REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE SAAD DAHLAB DE BLIDA 1
FACULTE DES SCIENCES

PROJET DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME MASTER

SPECIALITE :

SECURITE DES SYSTEMES D'INFORMATION

Mémoire réalisé par :

El-Robrini Redouane

Belkorane Hedjala Mohamed

THEME :

Mise en place d'un proxy SSH pour la centralisation et la surveillance des accès à privilèges.

Organisme d'accueil : Decima Technologies

Promotrice : Boustia Narhimene

Encadreur : Belmahdi Emir Fares

Les membres du jury :

fr : Gheribi

EX : Djedddar

Année Universitaire : 2017-2018

Remerciement

On remercie d'abord ALLAH le tout puissant qui nous a guidé et qui nous a donné la force et la volonté de réaliser ce travail.

Nos pensées vont vers nos parents, nos sœurs,
Nos frères, nos amis.

C'est grâce à leur soutien que nous avons pu réaliser ce travail.
Ils savent déjà combien on leur doit.

Comme on remercie notre promotrice madame Narhimene Boustia de nous avoir pris en charge et pour son aide.

On remercie aussi notre encadreur monsieur Belmahdi Emir Fares pour les efforts qu'il a fait afin de nous bien encadrer.

Nos remerciements les plus sincères à toutes les personnes qui auront contribué de près ou de loin à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Enfin, nous tenons aussi à remercier les jurés pour avoir accepté d'examiner et de juger notre travail.

Merci à tous et à toutes.

Résumé

Résumé

Petit à petit les entreprises ainsi que les administrations sont entrain de convertir leurs anciens systèmes d'information vers les nouveaux systèmes d'information en utilisant l'outil informatique. Avec l'apparition de l'internet et l'évolution des technologies, ces entreprises sont devenues dépendante de l'internet afin de fournir leurs services ou finir leurs tâches pour cela, les employés de ces entreprises auront besoin d'un accès à internet pour faire leurs travails. Notre travail se résume à la réalisation d'un proxy qu'on va positionner entre les employés et le réseau externe pour établir des droits d'accès à ces employés pour qu'ils ne puissent accéder qu'aux sites dont ils auront besoin. Dans ce contexte, on va rajouter deux tunnels SSH le premier sera entre le client et le proxy et le deuxième sera entre le proxy et l'internet pour sécuriser la liaison, et on va terminer notre travail par la réalisation d'une interface web au niveau du proxy que seulement l'administrateur de réseau a le droit d'y accéder afin d'éditer les droits d'accès des employés et superviser le trafic (les requêtes) des employés vers l'internet.

Mots clefs: system d'information, internet, proxy, tunnel SSH, interface web.

Abstract

Step by step companies and also administrations are converting their old information systems into modern information systems that are based on the use of the informatics tool, and with the appearance of the internet and the evolution of things those companies have become dependent on the internet so they can finish their tasks or deliver their services, which means that employees of those companies would need access to the internet in order to do their jobs. Our project would be to implement a proxy between the employees and the internet so that we can establish some access rights for the employees so they can only access the sites they need to, then we will add two SSH tunnels the first one would be between the employees and the proxy and the second one would be between the proxy and the internet in order to secure the link, then we will finish our work by implementing a web interface located in the proxy witch only the network administrator have the right to access in order to edit the access rights for the employees and supervise the traffic (queries) of the employees into the internet.

Keywords: information system, internet, proxy, SSH tunnel, web interface.

Sommaire

Liste des Figures

Liste des tableaux

INTRODUCTION GENERALE

L'évolution des entreprises est liée directement à l'évolution du domaine d'informatique vu que pas mal d'entreprises ont besoin d'accéder à des services ou récolter des informations qui se trouvent sur le Word Wide Web. Les employées de l'entreprise vont avoir un accès direct à des serveurs qui se trouvent soit sur internet soit sur des réseaux locales pour faire leurs travails ce qui met le système d'information de l'entreprise dans une certaine zone de risque. les entreprises donc ont vu la nécessité de centraliser les accès de leurs employés vers ces serveurs pour pouvoir les surveiller ainsi qu'à établir des autorisations ou ce qu'on appelle des accès privilèges, et cela pour surveiller les activités des employés au sein de l'entreprise.

Pour remédier à ce problème les chercheurs proposent de changer les system informatiques des entreprises afin d'établir un contrôleur d'accès qui vérifient si les employés ont le droit d'accéder à un certain serveur et concernant la notion de la centralisation ils ont proposé de changer la topologie de l'entreprise afin de laisser un seul chemin vers les serveurs.

Pour cela, on propose d'établir le contrôleur d'accès entre l'entreprise (l'employé) et le serveur (l'internet) à travers un proxy.
De cette façon on aura des accès des employés qui sont privilège d'un employé à un autre et ils vont être centraliser au niveau de proxy.

Ce projet consiste à établir un contrôleur des droits d'accès au niveau du proxy entre l'ensemble des employés d'une entreprise et l'ensemble des serveurs dont ils auront besoin d'accéder afin de faire leurs travails. Une interface web est rajoutér pour permettre à l'administrateur de gérer les droits d'accès.
Afin de sécuriser la liaison le protocole SSH est utilisé.
Cette solution ne fait pas l'unanimité entre les entreprises, vu qu'elle refusent que des modifications soient portées à leur système informatiqe.

Ce document est organisé comme suit :

- **Chapitre I :** est un état de l'art des différents concepts nécessaire à la réalisation de ce projet.
- **Chapitre II :** Conception de la solution
- **Chapitre III :** Réalisation de la solution

Nous terminerons ce document avec une conclusion générale et des perspectives

Chapitre I : Etat de l'art

1_Introduction

Avec la grande évolution du domaine d'informatique la plupart des entreprises sont obligées maintenant de sécuriser leur système, de protéger leurs informations, de créer leurs systèmes centralisés, de gérer les accès des employées, et rendre les accès plus sécurisé. Pour rendre leur système plus sécurisé et fiable la plupart des solutions choisissent la centralisation des accès des employées à travers l'utilisation d'un proxy et pour que ces accès soient plus sécurisés les solutions utilise le protocole SSH en implémentant le proxy pour avoir un proxy-SSH, et rajoute la cryptographie pour crypter les informations.

Ce chapitre sera structuré de la manière suivante : Nous présentons tout d'abord tout ce qui concerne le Proxy, nous allons décrire par la suite le protocole SSH et expliquer son fonctionnement, puis nous terminons avec un aperçu sur la cryptographie que nous avons utilisé au sein de la réalisation de notre projet.

2_Proxy

2.1_Définition du proxy :

Un proxy est un composant logiciel informatique qui joue le rôle d'intermédiaire en se plaçant entre deux hôtes pour faciliter ou surveiller leurs échanges.

Dans le cadre plus précis des réseaux informatiques, un proxy est alors un programme servant d'intermédiaire pour accéder à un autre réseau, généralement internet. Par extension, on appelle aussi « proxy » un matériel comme un serveur mis en place pour assurer le fonctionnement de tels services.[1]

2.2_Proxy Réseau :

Dans l'environnement plus particulier des réseaux, un serveur proxy ou « serveur mandataire » est une fonction informatique client-serveur qui a pour fonction de relayer des requêtes entre une fonction cliente et une fonction serveur (voir *Fig. I.1*). .

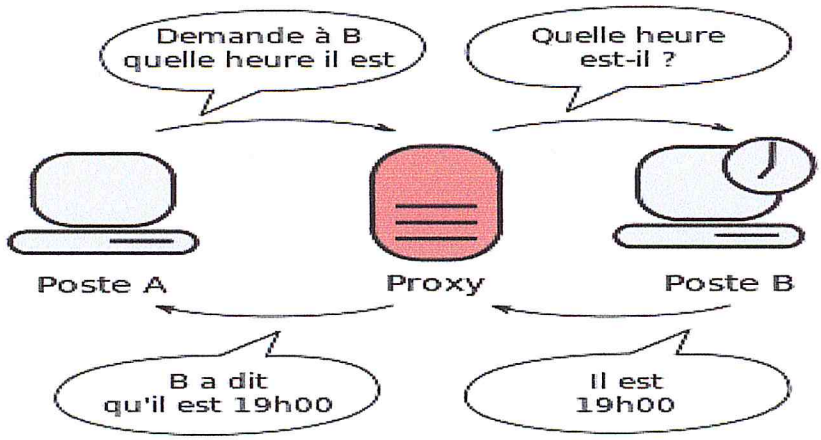

Figure 1: Principe du proxy réseau. [2]

Un proxy peut être utilisé pour plusieurs usages :

- Le premier usage du serveur proxy est d'abord d'assurer le filtrage de l'accès internet dans une entreprise. Cela permet d'empêcher l'accès à des sites potentiellement dangereux pour la sécurité du SI ou alors non conforme à la politique de l'entreprise (par exemple facebook)
- Le second usage du proxy est de permettre aussi paradoxalement de contourner les filtrages. Prenons l'exemple de la Chine qui bloque l'accès à facebook via pare-feu. Il est possible de contourner ce filtrage en passant par un proxy, situé au-delà du pare-feu dans un autre pays. [2]

2.3_Fonctionnalités du proxy :

- La première fonctionnalité du serveur proxy est de permettre l'anonymisation de l'internaute grâce au masquage de l'IP du client : lorsque le proxy va s'adresser au serveur distant, il va utiliser une adresse IP publique et masque donc l'IP de l'internaute qui a fait la requête initiale. La réponse du serveur distant à la requête passe aussi par le proxy qui est donc le seul à connaître l'adresse IP de l'internaute.
- Le serveur proxy permet une accélération de la navigation : principalement par les compressions des données, le filtrage des contenus lourds mais aussi via la fonction de cache. La fonction cache est la capacité à garder en mémoire les pages les plus souvent visitées afin de pouvoir les leur fournir le plus rapidement possible.
- Enfin, le proxy permet d'assurer un suivi des connexions via les *logs*. En effet, le proxy enregistre les requêtes des utilisateurs lors de leurs demandes de connexion à Internet. Lorsque le filtrage est réalisé selon une liste de requêtes autorisées, on parle de *liste blanche*, et lorsque le filtrage se fait à partir d'une liste de sites interdits on parle de *liste noire*. [2]

3_Protocole ssh

SSH qui est une abréviation de 'Secure Shell' est un programme informatique ainsi qu'un protocole de communication sécurisé, impose un échange de clés de chiffrement en début de connexion par la suite tous les segments TCP sont authentifiés et chiffrés ce qui rend l'utilisation d'un sniffer sans aucune utilité. Il a été conçu dont l'objectif de remplacer les différents protocoles non chiffrès comme « rlogin , telnet , rcp , ftp , rsh »[3]

3.1_définition :

Il existe deux versions majeurs du protocole SSH :

- La version 1.0 : elle permet de se connecter à distance à un ordinateur afin d'obtenir un shell (Une interface système qui fournit l'interface utilisateur d'un système d'exploitation.) Mais cette version souffrait de problèmes de sécurité concernant la vérification de l'intégrité des données envoyées ou reçues et dont elle été vulnérable à des attaques actives,
  En outre, cette version implémentait un système sommaire de transmission de fichiers, et du port tunneling (Un tunnel est une encapsulation de données d'un protocole réseau dans un autre, situé dans la même couche du modèle en couches, ou dans une couche de niveau supérieur.)
- La version 2.0 : elle était à l'état de brouillon jusqu'en janvier 2006, elle est beaucoup plus sûre au niveau cryptographique et possède en plus un protocole de transfert de fichiers complet (SSH file trasnfer protocol) [3]

Habituellement le protocole SSH utilise le port TCP 22, il est particulièrement utilisé pour ouvrir un shell sur un ordinateur distant.

SSH peut également être utilisé pour transférer des ports TCP d'une machine vers une autre, créant ainsi un tunnel. Cette méthode est utilisée afin de sécuriser une connexion non sécurisée en la faisant transférer par le biais du tunnel chiffrés SSH.

Il est également possible de faire plusieurs sauts entre consoles SSH, c'est-a-dire ouvrir une console sur un serveur puis, à partir de cette console, on ouvre une autre sur un autre serveur.

3.2_SSH avec authentification pour clés :

L'authentification peut se faire sans l'utilisation de mot de passe ou de phrase secrète en utilisant la cryptographie asymétrique, la clé publique est distribuée sur les systèmes auxquels

on souhaite se connecter. La clé privée, qui doit être protéger par un mot de passe, reste uniquement sur le poste à partie duquel on se connecte. L'utilisation d'un « agent SSH » permet de stocker le mot de passe de la clé privée pendant la durée de la session utilisateur.[4]

3.2_Fonctionnement du protocole SSH :

SSH utilise les deux chiffrements : asymétrique et symétrique. Cela fonctionne dans cet ordre [4]

1. D'abord on doit utiliser le chiffrement asymétrique pour s'échanger discrètement une clé secrète de chiffrement symétrique.
2. Ensuite, on utilise tout le temps la clé de chiffrement symétrique pour chiffrer les échanges.

Les étapes de la création d'un canal sécurisé avec SSH :

Pour échanger une clé de chiffrement symétrique entre un client et un serveur, on doit chiffrer la clé grâce au chiffrement asymétrique.

- Le serveur envoie la clé publique en clair au client pour qu'il puisse chiffrer (figure suivante).

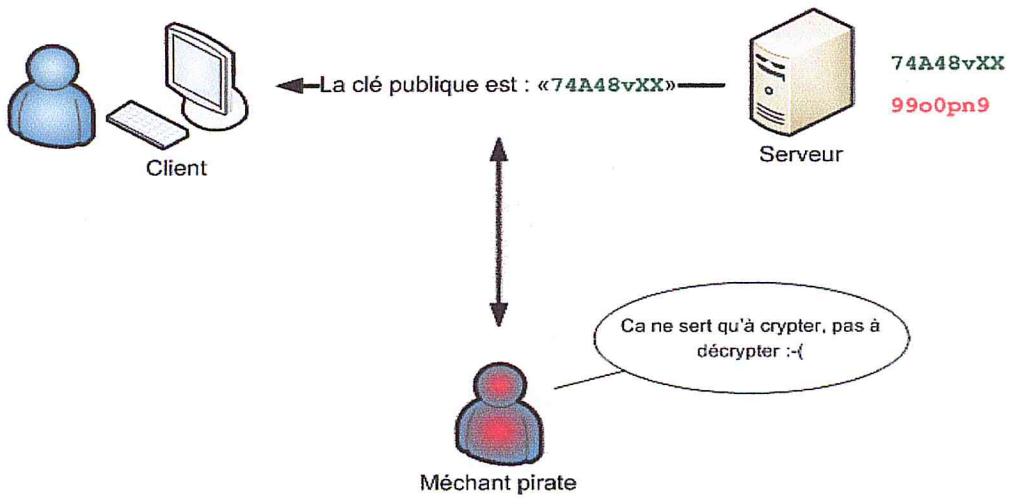

Figure 2: destrubition de clé public [4]

- Le client génère une clé de chiffrement symétrique (par exemple « topsecret ») qu'il chiffre grâce à la clé publique qu'il a reçue (figure suivante).

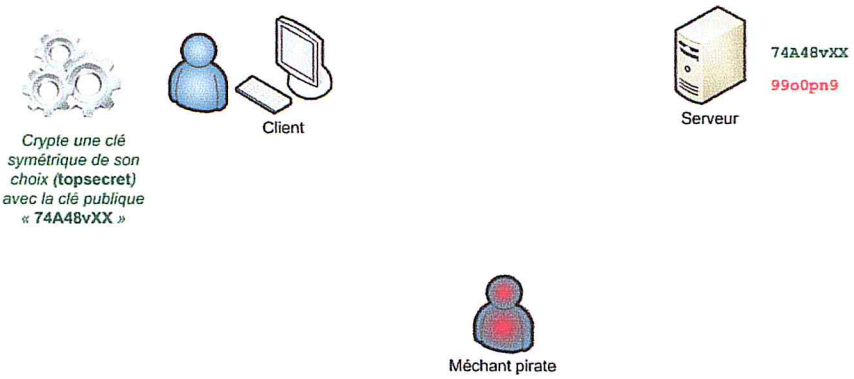

Figure 3: chiffrement de mot passe avec de clé public [4]

- Le client envoie la clé symétrique chiffrée au serveur. Le pirate peut l'intercepter, mais ne peut pas la déchiffrer car il faut pour cela la clé privée, connue seulement du serveur (figure suivante).

