République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Béjaïa
Faculté des sciences exactes
Département d'Informatique

Mémoire de Master
Option : Administration et sécurité des réseaux

Thème

Authentification unique avec CAS et LDAP

Présenté par :
ADJAOUD Yougourthen
KEHOUL Tarek

Devant le jury composé de :

Président : Mr. A.ACHROUFENE
Examinateur : Mr. R.OUZZEGGANE
Examinatrice : Mlle. K.GHIDOUCHE
Promotteur : Dr. A.R.SIDER

2011 - 2012

Dédicaces

Nous dédions ce travail à :

Nos chers parents.

Nos chers frères et sœurs.

Nos familles.

Tous nos amis(es).

Tarek et Yougourthen

Remerciements

Nous remercions avant tous DIEU tout puissant qui nous a donné la force, le courage et la volonté pour réaliser ce travail.

Un grand merci à nos familles surtout nos parents pour leurs encouragements et leurs suivis avec patience tout au long de notre projet.

Nous tenons à remercier vivement notre promoteur Dr A.SIDER d'avoir accepté de nous guider tout au long de ce travail.

Nous remercions également nos très chers amis(es), camarades et tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Enfin, nous tenons à remercier les membres de jury qui ont accepté de juger notre travail.

Merci à tous.

Table des matières

Table des figures

Glossaire

AAA Authentication Authorization Accounting.
ACL Access Control Lists.
ANSI American National Standards Institute.
ASCII American Standard Code for Information interchanges.

CAS Central Authentification Service.
CPU Central Processing Unit.

DES Data Encrypting System.
DNS Domain Name Service.

FTP File Transfer Protocol.
FAI Fournisseur d'Accès à Internet.

HMAC Hash Message Authentification Code.
HTTP Hyper Text Transfert Protocol.
HTTPS Hyper Text Transfert Protocol Secure.

IBM International Business Machines.
IEEE Institute of Electrical and Electronics Engineers.
IETF Internet Engineering Task Force.
IP Internet Protocol.
IPSec Internet Protocol Secure.
ISO International Standard Organization.

JDK Java Developement Kit.

LDAP Lightweight Directory Access Protocol.
LDIF Lightweight Data Interchange Format.

MIT Massachusetts of Technology.
MD5 Message Digest version 5.

NAS Network Access Server.
NIST National Institute of Technology.
NSA National Security Agency.

OCSP Online Certificate Status Protocol.
OSI Open Systems Interconnection.

PGT Proxy Granting Ticket.
PING Packet INternet Groper.
PKI Public Key Infrastructure.
PPP Protocole Point à Point.
PT Proxy Ticket.

RADIUS Remote Access Dial In User Service.
RSA Rivest Shamir Adelman.

SA Serveur d'authentification.
SEC Secure Electronic Transaction.
SHA-1 Secure Hash Algorithm version 1.
SI Système d'information.
SMTP Simple Mail Transfer Protocol.
SOAP Simple Object Access Protocol.
SSL Secure Socket Layer.
SSO Single Sign On.

TGT Ticket Granting Ticket.
TGC Ticket Granting Cookie.
TSL Transport Socket Layer.
TGS Ticket Granting Service.

UIT Union International des Télécommunications

Introduction générale

De nos jours, la sécurité informatique est une grande préoccupation des administrateurs de systèmes informatiques et les utilisateurs de ces derniers qui se soucient de leurs informations personnelles et les conséquences que peuvent entrainer des actions effectuées sous leurs identités. En effet, plus le service offert est important ou/et convoité par un grand nombre d'utilisateurs, plus il est exposé à des éventuelles attaques menées par des intrus au système. Pour minimiser ce risque, un service offert est généralement précédé d'une phase d'authentification imposée à tous ces utilisateurs enregistrés afin de distinguer les intrus.

Cependant, la phase d'authentification matérialisée par le fameux couple (identifiant, mot de passe) devient une tâche de plus en plus pénible à gérer. D'une part, les administrateurs du système, sont notamment dépassés par le surnombre des utilisateurs dont la spécification des droits d'accès reste à adapter pour chaque service. D'autre part, pour les utilisateurs eux même, à qui on exige de faire plusieurs étapes pour chaque service (remplir des formulaires d'identité pour s'enregistrer auprès du service, retenir le couple identifiant/ mot de passe approprié, fournir ce dernier à chaque fois qu'il souhaite y accéder). De plus, la multiplicité des services nécessitant des authentifications multiples et répétées génère une réaction d'agacement chez les utilisateurs. Ces derniers préfèrent laisser leurs sessions ouvertes rien que pour ne pas avoir à s'authentifier une autre fois et parfois accrochent des petits papiers ou fichiers, sur leurs postes, contenants les différents identifiants et mots de passes pour ne pas les oublier, ce qui facilite la tâche des intrus.

Notre objectif est de prévoir une solution d'authentification qui trouve un compromis entre sécurité, simplicité et pratique pour les usagers des applications web hébergées au sein du réseau de notre université (A.Mira de Bejaia) qui ne cesse d'évoluer.

Pour cela, nous avons à notre disposition plusieurs solutions d'authentifications toutes basées sur un même mécanisme d'authentification SSO (Single Sign On) ou authentification unique. Un tel mécanisme nécessite la combinaison de plusieurs composants, notamment un annuaire (LDAP) pour contenir l'ensemble des utilisateurs et unifier l'authentification des applications, un serveur d'authentification unique pour synchroniser l'accès aux applications et un protocole de sécurisation des échanges (SSL).

Notre mémoire est structuré en deux parties :

La première partie constituée de deux chapitres faisant l'objet d'une approche théorique sur les notions de base et les concepts de la mise en œuvre.

Le premier chapitre traite les notions et les aspects élémentaires de la sécurité, notamment les notions cryptographiques qui sont la base de tous système de sécurité. Ce chapitre nous introduit aussi au domaine de l'authentification et son utilisation pour le contrôle d'accès au réseau.

Le second chapitre traite le vif du projet ; il a pour but de présenter le fonctionnement des systèmes d'authentification unifiée avec LDAP et des systèmes d'authentification unique avec comme exemple une solution largement utilisée qui est CAS. Nous aurons aussi dans ce chapitre un petit aperçu sur les autres solutions d'authentification unique existantes.

La seconde partie du mémoire concerne la mise en place de ces concepts, qui est l'objectif du chapitre trois, il décrit les étapes mises en place afin de tester la solution retenue pour la gestion des authentifications unique des applications web de l'université de Bejaia. Cela pour, une éventuelle mise en place sur le terrain.

Enfin notre mémoire s'achève par une conclusion générale résumant les grands points qui ont été abordés dans ce mémoire ainsi que quelques perspectives pour le projet.

Chapitre

I Généralités sur la sécurité

Introduction

La sécurité des informations et le besoin de les rendre confidentielles a toujours été ressentie depuis l'antiquité. Avec l'apparition des ordinateurs et la numérisation de tous types de données, la sécurité de ces dernières devient le centre d'intérêt d'une part de leur propriétaire et d'autre part la cible des individus malveillants. Néanmoins, il existe plusieurs outils offerts par la cryptographie, afin de satisfaire les critères fondamentaux de la sécurité des données et qui permettent la mise en œuvre d'une politique de sécurité capable de faire face aux attaques éventuelles.

I.1 Fondamentaux de la sécurité

Tout système de sécurité jugé bon, doit répondre à tous les principes fondamentaux de la sécurité informatique afin de ne laisser aucune faille ou ambigüité qui pourra être exploité.

1. **L'authentification** est née du besoin de s'assurer de l'identité d'un utilisateur avant de lui donner l'accès à certaines données ou à des actions précises. A ne pas confondre avec l'identification qui consiste à reconnaitre quelqu'un en se basant sur un critère. Un exemple de la vie réelle illustre la différence. En effet, une personne est facilement reconnaissable par son apparence physique et par conséquent, on peut facilement être trempé par une personne imposteur qui a pris l'apparence de la vrai et le seul moyen pour faire la différence est de lui poser des questions que seul la vrai personne et vous connaissez les réponses.

   Donc l'authentification est l'identification plus la vérification de l'identité et pour cela les systèmes informatiques utilisent plusieurs moyens : un couple identifiant/-mot de passe, certificat numérique, système de questions/réponses, carte à puce, système biométrique, etc [5].

2. **La non-répudiation** associée à l'authentification, permet de garder une empreinte de toutes les actions qu'un utilisateur a effectué dans un système et ses données. En d'autre terme, après s'être authentifié un utilisateur ne pourra nier les actes qu'il a accomplis. On parle alors dans ce fondamental de signature numérique, ce par rapprochement a la signature de documents administratifs dans la vie réelle [5].

3. **L'habilitation :** un système sécurisé doit pouvoir affecter des privilèges pour chaque utilisateur, c'est-à-dire que le système doit gérer les utilisateurs en leurs donnant l'accès à accomplir des tâches en fonction de leurs statuts (administrateur, cadre, simple utilisateur, etc.).

4. **La confidentialité :** consiste à empêchait l'accès aux informations par des personnes non habilitées ou malintentionnées.

5. **L'intégrité :** terme décrivant le moyen de s'assurer que les informations d'origines ne sont pas altérées que se soit accidentellement ou par malveillance lors d'un échange.

6. **La disponibilité :** une application doit répondre à tout instant aux requêtes de l'utilisateur donc elle doit avoir un mécanisme résistant contre les pannes accidentelles et les attaques visant la saturation ou l'endommagement du service (déni de service).

7. **La traçabilité :** ce dernier fondamental, appelé aussi « preuve », il a toute son importance. En effet, en stockant des informations sur toutes les interactions des utilisateurs avec les applications, il permet la détection des attaques ou des dysfonctionnements. Ces traces peuvent servir à établir des responsabilités ou à déterminer la provenance d'une attaque [5].

I.2 Notions sur la Cryptographie

Afin d'assurer les services de confidentialité, d'intégrité et d'authentification dans un système informatique. On a besoin de connaitre certaines notions de cryptographie ainsi que les principaux systèmes de chiffrement.

Chiffrement : (*chiffrer* ou *crypter*) c'est l'action consistant à rendre les informations incompréhensible en l'absence de la méthode de *déchiffrement* qui est l'action inverse du chiffrement.

Cryptogramme : est un message chiffré qui est écrit en caractères secrets, en code, en langage chiffré.

Cryptographie : est la science qui consiste à représenter n'import-quelle information (texte, son, image ou vidéo) en la rendant inintelligible à ceux ne possédant pas la *clé* de déchiffrement ou les capacités pour la de déchiffrer.

Clé de chiffrement ou/et de déchiffrement est l'outil, la méthode ou l'indice utilisé pour chiffrer un message ou pour déchiffrer un cryptogramme. Elle varie d'un système de chiffrement à un autre. Dans certaine méthode de chiffrement, la clé de chiffrement diffère de la clé de déchiffrement mais dans d'autres elle reste la même.

I.2.1 Cryptographie moderne

Après la numérisation des données et l'apparition de la transmission sans fil (la radio), il a fallu concevoir des systèmes cryptographiques numériques. En effet, les échanges de messages par ondes radio par voie aérienne étaient inévitablement interceptés par l'ennemi. Cela-dit la confidentialité des informations n'été plus garantie.

On ne traite plus le message à chiffrer caractère par caractère comme autrefois, mais on transforme les caractères en binaire en utilisant le code ASCII *(American Standard Code for Information Interchange)*. Il est à savoir que l'ASCII code les caractères sur 7 bits.

Les premiers algorithmes se basaient sur les principes de permutation et de substitution directement sur le code binaire. Ce type de chiffrement a été concrétisé par la fameuse machine ENIGMA utilisée par les nazies lors de la seconde guère mondial. Cependant, l'évolution des calculateurs a permis de rendre ces algorithmes de chiffrement plus performant et plus robuste comme l'illustré l'algorithme LUCIFER qui fut inventé par IBM au début des années 70.

Les algorithmes de cryptographie numérique, ont apporté de nouvelles notions à la cryptographie notamment sur la méthode de chiffrement qui se base beaucoup plus sur problèmes mathématiques connus mais aussi sur le type de sa clé dont l'algorithme est classé selon ce dernier.

I.2.2 Système de chiffrement symétrique

I.2.2.1 Principe

Pour chiffrer ou déchiffrer des données, il faut avoir à sa disposition l'algorithme de chiffrement et sa clé. Si les deux opérations (chiffrement et déchiffrement) s'effectuent avec la même clé, le système est alors dit symétrique. Cette clé doit être gardée secrète par l'émetteur et le récepteur. Afin de rendre l'échange d'informations confidentiel comme l'explique la figure I.1 [2].

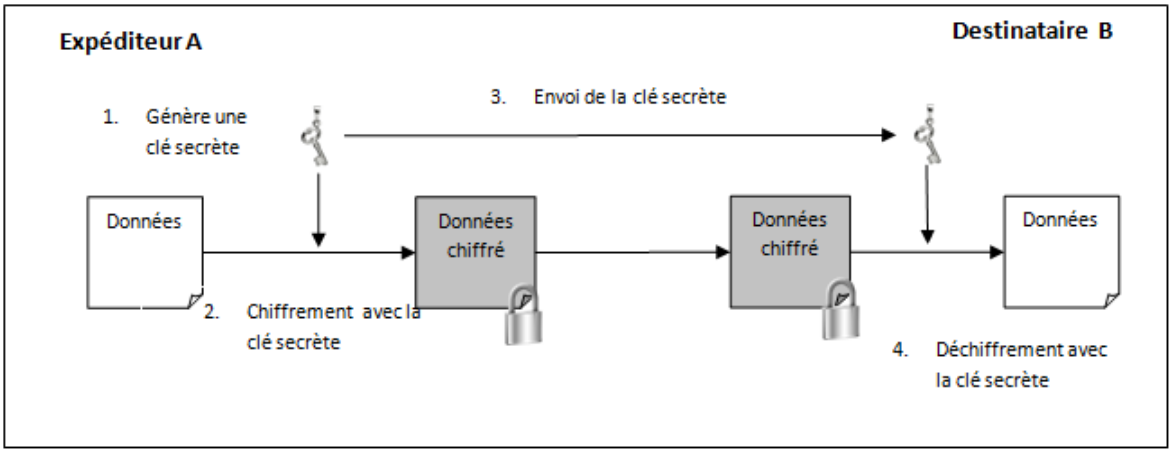


FIGURE I.1 – Chiffrement symétrique.

Comme l'indique la figure I.1, ce type d'algorithme nécessite un échange préalable de la clé secrète ce qui fait sa faiblesse. On remarque aussi que chaque entité possède autant de clés secrètes qu'elle a d'interlocuteurs. Ce qui devient vite impossible à réaliser et inadapté aux communications multipartenaires et aux services d'Internet.

I.2.2.2 DES (Data Encryption Standard)

Standard crée en 1976 à partir de Lucifer par la NSA (National Security Agency). Approuvé par le NIST (National Institute of Standard and Technology) en 1977, puis normalisé par l'ANSI sous le nom d'ANSI X3.92. Le DES chiffre les données par blocs de 64 bits avec une clé de 56 bits. Cet algorithme est largement répandu et utilisé pour les applications financières. Il est souvent mis en œuvre en un mode dit « chainage de blocs » ou le chiffrement d'un bloc dépend du précédent [2].

Il existe plusieurs variantes issus de l'algorithme DES, triple DES, DESX, GDES, RDES utilisant des clés plus longues, rendant ainsi l'algorithme plus puissant. En effet, comme l'indique son nom le Triple DES revient à l'utilisation de l'algorithme pour chiffrer trois fois de suite ce qui fait une clé effective de 168 bits.

I.2.2.3 Diffie - Hellman

Conçu par Whitfie Diffie et Martin Hellman en 1976, cet algorithme est venu afin de résoudre le problème majeur des systèmes de chiffrement symétrique qui est l'échange de la clé secrète. L'algorithme est aussi dit asymétrique car il est basé sur deux clés différentes pour effectuer l'échange d'une clé secrète tout en gardant sa confidentialité. Le chiffrement tire sa sécurité d'un problème mathématique (le logarithme discret) [5].

I.2.3 Le système de chiffrement asymétrique

Les systèmes de chiffrement symétriques posaient problème de gestion des clés et ne garantissaient certains critères de sécurité (intégrité, authentification et non-répudiation). C'est pour cette raison qu'un autre type de chiffrement dit asymétrique (ou à clé publique) a été mis en œuvre pour couvrir tous les besoins de la sécurité. RSA est l'algorithme le plus utilisé de ce type de chiffrement en raison de sa robustesse.

I.2.3.1 Principe

Un système de chiffrement asymétrique est basé sur l'utilisation d'un couple unique de deux clés complémentaires, calculées l'une par rapport à l'autre.

Chaque entité possède un couple de clé constitué d'une *clé publique* et d'une *clé privée*. Seul la clé publique peut être connue des autres entités, tandis que la clé privée reste confidentielle est un secret que l'entité propriétaire doit garder.

Pour envoyer un message à un destinataire, il faut d'abord connaitre sa clé publique, chiffrer le message avec celle-ci, pour enfin envoyer le message chiffré. Lors de la réception, le destinataire applique l'algorithme de déchiffrement en utilisant sa clé privée. De cette

manière, la confidentialité des données envoyées est garantie, de sorte que seul le vrai destinataire pourra les déchiffrer (cf.Figure I.2) [2].

