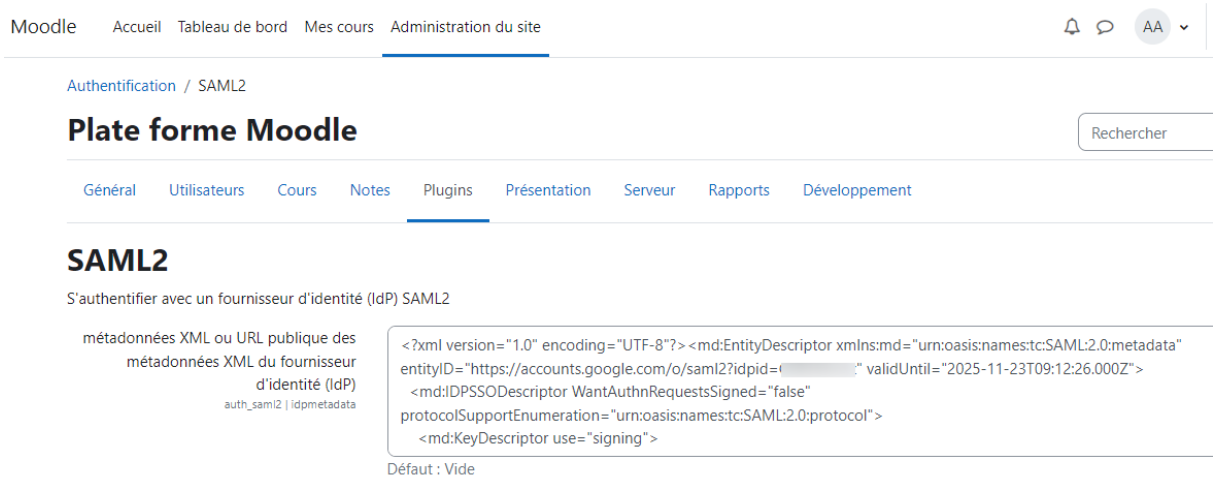


Figure 29: copier les métadonnées de l'IdP dans Moodle

De même, il faut extraire à partir des métadonnées de Moodle (Figure 30) les valeurs des élémentsACS URL (l'URL du service consommateur d'assertions) et Entity ID et les intégrer dans les paramètres de configuration du fournisseur de service au niveau du fournisseur d'identité (Figure 31):

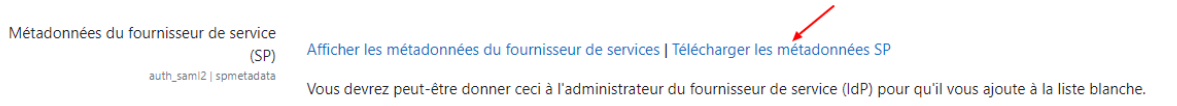


Figure 30: télécharger les métadonnées de Moodle

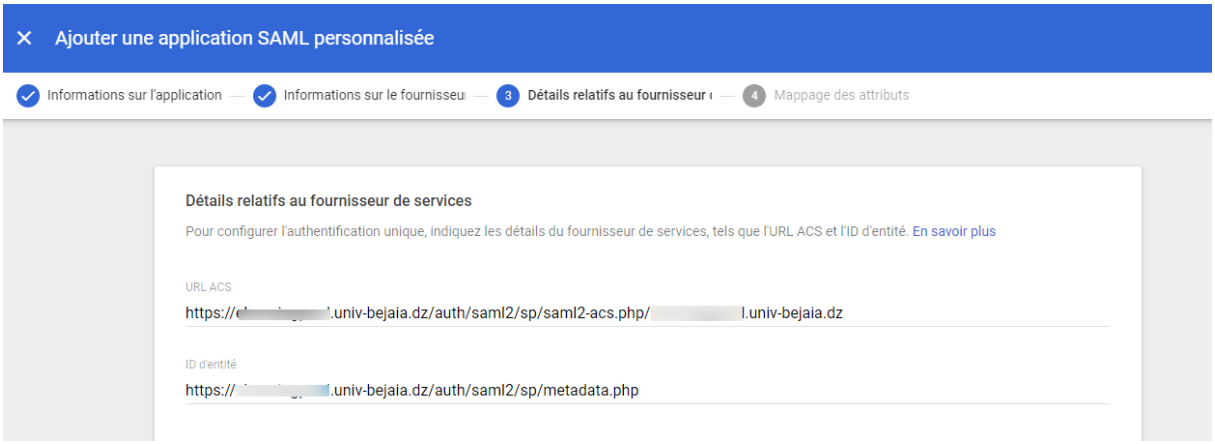


Figure 31: insertion des valeurs des paramètres URL ACS et Entity ID correspondant à Moodle dans l'IdP

Pour rappel, les métadonnées contiennent les configurations requises par le fournisseur de service et le fournisseur d'identité. L'échange de ces informations est aussi une façon d'établir la relation de confiance entre ces deux parties.

Étape 2 : comme pour OpenID Connect, il est nécessaire de mapper les attributs du fournisseur d'identité sur ceux du fournisseur de service ceci permet lors de la création d'un compte dans Moodle de renseigner les champs du profil utilisateur avec les attributs obtenus à partir du fournisseur d'identité.

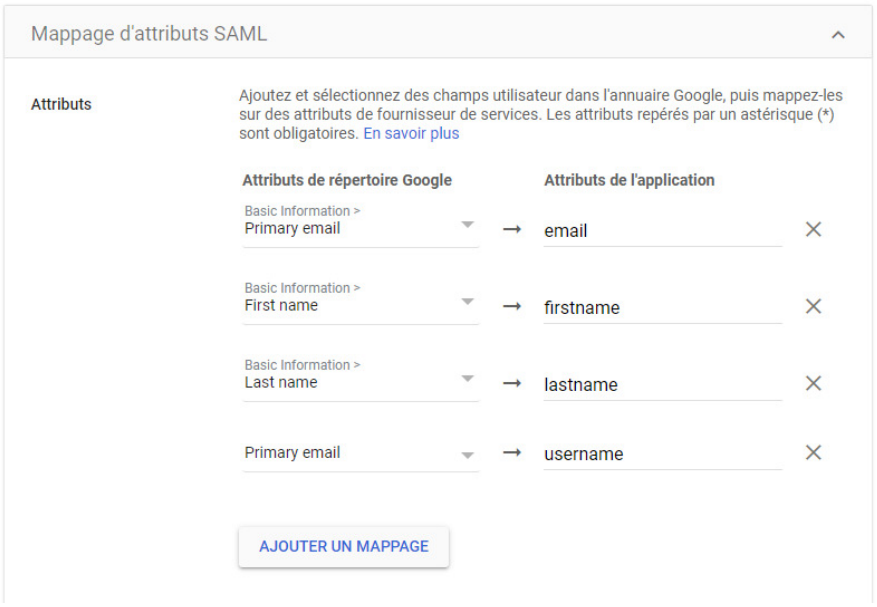


Figure 32: mappage des champs utilisateur au niveau de l'IdP

Le mappage d'attributs doit se faire aussi au niveau de Moodle, dans les paramètres de configuration du plugin SAML :

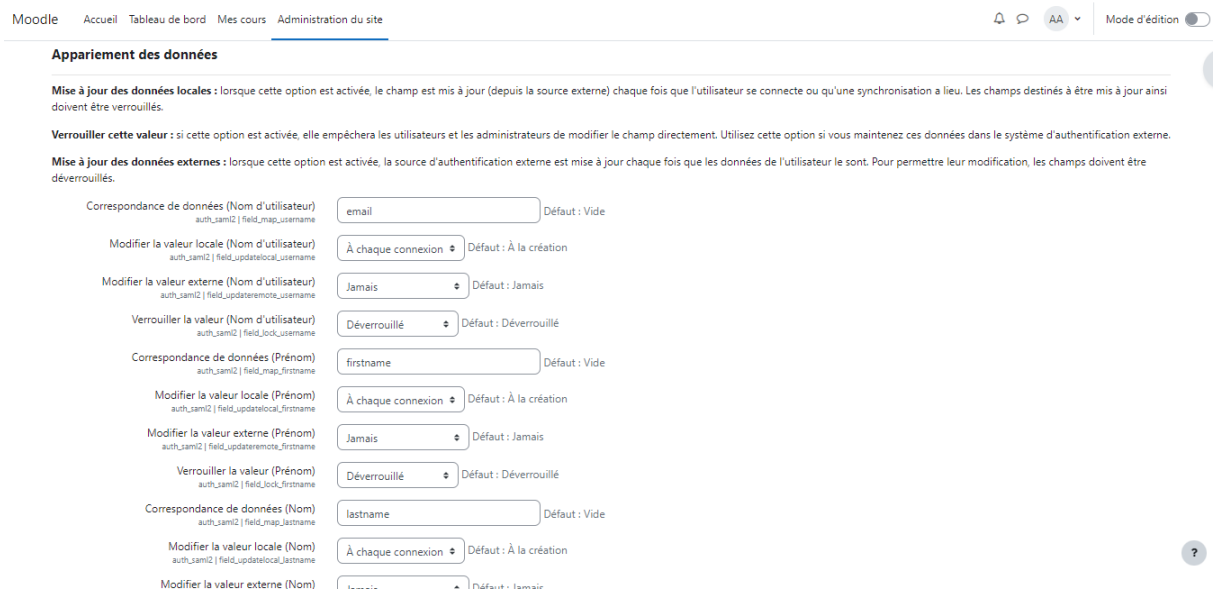


Figure 33: mappage des champs utilisateur au niveau de Moodle

Fonctionnement : lorsqu'un utilisateur souhaite se connecter à Moodle, il aura sur la page de connexion un bouton l'invitant à se connecter à partir de Google :