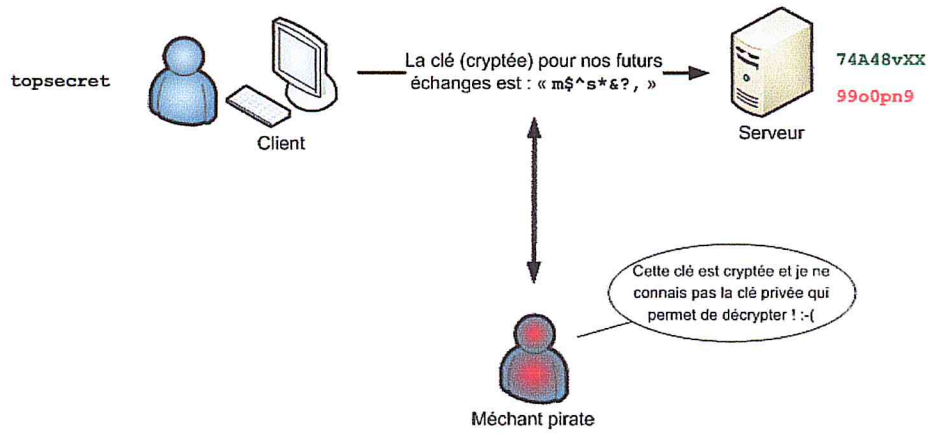

Figure 4: trenesemition de mot passe chiffré [4]

- Le serveur déchiffre la clé reçue grâce à sa clé privée qu'il a gardée à son niveau (figure suivante).

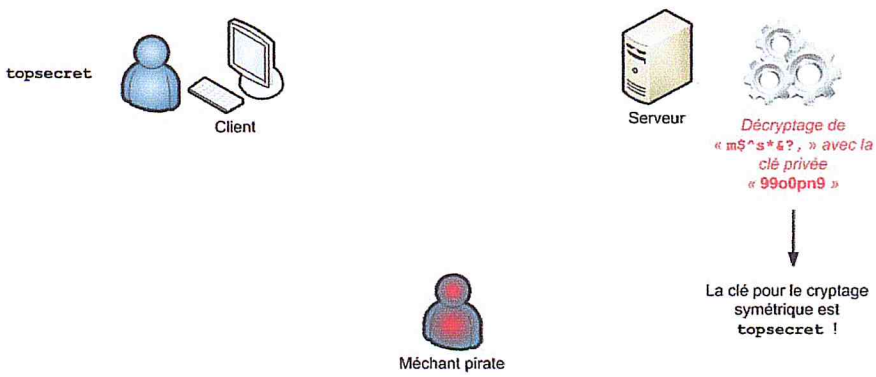

Figure 5 :Déchiffrement mot passe avec clé privé [4]

- Le client et le serveur connaissent maintenant tous les deux la clé symétrique « topsecret », et à aucun moment ils ne l'ont échangée en clair sur le réseau Ils peuvent donc s'envoyer des messages chiffrés de manière symétrique en toute tranquillité. Ce chiffrement est plus rapide et tout aussi sûr que le chiffrement asymétrique car le pirate ne connaît pas la clé (figure suivante)

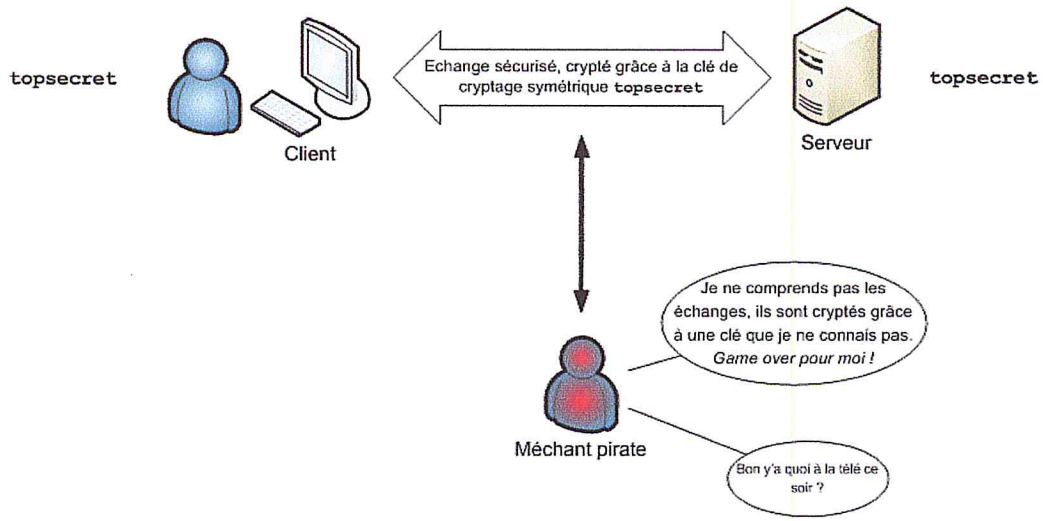

Figure 6 :Echange sécurise avec un cryptage symétrique [4]

4_Cryptographie

4.1_Définition Cryptographie :

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe crypter est parfois utilisé mais on lui préfèrera le verbe *chiffrer*.

Le fait de coder un message de telle façon à le rendre secret s'appelle chiffrement. La méthode inverse, consistant à retrouver le message original, est appelée déchiffrement.

Le chiffrement se fait généralement à l'aide d'une clef de chiffrement, le déchiffrement nécessite quant à lui une clef de déchiffrement. On distingue généralement deux types de clefs :

- **Les clés symétriques :** il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.

• **Les clés asymétriques** : il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

4.2_Buts de la cryptographie :

La cryptographie permet de résoudre quatre problèmes différents :

• **La confidentialité :** Le texte chiffré ne doit être lisible que par les destinataires légitimes. Il ne doit pas pouvoir être lu par un intrus.

• **L'authentification** : Le destinataire d'un message doit pouvoir s'assurer de son origine. Un intrus ne doit pas être capable de se faire passer pour quelqu'un d'autre.

• **L'intégrité :** Le destinataire d'un message doit pouvoir vérifier que celui-ci n'a pas été modifié en chemin. Un intrus ne doit pas être capable de faire passer un faux message pour légitime.

• **La non répudiation :** Un expéditeur ne doit pas pouvoir, par la suite, nier à tort avoir envoyé un message[5]

4.3_Mécanismes de la cryptographie :

Un algorithme de cryptographie ou un chiffrement est une fonction mathématique utilisée lors du processus de cryptage et de décryptage. Cet algorithme est associé à une clef (un mot, un nombre, ou une phrase). Afin de crypter une donnée avec des clés différentes le résultat du cryptage variera également. La sécurité des données cryptées repose entièrement sur deux éléments : l'invulnérabilité de l'algorithme de cryptographie et la confidentialité de la clef.

Un système de cryptographie est constitué d'un algorithme de cryptographie, ainsi que de toutes les clefs et tous les protocoles nécessaires à son fonctionnement.[6]

4.4_Algorithmes de la cryptographie :

4.4.1_Algorithmes symétriques (clef secrète) :

Un algorithme symétrique est un algorithme qui permet de transformer un texte en clair en texte chiffré en utilisant une clé et de retransformer le texte chiffré en texte en clair en utilisant la même clé.

Le secret de la communication est uniquement assuré par la clé qui est utilisée lors de la phase de chiffrement et de déchiffrement. L'algorithme utilisé ne fait pas partie du secret. [7]

On parle d'algorithmes symétriques car c'est la même clé qui sert à la fois au chiffrement et au déchiffrement du message.

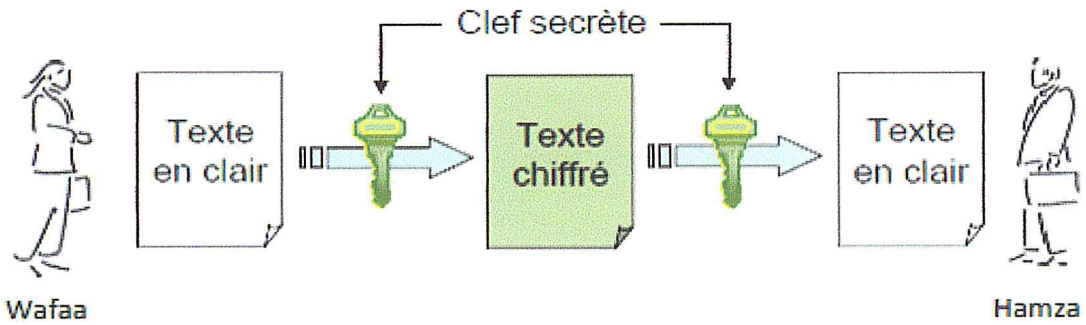

Figure 7 :cipe de l'algorithme symétrique [8]

4.4.2_Algorithmes asymétriques (clef publique)

Le problème des algorithmes symétrique est l'échange de clé : comment transmettre de manière fiable au destinataire la clé de chiffrement utilisée pour chiffrer le message que je lui envoie ? Il y a bien sûr le téléphone, mais il y a aussi les écoutes téléphoniques.

Les algorithmes asymétriques ont été inventés pour pallier précisément le problème de transmission sécurisée de la clé. [9]

On parle d'algorithmes asymétriques car ce n'est pas la même clé qui sert au chiffrement et au déchiffrement. Dans le cas de ces algorithmes, on parlera alors de clé privée et de clé publique. Ces deux clés, clé privée et clé publique, sont intimement liées par une fonction mathématique complexe.[10]

Les algorithmes asymétriques possèdent 2 modes de fonctionnement :

- Le mode chiffrement asymétriques dans lequel l'émetteur chiffre un fichier avec la clé publique du destinataire pour chiffrer. Le destinataire utilise sa clé privée pour déchiffrer le fichier. Dans ce mode, l'émetteur est sûr que seul le destinataire peut déchiffrer le fichier.

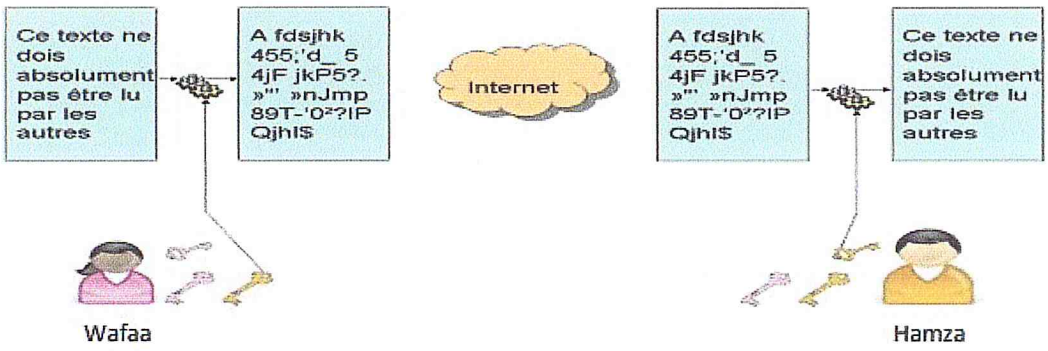

Figure 8: Chiffrement avec l'algorithme asymétrique. [5]

- Le mode signature dans lequel l'émetteur signe un fichier avec sa propre clé privée. Le destinataire utilise la clé publique de l'émetteur pour vérifier la signature du fichier. Dans ce mode, le destinataire est sûr que c'est bien l'émetteur qui a envoyé le fichier.

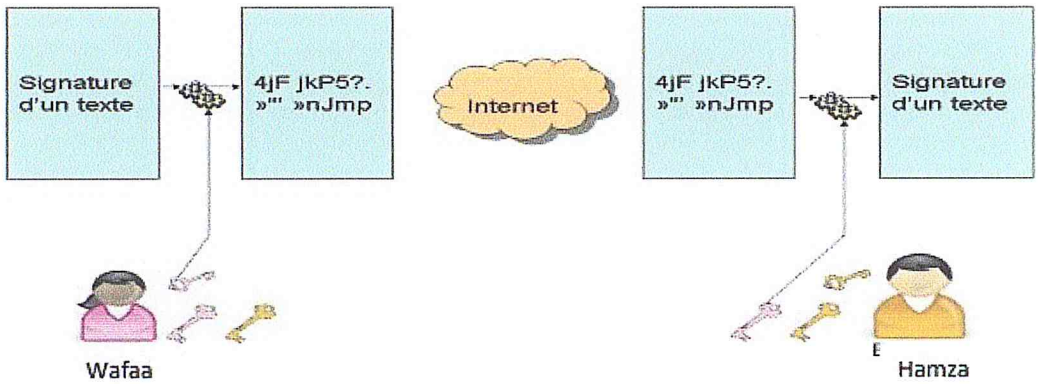

Figure 9 :Signature avec l'algorithme asymétrique. [5]

4.5_Définition de Fonction de hachage :

Une fonction de hachage (parfois appelée fonction de condensation) est une fonction permettant d'obtenir un condensé (appelé aussi condensat ou haché ou en anglais message digest) d'un texte, c'est-à-dire une suite de caractères assez courte représentant le texte qu'il condense.

La fonction de hachage doit être telle qu'elle associe un et un seul haché à un texte en clair (cela signifie que la moindre modification du document entraîne la modification de son haché). D'autre part, il doit s'agir d'une fonction à sens unique (one-way function) afin qu'il soit impossible de retrouver le message original à partir du condensé. S'il existe un moyen de

retrouver le message en clair à partir de l'haché, la fonction de hachage est dite « à brèche secrète ».[11]

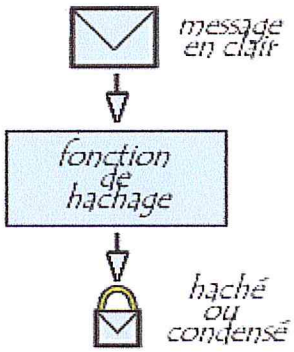

Figure 10 :Principe de fonction de hachage. [11]

4.6_Propriétés des fonctions de hachage :

Une fonction de hachage doit respecter les règles et propriétés suivantes :

1. La longueur de la signature doit être toujours la même *(quel que soit la longueur des données en entrée)*
2. Il n'est pas possible de trouver les données originales à partir des empreintes : Les fonctions de hachage ne fonctionnent que dans un seul sens.
3. Il ne doit pas être possible de prédire une signature. *(Il n'est pas possible d'essayer d'imaginer ce que pourrait être la signature en examinant les données)*
4. Et enfin, évidemment pour des données différentes : les signatures doivent être différentes. [11]

4.7_Utilisations des fonctions de hachage :

La liste ci-dessous est loin d'être exhaustive, car il existe bien d'autres utilisations des fonctions de hachage :

- Vérification de l'intégrité des fichiers ou des messages : Une application importante du hachage cryptographique est la vérification de l'intégrité d'un fichier (ou d'un message). Par exemple, la modification d'un fichier lors d'une transmission (ou toute autre activité) peut être prouvée en comparant la valeur de hachage du fichier avant et après la transmission.

- Vérification de mot de passe : Une autre application du hachage cryptographique est la vérification de mot de passe. Le stockage de tous les mots de passe utilisateur en clair peut entraîner une violation de sécurité massive si le fichier de mots de passe est compromis. Une façon de réduire ce danger est de stocker seulement la valeur de hachage de chaque mot de passe. Pour authentifier un usager, le mot de passe fourni par l'utilisateur est haché et comparé avec la valeur de hachage stockée. Avec cette approche, les mots de passe perdus ne peuvent pas être récupérés s'ils sont oubliés ; ils doivent être remplacés par de nouveaux mots de passe.
- La signature électronique (ou signature numérique) des documents. Les documents envoyés sont passés dans une fonction de hachage, et l'empreinte est envoyée chiffrée (avec le mécanisme de clés asymétriques) en même temps que les documents. Il suffit au destinataire de déchiffrer l'empreinte reçue et vérifier que celle-ci correspond bien au calcul de l'empreinte des données reçues.
- Pour le stockage des données : si la signature d'un fichier est présente, alors on peut dire que le fichier est présent : il n'est pas nécessaire de l'enregistrer à nouveau. Cela peut être intéressant car si le fichier est très gros, il faudra beaucoup moins de temps pour calculer sa signature que de l'enregistrer à nouveau.[11]

4.8_Fonctions de hachage célèbres

4.8.1_MD5 :

Cette fonction de hachage est toujours très utilisée bien qu'au niveau de la sécurité, il est recommandé de passer à des versions plus robustes car des suites de collisions ont été trouvées. Cette fonction renvoie une empreinte de 128 bits.

4.8.2_SHA1 :

Était la fonction remplaçante de MD5 car elle produisait des empreintes 160 bits et avec impossibilité de trouver des collisions … jusqu'en 2004-2005, date à laquelle des attaques ont prouvé des possibilités de générer des collisions. Depuis, cette date il n'est plus conseillé d'utiliser la fonction SHA1. Mais, elle est encore très utilisée. Les certificats numériques utilisant SHA1 ne sont plus plus valides au 31 décembre 2016.