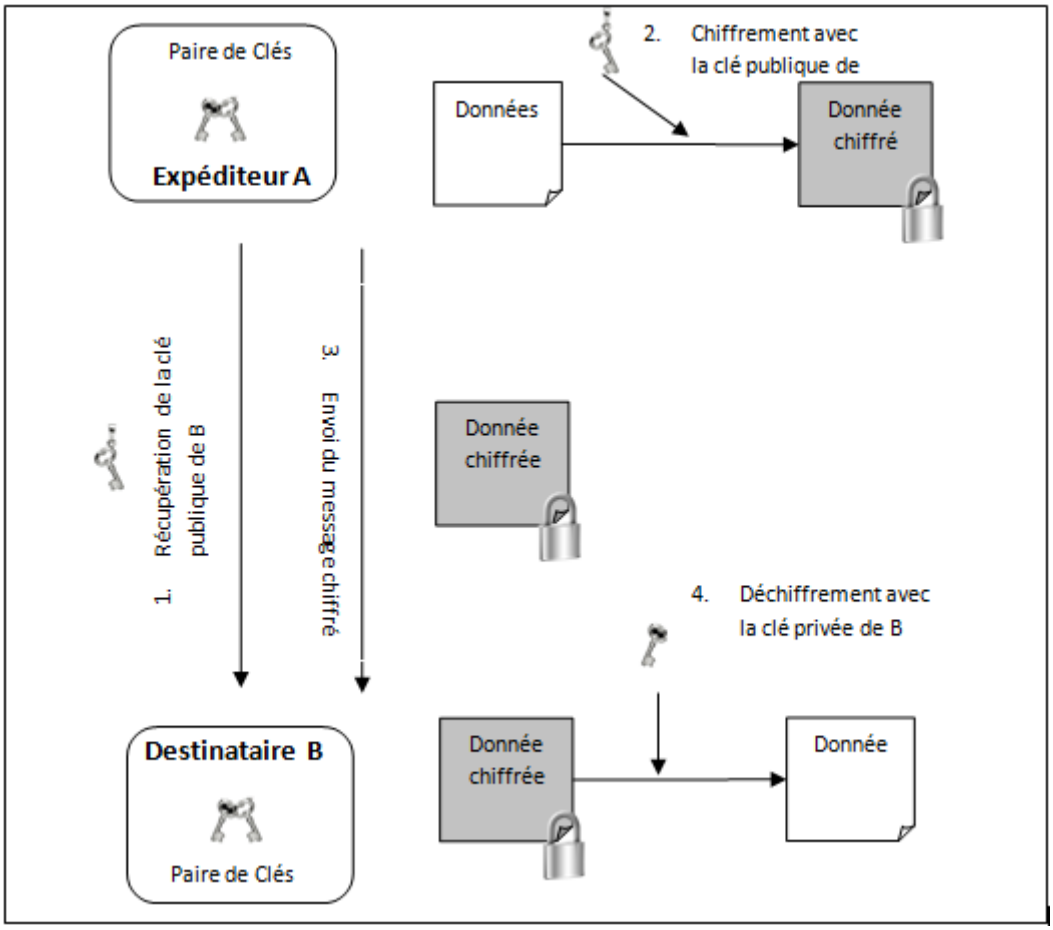


FIGURE I.2 – Chiffrement à clé publique.

Ce système offre aussi un autre service qui est la signature numérique. Elle permet de déterminer la provenance d'un message. Cependant, les systèmes de chiffrement asymétrique nécessitent une entité secondaire, intermédiaire entre les interlocuteurs, appelée autorité de certification qui effectue la gestion des clés publiques.

Si les chiffrements asymétriques présentent un niveau de sécurité très poussé et sont très répandus pour réaliser des échanges confidentiels. Par contre ils présentent des temps d'exécution relativement long les rendant moins performant lorsqu'il s'agit de chiffrer des données de grande taille.

I.2.3.2 RSA (Rivest, Shamir et Adelman)

Algorithme de cryptographie asymétrique, mis en oeuvre par Ron Rivest, Adi Shamir et Len Adelman en 1977. RSA a été breveté par le MIT (Massachusetts Institute of Technology) en 1983 aux États unis. En raison du niveau de sécurité présenté par l'algorithme notamment par les choix de la taille des clés utilisées (à savoir 512, 1024 ou 2048 bits),

il a trouvé place dans les domaines les plus sensibles à la sécurité des données, allant des simples applications web aux informations militaires classées, en passant par les systèmes bancaires.

La sécurité offerte par ce chiffrement est basé sur la complexité de factorisation des nombres premiers qui est utilisé pour générer la paire de clés. Le principe générale est le même présenté par la figure I.2.

Comme on l'a indiqué, les systèmes de chiffrement à clé publique comme RSA se focalisent sur la taille des clés pour renforcer le niveau de sécurité. Néanmoins plus la taille de la clé est longue plus les temps de chiffrement et de déchiffrement sont longs en les comparant aux temps d'exécution des systèmes symétrique. Afin de profiter des avantages des deux systèmes, on a combiné le système symétrique avec le système asymétrique. On parle alors de système *Hybride* qui combine sécurité et rapidité [10].

I.2.4 Fonction de Hachage

Lors d'un échange, les données peuvent subir des altérations que ce soit par des personnes malveillantes ou non-intentionnelles dûes au support de transmission. Pour cela la fonction de hachage est un mécanisme qui permet au destinataire de vérifier l'intégrité des données échangées.

Son principe consiste à faire accompagner la donnée envoyée par son haché (empreinte ou condensé). Cette empreinte est générée par un algorithme de hachage sur lequel les interlocuteurs se mettent d'accord. L'algorithme a comme entrée la donnée à envoyer et en sortie une chaine de caractères de taille fixe.

Il est à savoir qu'une donnée a un unique haché (en appliquant un même algorithme) et qu'une modification mineure de la donnée induit à un changement majeur dans son haché. De plus, le processus de hachage n'est pas inversible (on ne peut retrouver un message à partir de son haché). Ainsi la fonction de hachage permet de s'assurer de l'intégrité des données reçues (cf.figure I.3).

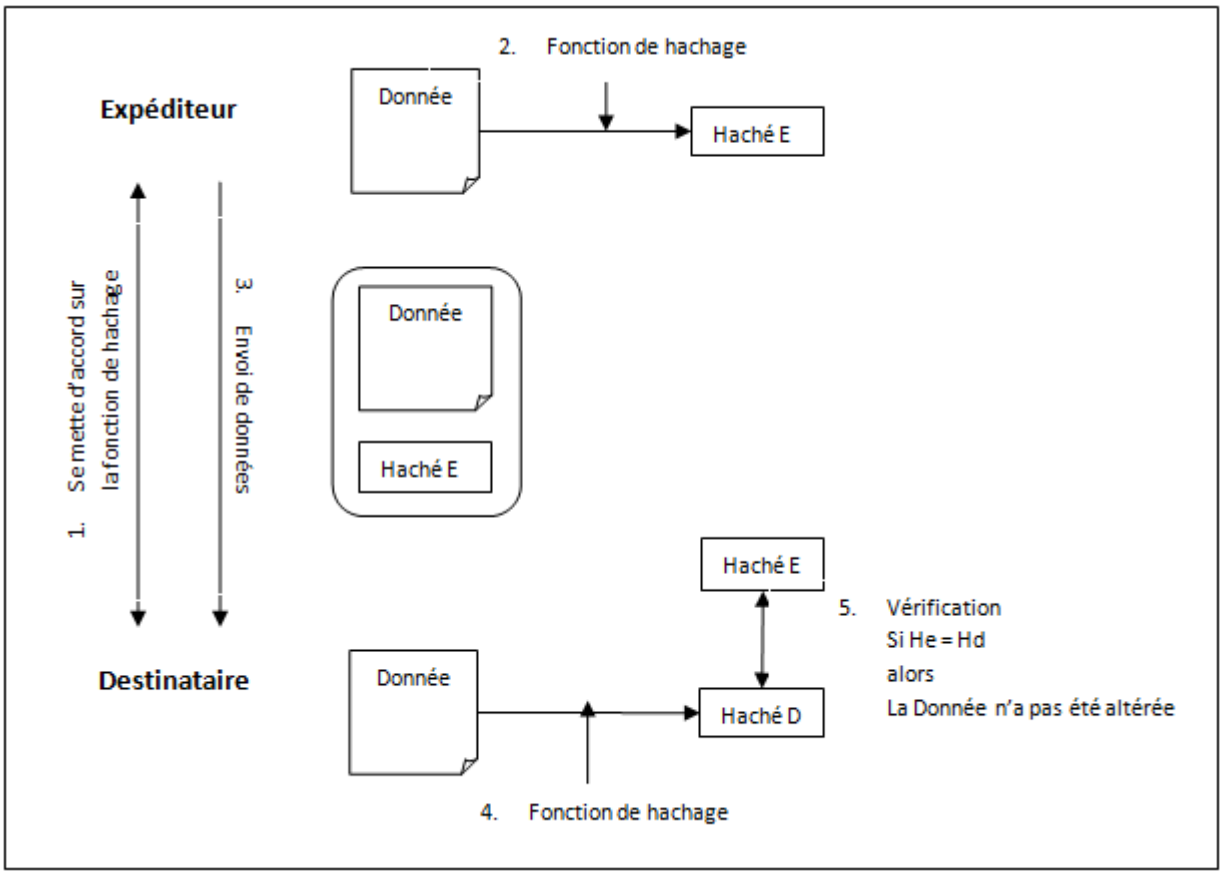


FIGURE I.3 – La fonction de hachage assurant l'intégrité des données.

Les fonctions de hachage les plus usuelles sont :

- **MD4** et **MD5**(Message Digest) furent développées par Ron Rivest. MD5 produit des hachés de 128 bits en travaillant les données originales par blocs de 512 bits.

- **SHA-1** (Secure Hash Algorithm 1), comme MD5, est basé sur MD4. Il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie. Il nécessite donc plus de ressources que MD5.

- **SHA-2** (Secure Hash Algorithm 2) est destiné à remplacer SHA-1. La différence principale réside dans la taille de haché possible : 256, 384 ou 512 bits.

- **Tiger** est une fonction de hachage cryptographique conçue par Ross Anderson et Eli Biham en 1996. Tiger fournit une empreinte sur 192 bits et des versions sur 128 et 160 bits existent aussi. Ces versions raccourcies prennent simplement les premiers bits de la signature de 192 bits.

I.2.5 Signature numérique

L'idée de la signature numérique vient de la signature des papiers qui prouve l'identité de leur auteur. Sur les documents numériques on applique l'inverse des algorithmes de chiffrements à clés publiques pour pouvoir déterminer leur expéditeur et ainsi garantir le service de *non-répudiation* dans les échanges (cf.figure I.4). Cela en utilisant la même fonction de hachage.

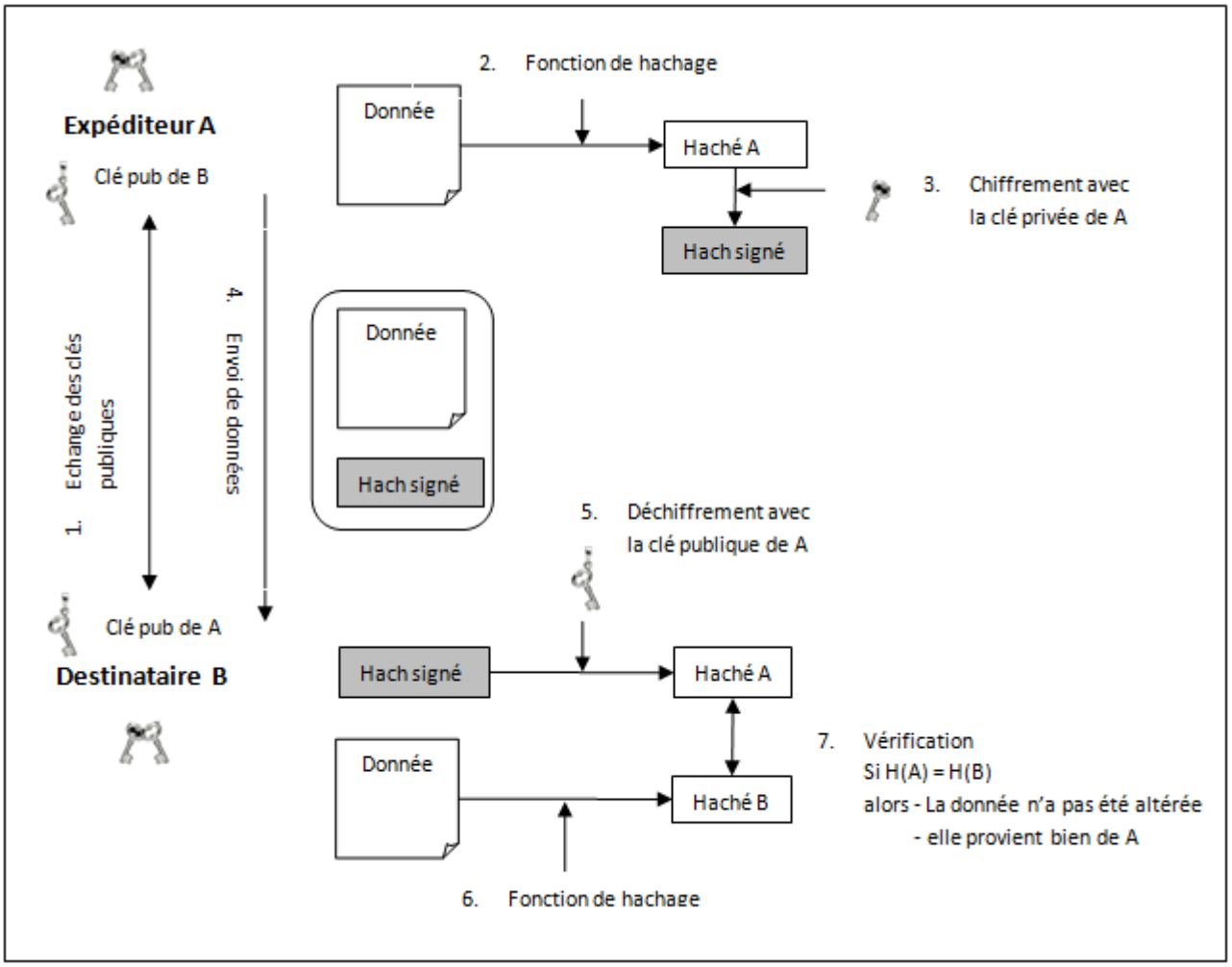


FIGURE I.4 – La signature numérique garantissant l'intégrité et la non-répudiation des données.

I.2.6 Autorité de certification

L'autorité de certification est un organisme de confiance nécessaire à toute mise en œuvre d'un système de chiffrement asymétrique. Appelé aussi PKI (*Public Key Infrastructure*), elle assure la gestion et la distribution des clés. Ce afin, d'éviter aux interlocuteurs de mémoriser l'ensemble des clés publiques de leurs correspondants ou de demander à ces derniers un échange préalable à chaque transmission.

Les principales fonctions supportées par une infrastructure de gestion de clés sont :

- La génération des couples de clés uniques (clé privée, clé publique) et leurs attributions aux entités.
- La création et la gestion des certificats numérique : signature, émission, validation, révocation et renouvellement des certificats.
- La sauvegarde des informations nécessaires à la gestion des clés : archivage de clé pour la récupération en cas de perte par son propriétaire ou en cas de demande par les autorités judiciaires.
- La diffusion des clés publiques aux entités qui les sollicites et qui sont habilitées de les avoir.
- La certification des clés publiques (par signature des certificats).

I.2.6.1 Le certificat numérique

Le certificat numérique (certificat électronique) constitue la carte d'identité numérique d'une entité (machine ou humaine) ou d'une ressource informatique à qui il appartient. Il contient entre autres, l'identification de son propriétaire, sa clé publique ainsi que l'identifiant de l'organisme qui l'a délivré.

La norme internationale X.509 élaborée l'UIT (Union International des Télécommunications) propose un cadre architectural pour la réalisation d'un service d'authentification basé sur l'usage de certificat. Dans celle-ci, une PKI délivre un certificat liant une clé publique à un nom distinct. Ce certificat est signé avec la clé privée de la PKI dont la clé publique peut être, à son tour, certifiée avec une autre signature (dans un autre certificat), formant ainsi une chaine de confiance [2].

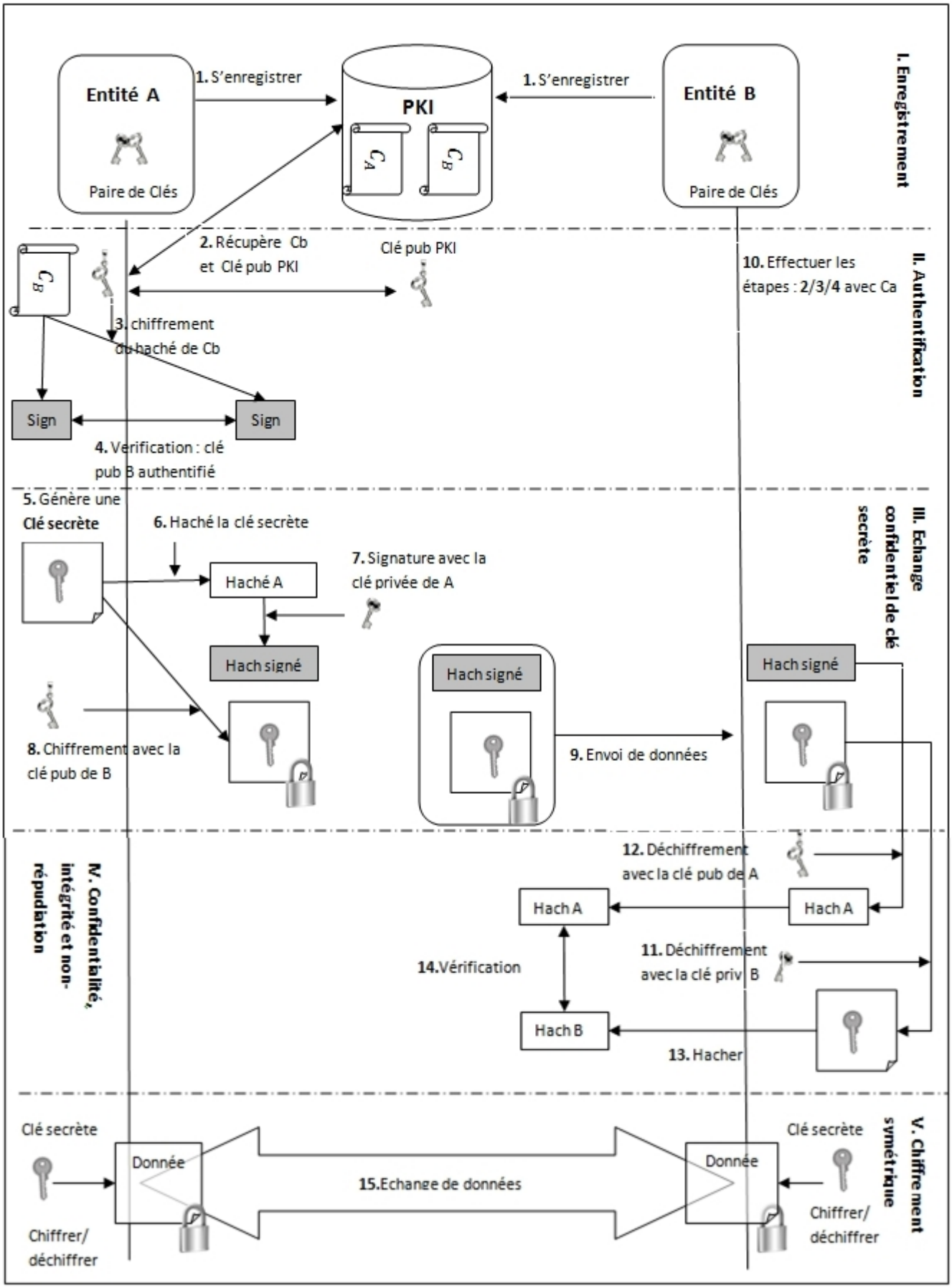
La figure I.5 présente les principaux champs et paramètres d'un certificat numérique selon la norme X.509. Cette structure de certificat est notamment adoptée par plusieurs protocoles de sécurité comme IPSec (Internet Protocol Security), SSL (Secure Socket Layers) et SET (Secure Electronic Transaction) [2].

