AUTHENTIFICATION D'UTILISATEUR
GESTION DES IDENTITES

UNIVERSITE SCIENCES ET TECHNOLOGIES BORDEAUX 1
351 Cours De La Libération
33400 TALENCE

Décembre 2007

SOLANKI Jigar
AKPAKPO Brunel

In God we Trust
All others must submit an X.509 certificate.

Table des matières

Table des figures

Liste des tableaux

Introduction

Lorsque l'on se connecte sur un réseau d'entreprise, d'université ; qu'on consulte nos emails ; qu'on allume notre téléphone portable ; ou encore qu'on effectue un paiement par carte bancaire, on rentre dans la plupart des cas un couple *login* et *password* (code pin, code de carte bancaire, mots de passe, etc.) :on s'authentifie.
Le mécanisme d'authentification permet de confirmer les droits et privilèges d'une entité, qu'il s'agisse d'un utilisateur, d'une machine ou encore d'un processus.

Qui : A qui/quoi autorise t-on l'accès ?

Quoi : Une fois l'accès autorisé, à quoi a t-on réellement accès suivant les privilèges qu'on a ?

Où : L'accès à un objet est-il autorisé de la même manière que l'on soit connecté en LAN ou en WiFi ?

Comment : Lorsque l'on change notre mot de passe, on modifie un fichier qui n'est accessible en écriture que par le super-utilisateur. Comment y avons nous accès ?

Néanmoins, avec l'accroissement des systèmes informatiques et surtout le nomadisme qui apparaît de plus en plus nécessaire dans les entreprises, les experts en sécurité informatique sont unanimes sur le fait qu'un couple *login* et *password* est insuffisant et comporte de nombreuses faiblesses. Il est donc nécessaire d'avoir des solutions plus robustes si l'on veut obtenir une preuve quasi-irréfutable de l'identité de toute personne ou plus généralement de toute entité qui éxécute une opération, qui accède à un contenu.

Notre but ici est de faire un état de l'art général en matière d'authentification et de sécurité de l'information, en exposant les principes de bases, afin de mieux cerner les briques sur lesquelles reposent certaines solutions robustes (ou qui le sont devenues). L'intérêt n'est pas d'adopter une solution plutôt qu'une autre, mais de décrire les plus parlantes en matière d'authentification afin de prendre le recul nécessaire et construire une bonne méthodologie en terme de politique de sécurité.

Dans cette optique, nous présenterons dans un premier temps une définition de l'authentification pour comprendre la nécessité d'avoir d'autres procédés que le login et le mot de passe pour confirmer l'identité d'une personne ; d'autant plus que les enjeux sont nombreux et à tous les niveaux. En outre, suivant différents mécanismes de preuves d'identités, nous verrons la manière dont sont implémentés certains procédés, notamment le projet PAM sous Linux, le projet Kerberos, ou encore Radius.

Chapitre 1

Authentification : Enjeux et Polémiques

Lorsque l'on établit une politique de sécurité des systèmes d'informations, il est, entre autres, nécessaire d'avoir une politique d'authentification robuste des utilisateurs. En effet, cela est indispensable si l'on veut avoir une bonne traçabilité des actions de chaque utilisateur et ainsi de pouvoir garantir une bonne confidentialité et intégrité du flux de données de l'entreprise. L'approche classique consisterait à attribuer un code d'accès à chaque utilisateur. Néanmoins, cela s'avère insuffisant lorsqu'il s'agit de mettre en place de grosses structures, comme par exemple un réseau inter-universitaire, ou une entreprise possédant plusieurs sucursalles. On définit alors plusieurs niveaux et procédés d'authentification en fonction des besoins.

1.1 Authentification Simple, Forte

Un **service d'authentification** repose sur l' **identification** qui définit l'ensemble des utilisateurs, et de l'**authentification** en elle-même dont le rôle est de confirmer l'identité présumée de l'utilisateur. Il s'agit donc d'un challenge (Fig.1.1) entre un *prouveur* (l'élément qui veut s'authentifier) et un $verifieur$ (l'élément qui authentifie). On les nomme ainsi pour éviter la confusion entre machines et utilisateurs car dans les deux cas, il peut s'agir de personnes physiques, de personnes morales, de machines, de programmes, de modules, de navigateur, etc.

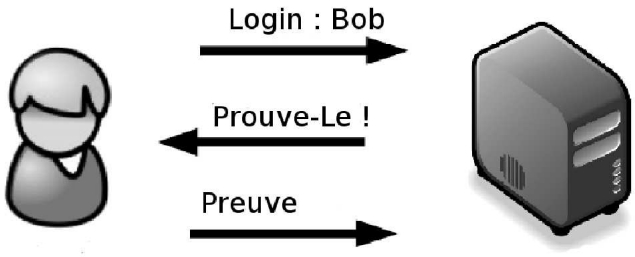

FIG. 1.1 – *Challenge d'authentification*

On classe alors les différentes preuves possibles en plusieurs catégories appellées **facteurs d'authentification** :

Ce qu'il sait : Il s'agit dans la plupart des cas d'un mot de passe, mais peut très bien être une toute autre information qu'il aurait en sa possession (son adresse, son numéro de sécurité sociale, etc.)

Ce qu'il possède : On parle ici de moyens d'authentification par hardware comme les clés USB, les cartes d'accès, des systèmes munis d'émetteurs qui déclenchent l'ouverture de porte dans un rayon determiné, etc.)

Ce qu'il est : Ceci comprend notamment les paramètres physiques de l'utilisateur comme une empreinte digitale, un scan rétinien, une reconnaissance vocale, etc.

Ce qu'il peut faire : Ceci englobe les gestes que l'utilisateur pourrait faire, les comportements qu'il pourrait avoir comme un signe de la main ou de la tête.

Ces différents moyens ont chacun leurs avantages et inconvénients, mais permettent d'avoir une flexibilité dans la mise en oeuvre d'une politique de sécurité. En effet, une empreinte digitale exigera la présence physique de l'utilisateur, tandis qu'un ticket Kerberos peut être prêté pour une durée determinée (3.1.2).

On parle d'authentification **simple** lorsqu'un seul de ces facteurs est nécessaire, et d'authentification **forte** lorsque plus de deux le sont. De nos jours, l'authentification forte est devenue un enjeu majeur, en particulier dans la protection de données sensibles ou encore dans l'accroissement des échanges éléctroniques.

1.2 Enjeux de l'authentification

Les enjeux sont très nombreux et répondent notamment à des besoins de confidentialité à tous les niveaux : Secret Défense, protection de la vie privée, développement

économique, etc.

1.2.1 Protection des droits d'auteurs

Dans le début des années 90, l'accès à des nouvelles technologies par le plus grand nombre a provoqué une recrudescence du piratage de l'information (principalement sur les supports CD/DVD) et surtout de sa diffusion sur la Toile. Il existe aujourd'hui 3 principaux médias où on authentifie l'utilisateur et on lui attribue certains privilèges, ou plutôt qu'on lui en retire certains. Il s'agit du *son* (musique principalement), de la *video* (films principalement) et des *logiciels* (jeux vidéos principalement). Plusieurs solutions basées sur de l'authentification ont été proposées et sont implémentées sur des supports numériques pour la protection des données contre la copie illégale.

DRM : Les **D**igital **R**ight **M**anagment correspondent à un mécanisme de droits et privilèges numériques. On authentifie l'utilisateur et on lui attribue des droits restreints (il ne peut lire que sur une plateforme en particulier par exemple).

SafeDisc : C'est une technologie(Fig.1.2) qui authentifie la validité des données en fonction d'une clé d'authentification présente sur le support et impossible à copier. Le CD doit donc être présent dans le lecteur pour la lecture et la gravure. S'il ne l'est pas, les données ne seront pas lisibles, même si elles sont copiées.

SecuROM : Produite par Sony (Fig.1.2), elle est également basée sur un système d'authentification. Ici le code d'identification éléctronique (CIE),impossible à copier, est gravé en dur dans les usines et l'exécutable de l'application installée sur le support est chiffré. En cas de copie, si le CIE n'est pas présent, l'application ne fonctionnera pas.

Les protections basés sur l'authentification vont au-delà de ces quelques exemples. D'autre part, le gouvernement a pris également des mesures nécessaires pour lutter contre les contenus illégaux sur la Toile, en particulier contre le Peer-2-Peer.

1.2.2 Loi pour la « confiance dans l'économie numérique »

Le 24 janvier 2006 était mise en vigueur la loi pour la confiance dans l'économie numérique (**LCEN**) dont le texte en entier est disponible à [5]. En effet, elle visait principalement à engager la responsabilité des hébergeurs et des fournisseurs d'accès

FIG. 1.2 – *Protections basées sur des méthodes d'authentifications*

internet en matière de notification de contenu illicites.
Il y eût également des polémiques par rapport à la surveillance de la correspondance privée pour le courrier éléctronique. En outre, cela obligeait les opérateurs de communications numériques (téléphonie mobile, FAI...) à archiver les identités, pseudonymes, mots de passe et toute autre information les concernant pendant une durée d'un an. Ceci montre bien les infrastructures mises en jeux en terme de gestion d'identité des personnes, d'authentification et de protection de la vie privée où la frontière devient de plus en plus floue.