4.8.3_SHA2 :

SHA256 et SHA512 sont 2 des grands standards utilisés actuellement, car il n'y a pas à ce jour d'attaques ayant trouvé des failles de sécurité sur ces fonctions de hachage. Elles produisent comme vous pouvez vous en douter des signatures de respectivement 256 et 512 bits.[11]

5_Conclusion

Notre but est de développer un proxy implémentant le protocole SSH et des fonctions de cryptographie, ces fonctions ont bien été définie dans ce chapitre.
Mais avant de commencer la conception de notre proxy, une analyse de l'existant doit être réalisée ce qui fait l'objet du chapitre suivant.

Chapitre II : Conception de la solution

1_Introduction :

Dans ce chapitre nous allons d'abord présenter la société qui nous a reçu. Nous allons abordé par la suite la problématique proposé par la société, nous allons ensuite présenter notre solution qu'on a implémenter pour cette problématique, nous allons parler aussi de l'étude des besoins qu'on a fait au siens de la société afin de récolter les informations brutes qu'on les a ensuite utilisé pour implémenter notre base de données qui va être une partie essentiel de notre projet, nous allons finir par donner une architecture globale de notre solution proposé et expliquer son fonctionnement et nous allons faire une petite étude comparative entre notre solution et les solutions existantes sur le marché qui répond à la même problématique.

2_Présentation de l'organisme d'accueil :

DECIMA Technologies : est une Société A Responsabilité Limitée (SARL) crée en aout 2017 et à partir de cette date elle a débuté son activité. Sa principale mission est d'offrir à ces clients une expertise reconnue dans le domaine des technologies de l'information en général et surtout dans le domaine de la sécurité IT. Le développement de Decima Technologies est animé par professionnalisme, proximité et réactivité avec un seul objectif qui est : « simplifier le quotidien de ces clients.

Son activité est basée sur les quatre pôles ci-dessous :

- Consultling (consultation) : elle offre à ces clients des audits, des conseils et une gestion des projets dans plusieurs domaines : infrastructure, réseaux, logiciels et sécurité.
- Sécurité : elle offre une expertise en matière de cyber sécurité et cyber défense grâce à des technologies de pointe ainsi qu'in savoir-faire éprouvé.
- Développement : elle offre une possibilité de développer des solutions et/ou des applications sur mesure et spécialement adaptées aux besoin du client.
- Infogérance : elle propose une offre globale de maintenance s'articulant autour de contrats personnalisés et modulables afin de répondre à tout type de demande.

Notre stage de fin d'étude s'est déroulé au sein de la société DECIMA qu'on décrira brièvement dans ce que suit.

2.1_Organigramme de la société :

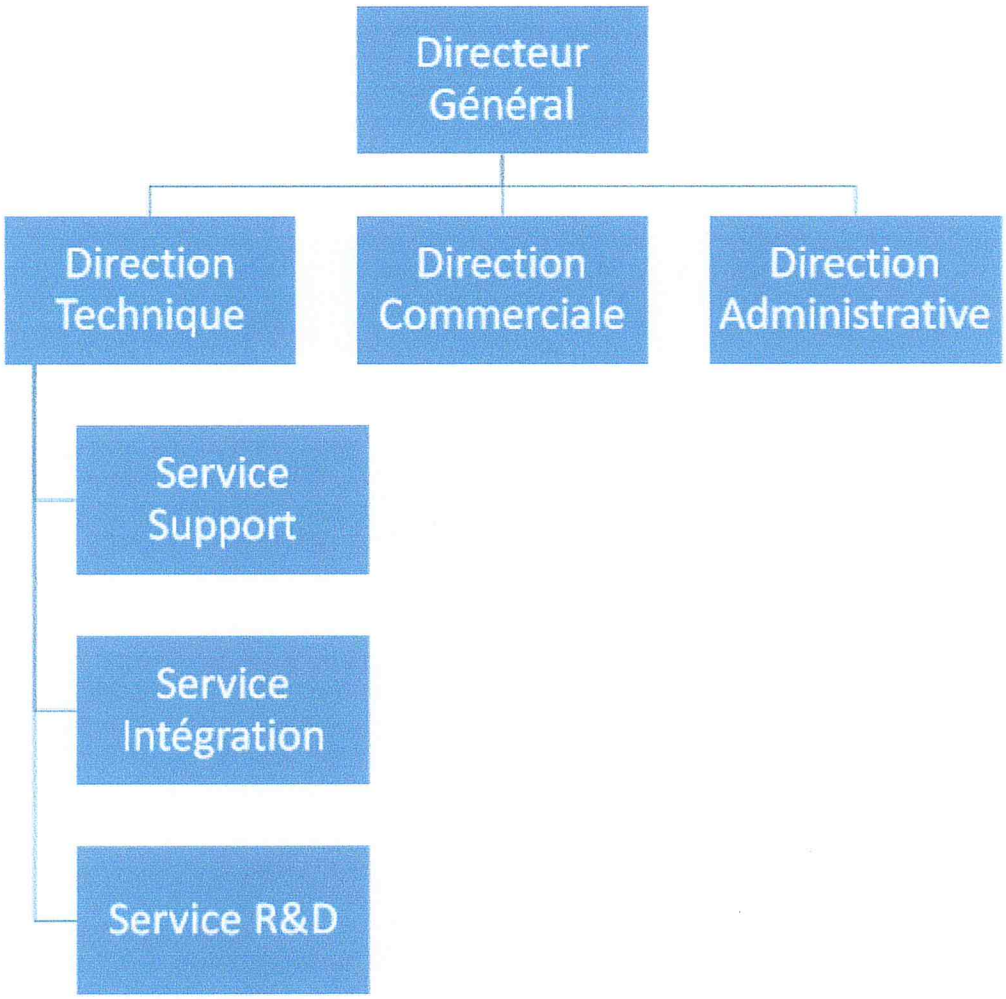

Figure 11: Organigramme de la société

La direction Technique de la société DECIMA Technologies était le lieu de travail (cadre du stage) pendant la durée du stage.

3_Présentation de la problématique :

Les employés de DECIMA Technologies utilisent des connexions sécurisées par le protocole 'SSH' vers des serveurs afin de faire leurs travails (demander des services, récupérer des

données, …). L'administrateur réseau de la société veux superviser les interactions entre les employés et ces serveurs et établir aussi des droits d'accès pour que chaque employé va avoir le droit de consulter seulement le serveur qu'il l'autorise de consulter tout en rajoutant une 'black-list' qui va contenir les commandes que l'employé ne peut pas exécuter dans le serveur qu'il est autorisé à consulter.
Plusieurs solutions ont été proposés par des développeurs pour remédier à ce problème, sauf que les employés vont être obligés de travailler avec les programmes développés par ces développeurs.
La société refuse d'adapter ces solutions et veux que ces employés travaillent avec les outils fournis par défaut avec le système d'exploitation (openSSH par exemple).
Donc la problématique est d'implémenter une solution qui va permettre de visualiser les interactions entre les employés de la société et les serveurs et aussi d'établir des droits d'accès pour ces employés toute en les laissant utiliser les outils fournit par défaut avec les systèmes d'exploitation.

4_Présentation de la solution :

Afin de visualiser un trafic dans un réseau (les interactions entre les employés et les serveurs) et d'établir des droits d'accès qui vont contenir eux même des 'blacklist', on devra implémenter un proxy qui se situe entre les employés et les serveurs, les employés vont donc passer par notre proxy qui va les rediriger vers les serveurs, c'est ainsi que les accès vont être centraliser (tous passe par le proxy) et surveiller par l'administrateur.

Les spécifications de notre proxy seront les suivant :

- Ça va être un proxy sécurisé par le protocole SSH afin de permettre les employés d'utiliser un client SSH fournit par défaut avec le system d'exploitation de leurs machines.
- Ce proxy va enregistrer un fichier journal qui contient les commandes envoyées par l'employé vers le serveur ainsi que les réponses de serveurs envoyés vers l'employé
- Ce proxy va être relier avec une base de données qui va contenir des informations nécessaires pour déterminer si un employé X a le droit de consulter un serveur Y, ensuite pour vérifier si l'employé X peut exécuter une commande spécifique dans ce serveur ou non.

Pour faciliter la tâche de l'administrateur réseau on implémente aussi une interface web qui va contenir un 'espace login' ensuite un 'Dashboard', qui aura comme but de manipuler la base de donnée (ajouter des employés, ajouter des serveurs, établir des droit d'accès…) et bien sur consulter les fichiers journal des employés.

Remarque : l'administrateur réseau peut désigner un employé qui va être un auditeur et donc il aura le droit d'utiliser l'interface web uniquement pour la consultation (consulter les informations des employés, consulter les informations des serveurs…).

5_Etude des besoins :

Dans le but de bien implémenter notre solution pour qu'elle répond exactement à la problématique proposée par la société, on fait une étude pour bien déterminer les besoins de la société concernant cette problématique et pour bien définir les informations dont on aura besoin au cours de la réalisation de notre projet.
Pour cela on s'est déplacé vers la société, et on a discuté avec les employés (ceux qui vont utiliser la solution) ainsi qu'avec l'administrateur réseau (celui qui va utiliser l'interface web pour gérer la solution), pour faire une récolte des informations.
Cette étude nous a permis de construire la base de données suivante.

6_Réalisation de la base de données :

6.1_ schéma conceptuel :

Le schéma conceptuel résultant de la récolte d'information est le suivant :

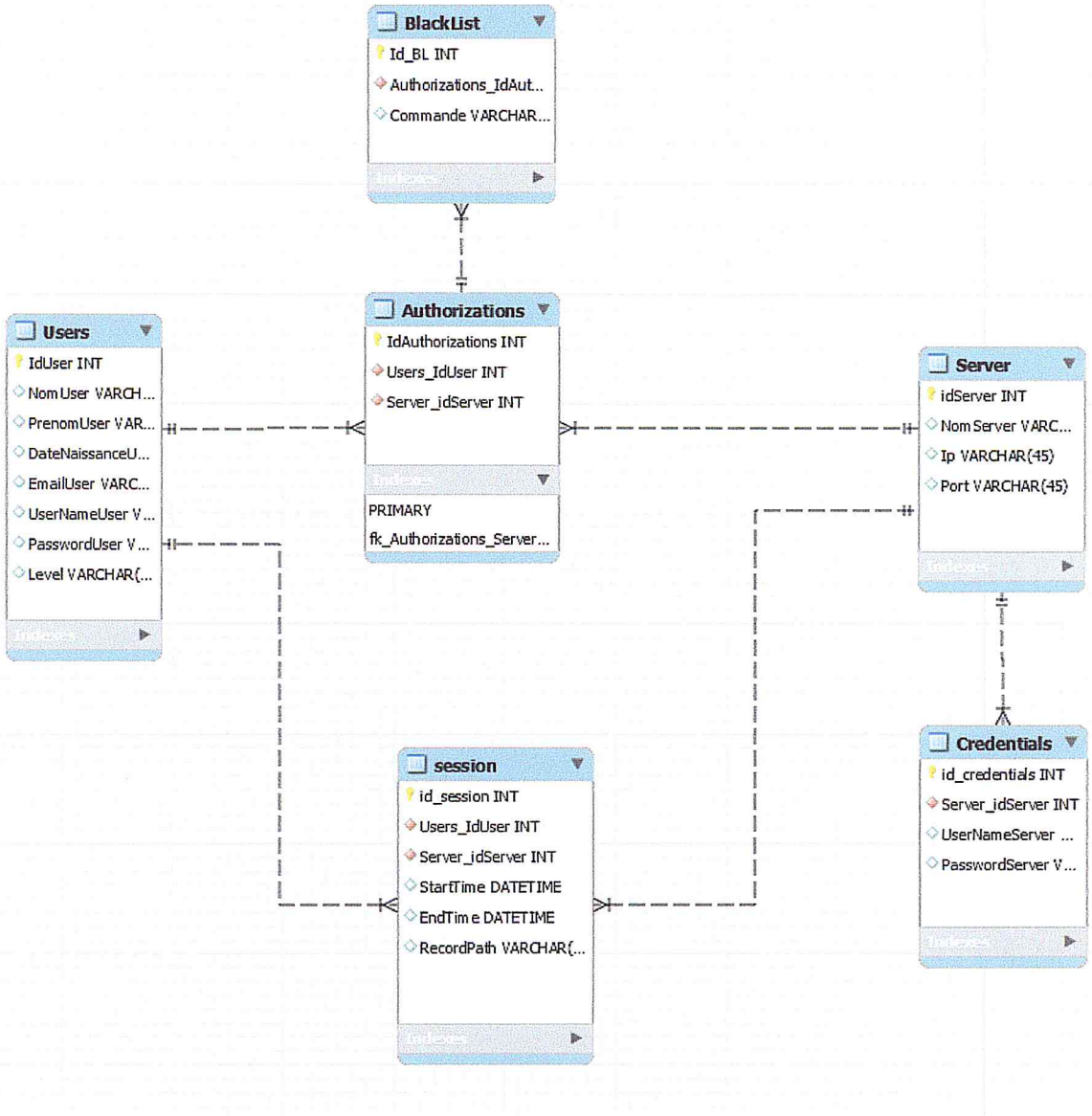

Figure 12 : schéma relationnel de la base de données

6.2_Dictionnaire de données :

classe	attributs	Définition
Users	IdUser	un attribut de type entier qui sera l'ID de chaque employé, il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	NomUser	un attribut de type 'VARCHAR' qui sera le nom de l'employé
	PrenomUser	un attribut de type 'VARCHAR' qui sera le prénom de l'employé
	DateNaissanceUser	un attribut de type 'DATE' qui sera la date de naissance de l'employé

	EmailUser	un attribut de type 'VARCHAR' qui sera l'email de l'employé
	UserNameUser	un attribut de type 'VARCHAR' qui sera le nom d'utilisateur pour l'employé qui va l'utiliser pour s'authentifier
	PasswordUser	un attribut de type 'VARCHAR' qui sera le mot de passe de l'employé qui va l'utiliser pour s'authentifier
	Level	un attribut de type 'VARCHAR' qui aura le but de différencier un employé simple et un administrateur.
Server	idServer	un attribut de type entier qui sera l'ID de chaque serveur, il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	NomServer	un attribut de type 'VARCHAR' qui sera le nom du serveur
	Ip	un attribut de type 'VARCHAR' qui sera l'adresse IP du serveur
	Port	un attribut de type 'VARCHAR' qui sera le port du serveur
Credentials	id_credentials	un attribut de type entier qui sera l'ID de chaque credentials (sa veux dire l'ID de chaque série d'informations liés à un serveur spécifique) , il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	Server_idServer	c'est la clefs primaire de la table serveur qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
	UserNameServer	un attribut de type 'VARCHAR' qui sera le nom d'utilisateur pour le serveur qu'on doit l'utiliser pour s'authentifier au serveur
	PasswordServer	un attribut de type 'VARCHAR' qui sera le mot de passe du serveur qu'on doit l'utiliser pour s'authentifier au serveur
Session	is_session	un attribut de type entier qui sera l'ID de chaque session, il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	Users_IdUser	c'est la clefs primaire de la table Users qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
	Server_idServer	c'est la clefs primaire de la table Server qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
	StartTime	un attribut de type 'DATE' qui sert pour déterminer la date de début d'une session
	EndTime	un attribut de type 'DATE' qui sert pour déterminer la date de fin d'une session
	RecordPath	un attrubut de type 'VARCHAR' qu'on va utiliser pour enregistrer le chemin vers le fichier qui contient l'historique d'une session

Authorizations	IdAuthorizations	un attribut de type entier qui sera l'ID de chaque autorisation, il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	Users_IdUser	c'est la clefs primaire de la table Users qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
	Server_idServer	c'est la clefs primaire de la table Server qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
BlackList	Id_Bl	un attribut de type entier qui sera l'ID de chaque commande interdite, il représente la clefs primaire de cette table et il aura la spécification d'être 'auto-incrément'
	Authorizations_ IdAuthorizations	c'est la clefs primaire de la table Authorizations qui sera dans cette table comme une clef étrangère pour établir la relation entre les deux tables.
	Commande	un attribut de type 'VARCHAR' qu'on va utiliser pour enregistrer la commande interdite

Tableau 1: dictionnaire de données

7_Architecture Globale :

Après avoir réalisé la base de données et avant de commencer la réalisation ou l'implémentation de notre solution on donne une architecture globale ensuite nous passons à l'explication du fonctionnement de notre proxy.