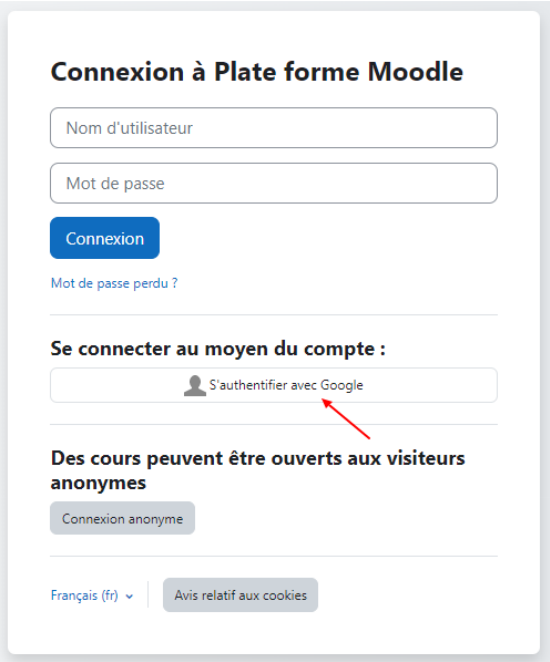


Figure 34: Bouton de connexion à partir de Google (dans le cas de l'utilisation du protocole SAML)

Comme avec OpenID Connect, lorsque l'utilisateur clique sur ce bouton, son navigateur est redirigé vers le fournisseur d'identité Google pour s'authentifier :

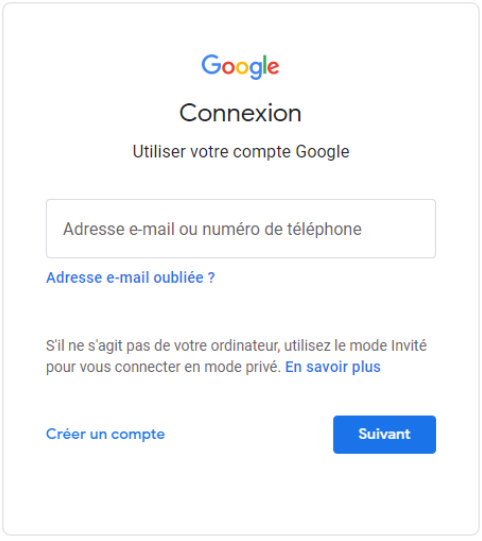


Figure 35: Authentification à partir de Google (dans le cas de l'utilisation de SAML)

Si l'utilisateur ne dispose pas de compte dans Moodle alors, un compte lui sera créé où les valeurs des champs du profil utilisateur seront ceux des attributs du fournisseur d'identité en raison du mappage d'attributs (étape 2).

3.3.3 Discussion

La mise en place d'un système d'authentification unique en utilisant Google comme fournisseur d'identité a ses avantages comme le fait que le fournisseur d'identité est déjà opérationnel, la seule chose à faire est de configurer les sites et les applications pour qu'elles utilisent Google comme système d'authentification des utilisateurs. Un autre avantage est que les administrateurs IT n'auront pas à se soucier des opérations de maintenance (matériel ou logiciel), de mise à jour, etc. Toutes ces taches sont réalisées par Google. L'inconvénient majeure de cette solution que ce soit avec OpenID Connect ou SAML est que le provisionnement et le déprovisionnement des utilisateurs par Google qu'ils soient manuels ou automatiques ne sont pas disponibles pour tout type d'application mais uniquement pour certaines applications Cloud indiquées dans [42].

C'est-à-dire que la création des comptes des utilisateurs dans les bases de données des sites se fait la première fois que ces utilisateurs se connectent à ces sites à partir de Google ce qui est un inconvénient pour certains sites ou applications comme par exemple Moodle, en effet, sans le provisionnement des utilisateurs, un enseignant ou un administrateur devra attendre la connexion des utilisateurs pour pouvoir les inscrire aux cours. L'absence du déprovisionnement est aussi un problème du fait que les comptes des utilisateurs qui quittent l'université seront toujours présents dans les bases de données des sites et des applications. Les administrateurs IT devront alors les supprimer eux même.

3.4 Proposition 2 : mise en œuvre d'un système SSO local avec OpenID Connect

Dans cette solution nous allons mettre en place un système d'authentification unique en local en utilisant OpenID Connect. Nous avons choisi OpenID Connect car premièrement, c'est un standard pouvant être implémenté sur différentes plateformes sans causer de problèmes d'interopérabilité. Deuxièmement, OpenID Connect est un protocole SSO inter domaines, c'est-à-dire qu'il peut prendre en charge les applications internes et externes. Les protocoles Kerberos et SAML vérifient également ces deux caractéristiques. Cependant, OpenID Connect est plus facile à utiliser et fonctionne dans un environnement REST. En plus de cela, il permet de prendre en charge tout type d'applications (web, mobile, etc).

Mise en œuvre de la solution :

Il existe plusieurs librairies open source ainsi que des services et des outils propriétaires qui ont implémenté ce protocole [43]. Pour ce mémoire, nous avons choisi MITREid Connect qui est une implémentation java et open source du protocole OpenID Connect (fournisseur d'identité et fournisseur de service (client)) [44]. OpenID Connect peut être configuré pour utiliser différentes sources de données MySQL, PostgreSQL, etc. Pour notre cas, nous avons choisi d'utiliser un annuaire LDAP[7]car il est idéal lorsqu'il est nécessaire que les données soient gérées, stockées et accessibles de manière centralisée via des méthodes standardisée [45]. De plus, la structure arborescente d'un annuaire LDAP facilite la représentation d'une organisation [15].

[7] LDAP (Lightweight Directory Access Protocol) est un protocole léger d'accès aux services d'annuaire. Un annuaire est une base de données spéciale dont les entrées sont sauvegardées d'une manière hiérarchique. Il est souvent utilisé pour l'authentification et le stockage d'informations sur les utilisateurs, les groupes et les applications [45].

Nous avons mis en place notre système sur une machine de test sur laquelle nous avons installé le système d'exploitation Debian 11. Nous avons mis en place un annuaire LDAP en installant OpenLDAP[8] :

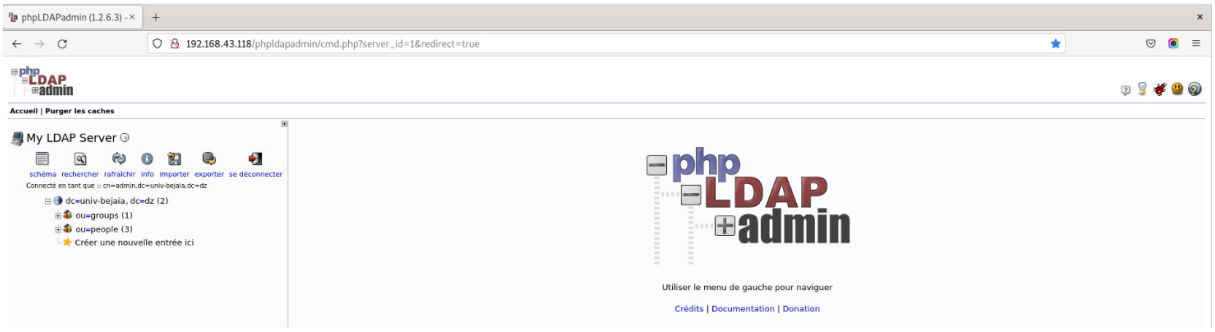


Figure 36: mise en place d'un annuaire LDAP

Nous avons également installé MITREid Connect et nous l'avons configuré pour qu'il utilise notre annuaire LDAP comme source de données, ce qui nous a permis de mettre en place un fournisseur d'identité OpenID Connect:

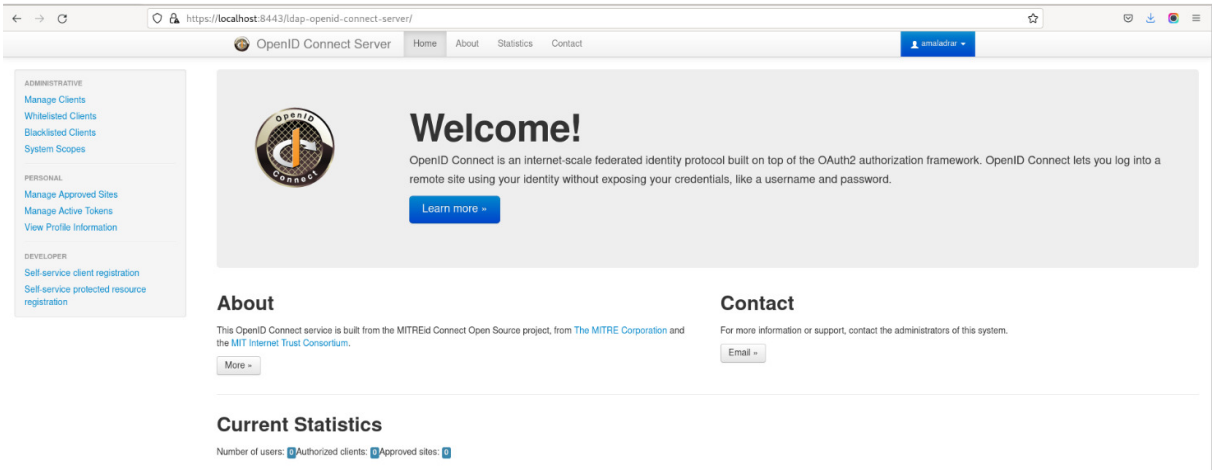


Figure 37: mise en place d'OpenID Connect

Pour tester le fonctionnement de notre système, nous avons installé une plateforme Moodle :

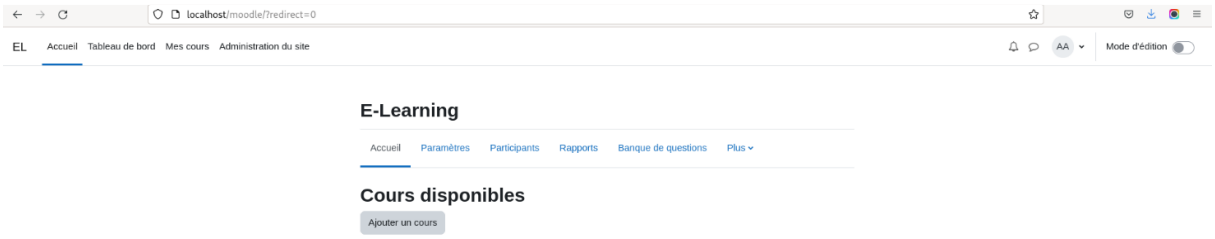


Figure 38: mise en place d'une Plate forme Moodle

OpenLDAP : https://www.openldap.org/

Nous avons suivi les étapes ci-dessous pour enregistrer Moodle (le client) au niveau de notre IdP afin d'obtenir l'ID client et le secret client:

Nous avons cliqué sur le bouton « New client » pour ajouter un nouveau client,

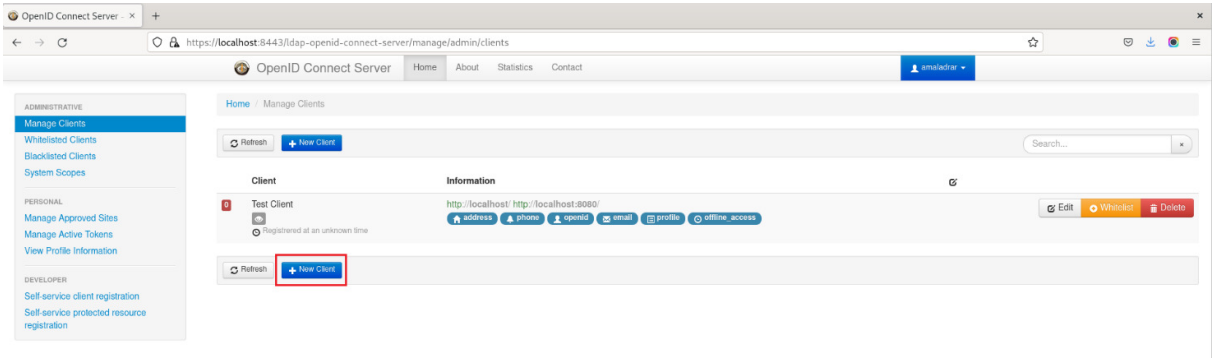


Figure 39: Enregistrement de Moodle au niveau de l'IdP

Ensuite sur la page qui s'affiche, nous avons renseigné les paramètres correspondants au client (Moodle), le plus important, étant l'URL de redirection de Moodle (http://localhost/moodle/admin/oauth2callback.php) vers laquelle l'utilisateur sera redirigé une fois authentifié auprès de l'IdP :

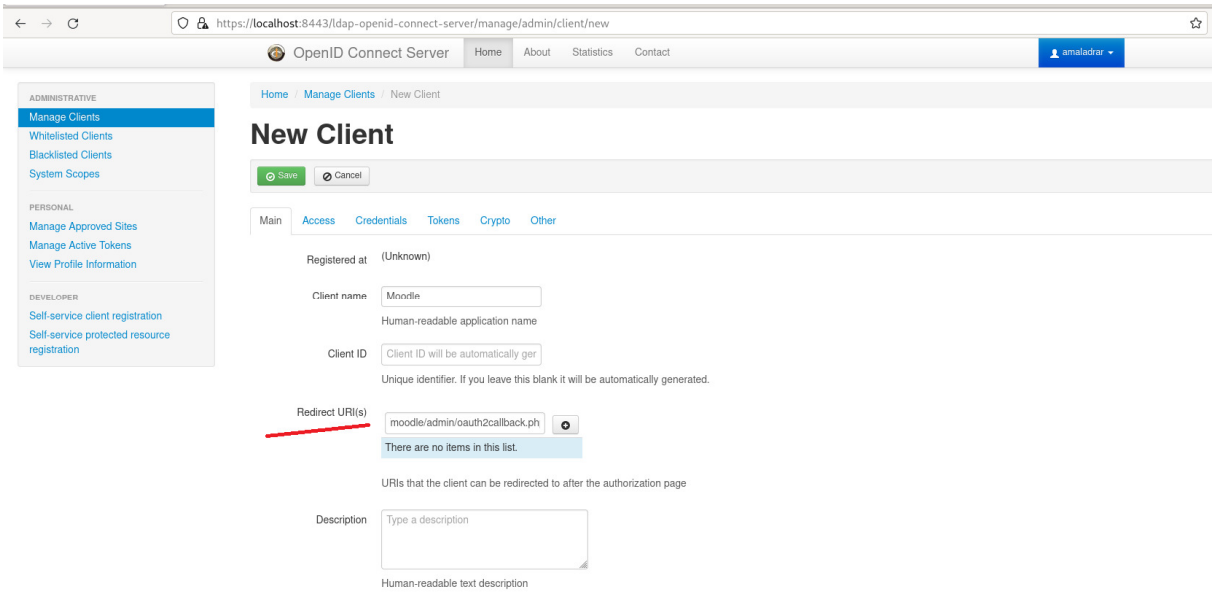


Figure 40: Configuration des paramètres du client (Moodle) dans l'IdP

Une fois les paramètres enregistrés, OpenID Connect nous a généré l'ID Client et le secret Client, nous aurons besoin de ces données lors de la configuration de l'IdP dans Moodle sachant que le secret ne doit être connu que de l'IdP et du client :

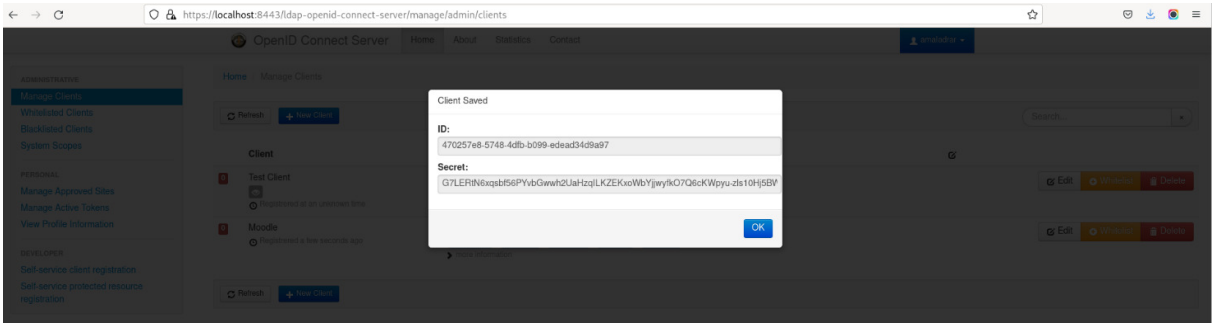


Figure 41: Génération de l'ID client et du secret client pour Moodle

On clique sur « Ok » et on constate l'ajout du client dans l'IdP :