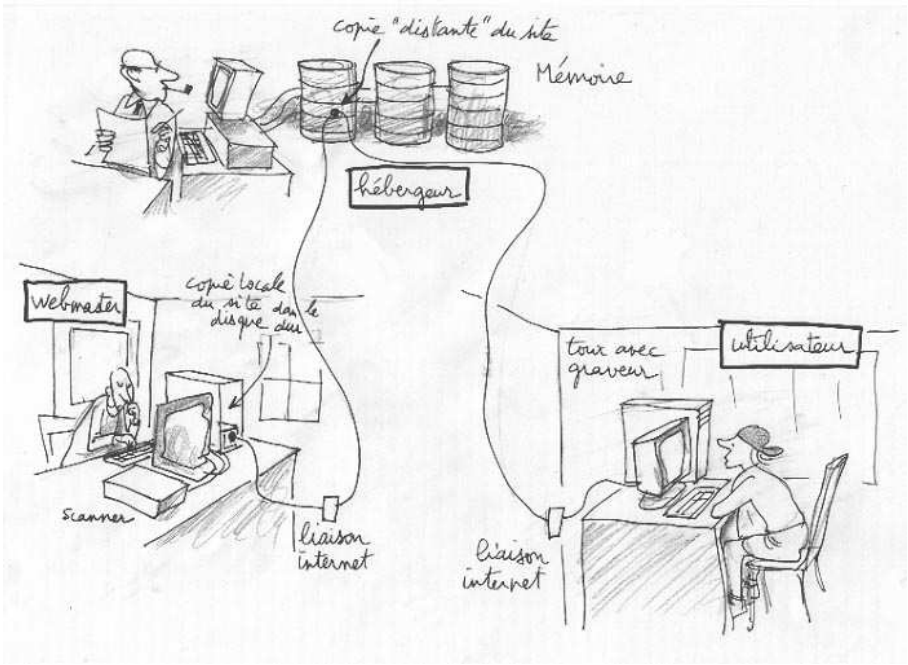

FIG. 1.3 – *Loi LCEN : Protection contre la diffusion de contenu illicite.*

1.2.3 RFID : La controverse autour des implants

La technologie **Radio Frequency Identification** (RFID) est souvent associée à une puce éléctronique ou une etiquette adhésive pour permettre de faire de l'authentification basée sur des ondes radios, c'est à dire recevoir/répondre à des requêtes radio entre un émetteur et un récépteur (Fig.1.4).

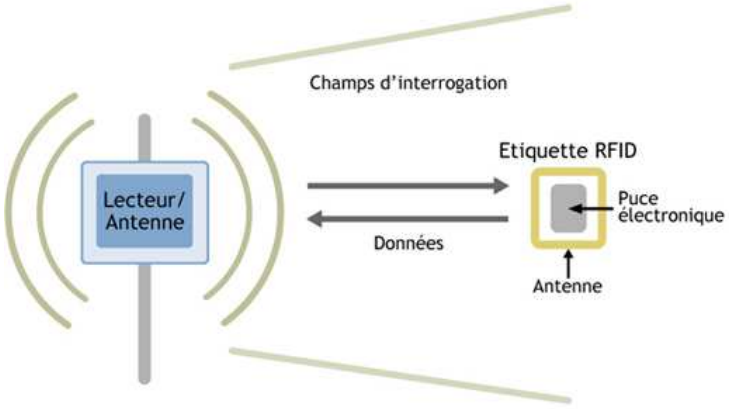

FIG. 1.4 – *Principe du fonctionnement de la RFID*

Cette technologie a d'abord été utilisée chez les vétérinaires, notamment aux Etats Unis à partir de 2004. La lecture des puces implantées sous la peau permettait d'accéder rapidement aux dossiers des animaux. Le firme américaine Applied Digital Solutions a alors développé une puce similaire pour le milieu hospitalier. Le concept était de pouvoir lire rapidement le dossier d'un patient quand il était par exemple inconscient. Outre le fait que se faire implanter une puce n'enchante pas le commun des mortels, un réel problème était posé par l'atteinte à la vie privée si jamais les données n'étaient pas protégées correctement sur ces puces. Il fallait avoir une authentification à la fois robuste et qui puisse tenir sur une petite puce ayant assez d'energie pour fonctionner.

En France, on comprend à présent, de part ces enjeux dans des domaines variés, pourquoi l'authentification fait partie des ***Technologies-Clés 2010***[1] . Beaucoup sont basées sur des $clefs$ d'authentification. Comment fonctionnent ces clés ? Où sont elles stockées ? Qui peut y accéder ?

[1]Technologies clés 2010 est une étude de la DIGITIP du Ministère Français de l'Industrie et des Finances, présidée par André Lebeau, chef de projet DIGITIP Phillipe Bourgeois.

Chapitre 2

Gestion des identités, PKI

Nous l'avons vu, depuis l'explosion des moyens de communications numériques, il a fallu trouver des solutions pour garantir la sécurité des données traitées et vérifier l'identité des personnes correspondantes.
Comment savoir si un e-mail à bien été envoyé par cette personne légitime ?
Comment garantir que le site web que l'on visite est bien le bon et non pas une redirection d'un pirate ?
On pourrait par exemple signer un mail numériquement. Que se passerait-il si la signature était falsifiée au cours du transport du mail ? Qu'arriverait-il si quelqu'un aposait ma signature sur un document ?
Les interrogations sont multiples et posent toutes un problème de gestion des identités des utilisateurs. Nous verrons dans ce chapitre à quoi sert le mécanisme de certificats et comment permet-il d'authentifier de manière irrévocable une personne, notamment à travers les *certificats X*509 et le logiciel *Pretty Good Privacy* (PGP).

2.1 Authentification et PKI

Lorsque l'on veut attester de l'identité d'une entreprise, d'une personne physique ou morale, on a souvent recours aux certificats numériques. Un certificats peut être assimilé à une carte d'identité de celui qui le détient. Les ***PKI-Public Key Infrastructure*** sont tout un ensemble de mécanismes et de procédés pour la gestion de certificats. Ceci permet notamment aux entreprises de protéger leur infrastructure par des transactions parfaitement authentifiées.

Une **PKI** est une autorité de confiance : elle doit pouvoir gérer les certificats (création, renouvellement, révocation, etc) et être capable d'identifier, authentifier et publier les listes d'utilisateurs (y compris les listes de révocation pour qu'un certificat révoqué ne soit pas utilisable). La notion de confiance est reportée sur un tiers (cette autorité) et l'ensemble de la communauté s'accordent sur le fait que cette autorité prédomine en matière d'attribution de certificat. Nous verrons que c'est le mécanisme adopté par la norme X509. D'autres mécanismes existement également pour ne pas avoir à repercuter la notion de confiance dans une entité qu'on connait finalement assez peu : c'est le mécanisme adopté par PGP, il s'agit de la notion $reseau\ de\ confiance$.

Cette autorité de confiance sert donc à signer les demandes de certificats des utilisateurs. L'autorité de confiance étant connue de tout le monde, il sera possible pour un utilisateur *lambda* de vérifier la signature et donc de confirmer l'identité de l'utilisateur. Nous verrons comme cela fonctionne un peu plus en détails dans la section suivante consacrée aux certificats X509.

Un certificat éléctronique est une donnée publique consultable par tout-un-chacun. A ce certificat est associée une clé privée détenue uniquement par le propriétaire du-dit certificat. On retrouve dans le certificat une clé publique et les informations d'identité de la personne ou entreprise : Nom, Raison Sociale, Date de naissance, etc. C'est l'ensemble clé publique + informations de l'utilisateur qui sont alors signés par l'autorité de certification (AC) qui en quelque sorte confirme, certifie l'identité et permet une authentification déterministe.

2.2 Certificat X509

L'objet d'un **certificat** est, pour résumer, de certifier une clé publique associée à l'identité d'un utilisateur. Autrement dit, pour utiliser la clé publique, il faut que l'on soit sur de l'identité de la personne. Pour cela, on peut soit avoir une relation de confiance directe avec l'interlocuteur, soit, comme souvent, avoir recours à un tiers de confiance qui certifie de l'identité de la personne en signant la clé publique : **X509** est un standard de certificat émis par ces tiers. Cette norme est utilisée, entre autre, dans SSL/TLS, IpSec, S/MIME, LDAP, etc.
Un certificat X509 ne dépend pas d'un protocole cryptographique en particulier. Il pose néanmoins la signature de la clé publique par une fonction de hachage (que l'on peut choisir).

Les principaux champs contenus dans X509 sont présentés dans le tableau 2.1.

Version : Contient le numéro de version du certificat. Version 1, 2 ou 3.

N° Série : Il s'agit d'un numéro de série unique et identifie le dit certificat chez l'AC.

Nom Créateur : est le nom du créateur du certificat.

Validité : Le certificat n'est valide ni avant, ni après la période indiquée.

Nom : Il s'agit du nom *usuel* du détenteur :Nom civil, nom de la société, nom du navigateur, de la passerelle . . .Il est donc possible qu'il y ait deux certificats ayant le même champ **Nom**.

Signatures : Il s'agit là du champ le plus **important** :la signature de l'AC. Cette signature est fabriquée à partir de la **clé privée** de l'AC (pour garantir que la signature provient bien de l'AC) et porte sur un hachage de l'ensemble des informations contenues dans le certificat, y compris la clé publique de l'utilisateur !