7.1_Vue Globale :

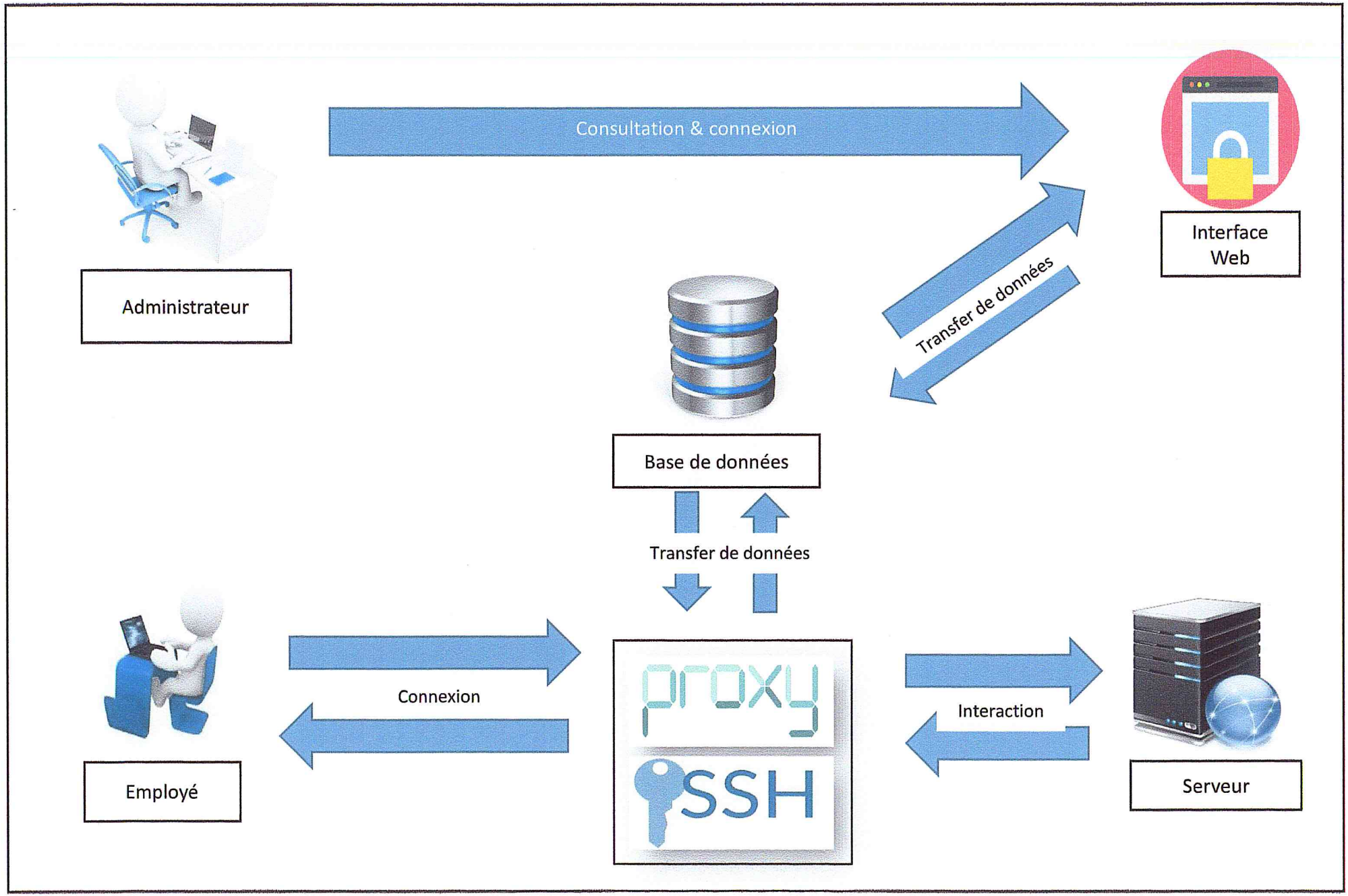

Figure 13: Vue globale de la solution

7.2_diagramme de cas d'utilisation :

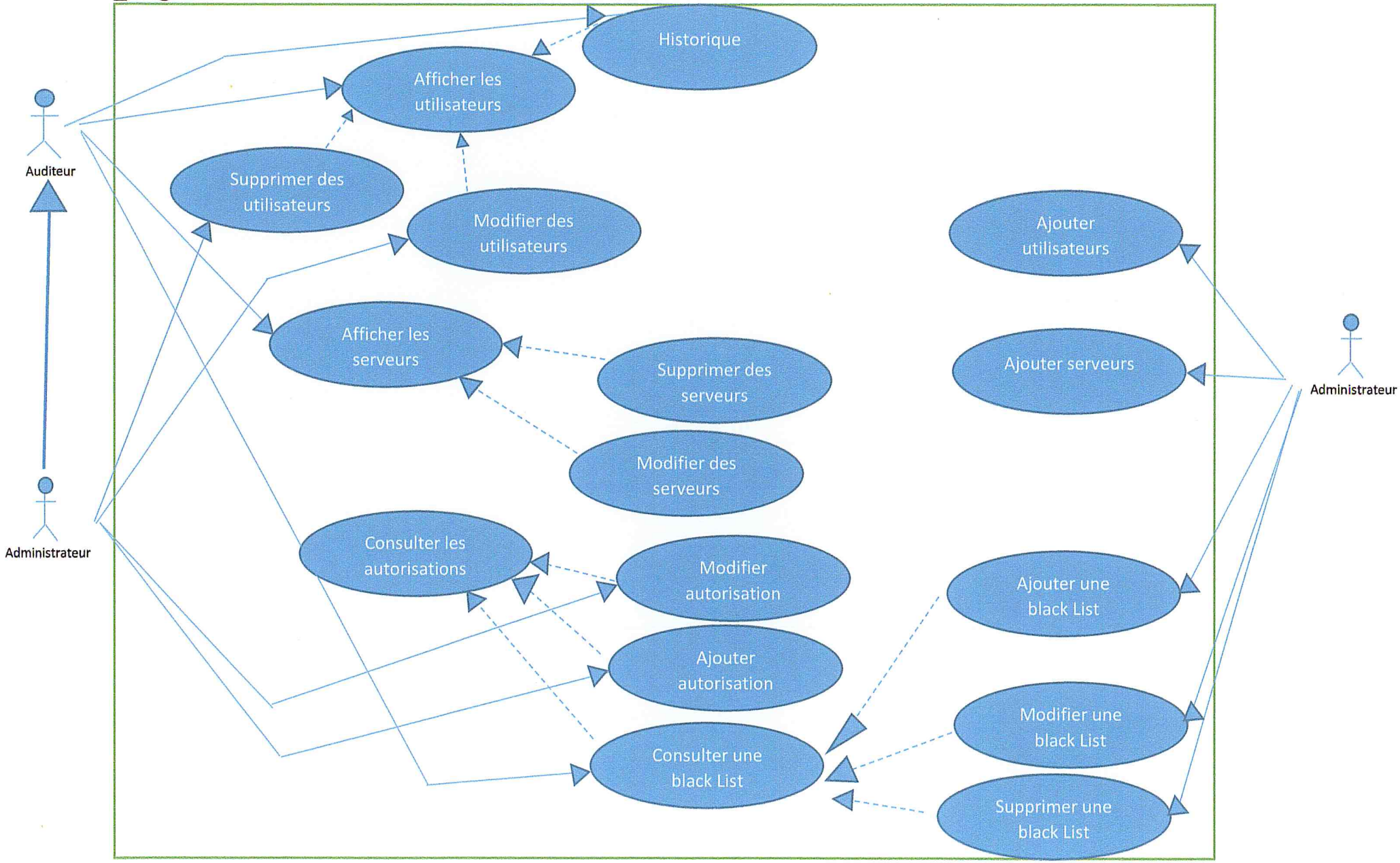

Figure 14: diagramme de cas d'utilisation

7.3 diagramme de séquences :

7.3.1 diagramme séquentiel qui explique l'utilisation de notre solution par un emplové de la société:

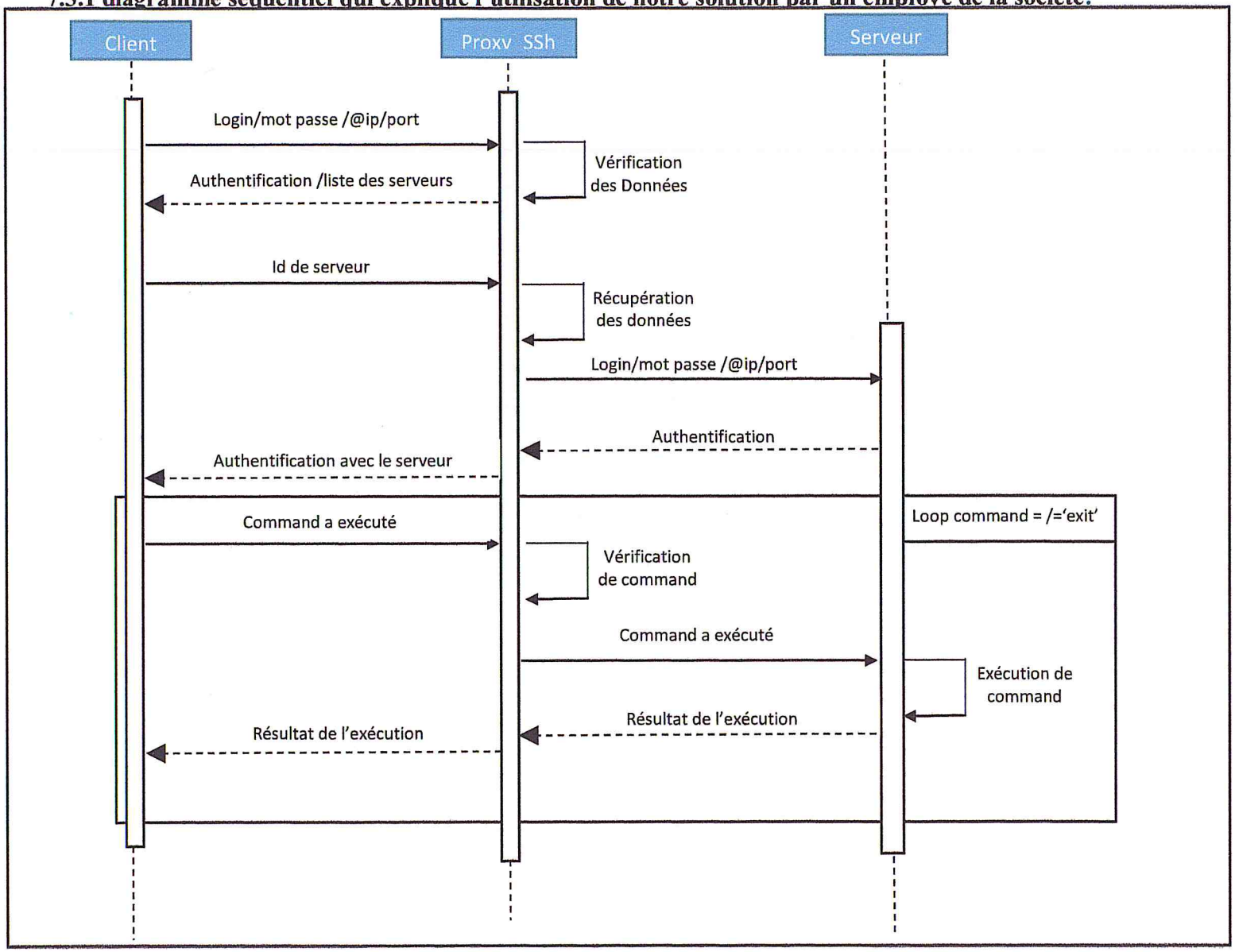

Figure 16: diagramme séquentiel pour un employé

7.3.2 diagramme séquentiel qui explique l'utilisation de notre solution par un administrateur :

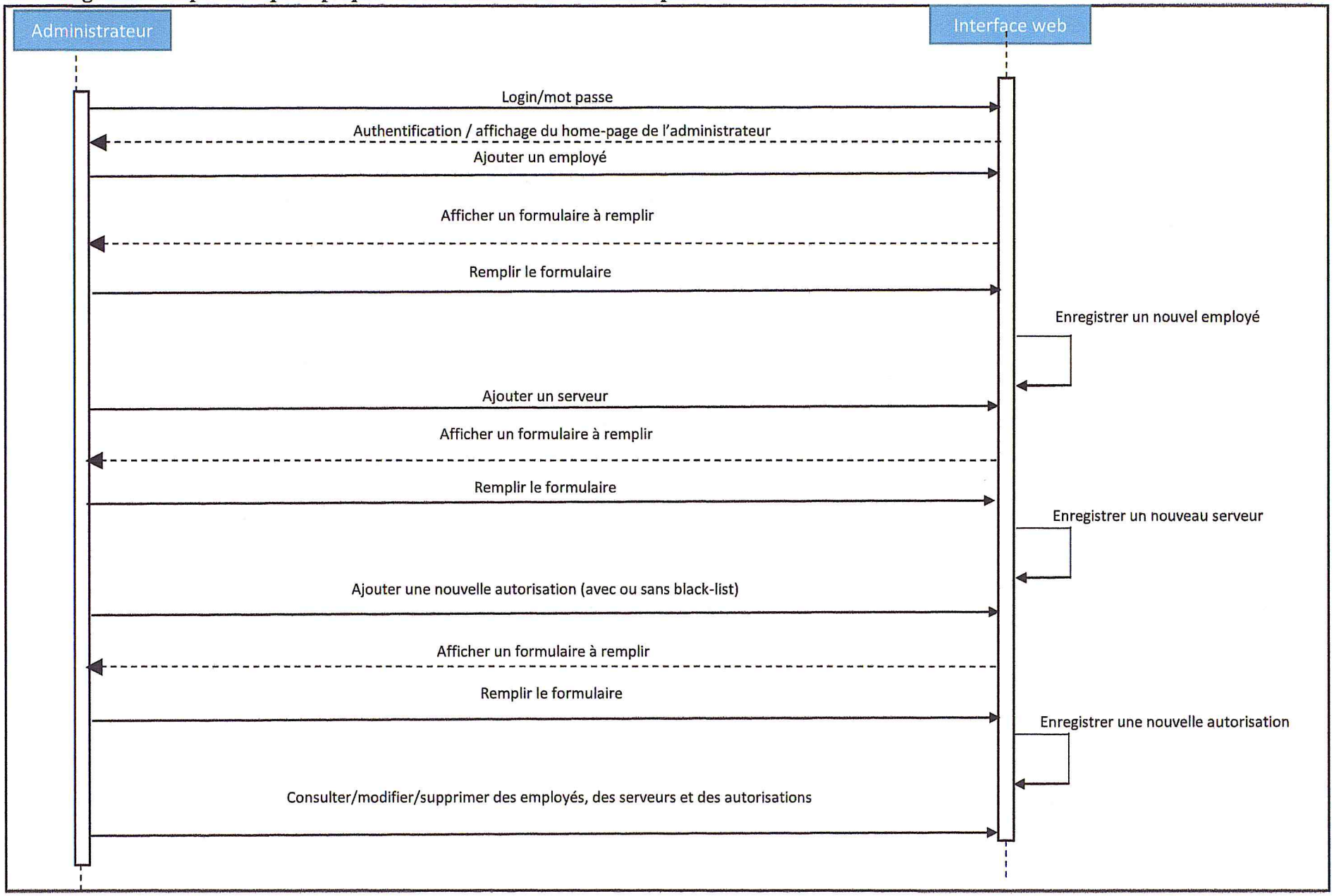

Figure 17: diagramme séquentiel pour un administrateur

7.3.3 diagramme séquentiel qui explique l'utilisation de notre solution par un auditeur:

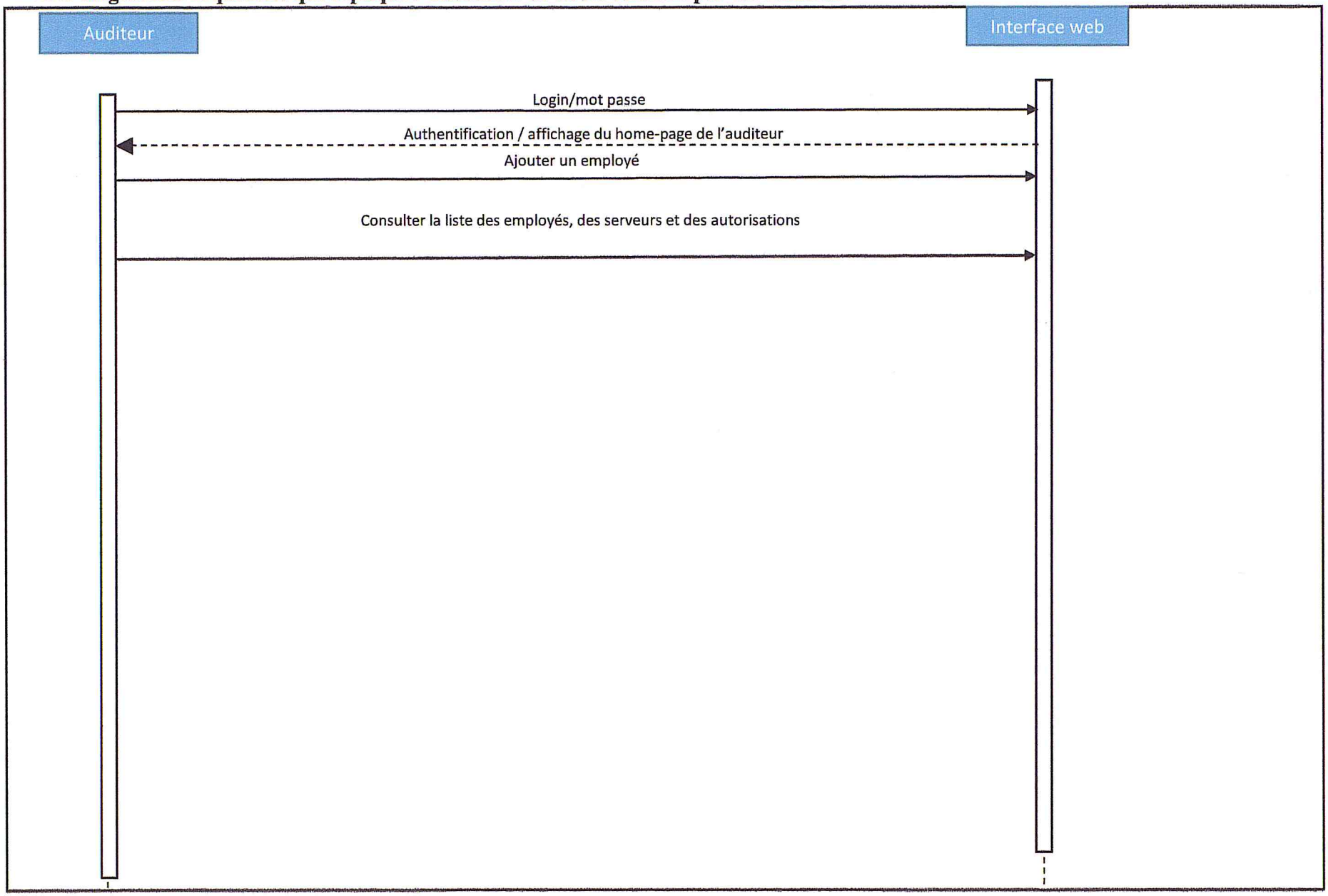

Figure 18: diagrame séquentiel pour un auditeur

7.4_Fonctionnement :

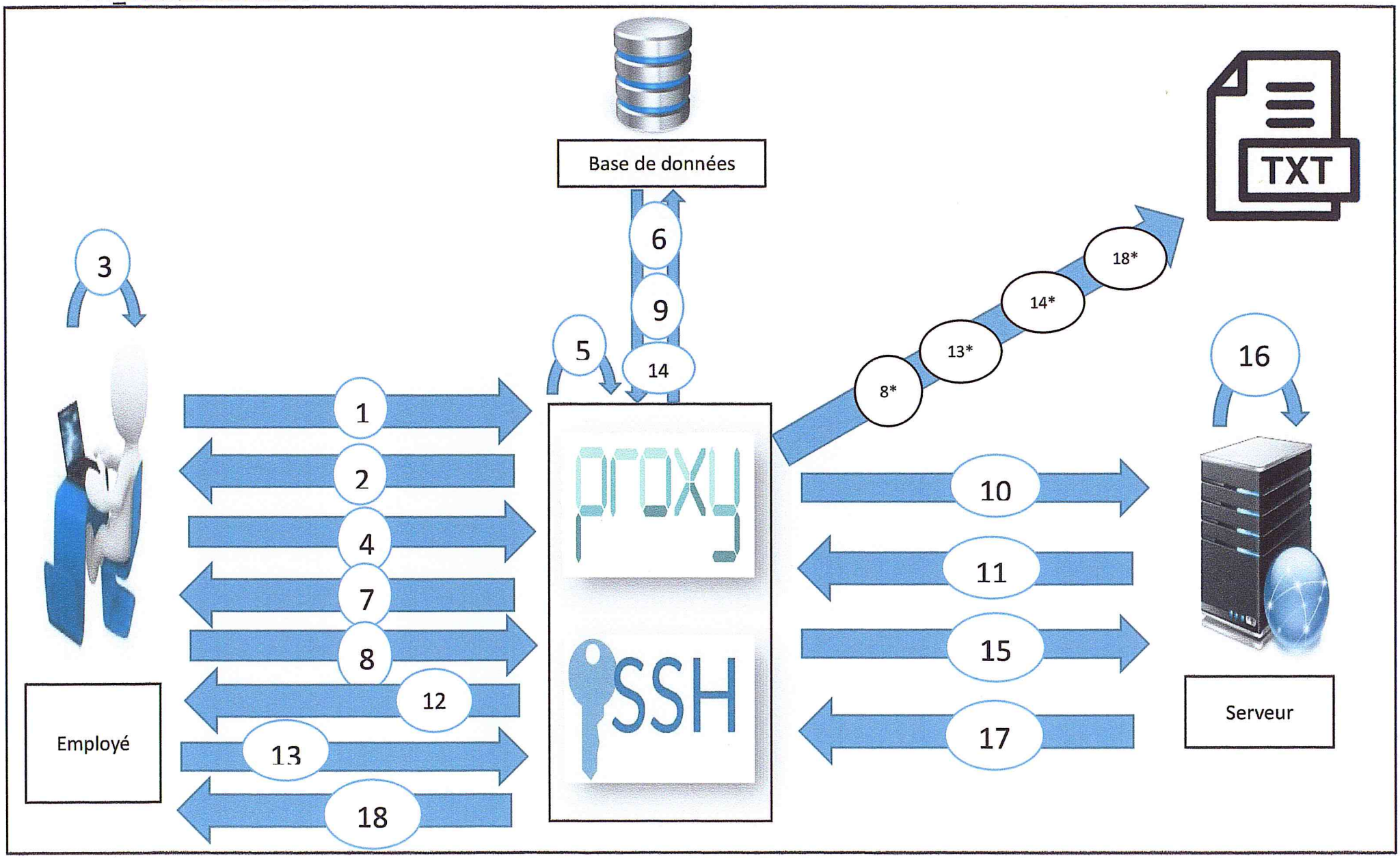

Figure 19: fonctionnement de la solution

Actions	Description
1	L'envoie d'une demande d'authentification au proxy de la part de l'employé. (pour s'authentifier)
2	L'envoie de la clé public du proxy a l'employé.
3	Chiffrement des informations (nom d'utilisateur et mot de passe) de l'employé en utilisant la clé public du proxy.
4	L'envoie des informations cryptés (nom d'utilisateur et mot de passe) au proxy.
5	Déchiffrement des informations reçus de la part de l'employé a travers la clé privée du proxy.
6	Utilisation des informations déchiffrés pour faire une vérification dans la base de donnée et voir s'il existe un employé qui porte ces informations. Dans le cas où le proxy trouve une correspondance il cherche à partir de la base de données les serveurs que cette employé peut accéder. Dans le cas contraire le proxy affiche un message d'erreur et donne la main a l'employé pour saisir une autre fois son nom d'utilisateur et son mot de passe.
7	L'envoie de la liste des serveurs à l'employé.
8	L'envoie du choix au proxy.
8*	L'enregistrement du choix de l'employé dans le fichier texte.
9	Récupération des informations nécessaires du serveur que l'employé a choisi pour accéder au serveur.
10	L'envoie des informations nécessaires pour l'authentification (nom d'utilisateur du serveur et son mot de passe)
11	L'envoie de la réponse de la part du serveur vers le proxy : _authentification avec succès : s'il s'agit des informations correctes. _authentification échoué : si les informations ne sont pas correctes.
12	L'envoie de la réponse obtenu de la part du serveur vers l'employé.
13	L'envoie d'une commande à exécuter dans le serveur vers le proxy.
13*	L'enregistrement de la commande envoyer par l'employé dans un fichier texte qui contient l'historique entre l'employé et le serveur.
14	Vérification si la commande n'est pas interdite (elle n'est pas interdite dans ce serveur de la part de cette employé)

14*	Enregistrement de la commande interdite de cette façon dans le fichier texte : 'l'employé (nom de l'employé) a essayé d'utiliser (la commande) qui est une commande interdite)
15	L'envoie de la commande à exécuter dans le serveur si cette dernière n'est pas interdite.
16	Exécution de la commande dans le serveur.
17	L'envoie du résultat de la commande vers le proxy.
18	L'envoie du résultat de la commande vers l'employé.
18*	L'Enregistrement du résultat de la commande dans le fichier texte.

Tableau 2: description des étapes du fonctionnement

8_Positionnement de notre solution :

On se positionne avec ce qu'il existe sur le marché qui répond à la même problématique pour qu'on évalue notre solution, et on peut dire que peux sont les solutions existantes concernant cette problématique qui offre la possibilité de travailler avec les outils fournis par le system d'exploitation, les solutions les plus connus solution qu'on a trouvée sur le web sont 'WALLIX' et 'BALABIT' et donc on se compare avec ces dernières d'une manière générale pour se positionner :

- Ces deux solutions ne sont pas 'open source' et donc pour pouvoir utiliser l'un de ces deux la société doit acheter la licence d'utilisation, et pour des raisons de budget la société n'a pas pu acheter l'une de ces solutions pour l'utiliser.
- Ces Solutions sont développées à l'étranger et donc en cas d'utiliser ces solutions on aura la notion de dépendance de l'étranger.
- Vu que notre solution est implémentée spécialement pour la société, elle peut être customiser selon les besoins réels de la société, par contre si la société utilise les solutions payantes elle va utiliser des solutions standards qui répond d'une manière générale à la problématique.
- Vu que DECIMA technologies est une société spécialisé dans la 'sécurity IT' elle va avoir le droit de distribution et de la commercialisation de notre solution.

9_Conclusion :

Après avoir terminé l'analyse de l'existant qui nous a aider à faire la conception de notre solution afin d'être prêts pour commencer la réalisation de notre proxy avec ces différentes fonctionnalités ce qu'il va être le sujet traiter dans le chapitre suivan

Chapitre III : Réalisation

1. Introduction :

A partir de la conception réalisée dans le chapitre précédent nous entamerons la réalisation de notre projet.

2. Outils de mise en œuvre :

Pour pouvoir commencer la réalisation de notre projet, il a fallu qu'on prépare nos machines qui utilisent le system d'exploitation 'windows' par défaut, donc on a utilisé une machine virtuelle pour pouvoir la configurer comment nous voulons, voici donc les étapes qu'on a suivi pour la préparation de notre machine :

2.1_Installation de VMware Workstation :

C'est le logiciel qui va nous permettre de créer des machines virtuelles sur nos machines réels (physiques), ceux-ci pouvant être reliés au réseau local avec une adresse IP différente, tout en étant sur la même machine physique. Il est possible de faire fonctionner plusieurs machines virtuelles en même temps, ce qu'on va faire en lançant la VM qui contient notre solution qui va jouer le rôle du proxy ainsi que d'autres VMS qui vont jouer le rôle des serveurs cibles.

Pour l'installation il suffit d'aller sur le site officiel de 'VMware' et télécharger la version trial compatible avec 'Windows' qui est gratuite ensuite l'installer le plus normalement du monde. [12]

2.2_Installation du system d'exploitation sur la VM :

On a choisi de travailler avec 'Ubuntu server' qui est un système d'exploitation basé sur le 'Kernel Linux', pour la simple raison qu'il est plus adaptative pour établir des réseaux et créer des serveurs. [13]

Pour l'installation on télécharger d'abord le fichier '.iso' depuis le site officiel 'www.ubuntu.com'

Ensuite nous créons une machine virtuelle en utilisant 'VMware Workstation', on spécifie le matériel nécessaire pour cette dernière et on installe 'Ubuntu server' en utilisant le fichier '.iso' télécharger depuis le site.

3.Réalisation de la solution

Notre solution se résume en un proxy qui se situent entre un client (qui représente l'employé de la société) et un serveur final, ce proxy sera sécurisé par le protocole 'SSH', et on va implémenter une interface web pour l'administrateur du réseau qui va lui permettre de gérer les serveurs externes, les utilisateurs et leurs droits d'accès, ainsi que consulter l'historique de chaque employé de la société. Et il y'aura aussi un auditeur qui peut utiliser l'interface seulement pour consulter (il n'aura pas le droit d'ajouter, modifier et supprimer des utilisateurs ou bien des serveurs ou bien des autorisations)
Pour cela on commence par la réalisation de la partie web qui va enregistrer les informations que l'administrateurs souhaite sauvegarder dans notre base de données, pour que la partie cœur de notre solution récupère les informations nécessaires pour son fonctionnement, ensuite nous passerons à l'implémentation de la partie cœur qui représente la partie la plus importante de notre solution d'où son nom.

3.1 Réalisation de la partie web :

Cette partie représente l'implémentation de l'interface web, on montre des exemples d'exécution en suivant deux scénarios le premier sera dédié à l'administrateur de réseau et le deuxième sera dédié à l'auditeur.

3.1.1_Utilisation de l'interface par un administrateur :

Nous allons traiter le cas où un administrateur va utiliser l'interface, et donc dès qu'il accède à l'interface il aura cette page que nous avons montré dans la figure suivante :

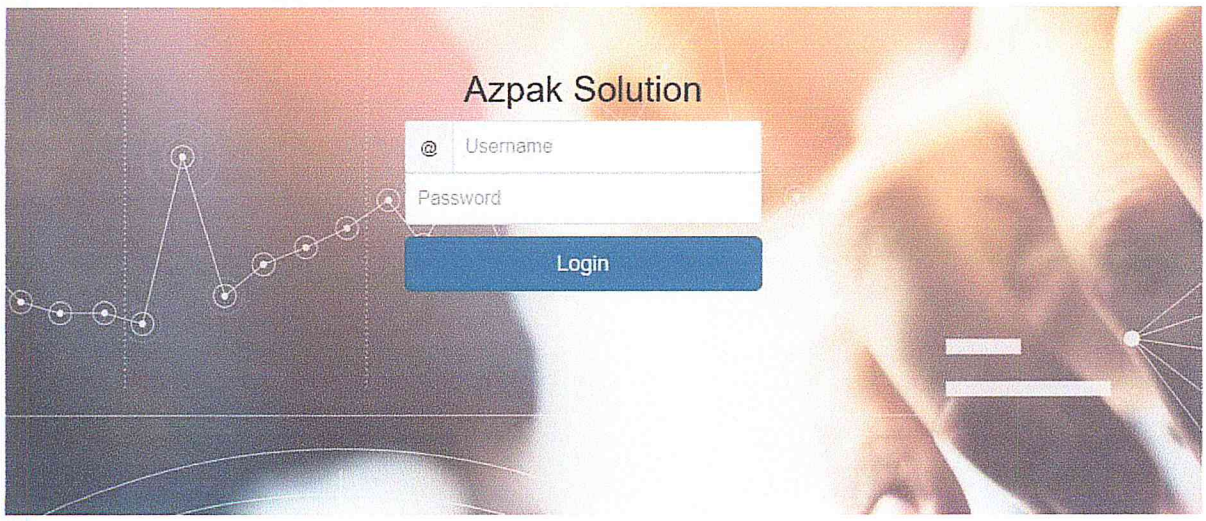

Figure 20: page de login

L'administrateur va donc introduire son nom d'utilisateur et son mot de passe pour s'authentifier et accéder à une autre page (montrer dans la figure suivante) où il peut éditer les informations des employés et établir des droits d'accès pour ces derniers ainsi qu'éditer les informations des serveurs.