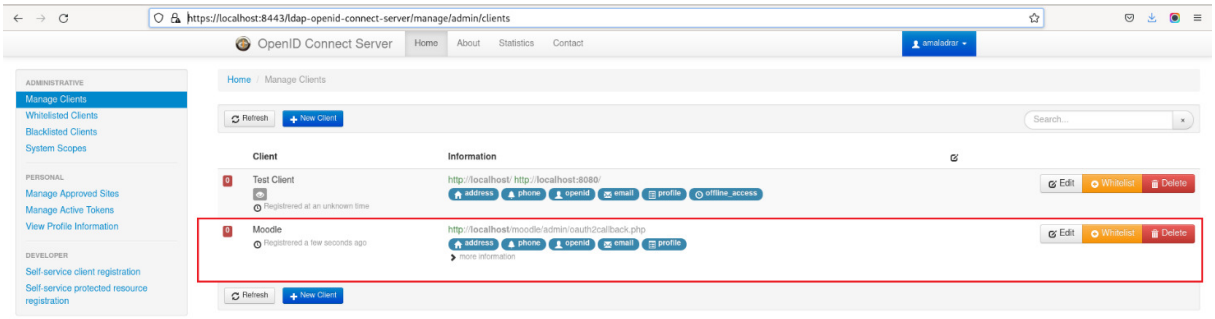


Figure 42: aperçu du client Moodle une fois enregistré dans l'IdP

Maintenant il est nécessaire de configurer Moodle pour qu'il utilise l'IdP OpenID Connect comme fournisseur d'identité. Pour cela, nous avons suivi les étapes suivantes :

Comme dans la première proposition, nous allons utiliser le plugin OAuth2 intégré dans Moodle pour configurer notre IdP pour cela sur la page de configuration des services OAuth 2 on clique sur le bouton « Personnalisé » :

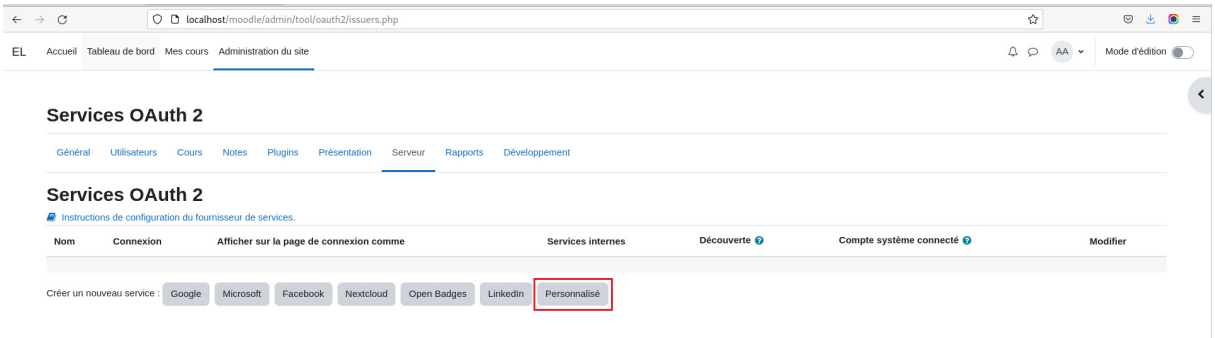


Figure 43: Ajout de l'IdP OpenID Connect dans Moodle

Sur la page qui s'affiche, on renseigne les paramètres ID client, le secret client et l'adresse de base de l'IdP qui est : https://localhost:8443/ldap-openid-connect-server/ grâce à cette adresse, Moodle va générer automatiquement les points de terminaisons de l'IdP :

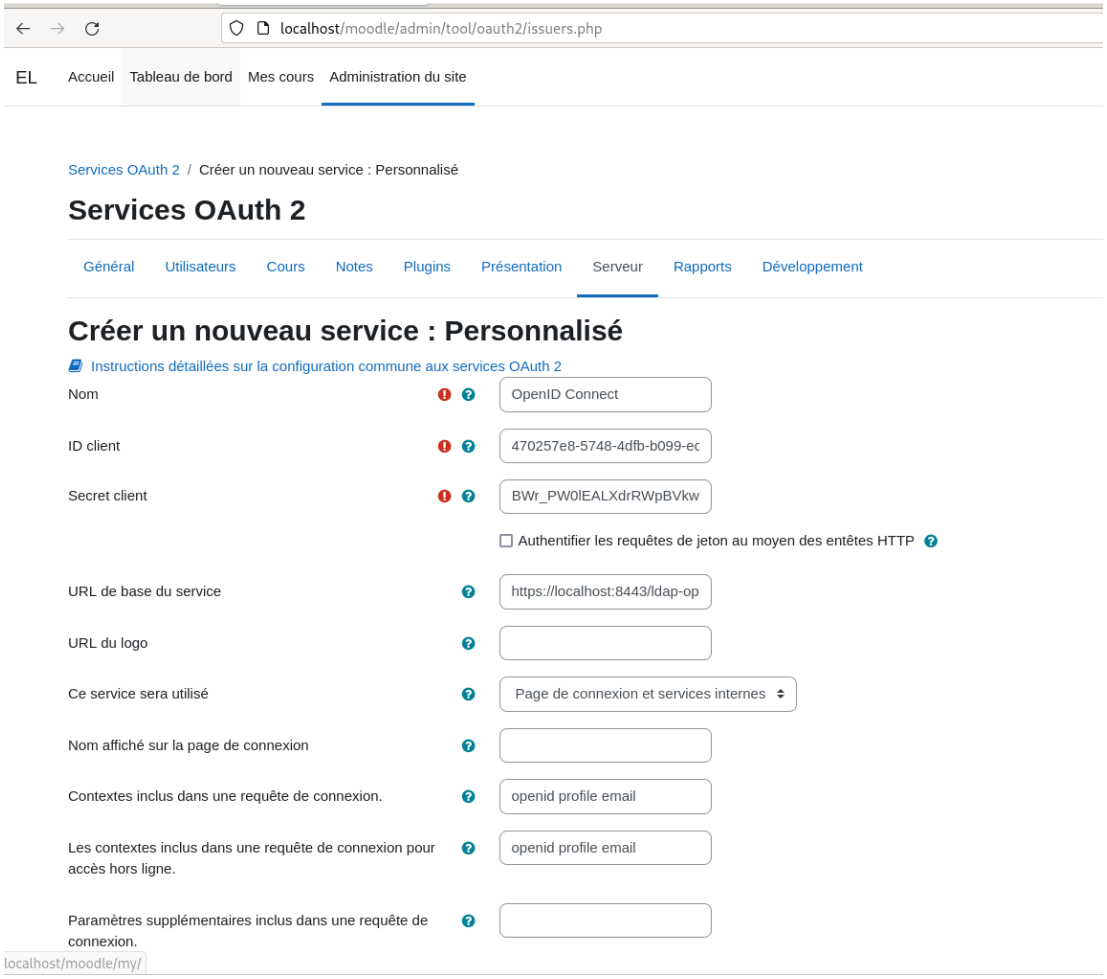


Figure 44: Paramétrage de l'IdP OpenID Connect dans Moodle

Une fois les paramètres enregistrés, on constate l'ajout de l'IdP OpenID Connetc :

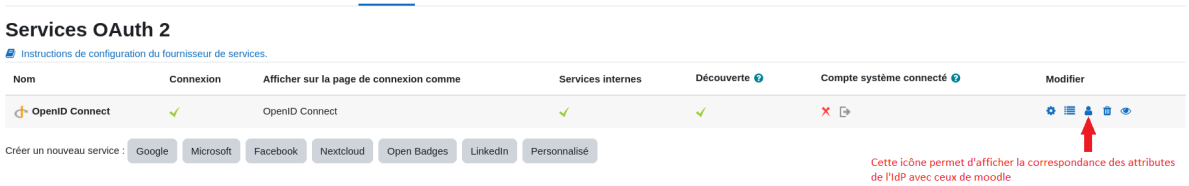


Figure 45: Aperçu de l'IdP OpenID Connect une fois ajouté dans Moodle

Moodle ajoute automatiquement le mappage des attributs de l'IdP OpenID Connect avec ceux de Moodle :