Version
N° Série
Algorithm de signature
Nom Créateur
Période de validité
Nom du sujet
Clé Publique & Infos complémentaires
ID *unique du créateur*
Extension
Signatures de l'AC

TAB. 2.1 – *Principaux Champs d'un certificat X509*

On pourrait souligner certains points spécifiques :

- Meme si l'on fait confiance à un tiers (AC) pour l'authentification, cette notion de confiance est repercutée en bout de chaîne où il existe forcément une entité qui s'auto certifie.Dans notre cas, il s'agit de l'Autorité de Certification.
- Tout ce mécanisme d'authentification par certificat évite d'envoyer des clés privées par des canaux non sécurisés. On pourrait utiliser des systèmes symétriques mais le problème reste le même : la transmission des clés.
- N'importe quel utilisateur peut vérifier l'authentificité d'un certificat avec la clé publique de l'AC.
- L'ensemble des clés publiques est signé par l'AC. Une interception et modification des informations sur le certificats sera détéctée car la signature ne sera alors plus

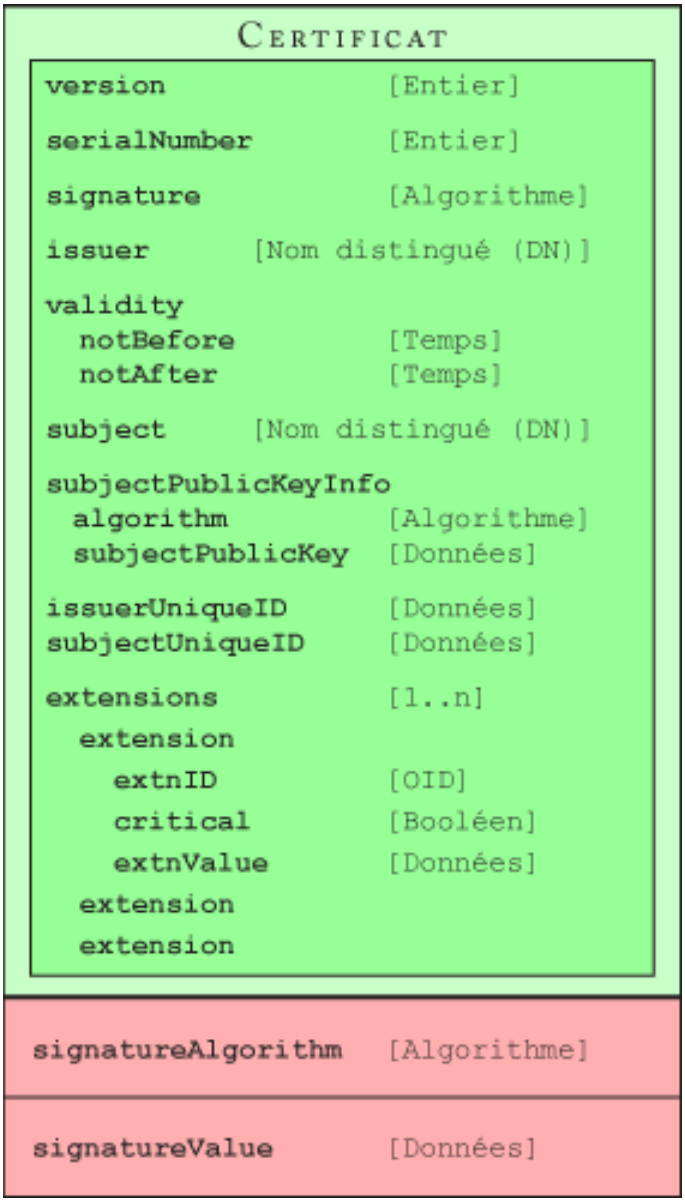

FIG. 2.1 – *Certificat X509*

valide. La signature étant fabriquée avec la clé privée de l'AC, elle n'est pas modifiable non plus. Autrement dit, aucune autre partie que l'AC ne peut modifier les données du certificat sans que cela ne soit détecté. Cela permet de réduire considérablement l'impact des attaques de type **M**an **I**n **T**he **M**iddle.

On peut voir un exemple de certificat X509 à la figure 2.2 et 2.3.

FIG. 2.2 – *Exemple de certificat X509, Interface WEB du serveur de News, Université Bordeaux1*

On dispose dorénavant d'un moyen d'authentification sûr permettant à deux personnes de communiquer sans que celles-ci ne soient inquiétées de l'usurpation de leur identité. Nous verrons un exemple concret de l'utilisation de ces certificats lorsque nous mettrons en place une authentification basée sur Kerberos. Néanmoins, obtenir un certificat X509 coûte cher, tout le monde ne peut pas en avoir un d'autant plus que les AC ne prennent en charge qu'un seul nom pour le détenteur de la clé et une seule signature numérique pour attester de la validité de la clé.

On peut alors se tourner vers le cryptosystème PGP.

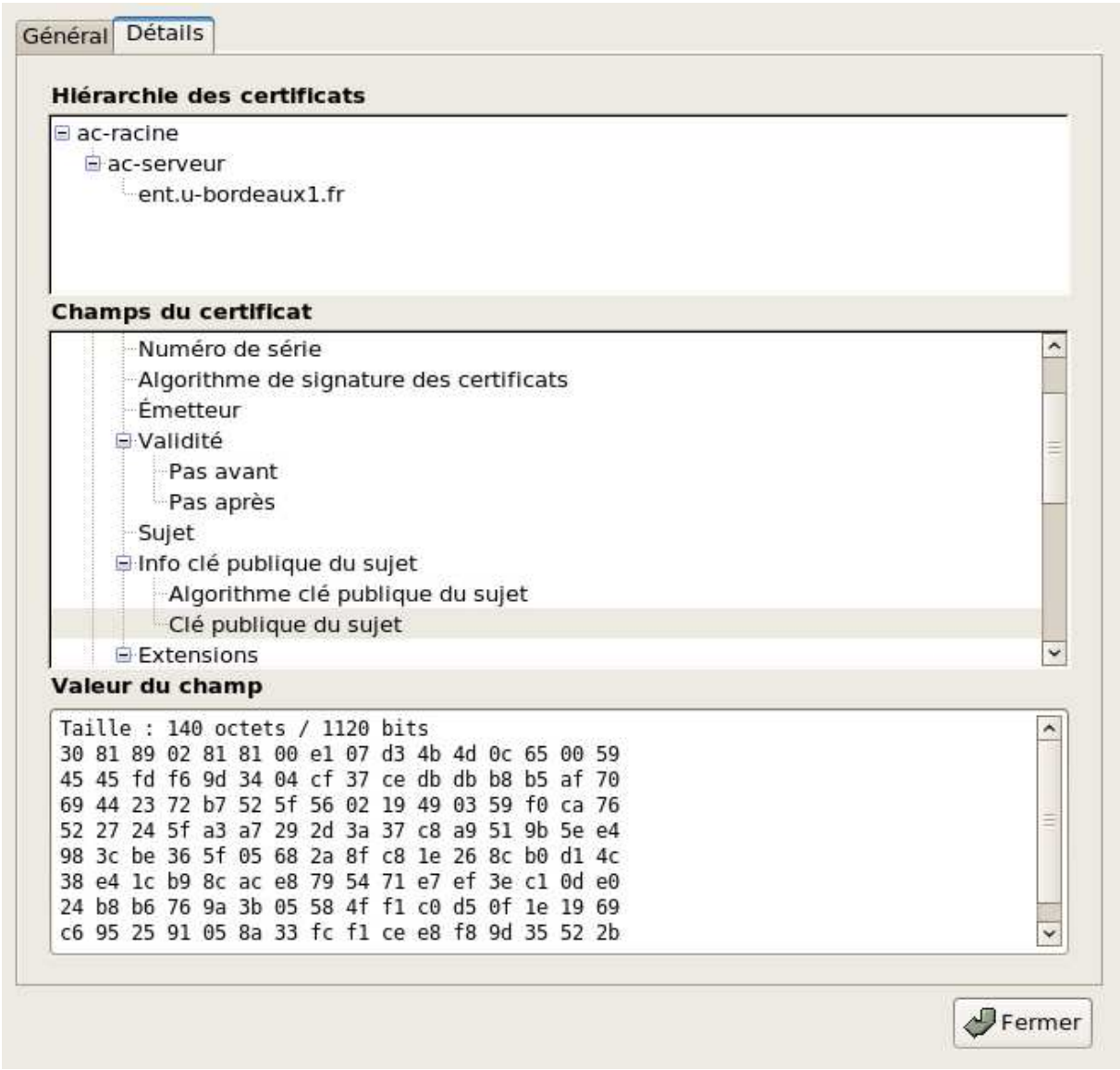

FIG. 2.3 – *Exemple de certificat X509, Interface WEB du serveur de News, Université Bordeaux1*

2.3 Authentification par PGP

L'application web la plus utilisée est, de loin, l'échange d'e-mails. Face aux besoins d'authentification et de confidentialité dans la correspondance éléctronique, est né PGP (« *Pretty Good Privacy* »).

PGP est un cryptosystème hybride utilisant la cryptographie à clé publique et à clé secrète. Il est très réputé dans l'échange de correspondance éléctronique de part sa rapidité d'éxécution. Le point fort de PGP est de pouvoir authentifier de manière sûre l'émetteur d'un message, sans que celui ci n'ait été altéré au cours de la communication.Comme pour la norme X509, PGP n'est lié à aucun protocole cryptographique en particulier. Il est gratuit, et permet d'avoir un large éventail d'applications possibles (sous les distributions Debian/Ubuntu, on dispose, par exemple, d'un mécanisme de recherche de paquets et de mise à jour du système basé sur *aptitude* (Synaptics pour l'interface graphique), les paquets disponibles sont signés en PGP).