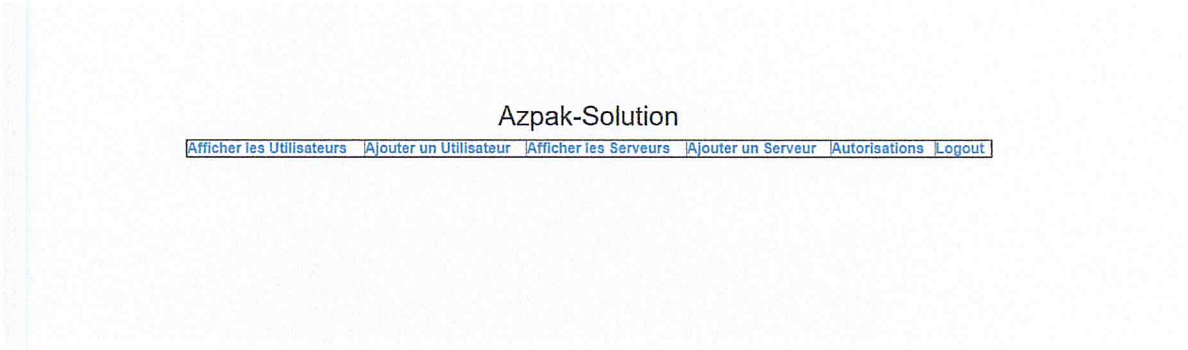

Figure 21: page d'acceuil pour l'administrateur

Une fois arriver à cette page l'administrateur aura le choix d'afficher les utilisateurs (employés), ajouter un utilisateur, afficher les serveurs, ajouter un serveur, ou bien établir des autorisations, et on ajoute un bouton nommé 'logout' qui permet de se déconnecter. Maintenant si l'administrateur souhaite modifier ou supprimer des utilisateurs (employés) existants il sera obligé d'afficher la liste des utilisateurs d'abord ensuite faire le choix entre modifier les informations d'un certain utilisateur ou bien le supprimer comme nous montrons dans la figure suivante :

Figure 22: affichage de la liste des utilisateurs

On ajoute une dernière case qui sera le lien vers une autre page qui contient l'historique de l'utilisateur en question. L'administrateur va se retrouver dans une page qui contient l'historique de chaque session entre l'utilisateur et le serveur dans un fichier texte, il n'aura qu'à choisir un fichier texte pour l'afficher. (Voir les deux figures suivantes)

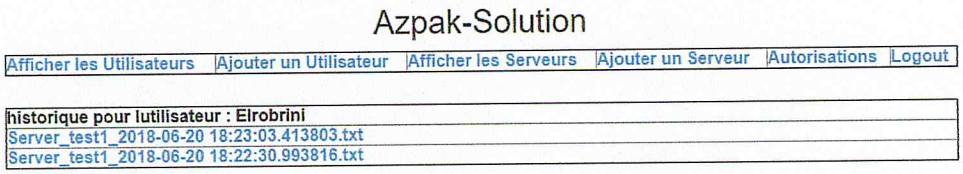

Figure 23: affichage de la liste des fichiers textes

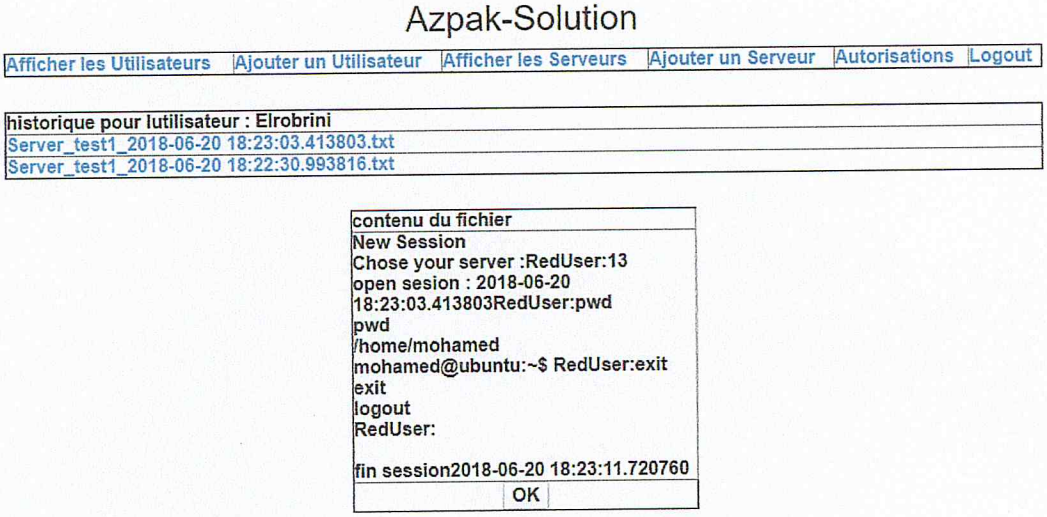

Figure 24: affichage de contenu de l'un des fichiers textes

La figure suivante montre le cas où l'administrateur veux ajouter un nouvel utilisateur :

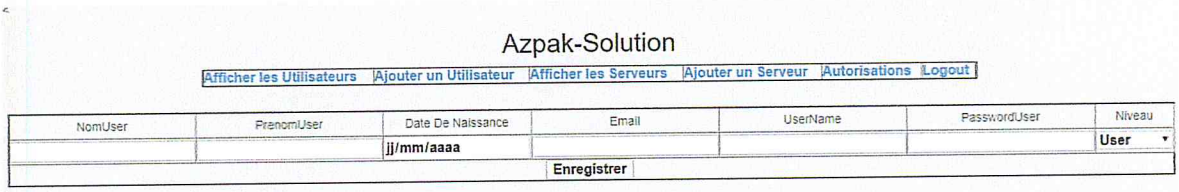

Figure 25: ajout d'un nouvel utilisateur

Remarque : l'édition des informations des serveurs ainsi que l'ajout d'un nouveau serveur se font de la même manière que l'édition des informations des utilisateurs.

Concernant l'établissement des autorisations, l'administrateur en cliquant sur l'onglet 'Autorisations' il va se retrouver dans la page que nous montrons dans la figure suivante :

Figure 26: affichage de l'onglet des autorisations

A partir de là l'administrateur aura le choix d'afficher les autorisations existantes, ajouter des nouvelles autorisations ou bien ajouter une 'black-list' pour une autorisation existante. Les trois (3) cas sont montrés dans les figures suivantes :

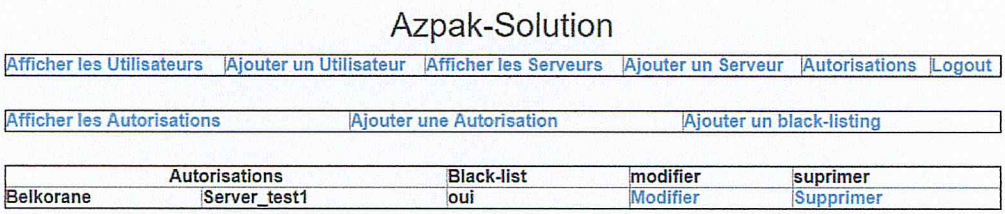

Figure 27: affichage du détails de l'autorisation

Remarque : la modification et la suppression se fait a travers l'affichage de la liste des autorisations.

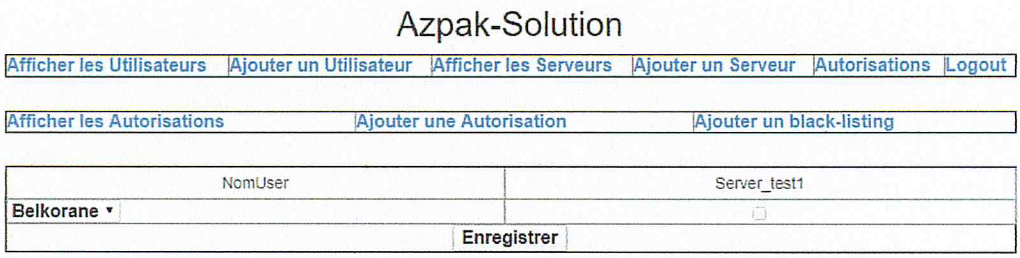

Figure 28: interface d'ajout d'une nouvelle autorisation

Figure 29: affichage de l'ongler du 'black-listing'

Remarque : à partir de cette onglet l'administrateur peut consulter la liste des commandes interdites pour cette autorisation, modifier une commande interdite existante ou la supprimer.

3.1.2_Utilisation de l'interface par un auditeur :

Tout comme l'administrateur, l'auditeur doit s'authentifier à travers son nom d'utilisateur et son mot de passe. Il se retrouvera par la suite dans une page qui va lui permettre uniquement de consulter la liste des utilisateurs (employés), la liste des serveurs, la liste des autorisations (avec leurs 'black-list' s'ils existent). Nous montrons dans les figures suivantes la page dédier pour l'auditeur avec ses fonctionnalités.

Figure 30: page d'accueil pour l'auditeur

Figure 31: l'affichage de la liste des utilisateurs dédier à l'auditeur

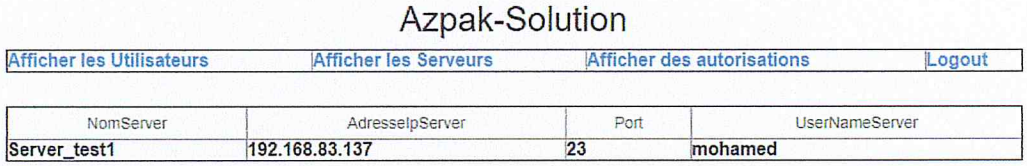

Figure 32: l'affichage de la liste des serveurs dédier à l'auditeur

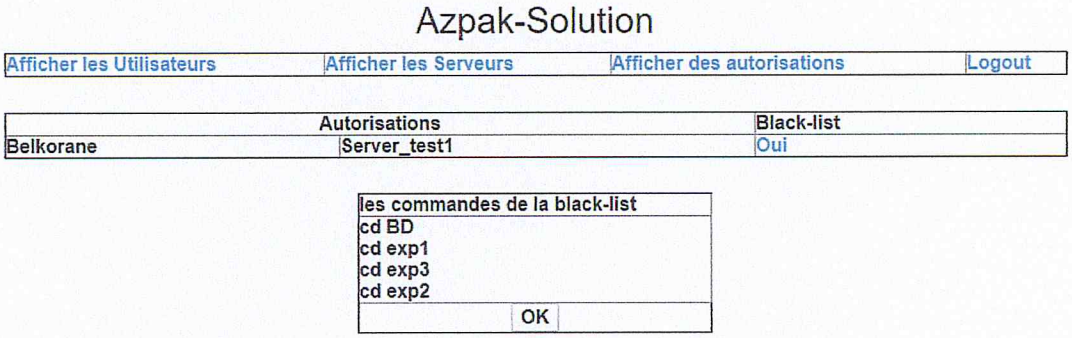

Figure 33: affichage des autorisations avec leurs 'black-list' dédier à l'auditeur

3.2 Réalisation de la partie cœur :

Cette partie qui représente l'implémentation de notre proxy sera divisé en deux sous-parties implémenter dans un même fichier avec 'Python' comme langage de programmation.
La première va représenter la partie serveur de notre proxy, c'est elle qui va activer la connexion entre l'employé et le proxy, et la deuxième sera la partie client de notre proxy et c'est elle qui se chargera d'activer la connexion entre le proxy et le serveur finale (celui que le client a choisi d'y accéder)

3.2.1 la sous-partie serveur :

Cette partie permet de vérifier l'authentification des employés pour les laisser accéder au proxy ou non. Comme notre proxy est de type 'SSH' ceci implique l'utilisation de la cryptographique asymétrique (la notion de clé publique et clé privée). Pour cela on utilise l'instruction suivante pour générer une clé publique ensuite la distribuer à chaque fois que le proxy reçoit une demande de connexion de la part d'un client de type 'SSH'

```
data = (b'AAAAB3NzaC1yc2EAAAABIwAAAIEAyO4it3fHlmGZWJaGrfeHOVY7RWO3P9M7hp'
        b'fAu7jJ2d7eothvfeuoRFtJwhUmZDluRdFyhFY/hFAh76PJKGAuslqIQKlkJxMC'
        b'KDqIexkgHAfID/6mqvmnSJfQb5W8v5h2pI/stOSwTQ+pxVhwJ9ctYDhRS1F0iT'
        b'UWT10hcuO4Ks8=')
good_pub_key = paramiko.RSAKey(data=decodebytes(data))
```

Figure 34 : Déclaration de la clé publique

Rappelons que le proxy envoie sa clé publique aux clients pour qu'ils l'utilisent pour chiffrer leurs informations ensuite les renvoyer dans un état chiffré au proxy, ce dernier va utiliser sa clé privée pour déchiffrer ces informations et les avoir en clair pour pouvoir les comparer avec les informations stockés dans la base de données afin de voir s'il y a des employés qui porte ces informations ou non (nom d'utilisateur et mot de passe).
Concernant la clé privée on utilise l'une des clés publiée sur internet, et donc on enregistre cette clé dans un fichier nommé 'test_rsa.key'.

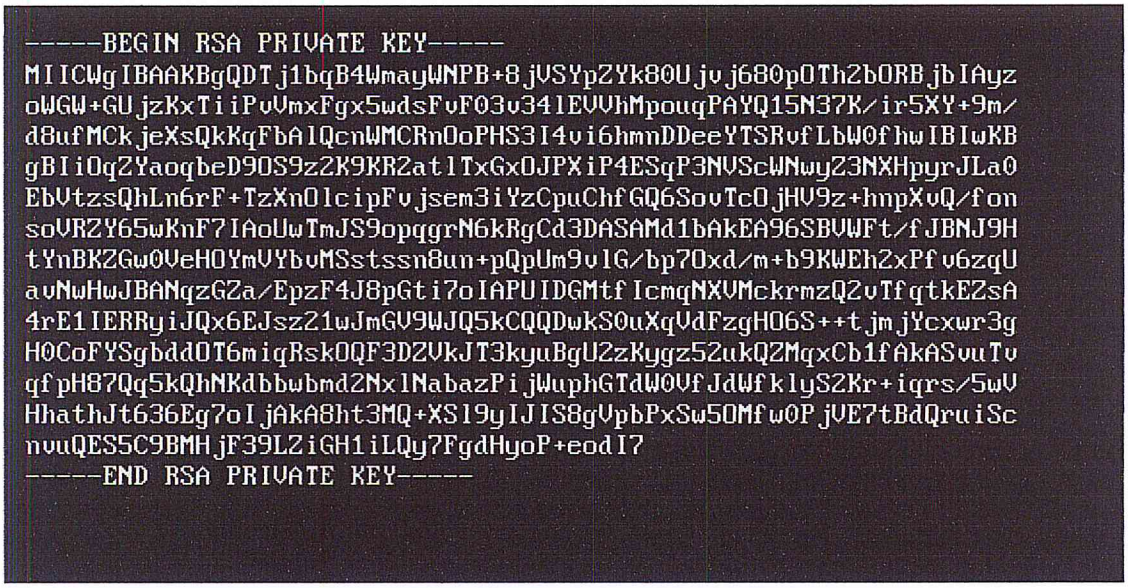

Figure 35 : la clé privée

Et pour l'utilisation de cette dernière on utilise l'instruction suivante :

```
host_key = paramiko.RSAKey(filename='test_rsa.key')
```

Figure 36 : Appel à la clé privée

On passe maintenant à l'explication de la manière dont l'authentification se fait, pour qu'on peut distinguer si l'employé a été bien authentifié ou non le package 'paramiko' fournit une fonction qu'on doit utiliser et le résultat retourné à partir de cette dernière va être un paramètre qu'on doit utiliser dans une autre fonction qui se chargera d'activer la connexion ou non selon ce paramètre :

```
def check_auth_password(self, username, password):
    if (verifait_user(username,password)==1):
        return paramiko.AUTH_SUCCESSFUL
    return paramiko.AUTH_FAILED
```

Figure 37 : fonction de l'authentification

On remarque que cette fonction dépend du résultat d'une autre fonction qu'on a programmés nous-même et c'est elle qui va se charger d'établir une connexion avec la base de données et

faire la vérification nécessaire :

```
def verifait_user(username,password):
        Mot=hashlib.sha256(password).hexdigest()
        conn = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
        cursor = conn.cursor()
        cursor.execute("""SELECT COUNT(*) , IdUser,PasswordUser  FROM users WHERE UserNameUser = %s$
        global belko
        global iduser
        belko=username
        rows = cursor.fetchall()
        for row in rows:
                conn.close()
        pas='{0}'.format(row[2])
        iduser = '{0}'.format(row[1])
        if(pas==Mot) :
          return 1
        return 0
```

Figure 38 : fonction de vérification des données

Dans une première instruction on applique l'algorithme de hachage 'SHA256' (le même algorithme qu'on a utilisé pour hacher les mots de passes des employés dans la base de données) sur le mot de passe que l'employé a saisie pour pouvoir le comparer avec les mots de passe enregistrés dans la base de données.

Ensuite, on utilise la fonction fournit par 'python' qui nous permet d'établir une connexion avec une base de données qui est 'mysql.connector.connect' qui prend le nom du hôte, le nom de l'utilisateur, le mot de passe de la base de données et le nom de la base de donnée comme paramètres.

Une fois ceci terminé, on fait la vérification et on retourne 1 dans le cas où il existe un employé qui porte ces informations, sinon on retourne 0 dans le cas contraire.