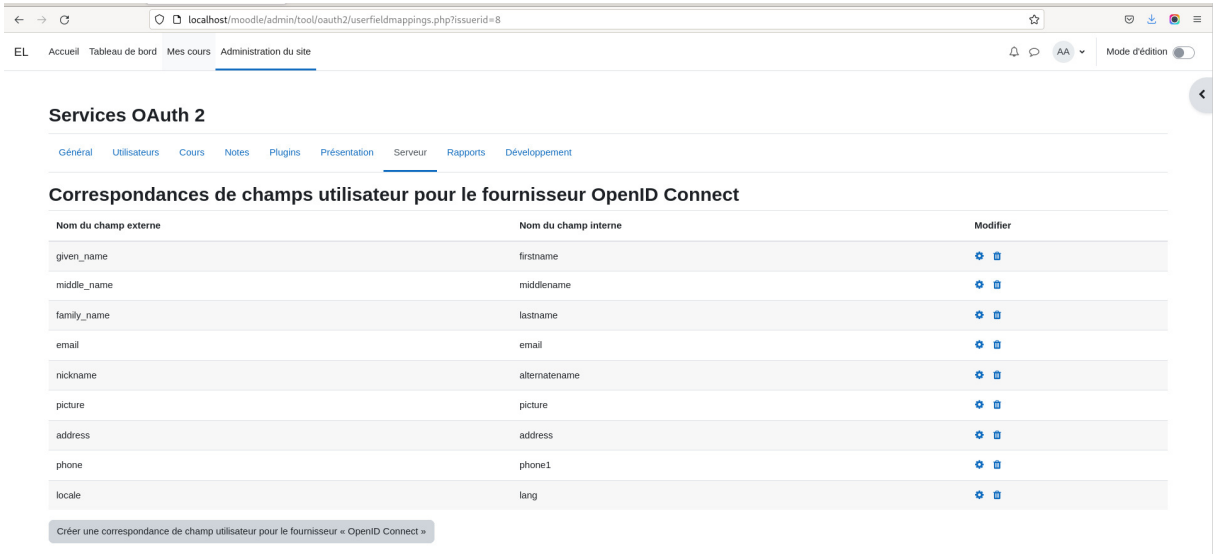


Figure 46: Mappage des attributs de l'IdP OpenID Connect avec ceux de Moodle

Après cette configuration, un bouton de connexion nous permettant de nous authentifier à partir d'OpenID Connect s'affiche sur la page de connexion de Moodle:

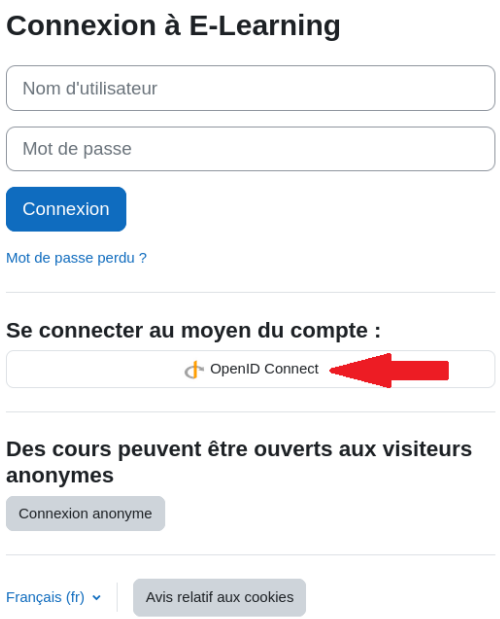


Figure 47: Bouton de connexion avec OpenID Connect ajouté à Moodle

Lorsqu'on clique sur ce bouton, le navigateur nous dirige vers l'IdP OpenID Connect afin de nous authentifier :

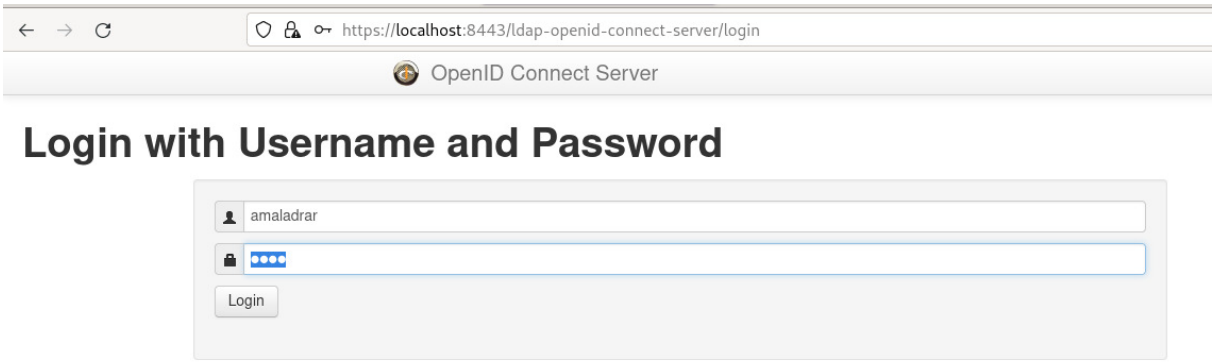


Figure 48: Saisi des identifiants dans l'IdP OpenID Connect

On doit saisir un nom d'utilisateur et un mot de passe valides et cliquer sur le bouton « Login », le navigateur nous dirige alors vers une page ou on doit donner notre consentement à l'IdP pour qu'il envoi à Moodle les informations affichées sur cette page, ces informations correspondent aux scopes (portées) requis par Moodle (openid, profile et email):

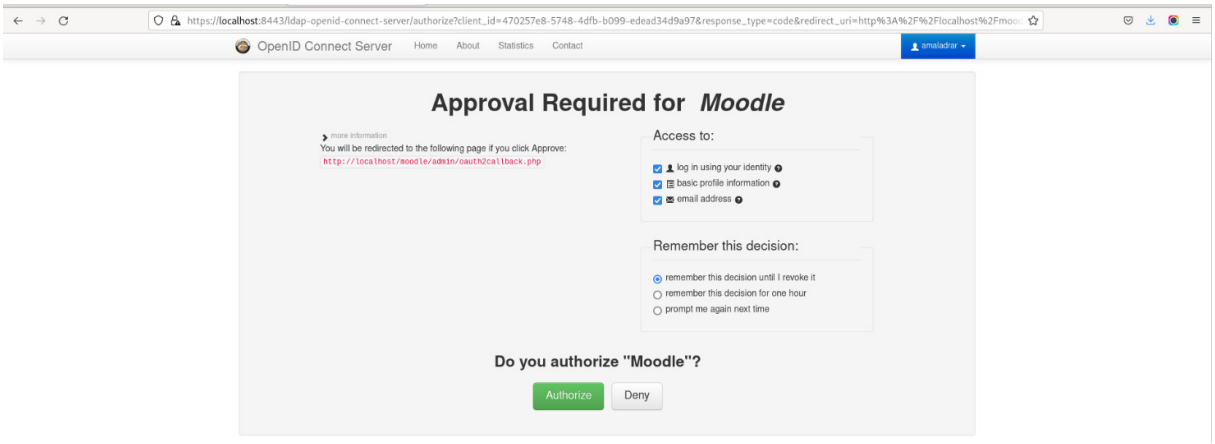


Figure 49: écran de consentement

Lorsqu'on donne notre autorisation, le navigateur nous dirige à nouveau vers Moodle ou on apparait authentifié :

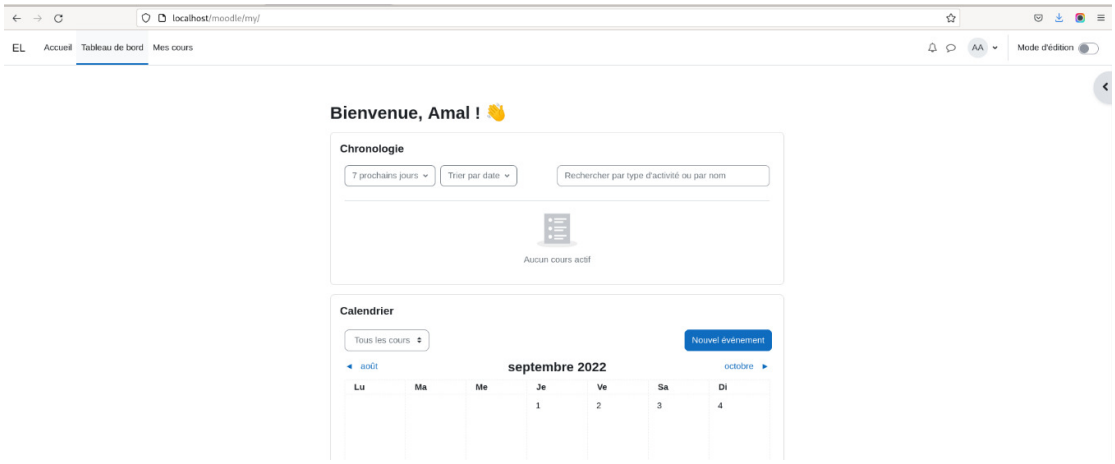


Figure 50: Connexion à Moodle réussie

Si on configure notre IdP OpenID Connect comme fournisseur d'identité dans un autre site ou dans une autre application alors, on accédera directement à ce site ou à cette application sans se connecter à nouveau.