L'ensemble des services fournis va de l'authentification d'utilisateur, à la compression des messages en passant par leur confidentialité. Il a été pensé pour être compatible et interopérable (beaucoup d'OS, de systèmes de mail, etc).

Le fonctionnement de (Fig 2.4) PGP diffère de celui des Certificats X509, en particulier lorsqu'il s'agit d'apposer une signature à un message : PGP utilise, en effet, des clés de sessions à usage unique. Une fois le message authentifié, la clé de session n'est plus utilisable.

Afin de garantir l'intégrité et l'authenticité d'un message, on procède de la manière suivante :

1. On écrit le message.
2. On calcule un hachage du message par une fonction de hachage, les plus utilisées étant la famille de fonction SHA, en particulier SHA-1.
3. On chiffre ensuite ce code de hachage (avec RSA par exemple mais ce n'est pas une obligation) en utilisant notre clé privée.
4. On apose le résultat à coté du message.

Le message est alors, d'une part, authentifiable grâce à notre clé publique, et d'autre part intègre car s'il est modifié la signature ne sera plus valide, on saura que le message

a été corrompu.
On peut voir la structure des paquets PGP sur le tableau 2.2 :

Clé publique du destinataire : Il s'agit de la clé publique du destinataire, servant notamment pour le chiffrement des messages.

Clé de session : Cette clé est chiffrée avec la clé publique du destinataire. Un intrus ne pourra donc pas y avoir accès sans posséder sa clé privée.

Horodatage : Date et heure à laquelle la signature a été produite. Ce champ permet d'éviter le rejeu des paquets. Si la clé de session a expiré et qu'on reçoit un paquet signé avec cette clé, il n'est plus valide.

Clé Publique Du Destinataire
Clé de session chiffrée
Horodotage
Clé Publique De L'Emetteur
Tête du résumé (2octets)
Hachage du message
Nom Fichier
Données

} Chiffré avec la clé de session (**non disponible en clair !**)

TAB. 2.2 – *Structure d'un paquet PGP*

On chiffre alors l'ensemble des champs (montré sur le tableau 2.2) avec la clé de session . On assure ainsi la confidentialité et l'authenticité de ce qui transite. En effet, la clé de session a, elle même, été chiffrée avec la clé publique du destinataire – seul lui peut la déchiffrer – ; ainsi une modification du message sera de suite détéctée, on saura qu'il a été corrompu. A noter que PGP permet également de chiffrer ces messages avant de les envoyer.

Contrairement à X509, PGP permet une autre notion de la confiance. En effet, PGP est basé sur un réseau de confiance en matière d'authentification de clés. Il met en oeuvre un mécanisme basé sur une chaîne de signature : on veut utiliser la clé de Monsieur X. On ne le connait pas, mais on connait Monsieur Y en qui on a confiance. Il s'avère que Monsieur Y a, quant à lui, signé la clé de Monsieur X. On peut alors utiliser la clé de Monsieur Y. Il s'agit d'un *reseau de confiance*, on affecte une fiabilité complète à la clé de Monsieur Y, qui lui a fait de même pour celle de Monsieur X. Cela permet, en particulier, de ne pas reporter notre confiance sur une autorité « faisant foi »en la matière mais dont on ne connait pas grand chose si ce n'est qu'il faut lui faire

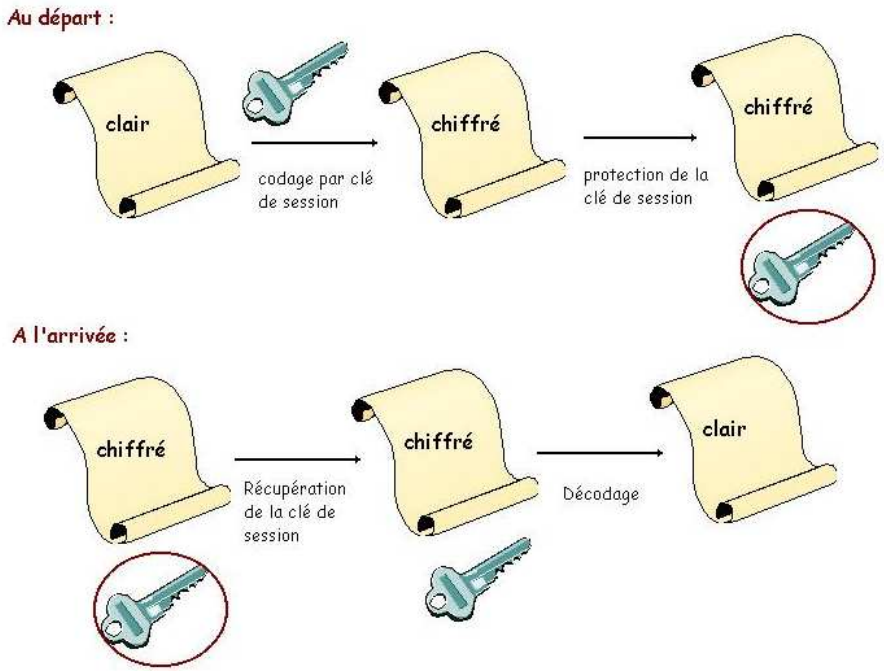

FIG. 2.4 – *Principe de l'authentification par PGP*

confiance ; et de batir notre propre réseau.

De nos jours, la généralisation des Infrastructures de Gestion de Clé est devenue « à la mode ». Il faut se méfier lorsque l'on met en place une telle infrastructure. En effet, il s'agit d'analyser l'ensemble des besoins en terme d'authentification et d'intégrité des personnes.

D'une manière générale, lorsque l'on utilise de multiples PKI, il devient vite impossible de gérer l'ensemble des flux des utilisateurs d'une manière centralisée du fait de l'hétérogénéité des solutions disponibles. On peut alors se tourner vers des protocoles dans l'authentification, robustes et stables. Leur rôle est uniquement d'authentifier un utilisateur et de passer le relai à l'application demandée (si celle ci nécessite un mot de passe (comme *ssh*) celui ci ne sera plus demandé).

Chapitre 3

Protocoles d'authentification spécialisés

Lorsque l'on se connecte sur un réseau, on rentre notre login et mot de passe. Une fois authentifié, on consulte nos emails, on s'authentifie une fois de plus. On se connecte alors sur une machine distante (un site de l'entreprise situé dans une autre ville par exemple), on s'authentifie de nouveau !

Certaines faiblesses sous-jacentes aux systèmes d'authentification traditionnels apparaissent alors. Dans un contexte où les attaquants peuvent monitorer le trafic réseau, intercepter des mots de passe, voler des sessions (usurpation de session TCP/IP basée sur les numéros de séquence par exemple), l'envoi d'informations d'authentification critiques doit être de plus en plus prohibé. Il est plus judicieux de mettre en place des méthodes d'authentification forte qui non seulement protègent les mots de passe, mais évitent également leur diffusion, même cryptée.

Comment faire alors pour ne plus avoir à s'authentifier plusieurs fois et donc ne plus avoir à envoyer autant de fois les mots de passe ?

D'autre part, plaçons-nous du coté des applications qui nous authentifient : les informations relatives à l'authentification des utilisateurs sont stockées sur le serveur d'authentification principal (un serveur NiS par exemple), sur le serveur POP/IMAP également, sur le serveur Apache, et sur une base de toute autre application ayant besoin d'authentifier les utilisateurs y accédant. Cela représente autant de duplications de

ces informations critiques en autant d'endroits différents.

Comment faire alors pour centraliser les informations d'authentification et assurer la liaison entre les applications qui en ont besoin d'une part et la base d'utilisateurs d'autre part ?

3.1 Service d'authentification Kerberos

Lorsque l'on désire avoir de l'authentification robuste, Kerberos se démarque par son fonctionnement atypique.

3.1.1 Présentation

Kerberos a été développé au MIT (Massachusetts) décrit dans la RFC 4120 depuis 2005. C'est un service destiné à faire uniquement de l'authentification d'utilisateur : il répond « oui »ou « non »à une demande d'authentification. L'éthymologie du nom est d'ailleurs évocatrice : Kerberos provient de « Cerbère », un chien de garde à trois têtes qui surveillait l'entrée Des Enfers dans la mythologie grecque.

Kerberos (Fig. 3.1) est apparu dans un contexte où beaucoup de services et les différents accès aux ressources sont de plus en plus décentralisés. Comment, en effet, assurer l'authentification d'un utilisateur lorsque l'on se trouve dans un réseau pas forcément sécurisé et que l'on veuille accéder à une ressource ? Comment s'assurer que le serveur qui répond est bien le bon ? Comment, par exemple, éviter le vol de session durant une écoute pendant une session d'authentification ?

Kerberos est donc pensé pour

- Authentifier les utilisateurs par rapport aux services.
- Authentifier les serveurs par rapport aux utilisateurs.

En ce sens, Kerberos doit répondre à un certain nombre de choses :

Sécurité : Une personne malveillante ne doit pas pouvoir se faire passer pour un utilisateur légitime juste en écoutant et monitorant le réseau.