Dans la figure suivante on montre un exemple d'un employé qui utilise 'Putty' qui est un client SSH pour connecter a notre proxy et dans cette étape il est en train d'entrer les informations nécessaire pour se connecter à notre proxy (l'adresse IP du proxy et son port)

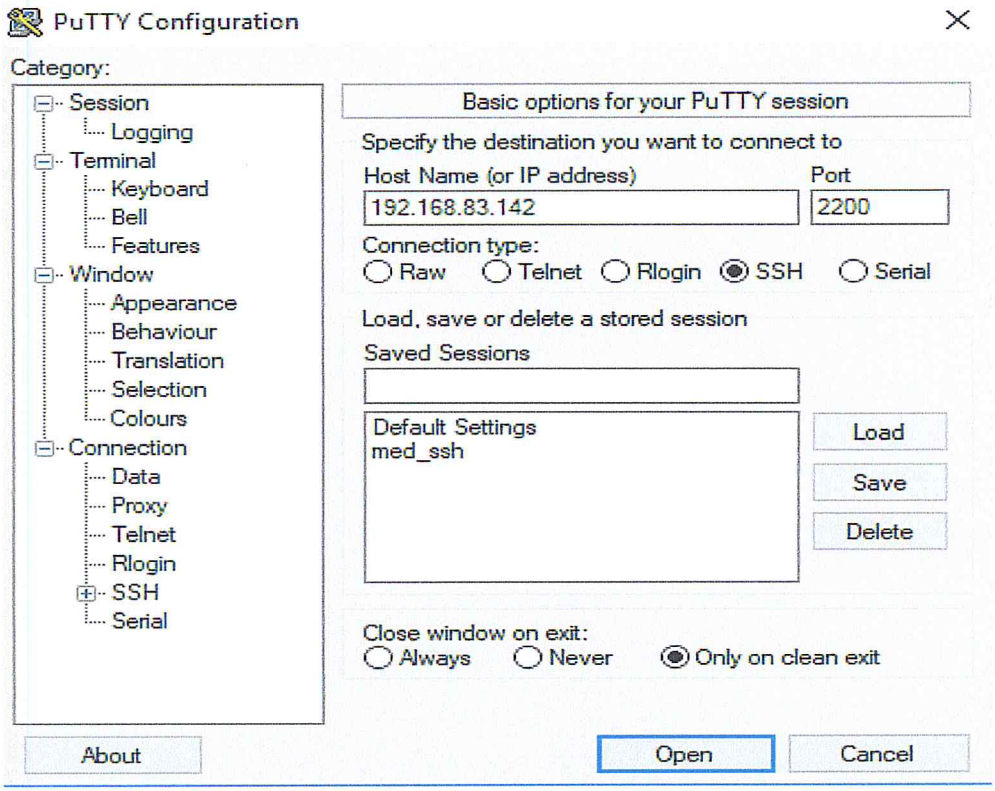

Figure 39 : interface de Putty (client SSH)

L'outil utilisé par l'employé dans cet exemple qui est 'Putty', établit une connexion à travers l'adresse IP de notre proxy et le port, l'employé va se retrouver dans une console où il sera obligé d'introduire son nom d'utilisateur ainsi que son mot de passe comme la figure suivante montre :

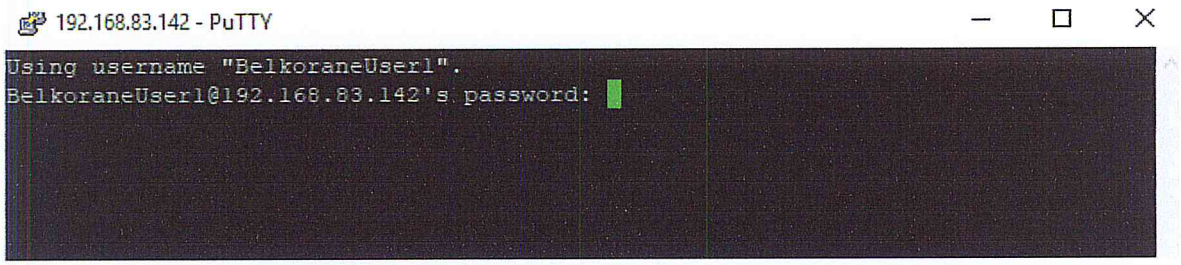

Figure 40 : terminale du client ssh (login)

Une fois l'authentification des informations de l'employé a été faite on passe à l'authentification de la clé publique comme nous montrons dans cette capture :

```
def check_auth_publickey(self, username, key):
    print('Auth attempt with key: ' + u(hexlify(key.get_fingerprint())))
    if (verifait_username(username)==1) and (key == self.good_pub_key):
            return paramiko.AUTH_SUCCESSFUL
    return paramiko.AUTH_FAILED
```

Figure 41 : fonction d'authentification des clés

Si les deux vérifications de l'authentification ont retourné un résultat positif, on ouvre une session entre le proxy et l'employé (on active la connexion entre eux) en utilisant cette fonction :

```
def check_channel_request(self, kind, chanid):
    if kind == 'session':
        return paramiko.OPEN_SUCCEEDED
    return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
```

Figure 42 : fonction d'ouverture de session

Ce que nous venons d'expliquer est le scénario normal d'une connexion entre un employé de la société et notre proxy sachant que parmi les buts pour lesquels on a développé ce proxy on a la notion de la centralisation qui veux dire que tous les employés doivent passer par le proxy ce qui implique que leurs demandes de connexion peuvent se faire en même temps.
Pour remédier à ce problème on a appliqué le principe du 'multi-threading' de cette façon : le proxy dans son état active va attendre les demandes des connexions (on appel sa écouter les connexions), dès qu'il reçoit une demande de connexion il initialise un 'thread' qui va s'occupé de cette demande (assurer l'authentification de cette employé et faire le traitement nécessaire selon les demandes de l'employé) alors que le proxy continu son exécution (écouter des tentatives de connexion des autres employés) et s'il reçoit une autre demande il initialise un autre 'thread' et c'est ainsi que notre proxy fonctionne.
L'initialisation du thread se fait a travers la fonction suivante :

```
def __init__(self,host,port):
    self.host=host
    self.port=port
    self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self. sock.bind((self.host,self.port))
```

Figure 43 : fonction d'initilisation de thread

Et voici un exemple qui montre l'exécution de plusieurs threads en même temps :

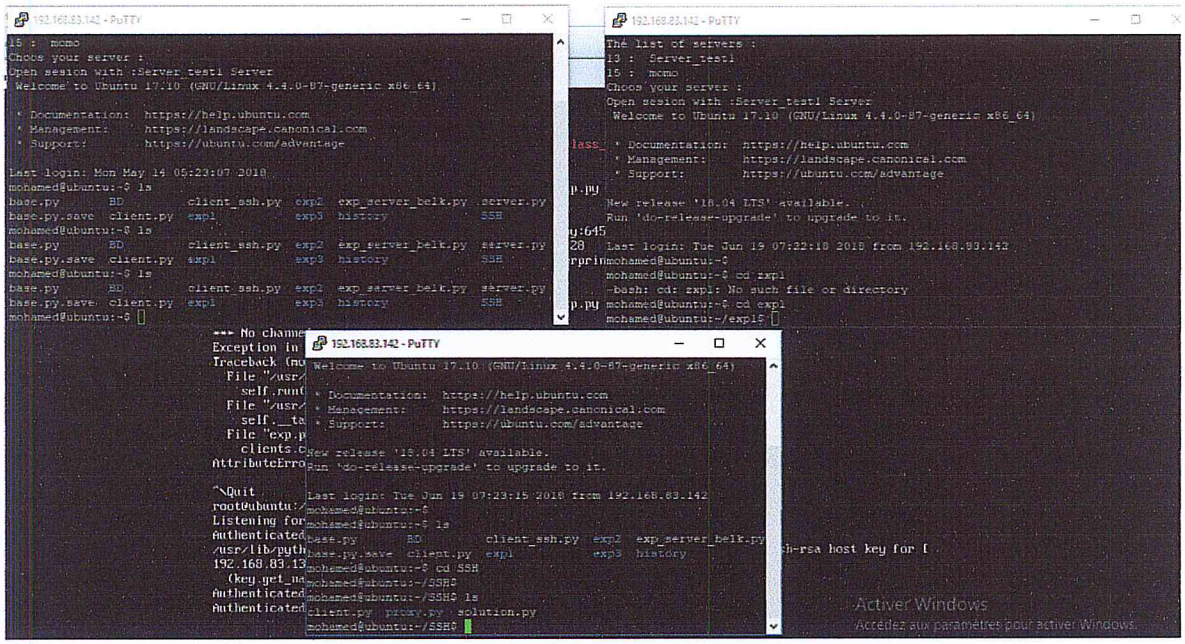

Figure 44 : exécution de plusieurs clients en même temps

Et la fonction qui va permettre au proxy d'écouter les demandes de connexion est la suivante :

```
def listen(self):
    self.sock.listen(100)
    print('Listening for connection ...')
    while True:
        clients, address = self.sock.accept()
        clients.settimeout(60)
        threading.Thread(target = self.listenToClient,args = (clients,address)).start()
```

Figure 45 : fonction d'activation de serveur proxy

Remarque : cette fonction dépend d'une autre fonction 'listenToClient' que le thread doit exécuter.

Dans le cas où le proxy reçoit une demande de connexion de la part d'un employé, il va initialiser un thread qui va s'occuper de cet employé, le thread va exécuter les fonctions précédentes pour assurer l'authentification de l'employé ensuite il exécute la fonction

suivante :

```
def listenToClient(self,clients,address):
 while True:
  try:
   try:
     self.t = paramiko.Transport(clients, gss_kex=DoGSSAPIKeyExchange)
     self.t.set_gss_host(socket.getfqdn(""))
     Crypto.Cipher.AES.new = fixed_AES_new
```

Figure 46 : fonction d'activation de tunnel ssh

Pour récupérer les informations nécessaires pour établir une session (adresse,port…) d'une façon sécurisé, par le suite le 'thread' va exécuter la fonction suivante qui a le but de vérifier si les informations récupérées de la part de l'employé sont correctes ou non.

```
clients = self.t.accept(20)

if clients is None:
    print('*** No channel.')
    sys.exit(1)
print('Authenticated!')
server.event.wait(10)
```

Figure 47: vérification des information de client

Une fois la session est initialisé le proxy (la première sous-partie) va récupère l'ID de l'employé ensuite consulter la base de donnée pour voir les serveurs que cet employé peut y'accéder, afficher la liste des serveurs a travers la fonction suivante :

```
f = clients.makefile('rU')
clients.send('Welcome to Azpak Solution  :     ')
clients.send(verifait_user_name(belko))
clients.send(' ')
clients.send(verifait_name(belko))
clients.send('\r\n')
clients.send('The list of servers :')
list_server = []
idid=verifait_user_id(belko)
conn = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
cursor = conn.cursor()
cursor.execute("""SELECT  Server_idServer FROM authorizations WHERE Users_IdUser = %s  """, ($
rows = cursor.fetchall()
for row in rows:
        clients.send('\r\n')
        clients.send('{0}'.format(row[0]))
        list_server.apend('{0}'.format(row[0]))
        k=int('{0}'.format(row[0]))
        clients.send('  :  ')
        clients.send(nom_server(k))
        conn.close()
```

Figure 48 : fonction qui affiche au client la liste des serveurs

Et pour récupérer le choix de l'employé on exécute cette fonction :

```
clients.send('\r\n')
clients.send('Chose your server : ')
id_server=int(f.readline())
if id_server not in list_server:
            sys.exit(1)
```

Figure 49 : fonction de choix de serveur

Et voici la façon dont la liste des serveurs sera affichée aux employés pour qu'ils fassent leurs choix :

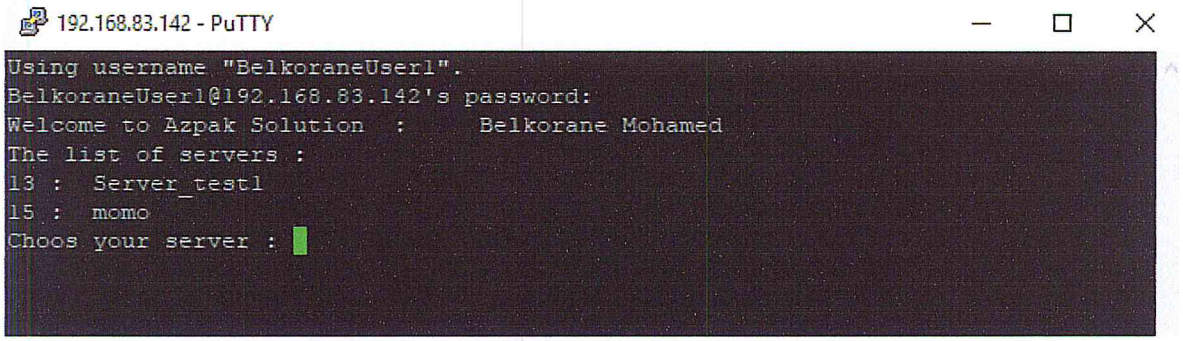

Figure 50 : terminale du client après l'authentification

Et par cette fonction on termine la sous-partie serveur de notre proxy car cette sous-partie est seulement responsable d'assurer qu'il s'agit des employés légitimes ensuite établir une connexion entre eux et notre proxy et récupérer le choix de l'employé.

3.2.2 la sous-partie client :

Cette sous partie va jouer le rôle d'établir la connexion entre notre proxy et le serveur que l'employé a choisi, pour cela elle ouvre une session entre le proxy et le serveur en suivant les mêmes étapes que la première sous-partie.

Afin d'établir la connexion avec le serveur on doit utiliser l'ID du serveur que nous avons récupérer directement à partir du choix de l'employé dans une requête 'SQL' pour récupérer tous les informations concernant ce serveurs (adresse IP, port, nom d'utilisateur…), ensuite les enregistrer dans des variables qui vont être des paramètres dans les fonctions qui suit.

On montre dans les deux captures suivantes le déroulement de cette procédure.

```
def server_information(srv):
        conn = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
        cursor = conn.cursor()
        cursor.execute("""SELECT  UserNameServer,UserNameServer ,PasswordServer FROM credentials WH$
        rows = cursor.fetchall()
        for row in rows:
                  conn.close()
        global nom_serveur
        nom_serveur= '{0}'.format(row[1])
        nbr='{0}'.format(row[0])
        MotPasse2='{0}'.format(row[2])
        MotPasse=decriptage(MotPasse2)
        global password_server
        password_server=MotPasse
```

Figure 51 : fonction de récupération des informations de serveur '1'

```
def server_info(srv):
        conn = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
        cursor = conn.cursor()
        cursor.execute("""SELECT  IdServer,NomServer,Ip,Port FROM server WHERE IdServer = %s  """, $
        rows = cursor.fetchall()
        for row in rows:
                  conn.close()
        nbr2=int('{0}'.format(row[0]))
        nbr3='{0}'.format(row[2])
        nbr4=int('{0}'.format(row[3]))
        MotPasse2='{0}'.format(row[1])
        global port
        global name_server
        global ip
        name_server=MotPasse2
        port= nbr4
        ip = nbr3
```

Figure 52 : fonction de récupération des informations de serveur '2'

Remarque : on doit exécuter deux requêtes pour récupérer toutes les informations car une partie de ces informations est enregistrée dans une table et l'autre partie est enregistrée dans une autre table.

Remarque 2 : le mot de passe du serveur est enregistré dans la base de données dans un état crypté où on a utilisé l'algorithme que nous avons programmé pour le crypter. Pour pouvoir l'utiliser pour s'authentifier au serveur on doit le décrypter en utilisant la fonction suivante :

```
def decriptage ( code ):
        liste = []
        list = []
        for car in code :
                  liste.append(ord(car))
        for ab in liste:
                  list.append(chr(ab)+5)
        b="".join(list)
        return b
```

Figure 53 : fonction de décryptage du mot de passe du serveur

Une fois les informations du serveur sont récupérées, on initialise une session entre le proxy et le serveur a travers ces instructions :

```
client = paramiko.SSHClient()
client.load_system_host_keys()
client.set_missing_host_key_policy(paramiko.WarningPolicy())
client.connect(hostname=ip, port=port, username=nom_serveur, password=password_server)
self.shell = client.invoke_shell()
```

Figure 54 : activation de la connexion ssh au serveur

Remarque : afin de garder la traçabilité des employés on doit exécuter cette fonction

```
date_open=datetime.now()
```

Pour récupérer la date d'ouverture de la session afin de l'enregistrer dans le fichier qui contient l'historique de cet employé.