Version du certificat
Numéro de série
Algorithme utilisé pour la signer le certificat
Nom de l'organisme qui a généré le certificat
Période de validité
Nom du propriétaire du certificat
Clé publique du propriétaire
Information additionnelles concernant le propriétaire ou les mécanismes de chiffrement
Signature du certificat

FIGURE I.5 – Principaux paramètres d'un certificat numérique selon la norme X.509.

I.2.7 Schéma complet pour une transmission sécurisée et rapide

La figure I.6 suivante représente les différentes étapes effectuées afin que deux entités puissent s'échanger des données tout en garantissant la confidentialité et l'intégrité des données, l'authentification et la non-répudiation des entités et cela en combinant les outils qu'offre la cryptographie moderne à savoir (chiffrement symétrique, chiffrement asymétrique, fonction de hachage, signature numérique et autorité de certification).



13

FIGURE I.6 – Système complet pour un échange sécurisé et rapide.

I.3 Politique de sécurité

On appelle politique de sécurité, toutes les mesures et précautions prises ainsi que les moyens déployés par une entreprise afin de garantir la sécurité de son système informatique. Étant le plan d'actions défini pour le maintien d'un certain niveau de sécurité dans une organisation, une politique de sécurité reflète la vision stratégique de la direction de l'organisme en matière de sécurité informatique [9].

La politique de sécurité est donc l'ensemble des orientations suivies par une organisation en termes de sécurité. A ce titre elle doit être élaborée au niveau de la direction de l'organisation concernée, car elle concerne tous les utilisateurs du système [9].

Sa mise en œuvre passe par les quatre étapes suivantes :

- Évaluation des ressources et analyse des risques : permet de déterminer les éléments critiques d'une organisation.

- Définition des objectifs stratégiques : pour chacune des ressources critiques, on visera à garantir la confidentialité, l'intégrité et la disponibilité.

- Développement et implémentation d'un plan d'actions : l'élaboration des mesures de protection contre des risques se fait en fonction des objectifs définie précédemment.

- Exploitation et évaluation de la politique : cette phase à pour but d'assurer la sécurité sur le long terme, il s'agit de veiller à ce que les actions menées remplissent les objectifs.

I.4 Les attaques informatiques

Tout ordinateur connecté à un réseau informatique est potentiellement vulnérable à une attaque. Une attaque est l'exploitation d'une des faiblesses (cf.figure I.7) d'un système informatique (système d'exploitation, logiciel ou bien même de l'utilisateur) à des fins non connues par l'exploitant du système. Les motivations des attaques peuvent être de différentes sortes :

- L'envoi et la recherche de chevaux de Troie.
- L'envoi de "bombes" logiques.
- La recherche de trous de sécurité.
- Le détournement d'identité (la mascarade).
- Changement des droits utilisateurs d'un ordinateur.
- La provocation d'erreurs non gérées.

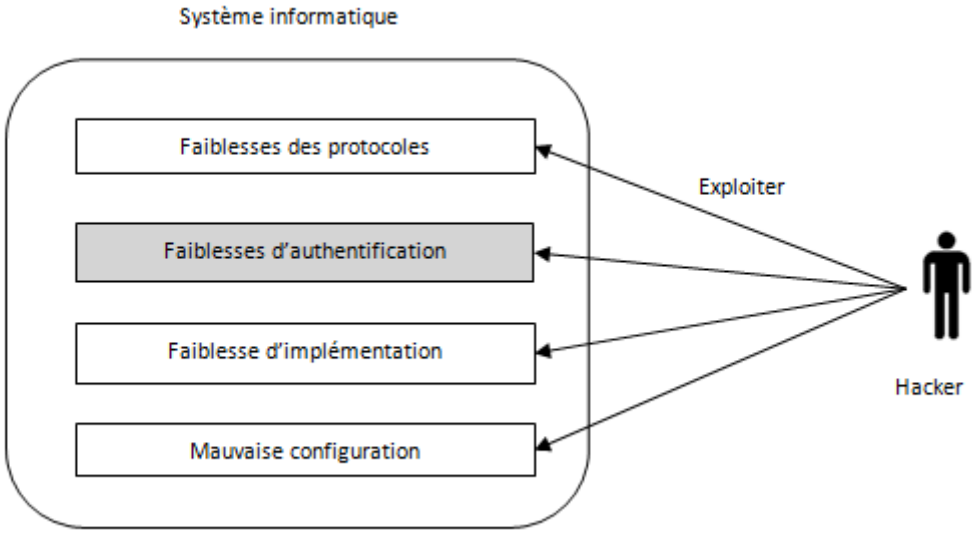


FIGURE I.7 – faiblesses exploitées par un hacker.

I.4.1 Différents type d'attaques

Le hacking est un ensemble de techniques informatiques, visant à attaquer un réseau. Les hackers utilisent plusieurs techniques d'attaques exploitant des failles trouvées dans la politique de sécurité adopté par l'entreprise. qui peuvent être regroupées en trois familles différentes [3] :

1. **Les attaques directes :** C'est la plus simple des attaques. Le hacker attaque directement sa victime à partir de son ordinateur. Par ce type d'attaque, il y'a de grandes chances pour que la victime puisse remonter à l'origine de l'attaque pour identifier l'attaquant.

2. **Les attaques indirectes par rebond :** Dans ce type d'attaque, les paquets d'attaques sont envoyés vers un ordinateur intermédiaire qui répercute l'attaque vers la victime. On utilise ce type d'attaque pour masquer l'identité (l'adresse IP) du hacker ou pour utiliser les ressources de l'ordinateur intermédiaire car il est plus puissant (CPU, bande passante. . .). Dans ce type d'attaque il n'est pas facile de remonter à la ressource.

3. **Les attaques indirectes par réponse :** Dans ce type d'attaque, le hacker va envoyer une requête vers l'ordinateur intermédiaire, la réponse à cette requête va être envoyée à l'ordinateur victime. Dans ce type d'attaque, il n'est pas possible de remonter à la source de l'attaquant.

I.4.2 Quelques techniques d'attaques

Il existe un grand nombre d'attaques permettant à une personne malintentionnée de s'approprier des ressources, de les bloquer ou de les modifier. Certaines requièrent plus de compétences que d'autres, nous nous intéressons aux attaques qui exploitent les faiblesses d'authentification [3] :

1. **le sniffer :** Est un dispositif logiciel ou matériel, qui permet de capturer les informations qui transite sur la machine ou il se trouve. Ils sont généralement utilisés pour récupérer les mots de passe dans un réseau. Pour cela, ils enregistrent les entêtes de paquets émis et reçus.

2. **L'IP spoofing :** Cette attaque consiste à modifier les paquets IP dans le but d'usurper l'identité d'une machine, et de passer inaperçu aux yeux d'un pare-feu qui les considérera alors comme provenant d'une machine de confiance.

3. **Le craquage des mots de passe :** Cette technique consiste à essayer plusieurs mots de passe afin de trouver le bon. Elle peut s'effectuer à l'aide d'un dictionnaire des mots de passe les plus courants, ou par la méthode de brute force (essayer toute les combinaisons jusqu'à trouver la bonne).

4. **Man in the middle :** comme son nom (homme au milieu) l'indique, cette attaque consiste à faire passer les échanges réseau entre deux systèmes par le biais d'un troisième, sous le contrôle du pirate. Ce dernier peut transférer des données à la volée, tout en masquant à chaque acteur de l'échange la réalité de son interlocuteur.

I.5 Outils et systèmes d'authentification

Pour mettre en place un système d'authentification, il existe plusieurs systèmes et protocoles qui ont des avantages et des limites selon le domaine d'application. Généralement un système d'authentification est toujours associé d'une part, à un ou plusieurs annuaires contenant l'ensemble des utilisateurs de ce dernier, d'une autre part, à un protocole d'échange de données sécurisé (SSL) reliant les utilisateurs à leur serveur d'authentification.

I.5.1 Les annuaires

I.5.1.1 Définition

Un annuaire électronique se base sur le même principe que les annuaires téléphoniques. Autrement dit, c'est une base de données classant les informations par un certain critère (ordre alphabétique, région, métier, etc.) afin de faciliter la recherche. Par conséquent le critère de recherche doit être puissant et simple d'utilisation et pour être objectif, il doit être choisit selon le domaine d'utilisation de l'annuaire.

I.5.1.2 Caractéristiques et apports d'un annuaire

- **Un aspect dynamique** qui permet d'une part un gain de temps lors de la recherche d'autre part, lors de mise à jour de l'annuaire. De plus on est presque sûr de l'actualité et de la validité des informations en délégant les mises à jour aux propriétaires même de celles-ci [8].

- **Une structure de données flexible** permettant l'ajout d'information sans porter attente sur les autres. Aussi la possibilité de modifier l'organisation des données ainsi que le critère de classement rendant la recherche plus pointue et rapide [8].

- **Le contrôle des habilitations** des utilisateurs offre une meilleure sécurité en limitant ou interdisant à ces derniers l'accès à des informations ou à des services [8].

- **La personnalisation** de l'annuaire a toute son importance. En effet, tenir compte des préférences de chaque utilisateurs leurs permet d'accéder plus rapidement aux informations les plus proches de leurs centres d'intérêt ou de leurs besoins [8].

I.5.1.3 Exemple d'annuaire

L'annuaire le plus utilisé des systèmes d'authentification est l'annuaire LDAP (*Lightweight Derectory Access Protocol*) qui est une base de données hiérarchique représentant les données sous forme d'un arbre. Ce standard a été normalisé par IETF. L'annuaire est conçu pour référencer toutes sortes d'informations : informations sur des personnes, sur des applications, sur un parc informatique.

LDAP propose un mécanisme de gestion d'authentification unifiée (centralisé) ainsi que la gestion des habilitations des utilisateurs enregistrés. (On verra le fonctionnement détaillé de ce standard dans le troisième chapitre)

I.5.2 SSL (Secure Socket Layer)

I.5.2.1 Présentation de SSL

SSL est un protocole de sécurisation des échanges, il a été développé à l'origine par Netscape qui a publié la version 2 de SSL en 1994. Ce dernier a été standardisé par l'IETF en 1996 et qui repris son développement.

C'est un protocole de chiffrement des échanges qui agit entre la couche application et la couche transport du modèle OSI. Il est utilisé comme tunnel de sécurité pour les protocoles applicatifs comme HTTP, SMTP ou FTP. Il utilise une authentification du serveur au travers d'un certificat X.509, ceci afin d'assurer la confiance de l'utilisateur sur le lien le reliant au serveur [5].

I.5.2.2 Principe et fonctionnement de SSL

SSL utilise les principes des chiffrements et d'authentification qui reposent sur les algorithmes à clé publique et les algorithmes à clé secrète décrits au premier chapitre. Il

permet ainsi de créer un tunnel chiffré entre deux entités. Les étapes d'initialisation du tunnel, appelé SSL Handshake, sont les suivantes [5] :

- Le client fait une requête SSL auprès du serveur.
- Le serveur présente alors son certificat afin de créer un contexte de confiance.
- Le client peut éventuellement lancer une requête OCSP pour vérifier la validité du certificat.
- Le serveur peut demander au client de lui présenter son certificat (optionnel).
- Les deux entités négocient, en fonction de leurs capacités cryptographique, les types et longueurs de clé qu'ils vont utiliser pour le chiffrement de échanges.
- Le client génère une clé secrète de session de façon aléatoire (cette clé sera inférieure à 56 bits en SSL V2, inférieure à 128 bits en SSL V3).
- Il chiffre cette clé secrète avec la clé publique du serveur et l'envoie à ce dernier.
- Le serveur déchiffre la clé secrète (clé de session) avec sa clé privée.
- En fin, les deux entités dialoguent en utilisant un algorithme à clé secrète pendant toute la durée de la session SSL (cf.figure I.8).

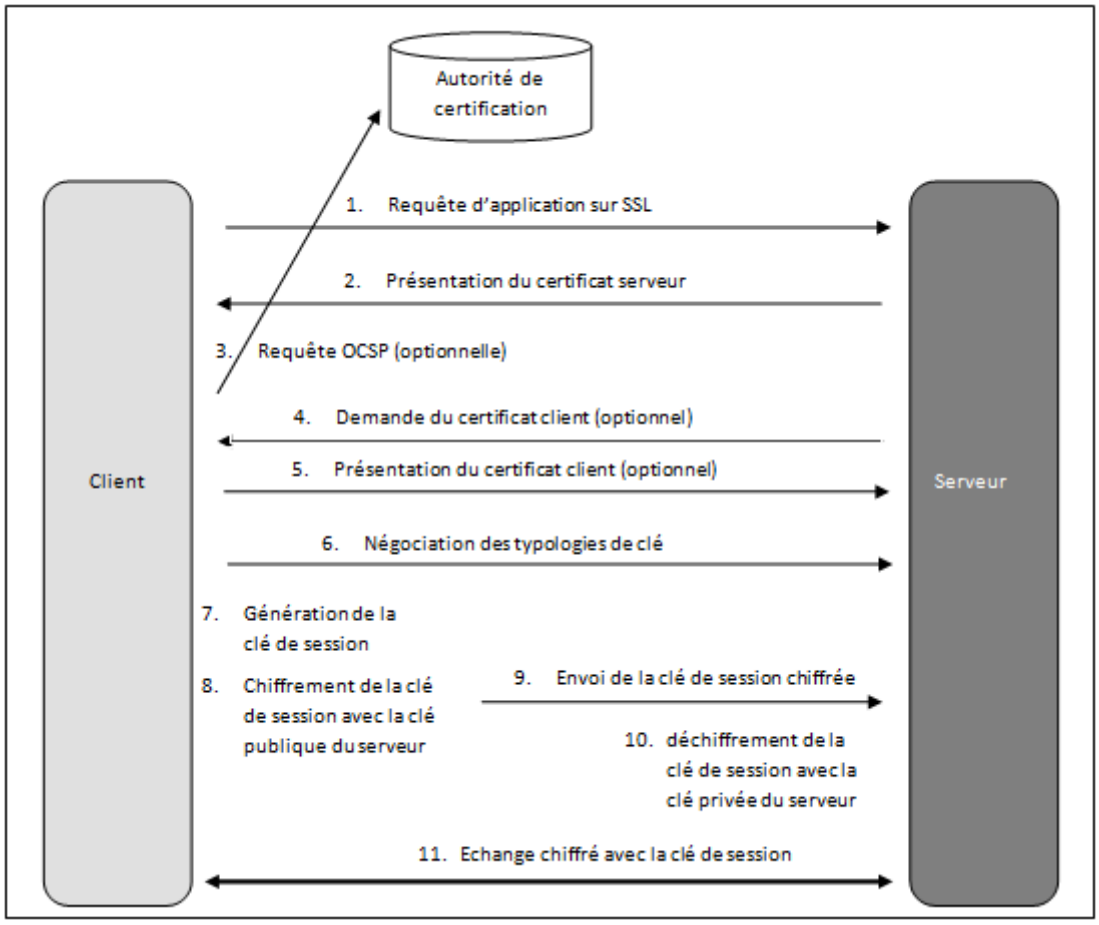


FIGURE I.8 – Étapes du SSL Handshake

Ainsi, SSL est basé sur un échange de clé en utilisant la cryptographie à clé publique, puis les échanges de données sur une session à base de cryptographie à clé secrète. SSL

profite donc de deux avantages, d'une part il dispense les clients de posséder une paire de clés, d'autre part, l'échange des données avec un chiffrement symétrique est beaucoup plus rapide.

Une session SSL est terminée par demande du client ou bien par timeout (paramètre du serveur précisant le temps rémanence de la session en cas d'inactivité du client).

La version 3 de SSL a apporté le contrôle d'intégrité des échanges en introduisant un hachage des données. Cette empreinte générée est aussi chiffrée avec la clé de session de la même façon dont les données sont chiffrées et ensuite elle est envoyée. Ce système est appelé HMAC pour *Hach Message Authentification Code*.

En fin, TSL (Transport Socket Layer) n'est que l'acronyme de SSL le rendant plus explicite. TSL version 1.0 est l'équivalant de SSL version 3.1 standardisé par l'IETF [5]

I.5.3 RADIUS

Le protocole RADIUS (Remote Authentication Dial-In User Service - RFC 2865 et RFC 2866) mis au point initialement par Levingston, c'est un protocole d'Authentification, d'Autorisation et d'Accounting (AAA), chargé du contrôle d'accès au réseau.

Le fonctionnement de RADIUS est basé sur un système client/serveur chargé de définir les accès d'utilisateurs distants à un réseau. Il s'agit du protocole de prédilection des fournisseurs d'accès à internet car il est relativement standard et propose des fonctionnalités de comptabilité permettant aux FAI de facturer précisément leurs clients.

Il permet de centraliser les données d'authentification : Les politiques d'autorisation, les droits d'accès et la traçabilité. Ce processus doit être relié à une source d'informations.

Le protocole RADIUS repose principalement sur un serveur (le serveur RADIUS), relié à une base d'identification (base de données, annuaire LDAP, etc.) et un client RADIUS, appelé NAS (Network Access Server), faisant office d'intermédiaire entre l'utilisateur final et le serveur. L'ensemble des transactions entre le client RADIUS et le serveur RADIUS sont chiffrées et authentifiées grâce à un secret partagé.
[4].

Son fonctionnement est simple :

1. L'utilisateur se connecte (via le protocole PPP) à un client RADIUS (ou NAS) qui est en général une passerelle/proxy.

2. Le client RADIUS demande à l'utilisateur son nom et son mot de passe, et il les communique de manière sécurisée à un serveur RADIUS relié à une base de données ou à un annuaire.

3. En fonction de la zone d'accès demandée et des droits de l'utilisateur, le serveur RADIUS peut exiger des informations supplémentaires pour l'authentification.

   - La réponse CHALLENGE permet d'éviter de transmettre le mot de passe. Le client envoie alors une autre requête répondant au Challenge pour s'authentifier.

   - Si l'identification réussie, le Serveur répond par un ACCEPT (REJECT dans le cas contraire).