FIG. 3.1 – *Cerberus, le Gardien Des Enfers*

Fiabilité : Même si l'authentification en elle-même est centralisée, les serveurs Kerberos doivent pouvoir travailler en coopération et se remplacer les uns les autres en cas de défaillance ou de panne.

Transparence : Un utilisateur ne doit pas se rendre compte que l'authentification est basée sur Kerberos : il continue à se logger de la même manière. Un service ne doit pas être réimplémenté ou réinstallé pour être compatible avec Kerberos.

Evolutivité : Il ne doit pas être rigide et figé, mais s'adapter aux évolutions et au développement des communications numériques.

Kerberos est disponible en open source sur le site du MIT, et certains éditeurs proposent des versions commerciales : Kerberos[1] est, par exemple, très utilisé dans l'authentification par Active Directory, l'annuaire des serveurs Windows de Microsoft. Il fournit un composant essentiel dans la sécurité globale des systèmes d'information.

3.1.2 Fonctionnement

Le mécanisme est basé sur le protocole Needham Schröder.

Source Wikipédia :

> « *Needham-Schroeder est un protocole d'authentification dans les réseaux informatiques, conçu pour être utilisé dans des réseaux*

[1] PKINIT est une extension de Kerberos utilisant les certificats numériques pour la phase d'authentification, pour les systèmes Microsoft.

> *non sûrs (Internet par exemple), et inventé par Roger Needham et Michael Schroeder en 1978. Le protocole permet à des tiers communiquant à travers un réseau de se prouver leur identité respective. »*

C'est donc un protocole d'authentification par un tiers : il s'agit de faire en sorte que deux systèmes communiquants (le client et la ressource) determinent une clé avec une troisième entité (authentification Kerberos) qui veille à ce que les choses se passent bien.

Protocole Basique :

Soit $\mathcal{K}$ une clé symétrique standard. On suppose que $\mathcal{A}$ veut communiquer avec $\mathcal{B}$ à travers un réseau pas forcément sûr. Le but est d'obtenir une clé de session. Soit $\mathcal{S}$ un serveur de confiance pour $\mathcal{A}$ et $\mathcal{B}$.
Deux clés $\mathcal{K}_{\mathcal{AS}}$ et $\mathcal{K}_{\mathcal{BS}}$ sont respectivement partagées par $\mathcal{AS}$, et $\mathcal{BS}$. Deux nombres $\mathcal{N}_{\mathcal{A}}$ et $\mathcal{N}_{\mathcal{B}}$ tirés aléatoirement pour les besoins de l'authentification.

Le processus d'authentification est alors le suivant :

1. $\mathcal{A}$ envoie à $\mathcal{S}$: Les informations relatives à $\mathcal{A}$ (login, mot de passe, IP, etc), à $\mathcal{B}$ (Service demandé, IP de $\mathcal{B}$, etc) et $\mathcal{N}_{\mathcal{A}}$.
2. $\mathcal{S}$ répond et envoie à $\mathcal{A}$ le bloc suivant : $\left\{\mathcal{N}_{\mathcal{A}}, \mathcal{K}_{\mathcal{AB}}, \mathcal{B}, \left\{\mathcal{K}_{\mathcal{AB}}, \mathcal{A}\right\}_{\mathcal{K}_{\mathcal{SB}}}\right\}_{\mathcal{K}_{\mathcal{AS}}}$. Ici, les indices $\mathcal{K}_{\mathcal{SB}}$ et $\mathcal{K}_{\mathcal{AS}}$ correspondent à la signature utilisant ces clés et partagées uniquement par les entités concernées. $\mathcal{K}_{\mathcal{AB}}$ correspond à une clé que Kerberos determine et envoie sur le réseau en la signant.
3. $\mathcal{A}$ initie alors la communication avec $\mathcal{B}$: $\mathcal{A}$ envoie à $\mathcal{B}$ le bloc $\left\{\mathcal{K}_{\mathcal{AB}}, \mathcal{A}\right\}_{\mathcal{K}_{\mathcal{SB}}}$. À noter que $\mathcal{A}$ ne peut pas modifier ce bloc car il a été signé par $\mathcal{K}_{\mathcal{SB}}$, clé partagée uniquement par $\mathcal{B}$ et $\mathcal{S}$.
4. $\mathcal{B}$ vérifie la signature, signe $\mathcal{N}_{\mathcal{B}}$ avec la clé $\mathcal{K}_{\mathcal{AB}}$ et envoie $\left\{\mathcal{N}_{\mathcal{B}}\right\}_{\mathcal{K}_{\mathcal{AB}}}$ à $\mathcal{A}$.
5. $\mathcal{A}$ vérifie la signature, calcule $\mathcal{N}_{\mathcal{B}} - 1$, le signe et envoie $\left\{\mathcal{N}_{\mathcal{B}} - 1\right\}_{\mathcal{K}_{\mathcal{AB}}}$ à $\mathcal{B}$. La session est alors prête.

Ce fonctionnement de Kerberos pose néanmoins un problème. En effet, il n'y a là aucune notion de durée de validité ou d'horodotage. Le bloc $\left\{\mathcal{K}_{\mathcal{AB}}, \mathcal{A}\right\}_{\mathcal{K}_{\mathcal{SB}}}$ peut être intercepté, gardé puis rejoué par quelqu'un. Tout le système risque alors de s'écrouler.

Attribution de Tickets :

Kerberos intègre alors la notion de validité par l'attribution de Tickets. Pour cela, il existe un **S**erveur d'**A**uthentification (SA) où sont stockés les codes de hachage de tous les mots de passe des utilisateurs enregistrés. Une clé secrète est alors partagée entre le SA et chacun des serveurs de service.

1. Le module d'authentification du client envoie alors au SA : L'identité de l'utilisateur (UserID), le mot de passe et les informations relatives au serveur demandé (ServerID).
2. Le SA authentifie l'utilisateur, génère le Ticket T = $\left\{$ ServerID, UserID, Addresse Réseau du Client $\right\}$, le chiffre avec la clé secrète partagée et l'envoie au client. Le client déchiffre le Ticket et peut accéder au service.

L'adresse réseau du client fait partie du Ticket pour éviter le rejeu des packets. En effet, il n'y a plus d'authentification par mots de passe entre le client et le serveur, le Ticket fait foi.

A remarquer que ce protocole n'est toujours pas sécurisé : Il faut, en effet, s'authentifier pour chaque service mais surtout à l'étape 1-, le mot de passe est envoyé en clair sur le réseau ! ! !

Protocole Kerberos Actuel :

Kerberos permet la saisie unique du mot de passe pour l'ensemble des services, l'authentification est faite une bonne fois pour toute pour tous les services, le Ticket est délivré et est valide pour toute sa durée. La philosophie de Kerberos est, en outre, d'éviter d'envoyer le mot de passe à travers le réseau.

On utilise alors un serveur d'octroi de Ticket (Fig. 3.2) : **T**icket **G**ranting **S**ervice (TGS).
On remarquera dans le protocole que le mot de passe ne transite pas sur le réseau. L'authentification est faite comme suit (On distingue le Client (Machine) de l'Utilisateur) :

1. Le Client demande au SA un ticket d'octroi de Ticket.

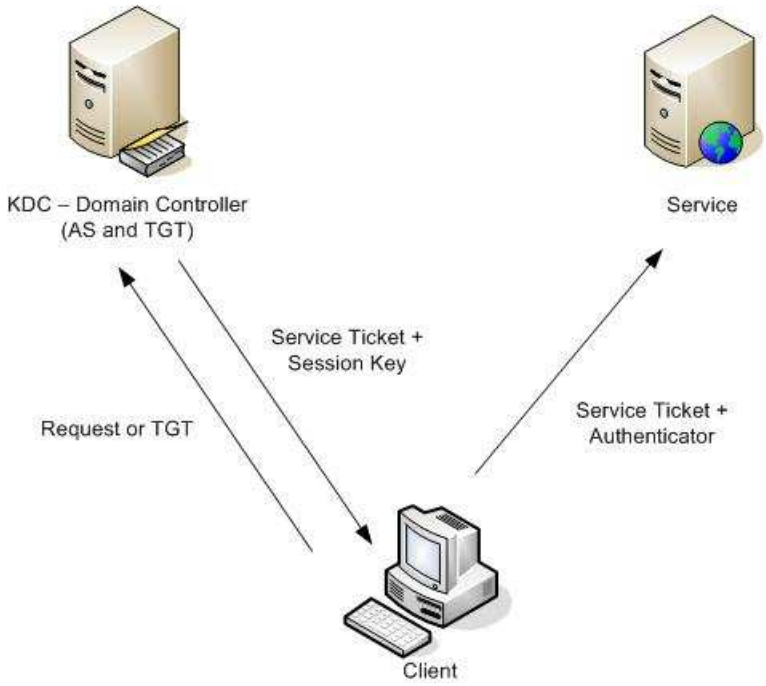

FIG. 3.2 – *Principe De Kerberos*

2. Le SA (qui possède toujours l'ensemble des codes de hachage des utilisateurs) forge un ticket chiffré avec le code de hachage de l'utilisateur et l'envoie au Client.
3. Le Client demande à l'Utilisateur de rentrer son mot de passe. Si celui ci est correct, le Client peut déchiffrer le ticket en calculant le code de hachage (le mot de passe n'est pas envoyé sur le réseau).
4. En cas de succès, le Client demande le ticket d'octroi de Service au TGS. La demande est T = $\left\{$ UserID, Addresse Réseau du Client, ticket d'octroi de Ticket $\right\}$. Le TGS déchiffre T et envoie le Ticket définitif au Client qui l'utilise pour authentifier l'utilisateur auprès du Service. Ce ticket est valable pour tous les services dont l'authentification se fait par Kerberos : il n'est plus nécessaire d'en demander un pour chaque service différent.