La figure suivante nous montre le début d'une session pour l'employé :

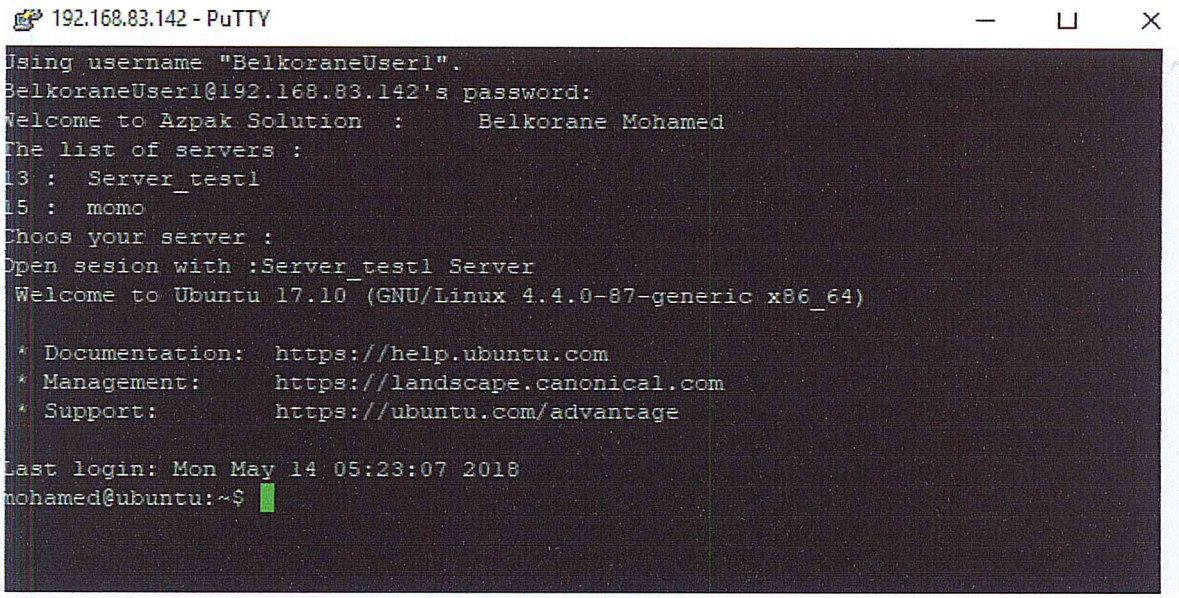

Figure 55 : terminale de client après l'ouverture de session avec le serveur

Avant de récupérer les commandes de l'employé pour les exécuter dans le serveur on utilise l'instruction suivante pour crée un fichier qui va contenir l'historique de l'employé en question :

```
if not os.path.exists("/var/www/html/Historique/"+belko):
    os.mkdir("/var/www/html/Historique/"+belko)
```

Figure 56 : ouverture du fichier

Une fois le fichier est créé on récupère la commande de la part de l'employé, on l'enregistre d'abord dans le fichier ensuite on vérifie si cette commande n'est pas interdite pour ce client avant de l'exécuter dans le serveur, on récupère le résultat de la commande de la part du

serveur, on l'enregistre dans le fichier ensuite on l'envoie à employer et c'est ainsi que notre proxy travaille.

la vérification des commandes interdites se fait de cette manière :

```
username =str(f.readline().strip())
self.shell.send(username)
time.sleep(0.01)
aaaa=self.shell.recv(2024)
clients.send(aaaa)
username =str(f.readline())
while username != 'exit':
    r=verifait_cmd(id_auto,username)
    if  r==1  :
          fichier.write(belko+":"+username)
          fichier.write("utilise une commande interdit ")
          clients.send("You can't use this commande")
          username =str(f.readline())
    else:
          fichier.write(belko+":"+username)
          self.shell.send(username)
          time.sleep(0.1)
          aaa=self.shell.recv(2024)
          clients.send(aaa)
          fichier.write(aaa)
          username =str(f.readline())
```

Figure 57 : fonction de vérification des commandes de client

avec une condition qu'on impose qui sera : 'tant que l'employé n'envoie pas « exit » au proxy continue l'exécution du programme', ce qui implique que si l'employé veut se déconnecter du proxy il n'aura qu'a envoyé la commande « exit », la fonction qui est imbriqué dans cette fonction est la suivante :

```
def verifait_cmd(id_auto,cmd):
      liste = []
      conns = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
      cursors = conns.cursor()
      cursors.execute("""SELECT  Commande FROM blacklist WHERE Authorizations_IdAuthorizations =%s $
      rows = cursors.fetchall()
      for row in rows:
               liste.append('{0}'.format(row[0]))
               conns.close()
      if cmd in liste :
        return 1
      else :
        return 0
```

Figure 58 : fonction de vérification des commandes (interdites ou non)

La figure suivante nous montre un cas d'une utilisation ordinaire par un employé de la société :

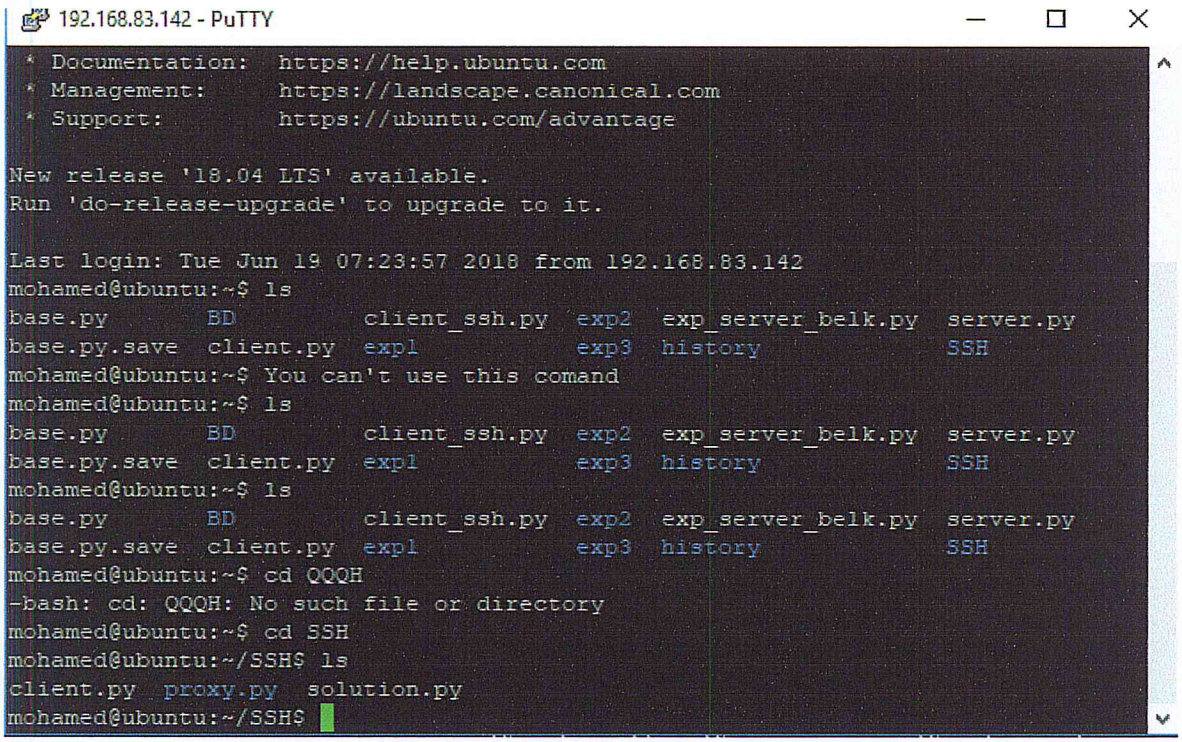

Figure 59 : terminale de client avec des commandes non interdites

Et dans la figure suivante on montre le cas où l'employé essaie d'exécuter une commande interdite :

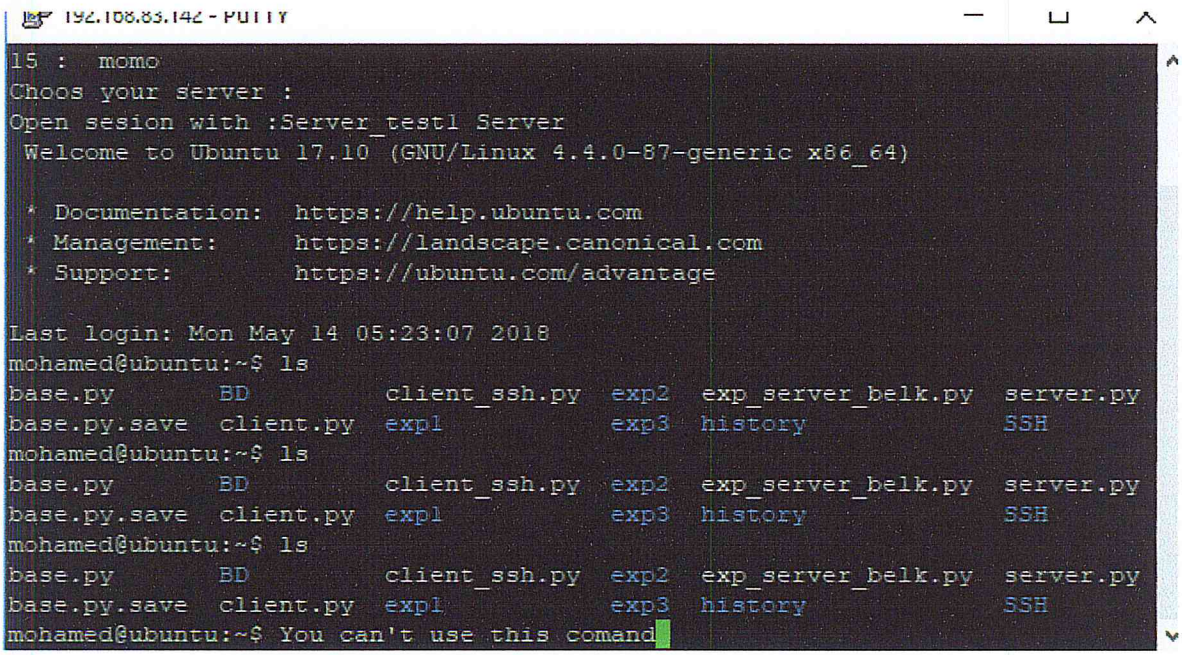

Figure 60 : terminal client avec une command interdites

Et voici la façon dont l'employé se déconnecte de notre proxy en exécutant le commande exit :

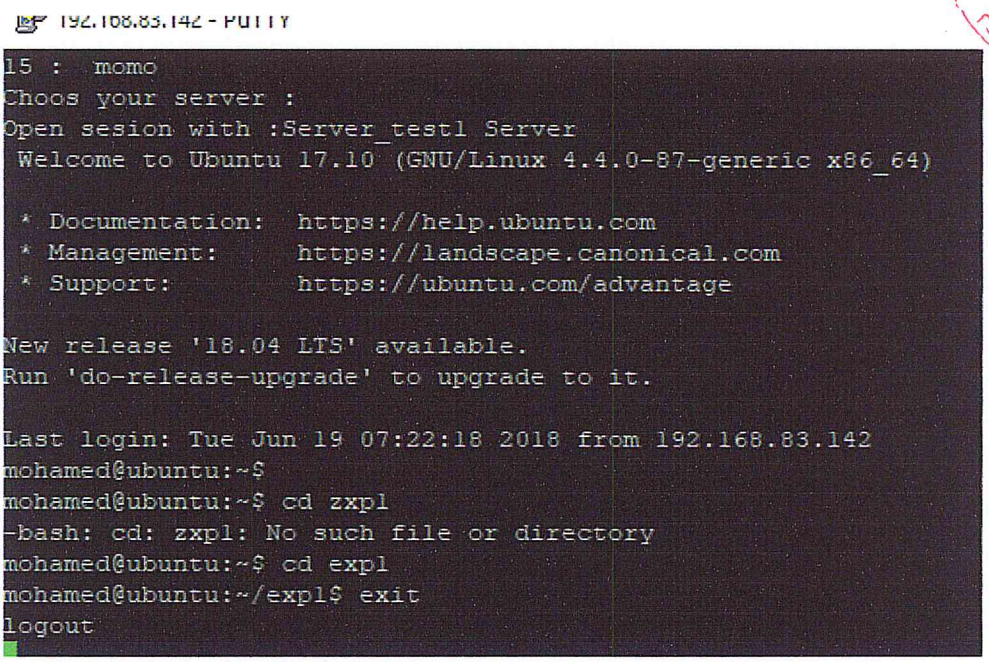

Figure 61 : terminale cliente en fin de session

Remarque : après la déconnection du l'employé on exécute l'instruction suivante `date_fin= datetime.now()` pour récupérer la date de fin de session entre l'employé et le serveur qu'il a choisi afin de l'enregistrer dans le fichier qui contient l'historique de cet employé.

Ensuite nous allons enregistrer les informations obtenus pour cette session qui seront (l'ID de la session, l'ID de l'employé, l'ID du serveur, la date de début pour cette session, la date de fin de session, et le chemin vers le fichier qui contient l'historique) dans la table 'session' pour assurer la traçabilité en exécutant cette fonction :

```
def fct_sesion(id_server,id_user,date_debut,date_fin,path):
        conn = mysql.connector.connect(host="localhost",user="root",password='',database="bd")
        cursor = conn.cursor()
        try:
                cursor.execute(""" insert into session (id_session,Users_IdUser,Server_idServer,Sta$
                conn.commit()
        except pymysql.InternalError as e:
            code, message = e.args
            print">>>>>>", code, message
            print e.errno
```

Figure 62 : fonction de la création de la session

4.Conclusion :

Dans ce chapitre, nous avons énumérer les outils et les langages utilisés dans la réalisation de notre solution que nous avons nommée 'Azpak Solution'. En effet nous avons achevé une implémentation d'une solution qui répond parfaitement à la problématique proposée et on a même ajouté le concept de la 'black-list'.

En d'autres termes, nous détenons maintenant une version acceptable du logiciel, installée dans notre environnement de développement.

Conclusion et Perspectives :

Les entreprises et leur employé ont toujours besoin d'accéder à des réseaux internes et externe afin d'exécuter leurs fonctions. Ces transactions ne sont pas sécurisées et si elles ne sont pas contrôlées, ils vont mettre le system d'information de la société dans une zone de risque.

Le proxy SSH est la meilleure solution adaptative pour remédier à ce problème afin de pouvoir centraliser les accès des employés (tous les accès passent par le proxy d'abord) et de surveiller ces accès qui seront des accès privilèges (établir un contrôleur d'accès au niveau de proxy), tout en sécurisant les liaisons entre l'employé et le proxy ainsi qu'entre le proxy et le serveur.

Pour ces raisons, nous avons choisi de mettre en place d'un proxy SSH pour la centralisation et la surveillance des accès à privilèges afin d'améliorer la sécurité au sein des sociétés et donné à l'administrateur de réseau un outil de surveillance pour surveiller les employés et aussi d'établir des droits d'accès pour ces employés.

La solution proposé dans le cadre de ce projet et de réaliser un proxy SSH qui vise à atteindre les objectifs suivants :

_établir des accès à privilèges à travers un contrôleur d'accès.

_centraliser et surveiller ces accès.

_sécuriser la liaison entre l'employé et le serveur à travers le protocole SSH (entre l'employé et proxy ainsi qu'entre le proxy et le serveur).

Nous avons même ajouté une option en plus qui permet à l'administrateur d'ajouter une liste noire qui contient des commandes interdites à exécuter dans un serveur particulier par un employé particulier.

Nous pouvons envisager plusieurs plans de future pour notre solution comme ajouter une partie RDP (Remote Desktop Protocole) qui permet à l'administrateur de contrôler la machine de l'employé.

On peut aussi ajouter une option qui permet à l'administrateur de voir la transaction entre l'employé et le serveur au temps réel et non pas a travers un fichier texte qui contient un historique enregistrer.

Bibliographie

- [1] « http://gdt.oqlf.gouv.qc.ca/ficheOqlf.aspx?Id_Fiche=8359717 » ,Pour la définition de proxy ,Consulté le 20-05-2018
- [2] «http://cookieconnecte.fr/2016/09/12/proxy-reverse-proxy-comprendre-lessentiel-5-minutes/ » , Pour le proxy réseau et son utilité et ses fonctionnalités , Consulté le 21-05-2018.
- [3] « https://www.commentcamarche.com/contents/214-ssh-protocole-secure-shell»,Pour la définition de SSH , consulté le 21-05-2018.
- [4] « https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/la-connexion-securisee-a-distance-avec-ssh », Pour les étape de la création d'un canal sécurisé avec SSH , consulté le 21-05-2018.
- [5] « https://ram-0000.developpez.com/tutoriels/cryptographie/ », Pour la cryptographie, consultée le 23-05-2018.
- [6] B. Beckett, introduction aux méthodes de la cryptologie, 1990.
- [7] F. BENHAMOU et I. DERRAR Etude comparative entre les algorithmes cryptographiques appliqués sur des fichiers textes et des fichiers images (BMP) basé sur une simulation d'une attaque exhaustive, 2006.
- [8] « http://www.labouret.net/crypto/#1 », Pour le Principe de l'algorithme symétrique, consulté le 23-05-2018.
- [9] « lasignature.com/algorithme/symetrique.htm », Pour l'Algorithmes asymétriques, consulté le 24-05-2018.
- [10] G. Zennor, Cours de cryptographie, 2000
- [11] « https://www.commentcamarche.com/contents/212-signature-electronique », Pour le hachage, consulté le 22-05-2018.
- [12] « https://www.vmware.com/products/workstation-pro.fr.html », pour la définition du Vmware workstation pro, consulter le 25-05-2018.
- [13] « https://www.techopedia.com/definition/4207/ubuntu-server », pour la définition du Ubuntu server, consulter le 25-05-2018.