4. Le serveur RADIUS envoie enfin les autorisations de l'utilisateur.

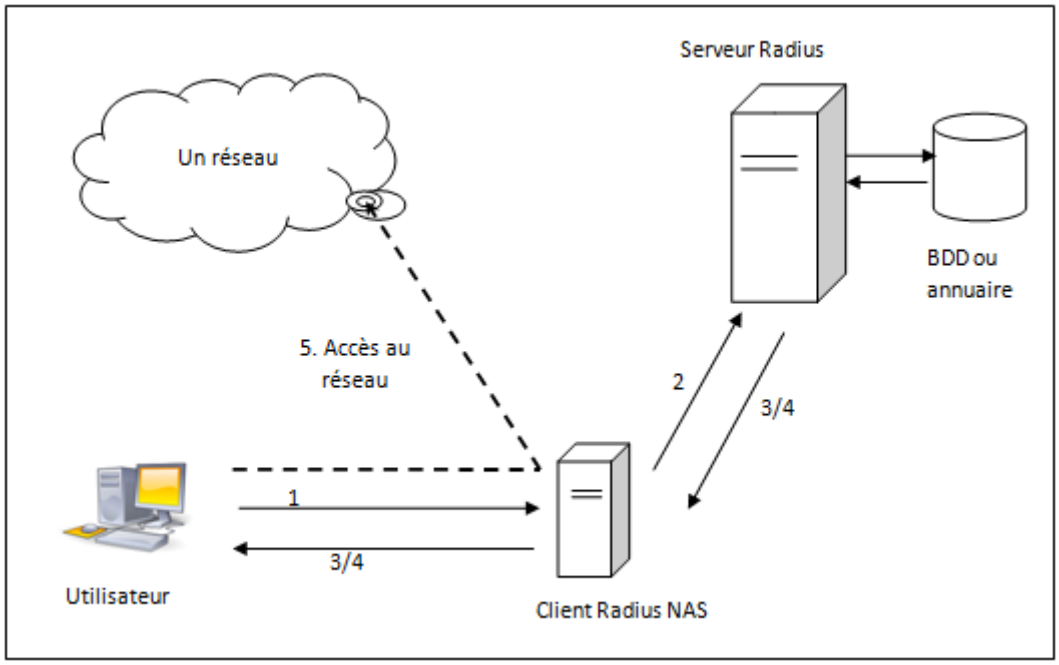


FIGURE I.9 – Fonctionnement de Radius

I.5.4 Kerberos

Kerberos est un système d'authentification développé par le MIT. La version 5 du protocole a été normalisée par l'IETF dans la RFC 1510 (septembre 1993) et RFC 1964 (juin 1996). Le nom « Kerberos » provient de la mythologie grecque et correspond au nom du chien (en français « Cerbère ») protégeant l'accès aux portes d'Hadès.

Kerberos utilise la notion de « ticket » pour éviter à un utilisateur de devoir s'authentifier constamment aux différents serveurs auxquels il se connecte. Un système kerberos est constitué d'un serveur d'authentification SA, d'un serveur de tickets TGS, de clients et de serveurs de services. Son fonctionnement est présenté sur la figure I.10 :
[3].

1. L'utilisateur émet une requête au serveur SA afin d'obtenir un ticket d'octroi de ticket.

2. Le serveur SA envoie en retour, après validation des droits d'accès, un ticket « tgs » et une clé de session chiffrés avec la clé dérivée du mot de passe de l'utilisateur.

3. Le poste de travail demande à l'utilisateur son mot de passe et l'utilise pour déchiffrer le message émis par le serveur SA. Il envoie alors au serveur TGS une requête de service en émettant le ticket «tgs » et un authentifiant (nom, adresse IP, adresse, heure). Le ticket « tgs », préalablement chiffré par le serveur SA avec une clé uniquement connue des serveurs SA et TGS, contient la clé de session.

4. Le serveur TGS déchiffre le ticket et l'authentifie, crée le ticket pour le service demandé et l'envoie au poste. Ces données sont chiffrées avec la clé de session partagée maintenant par le serveur TGS et l'utilisateur. Le ticket de service a été préalablement chiffré par le serveur TGS avec une clé uniquement connue du serveur de service et du serveur de tickets TGS.

5. Le poste envoie le ticket et l'authentifiant au serveur désiré.

6. Le serveur vérifie le ticket et l'authentifiant, puis octroie l'accès au service.

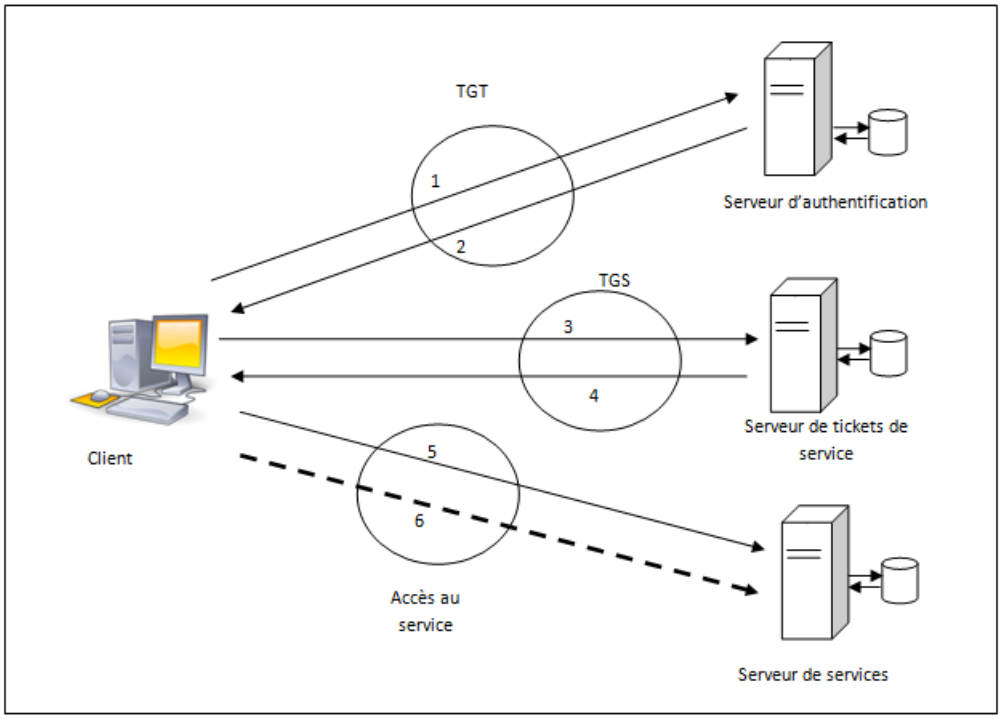


FIGURE I.10 – Fonctionnement de Kerberos

Conclusion

Dans ce chapitre, nous avons vu que pour assurer un certain niveau de sécurité il existe bien une multitude de mécanismes et d'outils cryptographique permettant chacun de couvrir une vulnérabilité dans un système. Cela dit, chaque outil présente bien des avantages pour lesquels il a été conçu ainsi que des limites lesquelles d'autres outils prendront en charge. Pour profiter des avantages de tous les systèmes, il est obligatoire de les combiner pour une sécurité optimale.

Ce chapitre nous apprend aussi que l'authentification des utilisateurs d'un système est primordiale pour la sécurité de ce dernier. Cela dit, la phase d'authentification peut parfois devenir encombrante pour les utilisateurs, d'une part, par la mémorisation de plusieurs couple (login /password) et d'autre part, par la répétition de celle-ci à longueur de journée pour différents services. C'est pour cela, nous allons consacré le chapitre suivant à l'étude de deux méthodes (authentification unique et unifiée) et leurs outils. Ces dernières améliorent le coté ergonomique, coté de gestion pour les administrateurs ainsi que le coté sécurité.

Chapitre

II Authentification unifiée et unique

Introduction

On a vu que la sécurité d'un système informatique commence par l'authentification des utilisateurs autorisés, cela afin d'éviter que des personnes malveillantes abusent facilement du système. On a vu aussi qu'il existe plusieurs systèmes pouvant assurer ce service, cependant l'importance de cette phase pour la sécurité et sa répétitivité que ce soit pour une même application ou plusieurs, nous ont conduits à nous intéresser à deux types d'authentifications bien particulières. L'authentification unifiée et l'authentification unique permettent bien d'avantages sur le plant de gestion et sur le plan ergonomique pour les utilisateurs.

II.1 L'authentification unifiée

On dit qu'une authentification est unifiée si celle-ci, permet à l'authentifié d'accéder à plusieurs services se trouvant sur un même système avec un même couple (nom utilisateur, mot de passe). D'une part elle évite le déploiement d'une base de données contenant la liste des usagés sur chaque service, d'une autre part, elle évite aux utilisateurs de retenir une multitude de mot de passes.

II.1.1 Présentation de LDAP

Apparu en 1993, LDAP ou Lightweight Directory Access Protocol est un protocole d'interrogation d'annuaire permettant d'effectuer des recherches et des modifications dans ces derniers. Il admet aussi bien la gestion des authentifications des utilisateurs ainsi que leurs habilitations dans un système informatique.

II.1.2 Utilité de LDAP

LDAP est aujourd'hui un standard incontournable. En effet par sa capacité à s'adapter à plusieurs cas d'utilisation, en plus de l'authentification, un annuaire LDAP devient pratiquement indispensable au cœur d'un système informatique (cf.figure II.1). Il est utilisé notamment pour [5] :

- Les outils de messagerie savent interroger un annuaire LDAP pour tirer leur carnet d'adresse.

- Les solutions d'infrastructure de sécurité (Firewall, proxy, etc.) peuvent y lire leurs paramétrages.

- Une majorité des solutions logicielles du marché (gestion de contenu, portails d'entreprise, progiciels de gestion, etc) utilisent LDAP pour l'authentification.

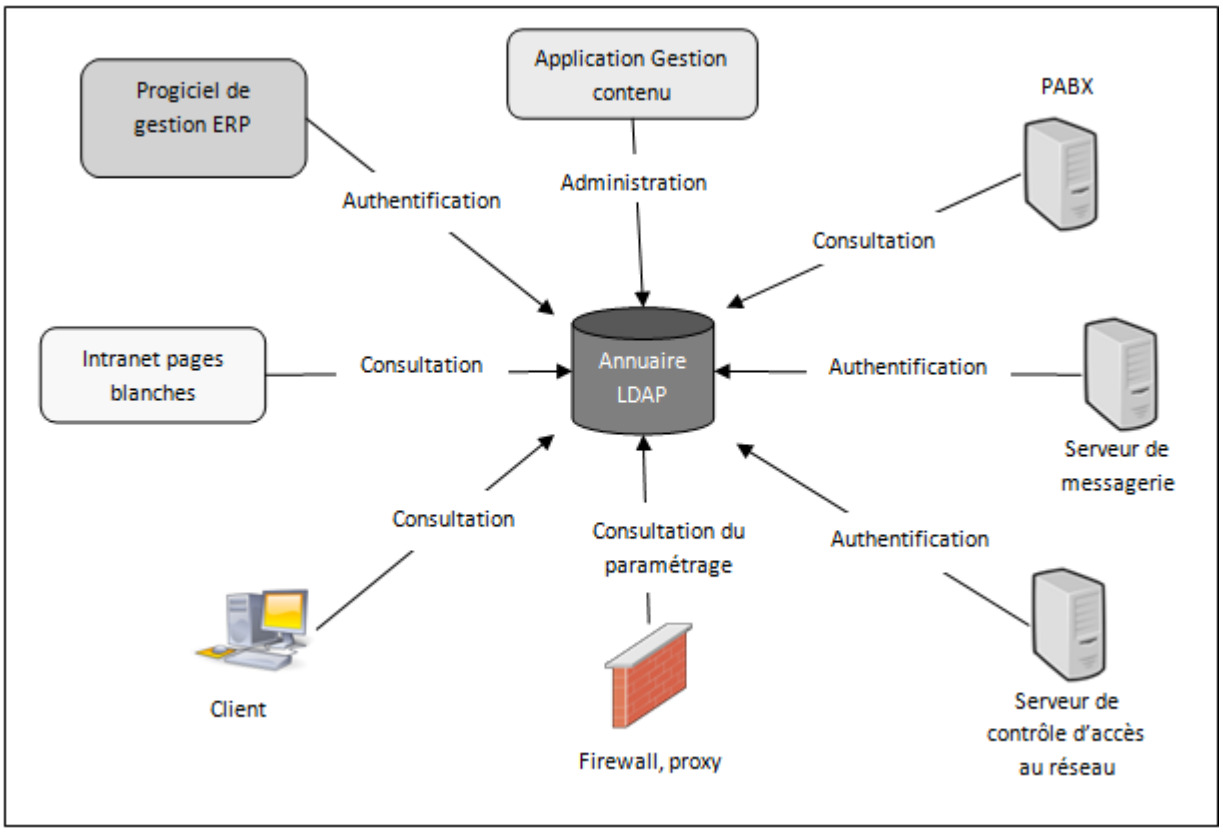


FIGURE II.1 – LDAP au cœur d'un système.

II.1.3 Services de sécurité proposés par LDAP

Dans le domaine de la sécurité LDAP propose des mécanismes pour gérer l'authentification et les habilitations des utilisateurs ainsi que la confidentialité des échanges.

II.1.3.1 Service d'authentification

Plusieurs méthodes d'authentification correspondant à différents niveaux de sécurité sont offertes par la norme LDAP :

- La connexion anonyme (anonymous) généralement limitée à la consultation de parties restreintes de l'annuaire.

- L'authentification par identifiant/ mot de passe.

- L'authentification par identifiant/ mot de passe avec hachage de ce dernier.
- L'authentification par identifiant/ mot de passe sur SSL.
- L'authentification par certificat X509.

L'existence d'un annuaire LDAP allège donc, les applications de l'étape d'authentification en utilisant l'une des méthodes précédentes.

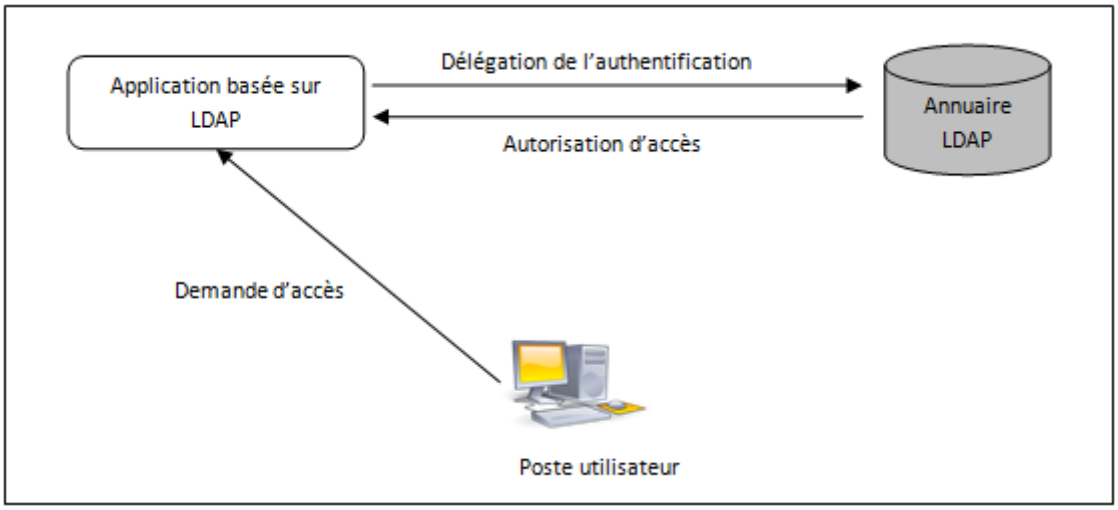


FIGURE II.2 – Authentification LDAP

Pour une meilleure sécurité, il vaut mieux utiliser la méthode d'authentification par identifiant/ mot de passe sur SSL. Cette dernière veille à ce que les échanges soient sécurisés, d'une part, entre l'utilisateur et l'application, d'autre part, entre l'application et l'annuaire LDAP. Pour ce faire, deux tunnels SSL sont nécessaires. Il faudra ainsi déployer un certificat sur l'application et sur le serveur LDAP [5] [8].

II.1.3.2 Service d'habilitation

On distingue deux types d'habilitations, habilitations concernant l'annuaire LDAP lui-même et habilitations concernant le système utilisant l'annuaire. Cependant LDAP propose deux mécanismes pour la gestion des deux types d'habilitations.

Le premier mécanisme gérant les habilitations sur les objets contenus dans l'annuaire consiste à utiliser des ACL *(Access Control Lists)* positionnées sur les nœuds de l'arbre LDAP et permettent de définir les droits d'interactions des utilisateurs avec les branches intérieures.

Le deuxième mécanisme consiste à définir des groupes et des rôles vis-à-vis les applications du système d'information. Un groupe LDAP permet de constituer des listes d'utilisateurs habilités à effectuer une action précise sur une application données. Les rôles aussi per-

mettent de définir des habilitations mais elles sont positionnées comme attributs dans la description des utilisateurs [5] [8].
Les deux mécanismes sont illustrés par la figure II.3.

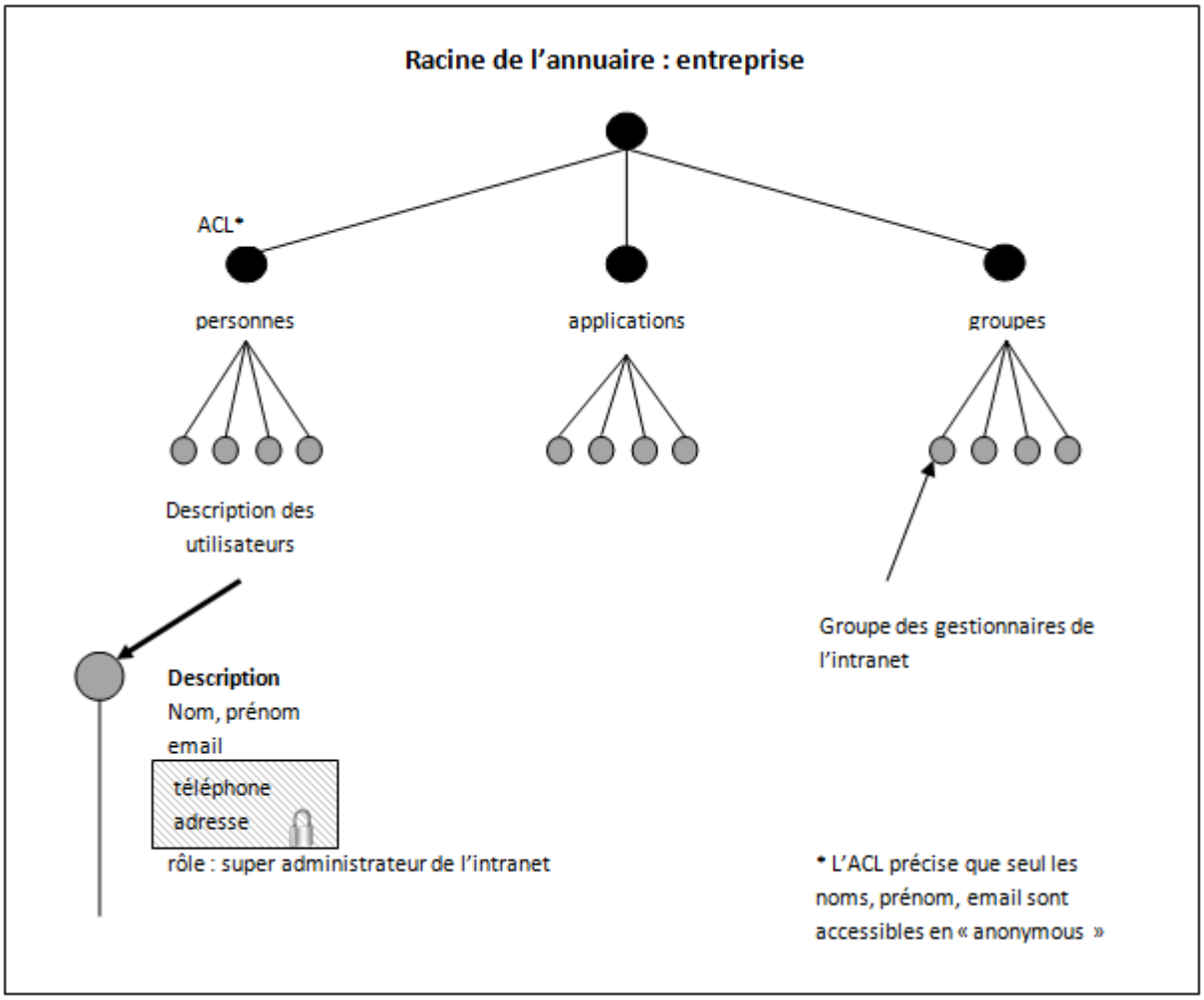


FIGURE II.3 – Gestion des habilitations.