L'avantage énorme qu'offre Kerberos est le fait de pouvoir prêter ce ticket pour une durée déterminée et sans révéler son mot de passe : il est possible de prêter sa session sans avoir à donner son mot de passe.
Ces tickets sont datés pour éviter le rejeu (Kerberos impose que l'heure de l'ensemble des machines aît un décalage minimum) et chiffrés avec une clé inconnue du Client.

D'autre part, il faut également assurer l'authentification des Services par rapport

aux Utilisateurs. Les serveurs Kerberos sont, en outre, la plupart du temps couplés avec des annuaires comme LDAP ou Active Directory afin d'avoir l'ensemble des informations relatives à l'identité des utilisateurs.

Dans un contexte où le nomadisme est en plein essor, les utilisateurs sont de plus en plus susceptibles de venir connecter leur ordinateur portable aux réseaux d'entreprises. Ces connexions échappent totalement au contrôle de l'administrateur système et peut poser des soucis faute d'une méthode d'authentification adéquate permettant de gérer le flux de connexion.

3.2 Radius et Authentification WiFi

Nous avons vu le fonctionnement de l'authentification par Certificats X509. Nous allons le mettre en pratique dans Radius et présenter les principaux axes de l'authentification par serveur Radius.

3.2.1 Description

RADIUS (***R**emote **A**ccess **D**ial **I**n **U**ser **S**ervice*) fait partie de la partie des serveurs NAS (Network Authentication Service), il est de type AAA (**A**uthentication (Qui) **A**uthorization (Droits accordés) **A**ccounting (Audit)). Il a été normalisé par l'IETF[2] et est décrit dans les RFC 2865 et RFC 2866. Il a été développé par la société Livingston Enterprises et conçu pour pouvoir gérer les connexions des utilisateurs distants, notamment en matière de politique d'authentification, de droits d'accès et de traçabilité des différentes connexions (contrairement à Kerberos qui offre de l'authentification forte dans un contexte géographiquement restreint). En ce sens, ce protocole a été très prisé des Fournisseurs D'Accès Internet car il permettait de gérer finement le temps de connexion à l'époque des forfaits ADSL. Il continue toujours d'être utilisé dans beaucoup d'autre domaines.

En effet, lorsque l'on met en place un réseau WiFi dans une université par exemple, il

[2]Internet Engineering Task Force est un organisme international dont la mission est entre autres de normaliser certains protocoles.

est primordial de pouvoir contrôler ce que font les utilisateurs sur le réseau, du fait qu'ils se connectent avec leur propre machine, il est souvent difficile d'établir une politique d'authentification à la fois efficace et stricte en particulier pour détecter et empêcher la connexion de périphériques sans-fil non autorisés.

On peut alors se tourner vers une authentification par RADIUS, protocole spécialisé dans l'authentification des clients distants. On peut le mettre en place de deux manières :

- Une authentification par addresse MAC.
- Une authentification par Certificats X509.

L'authentification par RADIUS en fait un composant essentiel de toute solution de sécurité basée sur 802.1X[3]. La disponibilé du réseau sans fil dépend donc du bon fonctionnement des serveurs RADIUS : ils doivent être fiables (pas de longue maintenance, disponibilité, peu de pannes) , réactifs (faire face à un nombre de demandes qui peut être très fluctuant et impossible à prévoir) et redondants (répartition de charge automatique entre les serveurs en cas de saturation) pour garantir des performances accrues (proches de celles des réseaux filaires). Ils peuvent, en outre, consigner différents types d'événements complémentaires comme les échecs de tentatives d'authentification et/ou les informations relatives au compte d'utilisateur.

3.2.2 Principe de RADIUS

Fonctionnement Général

RADIUS fonctionne sur un principe Client/Serveur (comme 90% des protocoles réseaux) : l'utilisateur se connecte (Fig. 3.3) dans un premier temps à un client RADIUS (qui peut être un Proxy RADIUS ou le serveur lui-même). Celui-ci lui demande ses identifiants, vérifie la validité en les communiquant à une base de données ou un annuaire LDAP auquel il est relié.

RADIUS manipule un certain type de paquets :

1. Le client demande une connexion au NAS (Network Authentication Service) qui est ici un client RADIUS : paquet **Acces - Request**, il contient l'identité de

[3]802.1X est une solution de sécurisation permettant d'authentifier un utilisateur accédant à un réseau (filaire ou non) et reposant sur le protocole EAP.

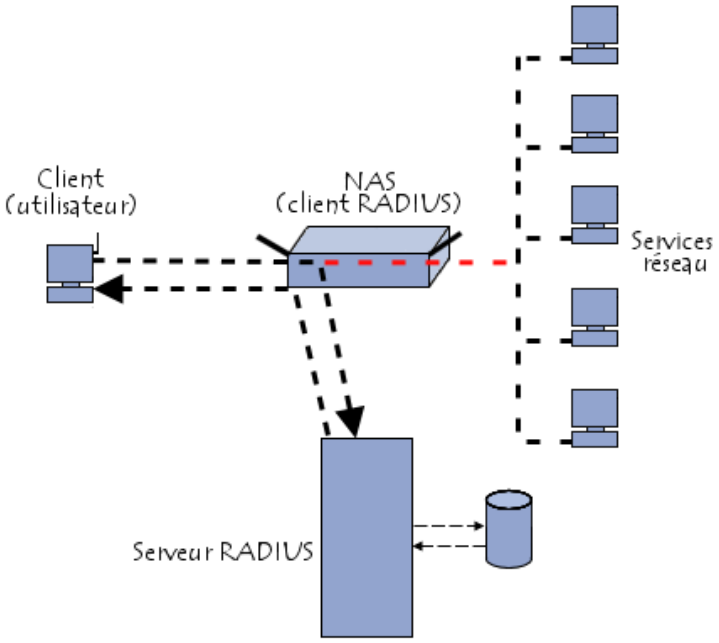

FIG. 3.3 – *Authentification par RADIUS*

l'utilisateur (username/password, certificat, addresse MAC).

2. Le NAS transmet la requête et les identifiants au serveur RADIUS. Celui-ci authentifie l'utilisateur grâce à l'annuaire (LDAP par exemple) auquel il est couplé et envoie la réponse au NAS. En cas de succès, il envoie un paquet **Access - Accept** contenant par exemple les services autorisés.
3. En cas d'échec, la connexion est refusée : le serveur envoie alors **Access Reject**. En principe, ce paquet peut être envoyé à tout moment pour couper une connexion déjà établie.
4. Si le serveur souhaite avoir des informations complémentaires, il envoie **Access - Challenge**. Ceci montre concrètement la notion de challenge expliquée plus haut. L'utilisateur prétend et le serveur demande une preuve.
5. Le NAS relaie les informations, connecte l'utilisateur et monitore l'ensemble du trafic en consignant l'ensemble des activités du client dans des logs afin de permettre un audit, une analyse du transit.

Comment ces différentes informations transitent-elles entre l'utilisateur et le client RADIUS ? Entre le Client RADIUS et le Serveur RADIUS ? Entre le Serveur et l'Annuaire ?

Authentification par addresse MAC et Certificats

RADIUS utilise nativement une implémentation des protocole PAP et CHAP : ***P**assword **A**uthentication **P**rotocol, **C**hallenge **H**anshake **A**uthentication **P**rotocol*. Il s'agit d'un mécanisme d'authentification très utilisé notamment dans les reseaux sans fils et les connexions Point-A-Point. Dans le cadre des réseaux sans fils, il existe une extension appellée EAP *(**E**xtensible **A**uthentication **P**rotocol)*. Les fabriquants de matériels sans fil intègrent ce protocole dans leur mécanisme d'authentification : WPA EAP-TLS ou WPA2 EAP-TLS[4].

EAP-TLS présente une authentification mutuelle client-serveur par certificat. L'intéret de s'authentifier par les certificats est que les NAS n'ont pas besoin de connaître tous les mécanismes, la liaison se faisant comme un tunnel vers un serveur qui, lui, implémente ces mécanismes. Le client peut également être authentifié par un username/password et le serveur par son certificat (Fig 3.4 en remplaçant les addresses MAC par des Certificats).

Une authentification par addresse MAC (Fig 3.4) permet d'éviter les connexions des postes non autorisés. L'annuaire tient à jour une base d'addresses MAC autorisées à se connecter sur le réseau et le serveur la consulte pendant la tentative de connexion tandis que les switch relayent les différentes addresses MAC.

Une authentification par certificats X509 utilise les protocoles de la famille EAP (EAP/TLS, EAP/PEAP ou EAP/TTLS) basés sur 802.1x. Une autorité de certification délivre différents certificats pour les machines utilisateurs, et ceux-ci font foi au moment de la tentative de connexion. Il est possible de coupler les deux types d'authentification (Fig. 3.5) lorsque l'on désire par exemple mettre en place RADIUS sur un réseau sans fil et filaire. Pour les réseaux filaires, il est possible d'attribuer dynamiquement un numéro de VLAN[5] à la machine cliente après son authentification : il s'agit ici de 802.1x également.