3.4.1 Discussion

Avec cette solution, il suffit d'alimenter l'annuaire avec des comptes utilisateurs et de connecter tous les services internes et externes à ce système. Les utilisateurs pourront alors se connecter une seule fois à ce système puis accéder à tous les services. Contrairement à la première solution, ce système est capable de provisionner/déprovisionner les services avec les comptes des utilisateurs. Cependant, le système étant mis en œuvre en interne, l'équipe IT devra prendre en charge toutes les taches de maintenance, mise à jour et sauvegarde de ce système.

3.5 Conclusion

Nous avons présenté dans ce chapitre deux solutions de mise en œuvre de l'authentification unique. Chacune à ses avantages et ses inconvénients. À notre avis, la première solution est plus adaptée à des services où le provisionnement/déprovisionnement n'est pas essentielle ou bien l'utiliser comme un moyen d'authentification supplémentaire. La deuxième solution est mieux adaptée pour les organisations où il est nécessaire d'avoir le contrôle total du système de gestion des identités qui est le cas de l'université de Béjaia et des universités en générale.

Conclusion générale et perspectives

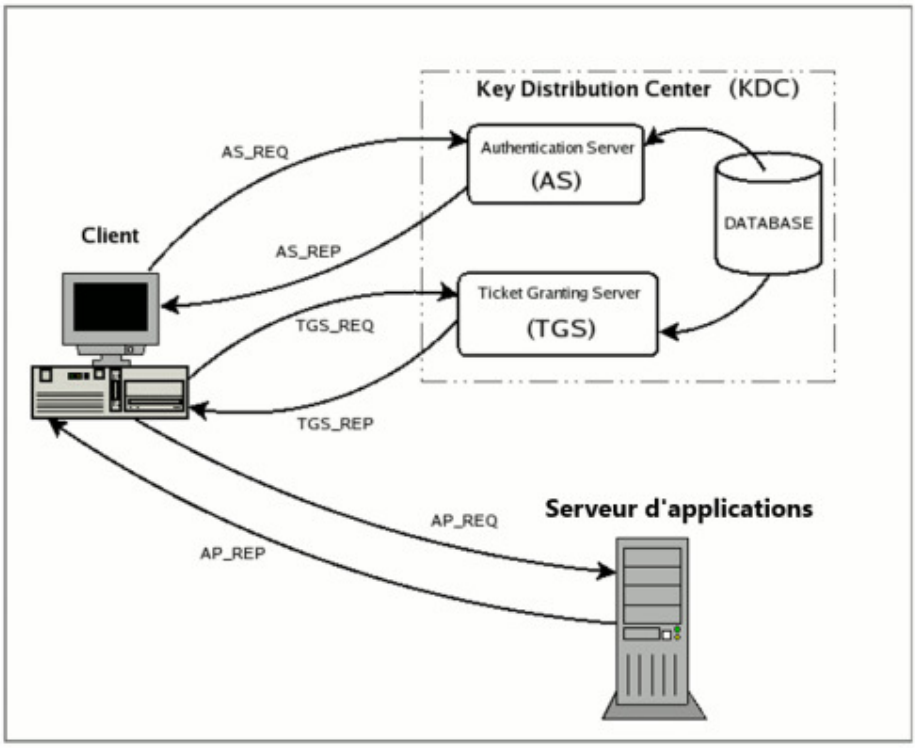
Nous avons donné dans le chapitre 1 quelques définitions de certains concepts qui sont l'identité, l'authentification, le Cloud computing et la gestion des identités et des accès car la compréhension de ces concepts est essentielle pour l'assimilation des autres chapitres de ce mémoire. L'identité est l'ensemble d'informations permettant de représenter une entité. L'authentification est le processus de vérification de l'identité. La gestion des identités et des accès est la pratique qui consiste à gérer les identités numériques des entités et leur accès aux ressources de l'organisation. Ces ressources peuvent être internes ou externes à l'organisation d'où l'introduction du Cloud computing qui permet de fournir aux individus et aux organisations des services à la demande et évolutifs. L'authentification est une brique essentielle d'un système IAM (Identity and Access Management) car elle permet d'assurer que seules les personnes autorisées pourront accéder aux ressources du système, elle s'appuie pour cela sur un ou plusieurs facteurs d'authentification. L'authentification unique est le mécanisme permettant à un individu de s'authentifier une seule fois pour accéder aux ressources de l'organisation.

Dans le chapitre 2, nous avons parlé des objectifs de l'authentification unique et de son principe de fonctionnement. Nous avons ensuite décrit le fonctionnement de quelques protocoles d'implémentation du SSO qui sont Kerberos, SAML et OpenID Connect. Kerberos se base sur la cryptographie symétrique pour l'authentification mutuelle et le chiffrement des communications. Le protocole SAML se base sur XML et permet l'échange d'informations d'authentification et d'autorisation entre entités sous forme d'assertions. Le protocole OpenID Connect fourni une couche d'authentification en dessus du protocole d'autorisation OAuth2.0. Après ça, nous avons résumé quelques travaux menés sur l'utilisation de ces protocoles pour l'authentification et le SSO suivi d'une discussion. Nous avons terminé ce chapitre en citant les avantages et inconvénients de l'implémentation d'un système d'authentification unique au sein d'une organisation. Vu les avantages qu'il offre, le système SSO mérite d'être implémenté et ce, en dépit de quelques inconvénients qu'il peut présenter car ceux-ci peuvent être traités ou atténués.

Dans le dernier chapitre (chapitre 3), nous avons parlé de la nécessité de la mise en place d'un système SSO au sein de l'université de Béjaia et nous avons proposé deux solutions. La première consiste à utiliser Google comme fournisseur d'identités. Malgré ses avantages, l'absence de la fonctionnalité de provisionnement des comptes des utilisateurs rend cette solution inadaptée pour tout type de site ou application. La deuxième solution que nous avons proposée consiste à la mise en place d'un système d'authentification unique local (sur site), cette solution a l'avantage d'avoir le contrôle total sur la gestion des identités dont le provisionnement et le déprovisionnement des sites et des applications avec les identités des utilisateurs. Nous avons mis en œuvre ce système en utilisant le protocole OpenID Connect et un annuaire LDAP pour centraliser les données des utilisateurs et nous l'avons testée sur une plateforme Moodle de test. Comme perspectives, nous souhaitons proposer un schéma de l'annuaire LDAP qui répondra aux besoins de notre université et de l'université algérienne en générale et aussi, proposer une solution de gestion de ce système.

Annexe A

Ci-dessous, l'explication détaillée de chaque message échangé entre le client, le KDC et le serveur d'applications lors de l'authentification.Sachant que les parenthèses () veulent dire que le message n'est pas crypté alors que les accolades { } signifientque le message est crypté avec une clé secrète indiquée juste après l'accolade fermante:



Messages échangés entre le client, le KDC et le serveur d'applications lors de l'authentification

Requête destinée au serveur d'authentification (AS_REQ)

Dans cette phase, appelée requête d'authentification initiale, le client demande au KDC (plus précisément à l'AS) un Ticket Granting Ticket (TGT). La requête n'est pas chiffrée et ressemble à ceci :

$$AS\_REQ = (Principal_{Client}, Principal_{Service}, IP\_list , Lifetime)$$

Où :

$Principal_{Client}$ est le principal associé à l'utilisateur cherchant à s'authentifier.

$Principal_{Service}$ est le principal associé au service pour lequel le ticket est demandé. Dans ce cas $Principal_{Service}$ correspond au principal associé au serveur d'octroi de tickets (TGS) étant donné que pour y accéder il faut présenter le TGT. Le TGS peut donc être considéré comme un serveur d'application qui assure comme service l'émission de tickets de service.
IP_listest une liste d'adresses IP qui indiquent la machine où il est possible d'utiliser le ticket qui sera émis.

Réponse du serveur d'authentification (AS_REP)

Lorsque le service d'authentification (AS) reçoit la requête précédente, l'AS vérifie si $Principal_{Client}$ et $Principal_{Service}$ existent dans la base de données du KDC.Dans ce cas, $Principal_{Service}$ existe forcement car il s'agit du principal associé au TGS. Donc, il suffit de vérifier si $Principal_{Client}$ existe, si ce n'est pas le cas, un message d'erreur est envoyé au client, sinon le Serveur d'Authentification (AS) traite la réponse comme suit :

- Il crée aléatoirement une clé de session qui sera le secret partagé entre le client et le TGS. Appelons la SK_{TGS} ;
- Il crée le Ticket Granting Ticket (TGT) qui contient le principal du client, le principal du TGS, la liste d'adresses IP (ces trois premières informations sont copiés à partir de la requête AS_REQ), la date et l'heure (du KDC) au format timestamp (c'est la date et l'heure auxquelles la validité du ticket commence), le lifetime (il s'agit de la durée de vie maximale du ticket) et enfin la clé de session. SK_{TGS}. Le Ticket Granting Ticket (TGT) se présente donc comme suit :

$$TGT = (Principal_{Client} , Principal_{TGS} , IP\_list , Timestamp , Lifetime , SK_{TGS})$$

- Il génère et envoie la réponse AS_REP contenant : le TGT précédemment créé chiffré à l'aide de la clé secrète du TGS (appelons-la K_{TGS}). Le principal associé au TGS, le timestamp, le lifetime et la clé de sessionSK_{TGS}, tous chiffrés à l'aide de la clé secrète de l'utilisateur (appelons-la K_{User}). En résumé:

$$AS\_REP = \{Principal_{TGS} ,Timestamp ,Lifetime , SK_{TGS} \}K_{User} \{ TGT \}K_{TGS}$$

Lorsque le client reçoit la réponse AS_REP, il demande à l'utilisateur d'entrer son mot de passe. Puis, génère la clé secrète de l'utilisateur K_{User} (la clé secrète est dérivée du mot de passe de l'utilisateur). Avec la clé secrète résultante, le client tente de déchiffrer la partie du message chiffrée par le KDC à l'aide de la clé secrète de l'utilisateur stockée dans la base de données. Si l'utilisateur est vraiment celui qu'il prétend être et a donc saisi le bon mot de passe, l'opération de décryptage sera réussie. La clé de session pourra alors être extraite. Le TGT reste crypté (car il est crypté à l'aide de la clé secrète du TGS qui n'est pas connue duclient) et est stocké dans le cache des informations d'identification de l'utilisateur.