II.1.3.3 Service de confidentialité

LDAP prend également en charge la sécurisation des échanges entre lui et ces clients, en utilisant le protocole SSL.

II.2 Authentification Single Sign On

On a constaté que du coté pratique pour les utilisateurs la solution apportée par les annuaires LDAP pouvait être améliorer. En effet, si LDAP offrait un référentiel utilisateur centralisé permettant l'établissement d'un identifiant/ mot de passe unique pour toutes les applications d'un système d'information, ce dernier oblige les utilisateurs à s'authentifier sur chacune des applications avec un même identifiant/mot de passe ce qui n'est pas

pratique. Ainsi est née l'idée de l'authentification unique (Single Sign On), qui a été inspiré du système Kerberos (présenté auparavant) et utilise notamment les mêmes termes.

II.2.1 Objectifs de SSO

- Au plan ergonomique, faciliter la navigation des utilisateurs entre les applications et services d'un système sans les obliger à s'authentifier à chaque fois.

- Simplifier la gestion des authentifications ainsi que les droits d'accès en centralisant les informations des utilisateurs et leurs habilitations sur un même serveur d'authentification.

- Une meilleure sécurisation de l'authentification, d'une part, en minimisant la circulation des mots de passes, d'autre part, en réduisant leurs nombres à un seul pour chaque utilisateur. Lors de la multiplicité des authentifications, les utilisateurs ont tendance à utiliser des mots de passes faibles et similaires pour faciliter leurs mémorisations.

II.2.2 Principes de SSO

Le principe de SSO consiste à mettre en œuvre un mécanisme garantissant la propagation de session d'une application à d'autres, si l'utilisateur désir y'accéder. Pour ce faire une architecture SSO est toujours basée sur un agent tiers « Moteur de SSO » à qui les applications délèguent l'authentification des utilisateurs, leurs habilitations ainsi que la propagation de session. Si un utilisateur souhaite accéder à une application et qu'il n'est pas authentifié de manière globale, il est obligé à s'authentifier auprès du moteur SSO. Ce dernier prend en charge les authentifications secondaires de l'utilisateur auprès des applications auxquelles il a accès (cf. figure II.4). Le mécanisme de l'authentification secondaire est totalement transparent à l'utilisateur et diffère d'une implémentation à une autre mais se base principalement sur des échanges de tickets [5].

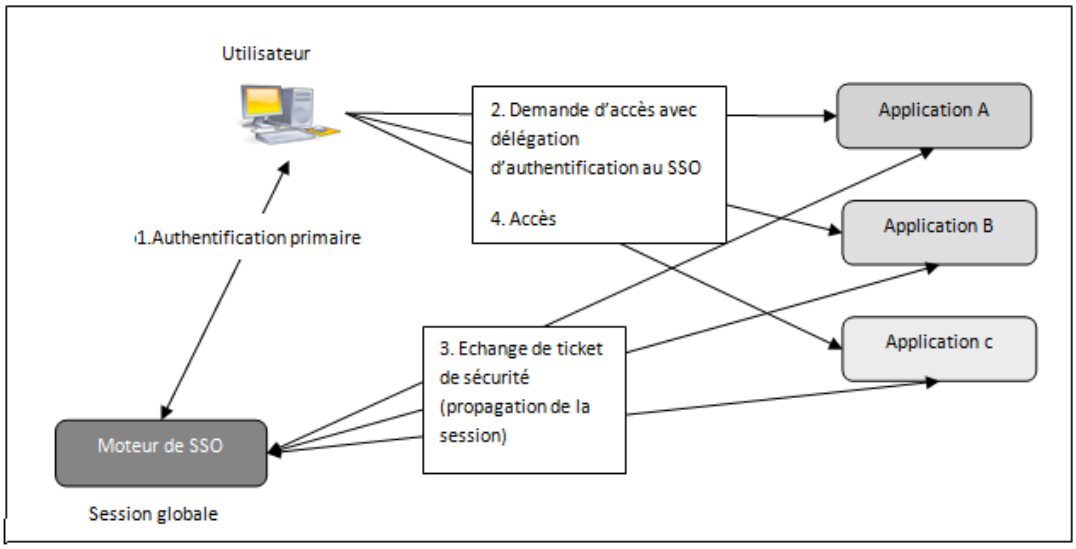


FIGURE II.4 – SSO pour délégation d'authentification.

II.2.3 SSO client/serveur

Le principe du SSO client/ serveur est le suivant (cf.figure II.5) : le moteur de SSO joue le rôle de serveur d'authentification et centralise les authentifications, les habilitations et les traces des utilisateurs. Le rôle de client est joué par un agent SSO déployé sur chaque application pour communiquer avec le serveur d'authentification.

Les agents permettent de rediriger l'utilisateur vers le moteur SSO si celui-ci n'est pas authentifié de façon globale, puis de collecter ses habilitations pour le compte de l'application. Si l'utilisateur est authentifié il se contente de collecter ses habilitations et les transmet à l'application.

Le moteur SSO utilise pour le stockage des identités et les habilitations correspondantes un annuaire LDAP ou une base de données.
[5]

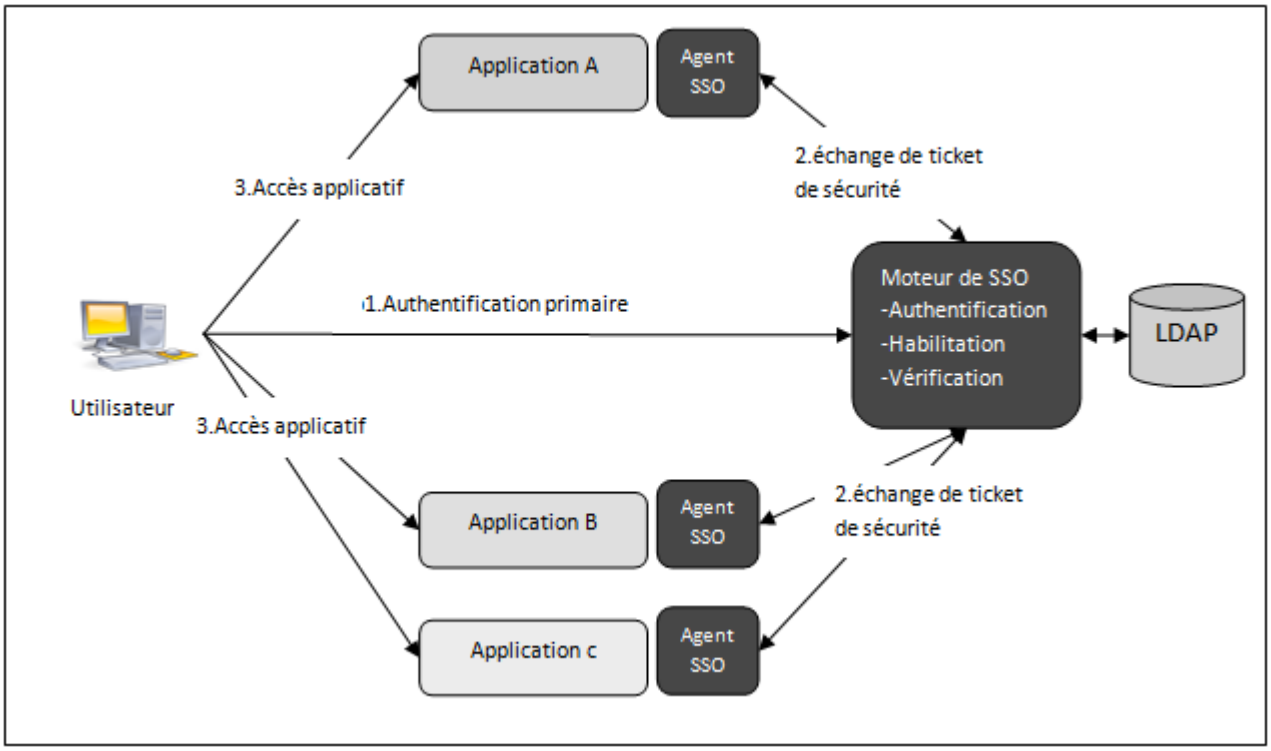


FIGURE II.5 – Fonctionnement du SSO client/ serveur.

Cette approche client/serveur est la plus courante dans le cadre d'applications hébergées sur un réseau car elle permet moins de charge sur le moteur SSO, celui-ci n'est sollicité qu'au moment d'ouverture des sessions. En revanche elle est très intrusive sur les applications, il faut supprimer le mode d'authentification de celles-ci et leurs inclure l'agent SSO, ce qui peut se révéler complexe ou impossible pour certaines applications. Cette approche est inadaptée aux réseaux extranets car les serveurs d'application sont directement accessibles sans aucune mesure de sécurité. Cependant, il existe une alternative pour renforcer

la sécurité des applications. Elle consiste à mettre le moteur de SSO comme proxy devant les applications, ainsi l'authentification devient un passage obligé avant même de voir le choix d'application offert par le système.

II.2.4 Central Authentification Service (CAS)

CAS (**Central Authentification Service**) a été développé en 2004 par l'Université de Yale. C'est un système d'authentification unique : on s'authentifie sur une application et on est alors authentifié sur toutes les applications qui utilisent le même serveur CAS. Il est composé de servlets java, qui fonctionnent sur tout moteur de *servlets* (*Tomcat* par exemple), et dont ses points forts sont les suivants :

- La sécurité est assurée par les dispositifs suivants :
  - Le mot de passe de l'utilisateur ne circule qu'une seul fois entre le navigateur client et le serveur d'authentification.
  - Les réauthentifications suivantes sont faites de manière transparente à l'utilisateur.
  - L'application reçoit du serveur d'authentification un « ticket opaque » qui ne transporte aucune information personnelle.

- CAS propose un mécanisme de mandataire (proxies). Des tickets dédiés permettent à des applications tierces d'être assuré de l'authentification de l'utilisateur.

- Des librairies clientes en Java, Perl, JSP, ASP, PL/SQL et PHP sont livrées. Cela permet une grande souplesse sur les serveurs d'applications.

[6] [7].

II.2.4.1 Architecture de CAS

L'authentification est centralisée sur une machine unique, le serveur CAS. Ce serveur est le seul acteur du mécanisme CAS à avoir connaissance des mots de passe des utilisateurs. Son rôle est double :

- Authentifier les utilisateurs ;
- Transmettre et certifier l'identité de la personne authentifiée (aux clients CAS).

II.2.4.2 Fonctionnement de base

1. **Authentification d'un utilisateur :**

Un utilisateur non précédemment authentifié, et qui accède au serveur CAS se voit proposer un formulaire d'authentification, dans lequel il est invité à entrer son couple login/-password :

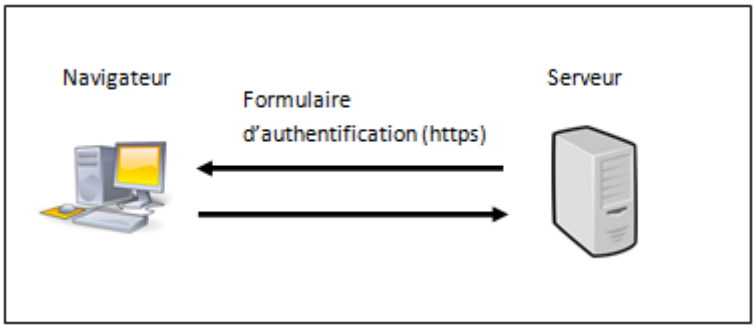


FIGURE II.6 – premier accès d'un navigateur au serveur CAS.

Si les informations sont correctes, le serveur renvoie au navigateur un cookie appelé TGC (Ticket Granting Cookie) :

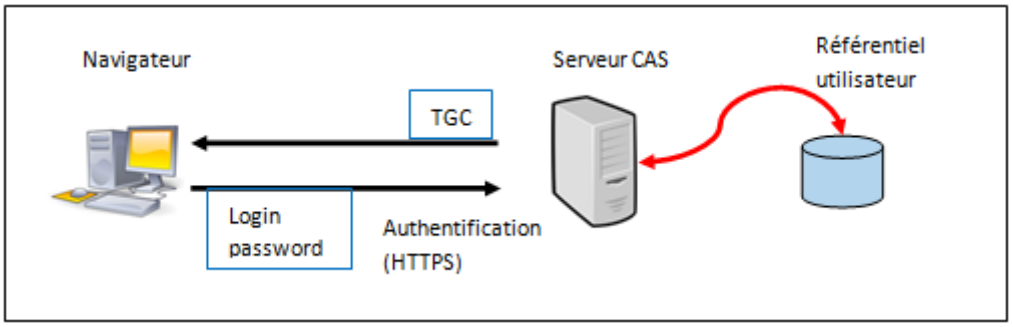


FIGURE II.7 – Authentification d'un navigateur auprès d'un serveur CAS.

Ticket Granting Cookie (TGC) est le passeport de l'utilisateur auprès du serveur CAS. Ayant une durée de vie limitée, TGC est le moyen pour les navigateurs d'obtenir auprès du serveur CAS des tickets de services sans avoir à se ré-authentifier. C'est un cookie privé et protégé. Il est aussi opaque tout comme les autres tickets utilisés dans le mécanisme CAS [7].

2. **Accès à une ressource protégée après authentification :**
    1. Envoi de la requête.
    2. Redirection de la requête vers le serveur CAS.
    3. Le navigateur, précédemment authentifié, présente le TGC auprès du serveur CAS.
    4. Le serveur CAS délivre un Service Ticket(ST) au navigateur.

5. Le navigateur redirige le ST vers le service demandeur.
6. Le Service Ticket est alors validé auprès du serveur CAS par le client CAS.
7. Si le ST est validé, la ressource peut être délivrée au navigateur.

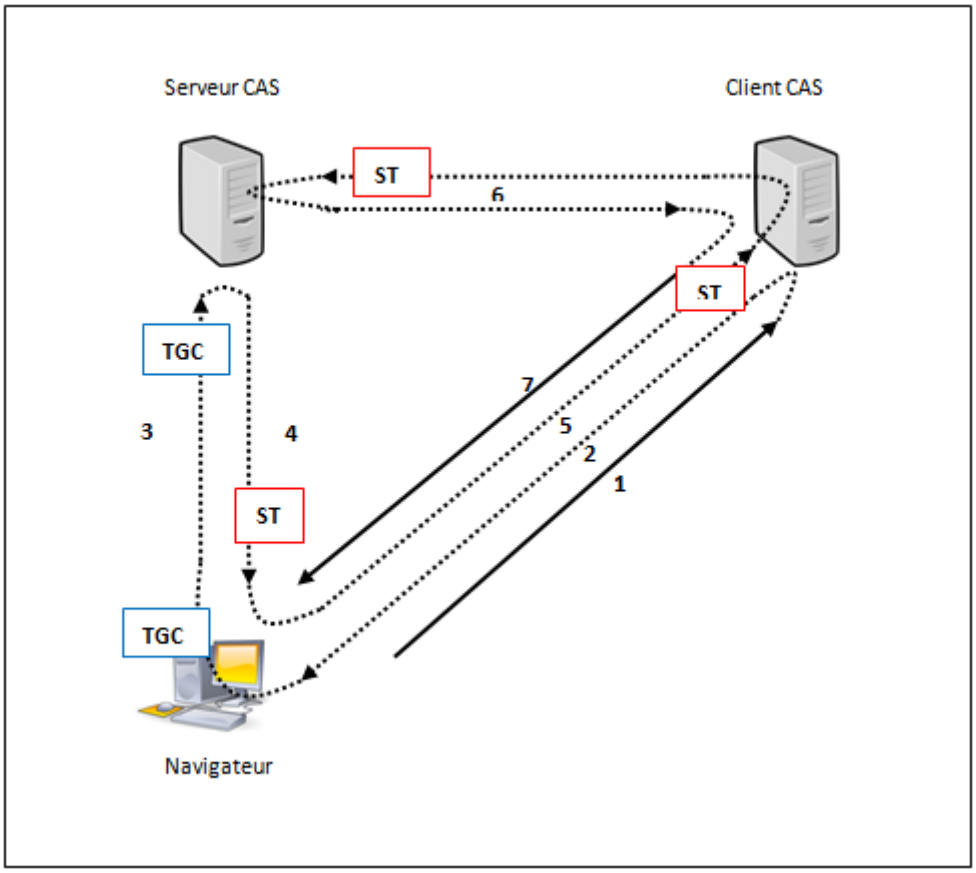


FIGURE II.8 – Fonctionnement de base de CAS.

Le Service Ticket (ST) est le passeport de l'utilisateur auprès du client CAS. Il est limité à un seul client CAS et sa durée de vie est très limitée.
[7]

II.2.4.3 Fonctionnement multi-tiers

Différemment au fonctionnement de base, le navigateur accède à un client CAS à travers un mandataire (proxy) CAS. Le mandataire CAS se comporte comme étant un navigateur. Cependant, le mécanisme de redirection n'est plus applicable.