Les mécanismes d'authentification sont aussi diverses que nombreux. En effet, une clé PGP n'est pas traitée de la même manière par la machine qu'un Ticket Kerberos ou qu'une addresse MAC pour RADIUS.

[4]Microsoft Windows SP2 ne gère pas WPA2, il faut lui ajouter un correctif : WindowsXP-KB893357-v2-x86-FRA.exe à télécharger sur le site de Microsoft.

[5]Tous les switchs ne supportent pas le VLAN, seuls les switchs administrables au dessus du niveau 2.

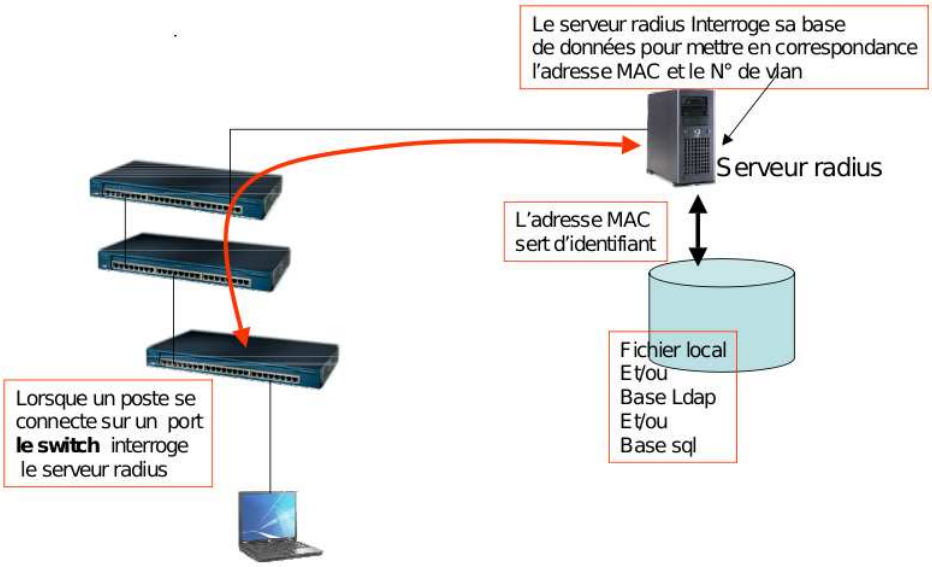

FIG. 3.4 – *Authentification par addresse MAC*

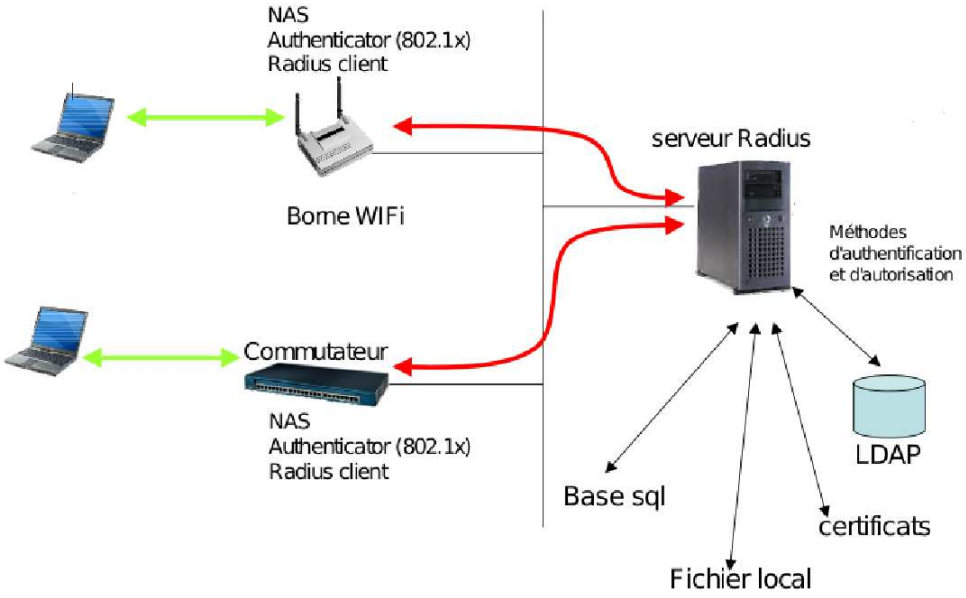

FIG. 3.5 – *Authentification Filaire et SansFil par RADIUS*

Si on se place du côté de la machine qui authentifie, il faut gérer tous ces processus en concurrence et faire en sorte que l'authentification se déroule correctement dans tous les cas. Si l'administrateur désire ajouter une nouvelle fonctionnalité nécessitant une authentification (mettre en place un tunnel VPN entre différents sites par exemple), il faut l'ajouter au noyau du système. Gérer manuellement et indépendemment ces mécanismes devient assez rapidement pénible.

Comment faire en sorte que les différents types d'authentification puissent cohabiter au sein d'un système ? Nous prendrons ici le cas de Linux : le Projet PAM, sous GNU/Linux permet de répondre à ces attentes et constitue le coeur de la gestion des multiples procédés d'authentification au niveau du noyau du système d'exploitation.

Chapitre 4

Le Projet PAM

Il y a quelques années, lorsque l'on se connectait à une machine, on rentrait le plus souvent un username et password et le système d'exploitation vérifiait si le mot de passe était le bon (sous Linux, le fichier `/etc/passwd` contenait les codes de hachages des mots de passe). De nouveaux moyens d'authentification sont alors apparus et les programmes nécessitant une authentification des utilisateurs devaient être modifiés pour la prendre en compte.
PAM fournit alors un moyen de s'abstraire du mécanisme d'authentification des applications en reléguant ceci à un niveau inférieur.

4.1 Présentation

PAM (**P**luggable **A**uthentication **M**odule) ou Modules d'Authentification Enfichables est un processus d'authentification centralisé et modulaire : les applications nécessitant une authentification d'utilisateur peuvent passer par PAM et ne sont pas obligés d'intégrer une politique d'authentification. Le PAM a été pensé en ce sens. Il utilise une architecture modulaire et permet donc à l'administrateur système d'être très flexible en matière d'authentification.

Si différents services implémentent différentes méthodes d'authentification (Kerberos, Radius, Droits Unix/Linux, IpSec, etc), PAM leur offre un support en centralisant tout le processus sous forme modulaire (Fig. 4.1) et permet d'avoir un contrôle sur l'intégrité du système quant à l'établissement d'une politique d'authentification.

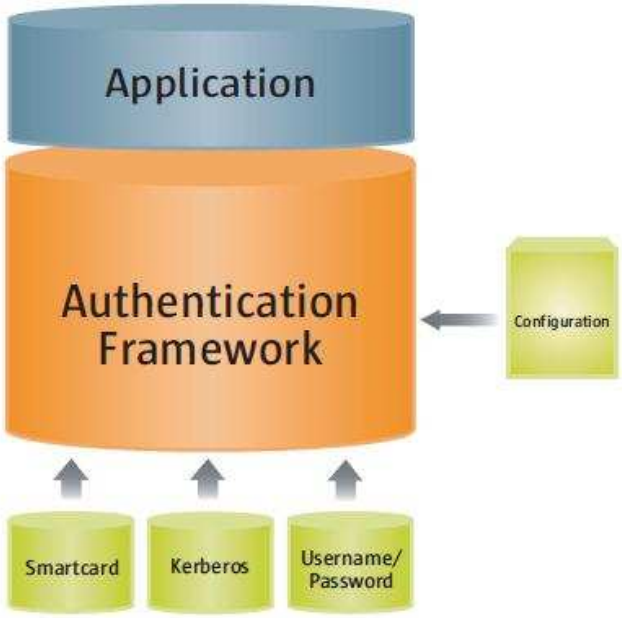

FIG. 4.1 – *Principe Basique des PAM*

On configure le framework d'authentification pour qu'il supporte l'ensemble des mécanismes d'authentification. L'application n'a plus besoin d'implémenter un mécanisme en particulier.

Les avantages de PAM sont donc nombreux : il peut être utilisé avec une multitude d'applications, il offre un degré de flexibilité élevé tant pour l'administrateur que pour le developpeur qui peuvent implémenter des applications sans avoir à définir leur propre système d'authentification.

4.2 Fichiers de configuration

Le répertoire `/etc/pam.d/` contient l'ensemble des fichiers de configuration de PAM : pour chaque application prenant en charge PAM, il existe un fichier dans ce répertoire. Auparavant, on utilisait le fichier `/etc/pam.conf`, son utilisation a finalement été abandonnée et il n'est lu que si le répertoire `/etc/pam.d/` n'existe pas (par exemple, pour IpSec :`/etc/pam.d/ipsec.conf`).

Chaque fichier de configuration est construit suivant le modèle :

```
<module interface>  <control flag>   <module name>   <module arguments>
```

Pour PAM, il existe quatres types de modules :

`auth` Assure l'authentification réélle des utilisateurs, en demandant un mot de passe, définissant un certificat ou un ticket Kerberos.

`account` Vérifie que l'accès est effectivement autorisé (expiration du compte, heure de la journée ...).

`password` Gère, définit et vérifie les différents mots de passe.

`session` Une fois l'utilisateur authentifié, ce module est permet à l'utilisateur de gérer son compte.