Requête au service Ticket Granting Server (TGS_REQ)

A ce stade, l'utilisateur a déjà prouvé qu'il est celui qu'il prétend être il dispose donc dans son cache d'un TGT et une clé de session SK_{TGS}, il veut maintenant accéder au service mais n'a pas encore de ticket adapté. Il envoie alors une requête (TGS_REQ) au service d'octroi de tickets (TGS) en la construisant comme suit :

- Il crée un authentificateur avec le principal associé à l'utilisateur, le timestamp de la machine cliente, le tout crypté avec la clé de session partagée avec le TGS, c'est-à-dire :

$$Authenticator = \{Principal_{Client} , Timestamp \}SK_{TGS}$$

- Crée une requête TGS_REQ contenant : le principal associé au service pour lequel le ticket est nécessaire(dans ce cas, il s'agit du principal associé au serveur d'application) et le lifetime sachant que les deux ne sont pas chiffrés. Le Ticket Granting Ticket qui est déjà chiffré avec la clé du TGS ; et l'authentificateur qui vient d'être créé. En résumé:

$$TGS\_REQ = (Principal_{Service} , Lifetime , Authenticator) \{ TGT \}K_{TGS}$$

Réponse du TGS (TGS_REP)

Lorsque le TGS reçoit la requête TGS_REQ, il vérifie d'abord que le principal associé au service demandé ($Principal_{Service}$) existe dans la base de données du KDC, s'il existe, alors, il ouvre le TGT à l'aide de sa clé secrète et extrait la clé de session (SK_{TGS}) avec laquelle il déchiffre l'authentificateur. Pour que le ticket de service soit émis, il vérifie que les conditions suivantes sont vérifiées :

- le TGT n'a pas expiré ;
- Le $Principal_{Client}$ présent dans l'authentificateur correspond à celui présent dans le TGT ;
- L'authentificateur n'est pas présent dans le cache et n'a pas expiré ;
- Si IP_list n'est pas nul il vérifie que l'adresse IP source du paquet de requête (TGS_REQ) fait partie de celles contenues dans la liste ;

Les conditions précédemment si elles sont vérifiées, elles prouvent que le TGT appartient bien à l'utilisateur qui a fait la demande et donc le TGS commence à préparer la réponse comme suit:

- Il crée aléatoirement une clé de session qui sera le secret partagé entre le client et le service. Appelons la $SK_{Service}$;
- Il crée le ticket de service$T_{Service}$, mettant à l'intérieur le principal associé à l'utilisateur, le principal associé au service, la liste des adresses IP, la date et l'heure (du KDC) au format timestamp, le lifetime (au minimum entre le lifetime du TGT et celui associé au $Principal_{Service}$) et enfin la clé de session $SK_{Service}$.

$$T_{Service} = (Principal_{Client} , Principal_{Service} , IP\_list , Timestamp , Lifetime , SK_{Service})$$

- Il envoie la réponse TGS_REP contenant : le ticket créé précédemment ($T_{Service}$), chiffré à l'aide de la clé secrète du service (appelons-la $K_{Service}$). Le principal associé au service, timestamp, lifetime et la nouvelle clé de session$SK_{Service}$, le tout chiffré à l'aide de la clé de session extraite du TGT. En résumé:

$$TGS\_REP = \{ Principal_{Service} , Timestamp , Lifetime , SK_{Service} \}SK_{TGS} \{ T_{Service} \}K_{Service}$$

Lorsque le client reçoit la réponse, ayant dans le cache des informations d'identification la clé de session SK_{TGS}, il peut déchiffrer la partie du message contenant l'autre clé de session $SK_{Service}$ et la mémoriser avec le ticket de service $T_{Service}$ qui, lui, cependant, reste chiffré (car il est crypté avec la clé secrète du service qui n'est pas connue du client).

Requête pour l'application (AP_REQ)

Le client, disposant des informations d'identification nécessaires pour accéder au service (c'est-à-dire le ticket $T_{Service}$ et la clé de session associée $SK_{Service}$), peut demander au serveur d'application l'accès à la ressource via un message AP_REQ

Il faut garder à l'esprit que, contrairement aux messages précédents (les requêtes et les réponses) où le KDC était impliqué, la requête AP_REQ n'est pas standard, mais varie selon les applications. Ainsi, le programmeur d'application a pour tâche d'établir la stratégie avec laquelle le client utilisera ses informations d'identification pour prouver son identité au serveur. Cependant, on peut considérer la stratégie suivante à titre d'exemple.

Le client crée un authentificateur contenant le principal associé à l'utilisateur et le timestamp, les deux chiffrés avec la clé de session $SK_{Service}$ qu'il partage avec le serveur d'application:

$$Authenticator = \{ Principal_{Client} , Timestamp \}SK_{Service}$$

Il crée une requête contenant le ticket de service $T_{Service}$ qui est chiffré avec clé secrète du service et l'authentificateur qui vient d'être créé:

$$AP\_REQ = Authenticator \{ T_{Service} \}K_{Service}$$

Lorsque le serveur d'application reçoit la requête AP_REQ, il déchiffre le ticket $T_{Service}$ en utilisant la clé secrète du service et extrait la clé de session $SK_{Service}$ qu'il utilise pour déchiffrer l'authentificateur. Pour établir que l'utilisateur demandeur est authentique et ainsi accorder l'accès au service, le serveur vérifie les conditions suivantes :

- le ticket n'a pas expiré ;
- Le $Principal_{Client}$ présent dans l'authentificateur correspond à celui présent dans le ticket ;
- L'authentificateur n'est pas présent dans le cache et n'a pas expiré ;
- Si IP_list (extraite du ticket) n'est pas vide, il vérifie que l'adresse IP source de la requête (AP_REQ) fait partie de celles contenues dans la liste ;

Bibliographie

[1] I.Milenković, O.Latinović et D.Simić, USING KERBEROS PROTOCOL FOR SINGLE SIGN-ON IN IDENTITY MANAGEMENT SYSTEMS, Journal of Information Technology and Applications. 3 (2013) 27-33.

[2] K.D. LEWIS et J.E. LEWIS, Web Single Sign-On Authentication using SAML, IJCSI International Journal of Computer Science Issues. 2 (2009) 41-48.

[3] S. Boonkrong, Methods and Threats of Authentication, in: S. McDermott, L. Berendson, R. Fernando (Eds.), Authentication and Access Control: Practical Cryptography Methods and Tools, Apress, Thailand, 2020, pp.45-70.

[4] Harry, Guillaume (2018, 11 janvier). IAM - Gestion des identités et des accès: concepts et états de l'art. HAL science ouverte - La connaissance libre et partagée. https://hal.archives-ouvertes.fr/hal-00879556.

[5] RSA seguridad. amazon. Consulté le 15 aout 2022. https://www.amazon.com/-/es/seguridad-sid700-6-60-Securid-sid700-llavero/dp/B0017TMCQI

[6] N. M. Karie, V. R. Kebande, R. A. Ikuesan, M. Sookhak et H. S. Venter, Hardening SAML by Integrating SSO and Multi-Factor Authentication (MFA) in the Cloud, 3rd International Conference on Networking, Information Systems & Security, Association for Computing MachineryNew YorkNYUnited States, At: March, 2020.

[7] E. Sturrus, O. Kulikova, Identity and Access Management, in: S. Murugesan, I. Bojanova (Eds.), Encyclopedia of Cloud Computing, Wiley-IEEE Press, United Kingdom, 2016, pp. 396-405.