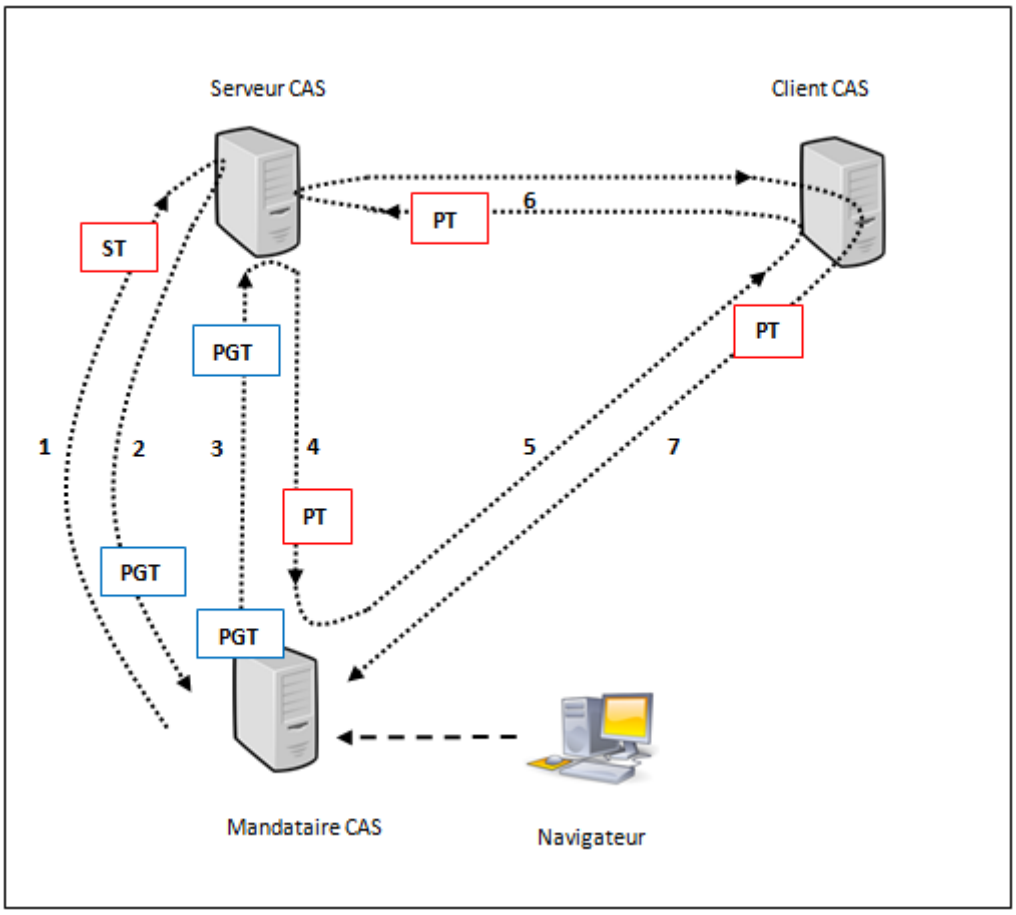


FIGURE II.9 – Fonctionnement n-tiers.

1. Le mandataire CAS valide un ticket ST, au même temps, il effectue une demande de PGT (Proxy Granting Ticket).
2. Le serveur CAS valide l'authentification en envoyant un PGT au mandataire CAS.
3. Le mandataire fait une demande de PT (Proxy Ticket) .
4. Le serveur CAS répond par un PT.
5. Le mandataire CAS redirige le PT vers le client CAS.
6. Le client CAS valide le PT auprès du serveur CAS.
7. Le client CAS renvoie le PT validé vers le mandataire CAS.

Proxy Granting Ticket (PGT) est le passeport d'un mandataire CAS, pour un utilisateur, auprès du serveur CAS. Comme le TGC (Ticket Granting Cookie), il est opaque et à une durée de vie limitée.

Proxy Ticket (PT) est le passeport de mandataire CAS auprès de clients CAS. Comme le Service Ticket (ST), il est à une durée de vie très limitée.

II.2.4.4 Authentification sous CAS :

CAS, dans sa distribution originale, ne propose pas de classe d'authentification : les méthodes d'authentification sont à la charge de l'administrateur du serveur CAS. Développée au sein du projet ESUP-Portail, la classe GenericHandler permet l'implémentation de plusieurs méthodes d'authentification : annuaires LDAP, bases de données, certificat X509, etc.

La configuration du serveur se fait simplement en renseignant un fichier au format XML. Les méthodes d'authentification souvent utilisées :

1. **Sur un annuaire LDAP :**

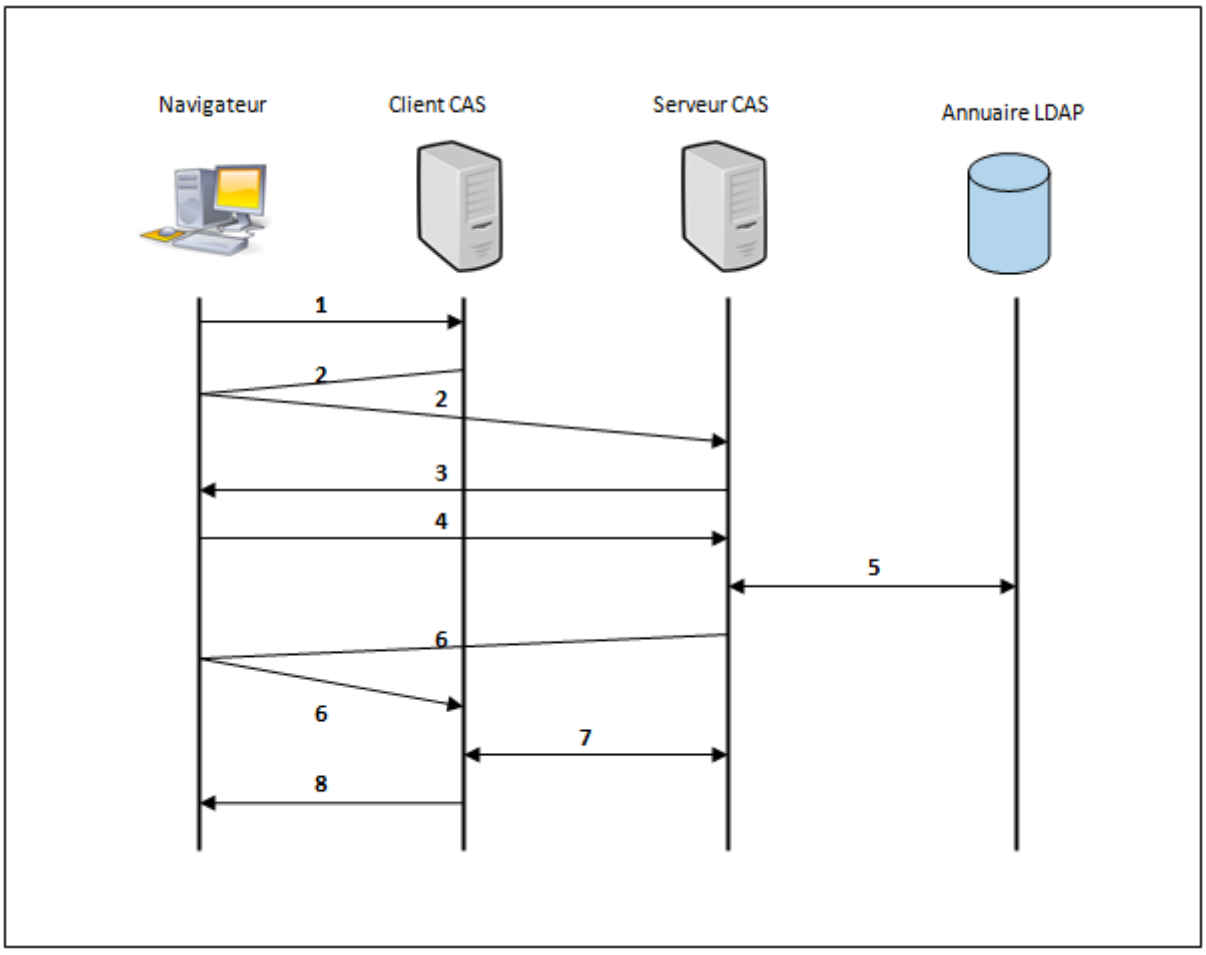


FIGURE II.10 – Authentification sur un annuaire LDAP.

1. Un internaute accède à une application web.
2. Le client CAS redirige le navigateur vers le serveur CAS.
3. Le serveur CAS envoie un formulaire d'authentification.
4. L'internaute s'authentifie.

5. Le serveur CAS vérifie le couple login/password grâce à l'annuaire LDAP.
6. Le serveur CAS crée un cookie de session (TGC) et redirige le navigateur vers l'application avec un ticket à utilisation unique(ST).
7. L'application valide le ticket (ST) en contactant directement le serveur CAS qui retourne l'identifiant de la personne.
8. L'internaute accède à la ressource demandée.

2. **Sur une base de données :**

Cette méthode est essentiellement destinée à des organisations dont certains utilisateurs, pour des raisons techniques ou organisationnelles, ne sont pas enregistrés dans leur annuaire LDAP mais dans une base de données. Deux modes sont proposés :

- **Mode « connexion » :** les utilisateurs à authentifier doivent être également utilisateurs déclarés de la base de données.

- **Mode « recherche » :** utilise une connexion privilégiée à la base, et les informations de connexion des utilisateurs sont stockées dans une table.

3. **Sur un certificat X.509 :**

Les certificats personnels X.509 sont communément utilisés pour l'authentification des applications web. Lorsqu'un certificat est transmis par le navigateur au serveur CAS, il peut servir d'authentification pour son utilisateur. Lorsqu'un utilisateur présente un certificat X.509, le serveur CAS :

- Contrôle la validité du certificat (au sens des PKI) ;

- Recherche l'identité de l'utilisateur, en s'appuyant sur un annuaire LDAP ou une base de données.

[7]

II.3 Quelques projets universitaires

Répondant à la problématique de l'authentification dans les applications web, de nombreuses universités en Europe et aux Etats-Unis, ont développé des solutions locales de SSO qui sont devenues des produits. Quelques uns de ces produits sont décrits ci-dessous ; pour la plupart distribués sous un régime de licence libre [6].

II.3.1 A-Select (Surfnet - Pays Bas)

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

II.3.2 Moria (FEIDE - Norvège)

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

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

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

II.3.3 PAPI (RedIris - Espagne)

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

II.3.4 Pubcookie (université de Washington)

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

II.4 Gestion de l'identité entre établissements

Les ressources web d'un établissement sont principalement destinées à ses personnels, enseignants, étudiants. Cependant des relations entre établissements se tissent pour le partage de ressources local qui doivent être accessibles à des personnes extérieures, ce qui requiert un contrôle d'accès donc une authentification. Pour cela, les architectures suivantes sont les pionnières [6] :

II.4.1 Liberty Alliance

L'authentification unique, dans le contexte Liberty Alliance, correspond à la possibilité pour l'utilisateur d'accéder, après s'être identifié à l'aide d'un compte unique, à des services proposés par différents fournisseurs appartenant à un même « cercle de confiance ». Ces spécifications techniques et fonctionnelles décrivent donc les mécanismes qui devraient d'une part simplifier l'usage d'Internet et d'autre part permettre à un individu de maîtriser la diffusion de ses données personnelles (nom, prénom, adresse électronique, etc.).

Une caractéristique spécifique de Liberty Alliance est la notion de fédération. Au lieu que ce soit un fournisseur de service qui décide si un utilisateur a le droit d'accéder à son service sans se ré-identifier, c'est l'utilisateur qui décide s'il veut accéder à ce service sans se ré-identifier. Cette possibilité ne peut être laissée à l'utilisateur que s'il doit au préalable s'identifier auprès d'un fournisseur d'identité reconnu par le fournisseur du service demandé. Cette caractéristique fait de Liberty Alliance un cadre pour la gestion d'identité, qui est utilisable dans un contexte d'applications d'entreprise étendue, où les utilisateurs délèguent la gestion des données personnelles à l'entreprise.

II.4.2 Shibboleth

Shibboleth est un mécanisme de propagation d'identités, développé par le consortium Internet2, qui regroupe 207 universités et centres de recherches. La propagation d'identités est la délégation de l'authentification à l'établissement d'origine de l'utilisateur et l'obtention de certains attributs de l'utilisateur.

La délégation de l'authentification réutilise les techniques de Single Sign On (redirection, cookies,...). Lors de l'accès initial à une ressource numérique, l'utilisateur est redirigé vers le service de découverte de la fédération, d'où il sélectionne son établissement d'origine ; il est ensuite renvoyé vers son fournisseur d'identités. Le pré-requis pour le fournisseur d'identités est de disposer d'un service d'authentification global tel que Central authentification Service (CAS).

À l'issue de la phase d'authentification, le fournisseur de services prend connaissance de l'identifiant de l'utilisateur qui lui permettra, lors d'une deuxième phase, d'obtenir ses attributs. Le fournisseur d'identités a la possibilité de définir, de façon différenciée pour chaque interlocuteur, quels attributs utilisateur pourront être dévoilés.

II.4.3 Microsoft Passport

Passeport est un moteur de SSO utilisable par tout site Web désireux de profiter du système d'authentification unique dont Les technologies utilisées sont les mêmes que les autres solutions de SSO web : redirections HTTP, cookies, tickets. La particularité est que les données utilisateurs sont stockées sur des serveurs hébergés chez Microsoft. Passeport n'est donc pas un produit mais un service.

Conclusion

Dans ce chapitre, nous avons constaté que pour apporter une meilleure solution attendue, nous allons mettre en place deux solutions d'authentifications existantes en utilisant, d'une part, le même référentiel utilisateur l'annuaire LDAP qui nous permettra d'unifier les informations nécessaires à l'authentification, d'autre part, un serveur d'authentification unique (CAS) pour unifier et centraliser les authentifications. CAS offre un très bon niveau de sécurité grâce à sa gestion de tickets.

Chapitre

III Mise en œuvre

Introduction

Ce projet s'inscrit dans le domaine de la sécurité informatique. Il vise à apporter une solution au problème de l'authentification diverse et répétitive auprès des applications existantes dans le réseau de l'université de Bejaia.

Notre travail consiste à implémenter un système d'authentification unique chargé de la centralisation des authentifications ainsi que l'accès des utilisateurs selon leurs statuts (enseignant, étudiant, employé simple, administrateur, etc.) avec une seule authentification pour toutes les applications concernées par le système. Afin d'atteindre notre but, on a eu recours à des outils et des méthodes déjà exposées dans les chapitres précédents (SSO CAS, annuaire LDAP, sécurisation SSL).

III.1 Spécification des besoins

Étant déjà nous-même des utilisateurs du réseau de l'université de Bejaia, on constate que nous avons un couple (nom utilisateur, mot de passe) pour chaque application et service web aux quel on a droit. En effet, le réseau héberge une plateforme "E-lerning", un service de messagerie Zimbra, un forum, des applications de gestion, ainsi que d'autres services envisageables, nécessitant tous une authentification lors de l'accès.

Une personne ayant droit d'accès à tous ces services et application se retrouve obligée d'une part à retenir une multitude d'identifiants et leurs mots de passes correspondants, et d'autre part, de s'authentifier autant de fois qu'il demande l'accès. De plus, la gestion des ces comptes utilisateurs (ajout, suppression ou modification) nécessite la mise à jour de plusieurs base de données. Il devient donc essentiel, de mettre en place un système répondant aux exigences suivantes :

- Centraliser les informations des utilisateurs sur une même base de données ou un annuaire.
- Unifier les authentifications et les habilitations de chaque utilisateur pour toutes les applications et services (un seul ID/mot de passe).
- Propager l'authentification d'une application vers les autres (authentification unique).

III.2 Notre travail

Afin de répondre aux attentes citées auparavant, notre travail consiste en la mise en œuvre d'une solution d'authentification unifiée et unique. Nous avons opté pour la solution CAS (Central Authentification Service) combiné à l'annuaire LDAP dont les fonctionnements ont été détaillés dans le deuxième chapitre. Le choix de cette solution est notamment dû aux points suivants :

- CAS est conçu pour rependre à des besoins similaires à nos objectifs.
- C'est une solution open source.
- CAS a déjà été essayé et fait ces preuves notamment dans d'autres universités (Rennes).
- Les modules CAS (client et serveur) sont légers et ne demandent pas trop de ressources.

Architecture de la solution

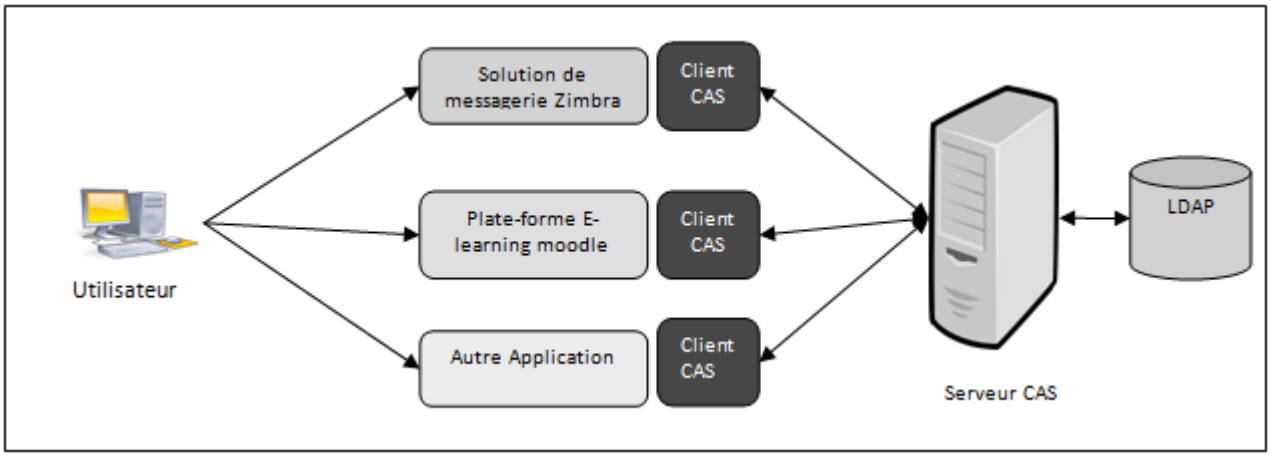


FIGURE III.1 – Conception de la solution.

Notre architecture est composée de :

- Un annuaire LDAP qui va contenir la liste de tous les utilisateurs avec la description de leurs droits d'accès.
- Un serveur CAS qui va assurer l'interrogation de l'annuaire LDAP, l'authentification des utilisateurs et la propagation de session.
- Un client CAS intégré à chaque application pour assurer la redirection de l'authentification vers le serveur CAS et faire le lien entre ce dernier et l'application.