On définit alors un module d'authentification pour un programme en ayant besoin :

```
#%PAM-1.0
auth       required  /lib/security/pam_securetty.so
auth       required  /lib/security/pam_pwdb.so shadow nullok
auth       required  /lib/security/pam_nologin.so
account    required  /lib/security/pam_pwdb.so
password   required  /lib/security/pam_cracklib.so
password   required  /lib/security/pam_pwdb.so shadow nullok use_authtok
session    required  /lib/security/pam_pwdb.so
```

On voit que lorsqu'un utilisateur se connecte, la vérification passe à travers les trois premiers modules (Fig. 4.2), y compris si le premier échoue (pour ne pas donner à l'utilisateur des informations relatives à l'échec de sa connexion). On peut changer ce comportement en changeant `required` par `requisite`.

`required` et `requisite` sont des indicateurs de contrôles. Un indicateur de contrôle indique aux modules PAM la manière de gérer une réussite ou un échec d'authentification (les PAM renvoient toujours un échec ou une réussite), il y en a quatre :

`required` Le module doit être validé pour que l'authentification réussisse. Si elle échoue, l'utilisateur n'en est pas averti tant que les autres modules n'ont pas été vérifiés.

`requisite` Le module doit être validé pour que l'authentification réussisse, à la différence de `required`, un message est envoyé à l'utilisateur.

`sufficient` Les vérifications de modules sont ignorées en cas d'échec.

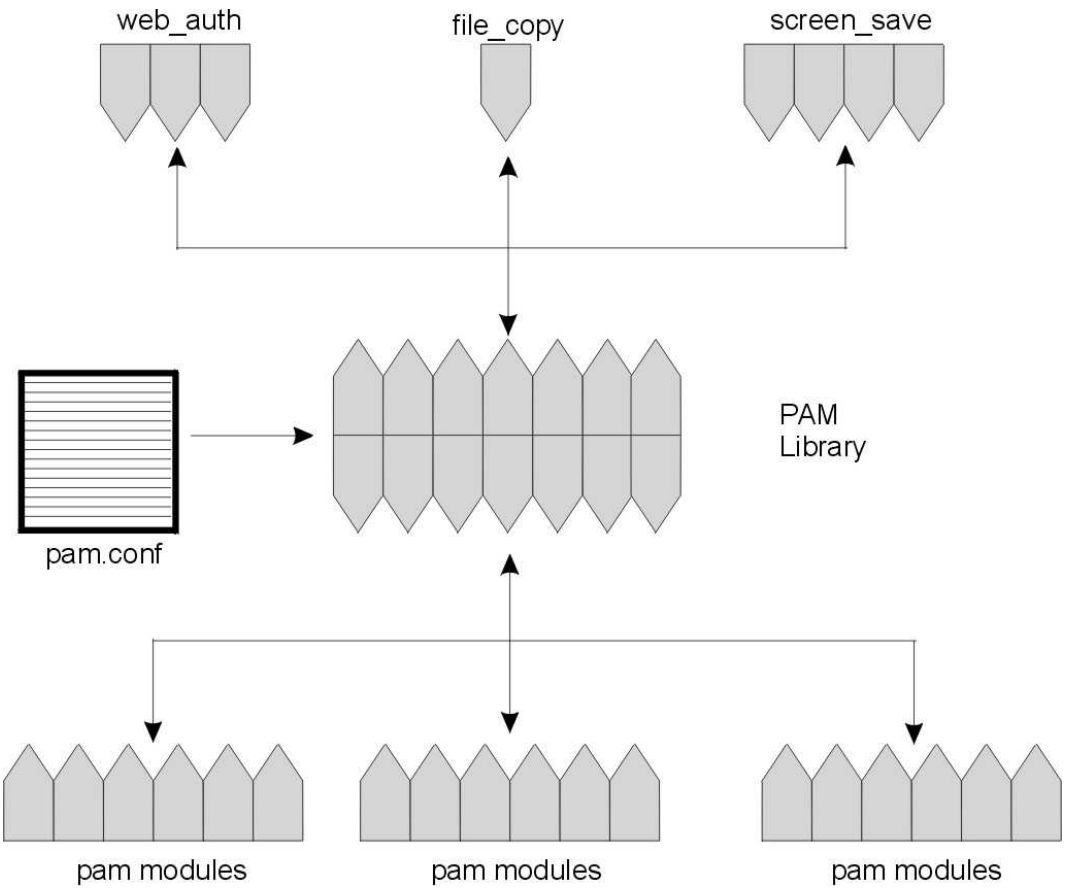

FIG. 4.2 – *Interprétation des modules PAM*

`optional` Les vérifications sont ignorées, mais devient nécessaire lorsqu'aucun autre module ne fait référence à cette interface.

Avec la multitude de solutions d'authentification disponibles, PAM représente un moyen de pouvoir faire cohabiter l'ensemble de ces procédés et d'assurer ainsi une authentification contrôlée par le système d'exploitation et non pas l'application qui l'implémente.

Conclusion

Les mécanismes d'authentification disponibles de nos jours répondent avant tout à des besoins des utilisateurs, des administrateurs, des entreptrises ou encore des universités pour garantir une traçabilité des différents événements, des accès aux données et aux ressources. En somme, il s'agit à tout moment de savoir qui fait quoi et où.

Les procédés vont du simple Certificats attestant de l'identité d'une personne à l'authentification par Kerberos qui propose une authentification robuste. Pour gérer cette diversité de solutions, les applications auraient dû les implémenter intrinsèquement si le projet PAM n'avait vu le jour : il permet, en effet, de jongler entre les différents moyens d'authentification, sous forme modulaire. Cela facilite la gestion des différents utilisateurs et permet d'avoir des applications détachées du processus d'authentification.

D'autres solutions sont évidemment disponibles et n'ont pas été détaillées ici (comme le VPN, IpSec, authentification par LDAP, etc) , le but n'étant pas de rédiger un manuel techniques, mais de montrer l'éventail que l'on peut avoir en matière d'authentification suivant les besoins et les attentes des différents organismes qui les mettent en place.

Dans un contexte où de plus en plus de services sont décentralisés, ou le nomadisme apparaît comme étant incontournable et ou l'hétérogéneité du matériel prédomine (lire les mails sur des téléphones mobiles, se connecter en GPRS, payer avec son téléphone, jouer en LAN sur une PSP, etc), l'authentification d'utilisateurs est un domaine très actif et n'est pas prêt de s'estomper.

Bibliographie

[1] **Authentification sous Linux** - Linux Spécial Sécurité : Securité Système, Données, PareFeu, Chiffrement, Authentification. Frank HUET. *Editions ENI.*, 2005.

[2] **Authentification Forte et Faible** - Sécurité des Systèmes d'informations, Ludovic Mé, Yves Deswarte. *Editions Lavoisier*. 2006.

[3] **Authentification Radius**, Authentification Réseau avec Radius - 802.1x, EAP, Radius. ; Serge Bordères., *Editions EYROLLES.*, 2000.

[4] **RFID :Polémique autour des implants**, *Sources La Presse Affaire, Montréal*, Michel Rousseau, disponible à : `http://www.filrfid.org/article-2629991.html`

[5] **Loi Sur La Confiance Dans L'Economie Numerique**, *Legifrance, Le service Public de l'accès au droit*, Décembre2007, disponible à : `http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=ECOX0200175L`

[6] **Les Protections Anti Copies**, Christophe RAFINA, *SupInfo Paris, Protection des Cds*, disponible à : `www.supinfo-projects.com/fr/2004/protection_cd/2/`

[7] **Kerberos, Certificats X509, PGP**, Adbou GUERMOUCHE, Sécurité des Réseaux, *Université Bordeaux 1*, disponible à : `http://dept-info.labri.fr/~guermouc/SR/`

[8] **Protocole d'authentification Réseaux, sans fil et filaire**, *Protocoles-auth-1page.pdf, S.Borderes*, disponible à : `http://raisin.u-bordeaux.fr/IMG/pdf/`

[9] **Authentification par Certificat X509**, Patrick CHAMBET, *Publications CHAMBET*, disponible à : `www.chambet.com`

[10] **X.509**, *Source Wikipédia*, disponible à : `http://fr.wikipedia.org/wiki/`

[11] **PGP Corporation Products**, disponible à :`http://www.pgp.com`

[12] **How PGP Works**, *Sylvain Lorain* disponible à : `http://www.pgpi.org/doc/pgpintro/`

[13] **Linux PAM Page**, Linux On-Line Documentation, disponible à : `http://www.kernel.org/pub/linux/libs/pam/`

[14] **Pluggable Authentication Modules**, Dag-Erling Smorgrav, *Networks Associate Technology, Inc* disponible à :`http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/`

[15] **Linux Kernel Programming**, Software Security Course 2007, *Emmanuel Fleury, Université Bordeaux 1*, disponible à : `http://www.labri.fr/perso/fleury/courses/SS07/`

[16] **Solaris PAM Documentation**, Slide Presentation, *Operating Systems And Platforms* disponible à : `http://www.sun.com/software/solaris/pam/`

[17] **PAM Pluggable Authentication Modules**, User Authentication HOWTO, disponible à : `http://www.linux.com/base/ldp/howto/`

[18] **Extensible Authentication Protocol**, *source Wikipépdia*, disponible à : `http://fr.wikipedia.org/wiki/`

Index

S

T