[8] Qu'est-ce que la gestion des identités et des accès (IAM)?. CISCO. Consulté le 04 juin 2022. https://www.cisco.com/c/fr_ca/products/security/identity-services-engine/what-is-identity-access-management.html

[9] Mell, Peter et Grance, Timothy (Septembre 2011). The NIST Definition of Cloud computing. NIST the National Institute of Standards and Technology. Consulté le 14 juin 2022. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[10] Z. KARTIT. Contribution à la sécurité du Cloud Computing : Application des algorithmes de chiffrement pour sécuriser les données dans le Cloud Storage. THÈSE DE DOCTORAT, Univertité Mohammed V Faculté des sciences Rabat, 22 Octobre 2016.

[11] Qu'est-ce que l'IAM ? . IBM. Consulté le 04 juin 2022. https://www.ibm.com/fr-fr/topics/identity-access-management

[12] Identity as a Service (IDaaS). PingIdentity. Consulté le 19 juin 2022. https://www.pingidentity.com/en/resources/content-library/articles/identity-as-a-service-idaas.html

[13] A Brief Introduction to XACML (2003, 14 march). OASIS OPEN. Consulté le 16 aout 2022. https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html

[14] T. Bazaz et A. Khalique, A Review on Single Sign on Enabling Technologies and Protocols, International Journal of Computer Applications. 151 (2016) 0975 – 8887.

[15] N. Dagli, M. Deorukhar, S. Sawant, K. Upadhyaya et N. Shaikh, Implementation of Single Sign on (SSO) for College websites, International Research Journal of Engineering and Technology (IRJET). 07(05) (2020) 1285-1289.

[16] L, Bastien (2019, 11 july). SSO (Single Sign-On) définition de l'authentification unique. Lebigdata. Consulté le 20 mars 2022. https://www.lebigdata.fr/single-sign-on-sso-definition

[17] NGUESSAN, Noel (2014, 02 fevrier). Classement des systèmes d\'authentification (social login). arobasenet.com. Consulté le 14 aout 2022. https://arobasenet.com/2014/02/classement-des-systemes-dauthentification-social-login.html

[18] V. Radha et D. Hitha Reddy, A Survey on Single Sign-On Techniques, Elsevier. 4 (2012) 134-139.

[19] I. Nongbri, P. Hadem et S. Chettri, A Survey on Single Sign-On, International Journal of Creative Research Thoughts (IJCRT). 6(2) (2018) 595-602.

[20] Kerberos: The Network Authentication Protocol. Massachusetts Institute of Technology (MIT). Consulté le 28 avril 2022. https://web.mit.edu/kerberos

[21] How Does Kerberos Work?. Massachusetts Institute of Technology (MIT). Consulté le 28 avril 2022. https://web.mit.edu/kerberos/kfw-4.1/kfw-4.1/kfw-4.1-help/html/how_kerberos_works.htm

[22]Ricciardi, Fulvio (11/27/2007) KERBEROS PROTOCOL TUTORIAL. MIT Kerberos Consortium. Consulté le 29 avril 2022. https://www.kerberos.org/software/tutorial.html.

[23] Security Assertion Markup Language (SAML) V2.0 Technical Overview (2008, 25 mars). OASIS. Consulté le 25 mars 2008. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[24] SAML Overview. https://developer.okta.com/docs/concepts/saml/. A modifier

[25] Cassam Chenaï, Rémi et Levillain, Olivier (2018, juillet). OpenID Connect : présentation du protocole et étude de l'attaque Broken End-User Authentication. diamond connect la documentation technique des pros de l'IT. Consulté le 08 avril 2022. https://connect.ed-diamond.com/MISC/misc-098/openid-connect-presentation-du-protocole-et-etude-de-l-attaque-broken-end-user-authentication

[26] Welcome to OpenID Connect. OpenID. Consulté le 08 avril 2022. https://openid.net/connect/

[27] Hardt,Dick (2012, October). The OAuth 2.0 Authorization Framework. IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc6749.

[28] SAKIMURA, Nat (2018, 10 Août). Secret of Authorization Code. https://youtu.be/xcT6OCbI77k

[29] Sakimura,Nat;Bradley,John; Jones,Michael B; de Medeiros,Breno et Mortimore,Chuck (2014, 08 Novembre). OpenID Connect Core 1.0 incorporating errata set 1. OpenID. Consulté le 23 Avril 2022. https://openid.net/specs/openid-connect-core-1_0.html

[30] Sakimura,Nat;Bradley,John; Jones,Michael B; de Medeiros,Breno et Mortimore,Chuck (2020, 24 07). OpenID Connect Basic Client Implementer's Guide 1.0 - draft 40. OpenID. Consulté le 08 aout 2022. https://openid.net/specs/openid-connect-basic-1_0.html

[31] Soni, Rakesh. (2019, 12 November). 7 Benefits of Single Sign-On (SSO) and Why Your Business Needs It. loginradius. Consulté le 20 mars 2022. https://www.loginradius.com/blog/start-with-identity/benefits-single-sign-on-sso/

[32] A. Sonakiya et M. Jain, Kerberos based Enhanced Authentication Protocol for Cloud Computing Environment, International Journal for Rapid Research in Engineering Technology & Applied Science. 6(6) (2019) 2455-4723.

[33]Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 (2005, 15 March). OASIS. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[34] T. Sujanani et S. Vinod, Implementation of OpenId connect and OAuth 2.0 to create SSO for educational institutes, International Journal of Engineering & Technology. 7 (2.6) (2018) 153-157.

[35] A. Purwinark, W. Hardyanto et M A. Adhi, Implementation of google single sign On (SSO) in the library management system, Journal of Physics: Conference Series. (2021).

[36] Presentations & Videos. OpenID. Conulté le 12 aout 2022. https://openid.net/foundation/presentations-videos/

[37] I. Indu, P.M. Rubesh Anand et V. Bhaskar, Identity and access management in cloud environment: Mechanisms and challenges, Engineering Science and Technology, an International Journal. 21 (2018) 574–588.

[38] OAuth 2 services (2021, 18 mai). Moodle. Consulté le 18 aout 2022. https://docs.moodle.org/400/en/OAuth_2_services

[39] OAuth 2 Google service (2021, 18 fevrier). Moodle. Consulté le 18 aout 2022. https://docs.moodle.org/400/en/OAuth_2_Google_service

[40] SAML2 Single sign on. Moodle. https://moodle.org/plugins/auth_saml2

[41] Set up your own custom SAML application.https://support.google.com/a/answer/6087519?hl=en

[42] About automated user provisioning. https://support.google.com/a/answer/7681608?hl=en

[43] Libraries, Products, and Tools. OpenID. https://openid.net/developers/libraries/

[44] MITREid Connect. https://github.com/mitreid-connect/

[45] OpenLDAP. . https://www.openldap.org/doc/admin24/intro.html

Résumé

Avec le développement des technologies informatiques, les organisations offrent de plus en plus à leurs utilisateurs des services nécessitant une authentification et accessibles par internet. Ces services peuvent être internes aux organisations ou fournies par des fournisseurs de services Cloud. Le nombre d'identifiants et de mots de passe que les utilisateurs devront mémoriser augmente avec l'augmentation du nombre de services mis à leur disposition. Dans ces cas-là, la mise en place d'un système de gestion des identités et des accès est importante pour gérer les identités des utilisateurs et leur accès aux différents services. L'authentification unique (Single Sign-On ou SSO) est une fonctionnalité importante d'un système de gestion des identités et des accès, elle permet aux utilisateurs de s'authentifier une seule fois avec un seul ensemble d'informations d'identification pour accéder à tous les services mis à leur disposition, leur évitant ainsi de mémoriser plusieurs identifiants et mots de passes et de s'authentifier séparément à chaque service. Dans ce mémoire, nous avons parlé de l'importance de la mise en place d'un système d'authentification unique au sein d'une organisation, nous avons ainsi décrit le fonctionnement de certains protocoles d'implémentation du SSO qui sont Kerberos, SAML et OpenID Connect et nous avons proposé une solution d'implémentation d'un système SSO pour l'université de Béjaia.

Mots-clés: IAM, Single Sing-On, authentification, Kerberos, SAML, OpenID Connect, Cloud computing

Abstract

With the development of computer technologies, organizations are increasingly offering their users services requiring authentication and accessible via the Internet.These services can be internal to organizations or provided by cloud service providers.The number of usernames and passwords that users will need to remember increases with the increase in the number of services available to them. In these cases, the implementation of an identity and access management system is important to manage the identities of the users and their access to the various services.Single Sign-On (SSO) is an important feature of an identity and access management system, it allows users to authenticate once with a single set of credentials to access all the services made available to them. with SSO users will no longer need to remember multiple usernames and passwords and they will no longer need to log in separately to each service.In this master's thesis, we have emphasized the importance of setting up a single sing on system within an organization, we have thus described the operation of certain SSO protocols which are Kerberos, SAML and OpenID Connect and we have proposed a solution for implementing an SSO system for the University of Béjaia.

Keywords: IAM, Single Sign-On, Authentication, Kerberos, SAML, OpenID Connect, Cloud Computing.