III.3 Installation et configuration

Pour tester la solution proposée, nous avons besoin de deux machines virtuelles dans lesquelles nous installons un système d'exploitation LINUX. Une machine va contenir le serveur CAS et l'annuaire LDAP tandis que l'autre va contenir le service de messagerie

ZIMBRA et le client CAS ainsi que un serveur DNS (Bind9), voir la figure III.2. Cependant, la mise en place et le fonctionnement de ces derniers nécessitent le déploiement de quelques outils de base.

- **Pour serveur CAS** : Debian 6 comme système d'exploitation, Tomcat6, Openjdk-6, Maven-2.2.1, OpenLDAPv3, PhpLdapAdmin-1.2.2 et cas-server-3.4.11.

- **Pour le client** : Debian 5 comme système d'exploitation, Bind9, Zimbra 6 et cas-client-3.3.1.

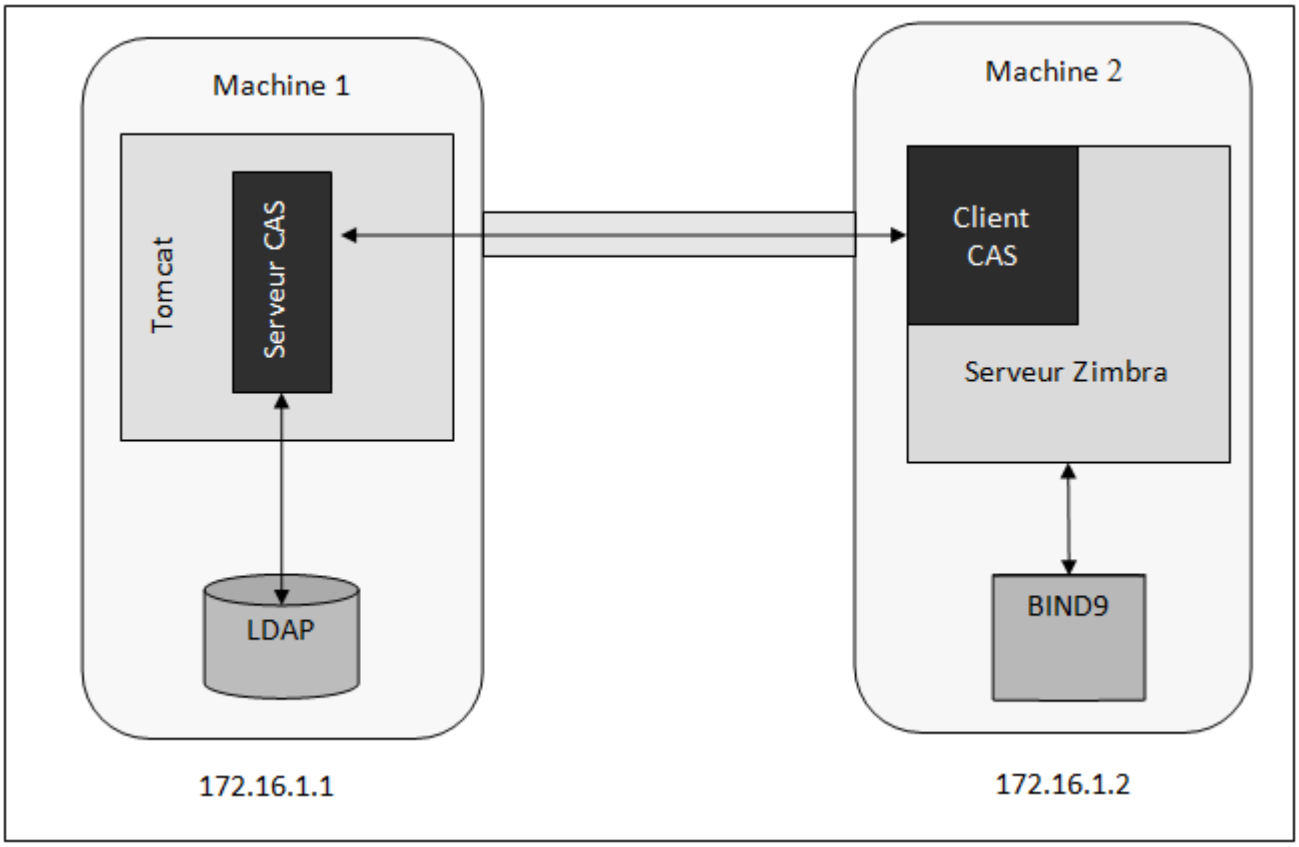


FIGURE III.2 – Architecture de test détaillée sur le service Zimbra.

Afin de connecter les deux machines, on leurs ajoute une carte réseau chacune, puis configure leurs interfaces comme suite : On édite le fichier /etc/network/interfaces pour rajouter :

1. **Pour la machine 1**

```
auto eth1
iface eth1 inet static
address 172.16.1.1
netmask 255.255.0.0
broadcast 172.16.255.255
```

Même chose pour la machine 2 avec l'adresse 172.16.1.2.

III.3.1 Sur la machine 1

Les points suivants nous présentent les composants pré-requis de la mise en place du serveur CAS, leurs utilités, ainsi que les détails de leurs installations et configurations.

A) **Apache Tomcat**

1. **Présentation :**

Apache Tomcat est un conteneur libre de servlets et JSP Java EE. Issu du projet Jakarta, Tomcat est un projet principal de la fondation Apache. Il implémente les spécifications des servlets et des JSP du Java Community Process1 . Il est paramétrable par des fichiers XML et de propriétés, et inclut des outils pour la configuration et la gestion. Il comporte également un serveur HTTP.

2. **Installation :**

Avant d'installer tomcat, nous devons vérifier si la JDK(java developement kit) est présent sur la machine :

```
# dpkg --get-selections | grep sun-java
```

Dans le cas ou la JDK n'est pas installée, il faut l'installer à partir d'un terminal par la commande suivante :

```
# apt-get install openjdk-6-jdk
```

Ensuite, nous installons la version la plus récente de Tomcat en utilisant la commande :

```
# apt-get install tomcat6
```

A ce stade, le serveur est installé, mais il n'est pas encore fonctionnel. En effet, Tomcat a besoin de la variable d'environnement JAVA_HOME et celle-ci n'est pas mise par défaut sur le système. Pour cela, nous éditons le fichier .bashrc en utilisant la commande :

```
# nano ~/.bashrc
```

Et nous rajoutons la ligne suivante :

```
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
```

A ce point, nous pouvons démarrer Tomcat. Pour le démarrer, il suffit de faire :

```
# /etc/init.d/tomcat6 start
```

Pour l'arrêter :

```
# /etc/init.d/tomcat6 stop
```

NB : nous devons ajouter un (manager-gui) pour pouvoir utiliser la page d'administration d'apache Tomcat, et cela en modifiant le fichier ./conf/tomcat_users.

B) **Maven**

1. **Présentation :**

Maven est indispensable afin de mettre en place CAS. Vu que CAS est un projet développé en Java et Maven est un outil open-source de gestion et d'automatisation de développement Java. Maven est utilisé pour compiler les sources du projet CAS et retourne en sortie un fichier (.war) qui est la servlet de déploiement du serveur CAS.

2. **Installation :**

Pour le mettre en place, il faut se rendre sur le site de Apache Maven :
http ://maven.apache.org/
Nous téléchargeons le fichier .tar.gz de la derniere version 2.2.1.
Nous décompressons le fichier apache.maven-2.2.1.tar.gz avec la commande :

```
# tar xvzf apache.maven-2.2.1.tar.gz
```

Puis, nous déplaçons le dossier dans /opt :

```
# mv  apache-maven-2.2.1 /opt
```

Après, nous ajoutons le bin directory au PATH :

```
# export PATH=/opt/apache-maven-2.2.1/bin:$PATH
```

Enfin, nous vérifions que Maven a bien été installé :

```
# mvn -version
```

C) **CAS**

Installation :
Pour le mettre en place nous devons récupérer les sources du projet, dans un terminal, on lance la commande :

```
# wget http://downloads.jasig.org/cas/cas-server-3.4.11-
release.tar.gz

# tar xvzf cas-server-3.4.11-release.tar.gz
```

On déplace le dossier dans /opt :

```
# mv cas-server-3.4.11 /opt
```

D) **OpenLDAP :**

1. **Présentation :**

OpenLDAP est une implémentation libre du protocole LDAP développée par The OpenLDAP Project. Nous utilisons la plus récente version qui est OpenLDAP v3.

2. **Installation :**

```
# apt-get install slapd ldap-utils
```

Pour démarrer Ldap, on utilise la commande :

```
# /etc/init.d/slapd start
```

Pour l'arrêter :

```
# /etc/init.d/slapd stop
```

3. **Configuration du serveur LDAP :**

Pour reconfigurer le serveur LDAP, nous utilisons la commande :

```
# dpkg-reconfigure slapd
```

4. **Tester le serveur LDAP :**

```
# ldapsearch -x -b dc=univ-bejaia, dc=dz
```

5. **Organiser l'annuaire :**

Pour organiser l'annuaire nous utilisons un fichier au format .LDIF
Le fichier init.ldif doit contenir le code suivant :

```
dn: ou=people, dc=univ-bejaia,dc=dz
objectclass: top
objectclass: organizationalUnit
ou: people
description: Branche gens

dn: ou=etudiants, ou=people, dc=univ-bejaia,dc=dz
objectclass: top
objectclass: organizationalUnit
ou: etudiants
description: Branche etudiants

dn: ou=personnels, ou=people, dc=univ-bejaia,dc=dz
objectclass: top
objectclass: organizationalUnit
ou: personnel
description: Branche personnel

dn: ou=services, dc=univ-bejaia,dc=dz
objectclass: top
objectclass: organizationalUnit
ou: services
description: Branche services

dn: ou=groupes, ou=services, dc=univ-bejaia,dc=dz
objectclass: top
objectclass: organizationalUnit
ou: groupes
description: Branche groupes
```

Dans les lignes précédentes de code, nous avons créé le nom de domaine "univ-bejaia.dz", dans le quel, on a ajouté des groupes d'utilisateurs.

Nous ajoutons le fichier ldif à l'annuaire avec la requête :

```
# ldapadd -x -f init.ldif -W -D cn=admin,dc=univ-bejaia,dc=dz
```

6. **Ajouter un utilisateur :**

Nous créons un fichier utilisateur.ldif qui contient le code suivant :

```
dn: uid=tarek, ou=etudiants, ou=people, dc=univ-bejaia,dc=dz
objectClass: top
objectClass: posixAccount
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tarek
cn: tarek
sn: tarek
givenName: tarek
uidNumber: 1100
gidNumber: 1111
homeDirectory: /home/tarek/ldap
loginShell: /bin/bash
userPassword: password
mail: tarek@univ-bejaia.dz
l: Bejaia
ou: mongroupe
```

Dans les lignes précédentes de code, nous avons ajouté un utilisateur "tarek" dans le sous groupe étudiants.

Nous ajoutons l'utilisateur dans l'annuaire :

```
# ldapadd -x -f utilisateur.ldif -W -D cn=admin,dc=univ-
bejaia,dc=dz
```

E) **PhpLdapAdmin**

A fin de simplifier l'utilisation de l'annuaire LDAP, nous avons installé PhpLdapAdmin qui est le client web permettant la gestion de l'annuaire LDAP. Il offre une vue hiérarchique sous forme d'arborescences et des fonctions avancées de recherche, qui facilite la navigation et l'administration de l'annuaire. Pour le mettre en place nous utilisons la commande :

```
# apt-get install phpldapadmin
```

Pour accéder à PhpLdapAdmin nous utilisons le lien suivant :
http ://localhost/phpldapadmin/

FIGURE III.3 – Page d'accueil PhpLdapAdmin.

III.3.1.1 Configuration de CAS-server pour supporter LDAP

a) **L'ajout de la dépendance**

```
# cd cas-server-3.4.11/cas-server-webapp

# gedit pom.xml
```

En ajoutant la portion de code XML suivante nous indiquons à CAS qu'on veut utiliser le module de connexion à un annuaire LDAP.

```
<!-- Dependance support LDAP -->

        <dependency>

        <groupId>org.jasig.cas</groupId>

        <artifactId>cas-server-support-ldap</artifactId>

        <version>3.4.11</version>

        </dependency>
```

b) **Configuration de la méthode d'accès à l'annuaire**

Nous allons maintenant configurer l'accès à l'annuaire. Pour cela, on édite le fichier deployerConfigContext.xml :

```
# gedit /opt/cas-server-3.3/cas-server-webapp/src/main/webapp/WEB-
INF/deployerConfigContext.xml
```

Et nous ajoutons le code suivant avant la dernière ligne « </beans> » :

```
<bean id="contextSource"
     class="org.springframework.ldap.core.support.LdapContextSo
urce ">
     <property name="pooled" value="false"/>
     <property name="urls">
     <list>
     <!—Nous insérons ici l'adresse de notre annuaire LDAP -->
     <value>ldap://127.0.0.1:389</value>
     </list>
     </property>
     <!--Cette ligne est à adapter selon le chemin où se trouve
     notre utilisateur -->
     <property name="userDn" value="dc=local"/>
     <!-- Renseignez ici le mot de passe de notre user -->
     <property name="password" value="festina"/>
     <property name="baseEnvironmentProperties">
     <map>
     <entry key="com.sun.jndi.ldap.connect.timeout"
value="3000" />     <entry key="com.sun.jndi.ldap.read.timeout"
value="3000"   />
     <entry key="java.naming.security.authentication"
     value="simple" />
     </map>
     </property>
</bean>
```

c) **Configuration de la méthode de recherche des login**

Il faut préciser la méthode utilisée pour rechercher un login dans l'annuaire. Supprimez cette portion de code dans le fichier deployerConfigContext.xml :

```
<bean
class="org.jasig.cas.authentication.handler.support.SimpleTestUser
namePasswordAuthenticationHandler" />
```

Nous le remplaçons par le code suivant :

```
<bean
     class="org.jasig.cas.adaptors.ldap.BindLdapAuthenticationHandl
     er">
     <property name="filter" value="uid=%u" />
     <!—Notre base c'est local donc DC=local -->
     <property name="searchBase" value="dc=local" />
     <property name="contextSource" ref="contextSource" />
     <property name="ignorePartialResultException" value="yes" />
</bean>
```

NB : Nous devons créer en parallèle un utilisateur dans la base de notre annuaire LDAP.

d) **Compilation de notre application CAS-server**

Les paramétrages sont terminés, il faut maintenant compiler l'application avec la dépendance du support d'authentification LDAP ainsi que les paramètres de connexion à l'annuaire.
Pour compiler l'application, nous utilisons Maven l'outil de management de projet Java. Nous devons retourner à la racine du projet, soit cas-server-webapp :

```
# cd /opt/cas-server-3.4.11/cas-server-webapp
```

Nous fixons Le PATH :

```
# export PATH=/opt/apache-maven-2.2.1/bin:$PATH
```

Nous devons corriger les chemins des fichiers de journalisations en modifiant le fichier log4j.xml :

```
# cd /opt/cas-server-3.4.11/cas-server-
webapp//src/main/webapp/WEB-INF/classes/
# gedit log4j.xml
<param name="File" value="/var/logs/tomcat6/cas.log" />
<param name="File" value="/var/logs/tomcat6/perfStats.log"/>
```

Ensuite, nous devons créer les deux fichiers de journalisation cas.log et perfStats.log :

```
# touch /var/logs/tomcat6/cas.log
# touch /var/logs/tomcat6/perfStats.log
```

En fin, nous lançons la compilation du projet Java : CAS

```
# mvn package install
```

La compilation va prendre un peu de temps car Maven calcule toutes les dépendances. La compilation doit se terminer ainsi :

```
------------------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------------------
Total time: 47 seconds
Finished at: Tue Jun 05 11:29:43 CEST 2012
Final Memory: 30M/53M
------------------------------------------------------------------------
```

e) **Déploiement de cas-server sur Tomcat**

A ce stade, nous pouvons copier le fichier cas.war qui est le résultat de compilation de notre application avec les nouvelles dépendances :

```
# cd /opt/cas-server-3.4.11/cas-server-webapp/
# cp target/cas.war /var/lib/tomcat6/webapps
```

Nous redémarrons Tomcat avec la commande :

```
# /etc/init.d/tomcat6 restart
```

Il faut s'assurer que notre annuaire LDAP a bien été démarré, dans le cas contraire, nous le redémarrons :

```
# /etc/init.d/slapd restart
```

Ensuite, nous pouvons tester notre application en utilisant le lien suivant :
http ://localhost :8080/cas/
Il doit nous rediriger vers la page d'authentification du CAS suivante :

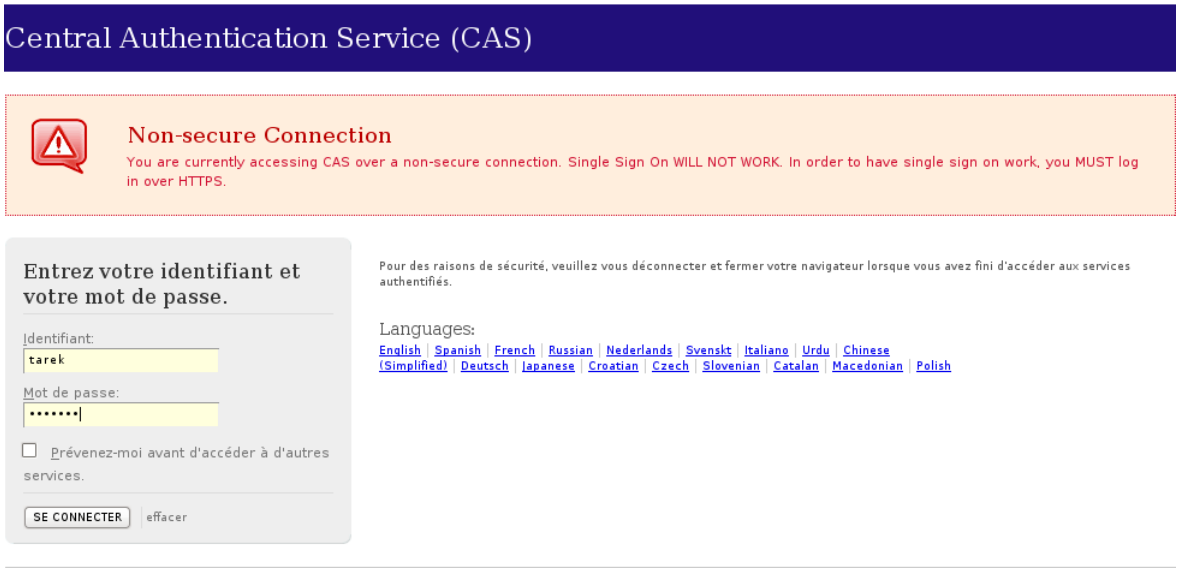


FIGURE III.4 – page d'authentification de cas-server.

III.3.2 Sur la machine 2

Avant de procéder à l'installation de Zimbra et pour que ce dernier soit fonctionnel, nous devons mettre en place un serveur DNS, dans notre cas, nous utilisons le serveur Bind9.

III.3.2.1 Bind9

1. **Présentation :**

BIND (Berkeley Internet Name Domain) fournit un service de résolution de noms grâce au service /usr/sbin/named. Il s'agit du serveur DNS de référence sur les systèmes Linux.

2. **Installation :**

```
# apt-get install bind9
```

3. **Configuration :**

- Modification du fichier named.conf :

Nous ajoutons les deux zones :

```
zone " univ-bejaia.dz " {

type master ;

file "  /etc/bind/zones/univ-bejaia.dz.hosts " ;

} ;

zone " 16.172.in-addr.arpa " {

type master ;

file "/etc/bind/zones/rev.172.16.in-addr.arpa " ;

} ;
```

◦ Créations des deux fichiers dans la zone :

```
# cd /etc/bind/

# mkdir zones

# cd zones/

# gedit univ-bejaia.dz.hosts
```

```
$ttl 86400
@ IN SOA zimbra.univ-bejaia.dz. root.univ-bejaia.dz. (
2012062003
21600
3600
604800
86400 )

;ENREGISTREMENT "A" DNS <-> IP CLASSIQUES
@ IN NS zimbra.univ-bejaia.dz.

        IN MX 10 zimbra.univ-bejaia.dz.
        IN A 172.16.1.2

mail IN A 172.16.1.2
zimbra IN A 172.16.1.2
casserver IN A 172.16.1.1
```

Toujours dans le dossier "zones" :

```
# gedit rev.172.16.in-addr.arpa
```

Nous ajoutons le code suivant :

```
@ IN SOA univ-bejaia.dz. admin.univ-bejaia.dz. (
2012062003
28800
604800
604800
86400)

@ IN NS zimbra.univ-bejaia.dz.

2.1 IN PTR zimbra.univ-bejaia.dz.
```

III.3.2.2 Zimbra

1. **Présentation :**

Zimbra est un logiciel serveur collaboratif qui permet à ses utilisateurs de stocker, organiser et partager rendez-vous, contacts, emails, liens, documents, etc. Un des nombreux avantages de Zimbra est qu'il installe (et utilise) beaucoup d'applications externes de manière très simple. Par exemple, lors de l'installation de Zimbra, un serveur mail Postfix est automatiquement installé et configuré. Dès lors, pour y accéder, on utilise le client mail inclus à Zimbra (ou un client externe via POP ou IMAP, également installés par Zimbra). De même, il installe et configure un serveur LDAP sans l'intervention de l'utilisateur.

En janvier 2007, l'éditeur de Zimbra affirmait avoir 6 millions d'utilisateurs. Le 17 septembre 2007, Yahoo a racheté Zimbra pour 350 millions de dollars. Depuis, Zimbra est disponible sous deux formes : une version payante et une version Open Source. Fonctionnalités principales de Zimbra, sont les suivantes :

- Serveur et client mail (ainsi que POP et IMAP).
- Calendrier
- Contacts partagés
- Calendriers partagés
- Notes partagées
- Gestion du multi-domaine
- Synchronisation Outlook, Apple iCal...
- Synchronisation PDA, iPhone, Blackberry.

2. **Installation :**

Après avoir installé et configuré notre serveur DNS, nous devons récupérer la source du projet du site officiel de Zimbra : www.zimbra.com/downloads/os-downloads.html
Après avoir téléchargé la version étant compatible avec notre système (Debian 5), nous devons l'extraire avec la commande suivante :

```
# tar xvzf zcs-6.0.16_GA_2998.DEBIAN5.20120404131543.tgz
```

Ensuite en rentre dans le dossier et on exécute le programme d'installation :

```
# cd zcs-6.0.16_GA_2998.DEBIAN5.20120404131543
# ./install
```

Voici la page d'authentification de Zimbra(cf. Figure III.5) :

FIGURE III.5 – Page d'authentification de Zimbra.

III.3.2.3 CAS-ification de Zimbra

La CAS-ification d'une application web est une chose aisée et légère. Des librairies clientes sont disponibles afin de faciliter cette opération.

A) **Importer la librairie cas-client**

Cette librairie est utilisable pour l'implémentation personnalisée des fonctionnalités du CAS, et pour simplifier la CAS-ification de l'application WEB en appliquant un filtre.

- Télécharger la librairie depuis http ://www.ja-sig.org/downloads/cas-clients La version client 3.1.x fonctionne bien avec Zimbra 6.0.x et serveur CAS 3.3.x..
- Copier le CAS-client-core-3.1.x.jar dans /opt/zimbra/jetty/common/lib

```
# cp CAS-client-core-3.1.x.jar /opt/zimbra/jetty/common/lib
```

B) **Modification du fichier web.xml**

1. **Zimbra webapp**

Nous ajoutons les lignes suivantes à /opt/zimbra/jetty/etc/zimbra.web.xml.in avant la première section <servlet> (environ la ligne 230), Les ports par défaut sont 8443 pour le serveur CAS et 443 pour le client Zimbra Web (ou 80 si HTTP est utilisé au lieu de HTTPS) :

```
<filter>
 <filter-name>CasSingleSignOutFilter</filter-name>
  <filter-class>
org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>


<filter-mapping>
 <filter-name>CasSingleSignOutFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>


 <listener>
  <listener-
 class>org.jasig.cas.client.session.SingleSignOutHttpSessionListene
 r</listener-class>
 </listener>
```

```
<filter>
 <filter-name>CasAuthenticationFilter</filter-name>
  <filter-
class>org.jasig.cas.client.authentication.AuthenticationFilter</fi
lter-class>
    <init-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value>http://casserver:8080/cas/login</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://univ-bejaia.dz/zimbra</param-value>
    </init-param>
</filter>
```

Nous avons indiqué l'adresse de la page *login* du serveur CAS et l'adresse de la page cliente de Zimbra.

```
<filter-mapping>
    <filter-name>CasAuthenticationFilter</filter-name>
    <url-pattern>/public/preauth.jsp</url-pattern>
</filter-mapping>
```

```
<filter>
    <filter-name>CasValidationFilter</filter-name>
    <filter-
class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketVal
idationFilter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>http://casserver:8080/cas </param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://univ-bejaia.dz/zimbra</param-value>
    </init-param>
    <init-param>
        <param-name>redirectAfterValidation</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>


 <filter-mapping>
     <filter-name>CasValidationFilter</filter-name>
     <url-pattern>/*</url-pattern>
 </filter-mapping>
```

Nous avons indiqué l'adresse du serveur CAS.

```
<filter>
    <filter-name>CasHttpServletRequestWrapperFilter</filter-name>
    <filter-
class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</f
ilter-class>
</filter>
```

```
<filter-mapping>
    <filter-name>CasHttpServletRequestWrapperFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

2. **ZimbraAdmin Webapp :**

Nous ajoutons les mêmes lignes précédente dans /opt/zimbra/jetty/etc/zimbraAdmin.web.xml.in avant la première section <servlet> (environ la ligne 230), Les ports par défaut sont 8443 pour le serveur CAS et 7071 pour la console d'administration de Zimbra.

C) **Création du fichier PreAuth key**

Nous exécutons la commande suivante :

```
# su - zimbra
# zmprov gdpak univ-bejaia.dz
```

D) **Création des fichiers preauth.jsp**

– **Zimbra Webapp**
le fichier preauth.jsp-zimbra (le télécharger à partir de
"https ://wiki.jasig.org/display/CAS/CASifying+Zimbra" dans "/opt/zimbra/jetty/webapps/zimbra/public/preauth.jsp".
Remplacer le DOMAIN_KEY avec la clé préalablement générer avec "zmprov" :

```
public static final String DOMAIN_KEY =”la clé déjà générée”
```

– Remplacer le nom de domaine à la ligne 90 (dans notre cas on met local).
– Exécuter la commande :

```
# chown zimbra:zimbra
/opt/zimbra/jetty/webapps/zimbra/public/preauth.jsp
```

– **ZimbraAdmin Webapp**
Meme étapes avec Zimbra Webapp sauf que ici nous utilisons le fichier preauth.jsp-zimbraadmin et le dossier /opt/zimbra/jetty/webapps/zimbraAdmin/public/

E) **Remplacer les URLs de login et de logout**

```
# zmprov md univ-bejaia.dz zimbraWebClientLoginURL
http://univ-bejaia.dz/zimbra/public/preauth.jsp
# zmprov md univ-bejaia.dz zimbraWebClientLogoutURL
http://casserver.local:8080/cas/logout
# zmprov md univ-bejaia.dz zimbraAdminConsoleLoginURL
https://zimbra:7071/zimbraAdmin/public/preauth.jsp
# zmprov md univ-bejaia.dz zimbraAdminConsoleLogoutURL
http://casserver.local:8080/cas/logout
```

F) **Redémarrer Zimbra**

```
# zmcontrol restart
```

Résultat final de l'exécution :
Nous essayons d'accéder au service de messagerie Zimbra avec l'URL :
http ://univ-bejaia.dz/zimbra.
le navigateur nous redirige vers la page d'authentification du serveur CAS (cf. Figure III.6).

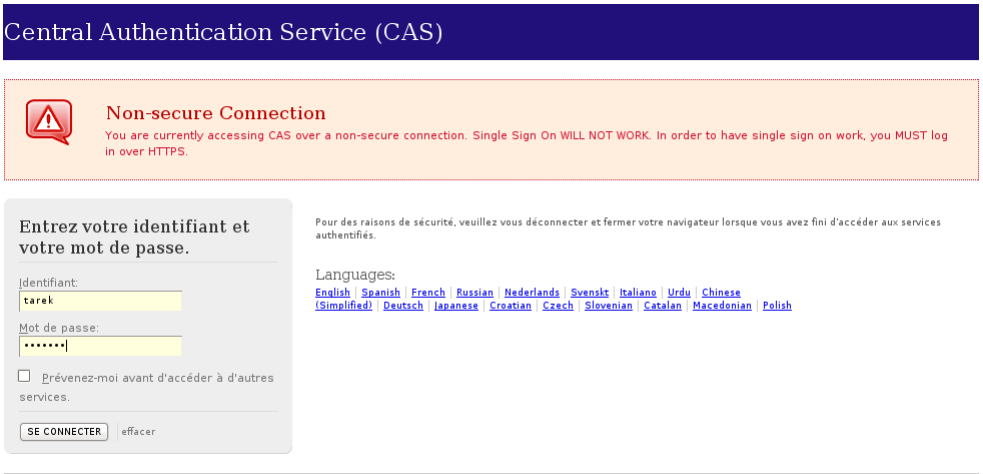


FIGURE III.6 – page d'authentification de CAS.

Si les informations transmises sont correctes, le navigateur nous redirige vers la page d'accueil de Zimbra (cf. Figure III.7).

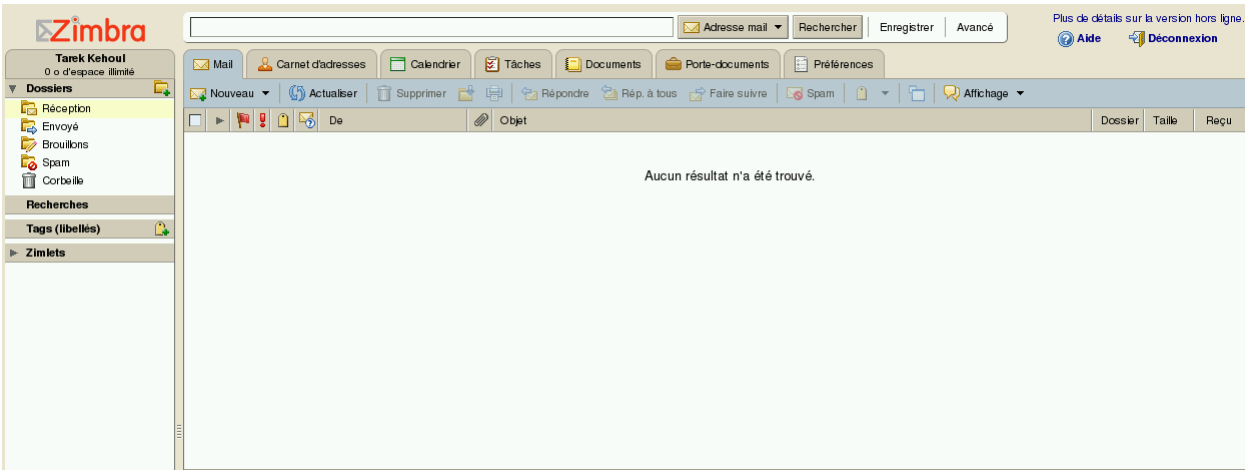


FIGURE III.7 – page d'accueil de Zimbra.

III.4 Difficultés rencontrées

Durant ce projet, nous avons rencontré quelques problèmes concernant :
- Le manque de documentations et spécialement sur le serveur CAS.
- Les dépendances entre les versions des composants installés.

III.5 Tolérance aux pannes

On constate après la mise en œuvre de CAS qu'il est l'unique point d'authentification du système d'information, une panne ou un arrêt de ce service SSO interdit donc tout accès au applications nécessitants une authentification. Pour y remédier à ce problème plusieurs alternatives existent, l'une d'elles consiste à mettre deux machines en cluster qui se surveille et partage des données grâce a des disques en raid réseau (drdb). En cas de panne d'une des machines, c'est l'autre qui prend le relais, cependant les utilisateurs doivent se ré-authentifier.

Conclusion

Dans ce chapitre, nous avons expliqué les étapes de la mise en œuvre de notre projet, commençant par l'installation et la configuration des outils nécessaire sur la machine hébergeant le serveur CAS et l'annuaire LDAP, en suite, nous avons mis en place une deuxième machine dans laquelle on a installé un service de messagerie. Comme étant un client, Zimbra nous permet de tester le bon fonctionnement de notre service d'authentification.

Conclusion générale et perspectives

Dans le domaine de la sécurité informatique, il est difficile de mettre en œuvre une solution sans lacunes qui répond parfaitement aux besoins ressentis. C'est pour cela que les administrateurs travaillent sans relâche, afin d'améliorer le niveau de sécurité du système dont ils sont responsables.

Au cours de notre étude qui a été consacrée au système de sécurité garantissant le service de l'authentification, nous avons constaté l'importance de celui-ci dans le contrôle d'accès aux services demandés par les utilisateurs et ainsi minimiser le risque d'intrusion.

Cette étude nous apprend aussi, qu'un système d'authentification unifiée est impérativement indispensable dès qu'un système informatique héberge une multitude d'applications nécessitant une authentification. Un tel système ne peut qu'être bénéfique d'une part pour le niveau de sécurité en minimisant la circulation des informations d'authentification, d'autre part pour les administrateurs en termes de gestion et les utilisateurs en termes d'ergonomie.

A l'issue de notre travail ayant pour objectif la mise en place d'une solution d'authentification SSO pour certaines applications web de l'université, nous constatons que l'alliance entre les deux outils LDAP et CAS répond bien à nos objectifs.

La mise en œuvre de ce projet nous a permis d'apporter une petite contribution à l'université. Mais aussi, de mettre à l'épreuve nos connaissances acquises durant notre cursus, ainsi que d'emmagasiner de nouvelles connaissances notamment dans l'art de la configuration sous la plateforme Linux, de plusieurs service tels que le réseau, Bind, Openldap, Tomcat, Zimbra, et les applications web Java.

Enfin, comme perspectives pour ce projet : Pour une meilleure tolérance aux pannes, nous prévoyons une architecture incluant une méthode de clustering. Cette architecture consiste à mettre en place deux serveurs d'authentification CAS, un serveur principal et un auxiliaire qui se surveillent. En cas de panne du serveur principal, le serveur auxiliaire prend le relai et devient principal.

Nous souhaitons également étendre ce projet en exploitant mieux les services qu'offre un annuaire LDAP, notamment relier tous les serveurs d'authentification possible y compris le contrôle d'accès au réseau ainsi que la gestion des habilitations des utilisateurs.

Bibliographie

[1] M. Pybourdin, E. Macé, D.Sztykman, V. Steenhoute , Ouvrage, Linux Administration système et réseau, DUNOD, 2 édition 2008.

[2] S. Ghernaouiti-Hélie, Ouvrage, Sécurité informatique et réseau, DUNOD, 2 edition 2008.

[3] C. Llorens, L. Levier, D. Valois, Ouvrage, Tableau de bord de la sécurité réseau, DUNOD, 2006.

[4] S. Bordères, Ouvrage, Authentification réseau avec radius, EYROLLES, 2006.

[5] G. Plouin, J. Soyer, M-E. Trioullier, Ouvrage, Sécurité des architectures web, DUNOD, 2004.

[6] O. Salaün, Article, Introduction aux architectures web de Single Sign-on, 15 Octobre 2003.

[7] V.Mathieu, P. Aubry, J. Marchal, Article, Single Sign-On open-source avec CAS, 12 Octobre 2003.

[8] M. Rizcallah, Ouvrage, Annuaires LDAP, EYROLLES, 2002.

[9] S. Natkin, Ouvrage, Les protocole de sécurité d'internet, DUNOD, mai 2002.

[10] Paul C. Kocher , article, Cryptography Research, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems ".

Résumé

Dans le domaine de la sécurité informatique, les administrateurs travaillent afin de rendre les systèmes plus robustes en essayant d'appliquer au mieux les principes de base de la sécurité.
Ce mémoire traite la problématique de l'authentification multiple auprès d'applications se trouvant dans le même système informatique. Notre objectif est l'implémentation d'une solution d'authentification unique pour des applications de l'université (A. Mira). Cependant, ce travail ne peut se réaliser sans faire un petit rappel sur les fondements de la sécurité informatique ainsi que les notions cryptographiques, cela afin de bien comprendre les concepts répondant à la problématique.
Pour la solution qu'on a implémentée sous Linux, nous avons choisi une méthode et des outils dont le fonctionnement est exposé dans la partie théorique, notamment une méthode SSO avec un serveur d'authentification CAS et pour stocker les informations concernant les utilisateurs on a utilisé l'annuaire LDAP.

Mots clés : CAS, LDAP, SSO, Authentification unique.

Abstract

In the field of IT security, administrators are working to make the systems more robust by trying to apply the most basic principles of safety.

This report addresses the problem of authentication for multiple applications in the same information system. Our goal is to implement a unified authentification solution for applications of the university (A. Mira). However, this work can ?t be achieved without doing a little recall of the fundamentals of IT security and cryptographic concepts, in particular, to understand the concepts spreading to the problem.

For the solution that we have implemented on Linux, we chose a method and tools whose operation is explained in the theoretical part, a particular SSO method with a central authentication server (CAS) and to store user information in an LDAP directory.

Keys words : CAS, LDAP, Sing Sign On.