Avant-Propos

Les travaux présentés dans ce mémoire sont effectués au Laboratoire de Recherche en Informatique et Télécommunication (LRIT), à la Faculté des Sciences de Rabat (FSR), sous la direction du Professeur Mohamed EL MARRAKI et le co-encadrement du Professeur Ali KARTIT.

Je tiens à exprimer ma gratitude à mon directeur de thèse, M. Mohamed EL MARRAKI, professeur de l'enseignement supérieur à la Faculté des sciences de Rabat (FSR). Je le remercie pour l'opportunité qu'il m'a offerte en m'accueillant dans son équipe de recherche. Merci d'avoir cru en moi et m'avoir montré les portes qu'il fallait ouvrir pour que ce travail soit à la hauteur.

Je tiens à remercier également mon co-encadrant, M. Ali KARTIT, professeur Habilité à L'École Nationale des Sciences Appliquées d'El Jadida (ENSAJ). Je le remercie pour sa disponibilité, ses encouragements et ses conseils pertinents.

Je remercie chaleureusement M. Ahmed Hammouch, professeur de l'enseignement supérieur à l'École Normale Supérieure de l'Enseignement Technique de Rabat (ENSET), d'avoir présidé ma soutenance.

Je remercie également M. Chafik NACIR, professeur de l'enseignement supérieur à L'École Normale Supérieure de l'Enseignement Technique de Rabat (ENSET), d'avoir accepté de rapporter ma thèse. Merci également pour le temps que vous avez consacré à lire ce mémoire et à évaluer ce travail.

Je remercie vivement M. Youssef FAKHRI, Professeur Habilité à la Faculté des Sciences de Kénitra (FSK), d'avoir accepté de rapporter ce travail. Je le remercie pour le temps qu'il a consacré à lire mon mémoire et évaluer mon travail. Une sincère gratitude pour l'être méthodique que vous êtes.

Un grand merci à Mme Dounia LOTFI, professeur assistant à la Faculté des sciences de Rabat (FSR), d'avoir fait partie du jury en tant qu'examinateur.

Je remercie aussi ma famille pour leurs encouragements, leur soutien moral, et d'avoir supporté mon absence, entre autres, durant la préparation de cette thèse.

Je remercie également mes amis de leurs soutiens et encouragements et toutes les personnes qui m'ont aidé de près ou de loin pour achever ce travail.

Résumé

Le Cloud Computing est l'une des tendances nouvelles et très prometteuses dans le domaine de la technologie informatique, elle fait l'objet de nombreuses recherches et des progrès continus sont réalisés. Ses principaux objectifs sont de fournir différents services aux utilisateurs, tels que l'infrastructure, la plate-forme ou le logiciel avec un coût raisonnable et de plus en plus décroissant pour les clients. Pour atteindre ces objectifs, certaines questions doivent être abordées, principalement l'utilisation des ressources disponibles de manière efficace afin d'améliorer la performance globale, tout en tenant compte de la sécurité et de la disponibilité du cloud. Par conséquent, parmi les aspects les plus étudiés par les chercheurs nous avons : l'équilibrage de charge, la détection d'intrusions, le stockage dans le cloud computing et les politiques de sécurité en particulier pour les grands systèmes de cloud distribués qui traitent de nombreux clients et de grandes quantités de données et de demandes.
Dans cette thèse, nous fournissons un algorithme de déploiement de politiques de sécurité de pare-feu type II qui surmonte les incohérences présentes dans l'algorithme "Greedy-2-Phase" omniprésent dans la littérature. Nous abordons aussi le sujet de l'équilibrage de charge et le stockage dans le cloud computing et nous présentons une architecture semi-centralisée et multi-cluster. Cette approche proposée garantit principalement une meilleure performance globale avec un équilibrage de charge efficace, une disponibilité continue et un aspect de sécurité mieux aiguisé. Nous abordons également, les inconvénients des IDS comportementaux et nous proposons un algorithme de clustering sous-jacent, qui peut traiter de manière optimale le clustering de données. Cet algorithme combine deux méthodes de clustering et d'optimisation, à savoir le K-means et Simulated Annealing, afin d'obtenir une classification globale optimale pour le profil concerné à apprendre et, par conséquent, d'éviter un blocage au niveau d'une solution optimale locale. Le K-Means dans ce travail est utilisé dans sa variante semi-supervisée afin de diminuer le nombre de fois que l'algorithme est appliqué et ainsi garder notre travail susceptible d'être utilisé dans un contexte en temps réel. L'algorithme développé a produit des résultats satisfaisants lorsqu'il est appliqué sur l'ensemble de données NSL-KDD, les tests révèlent que cette méthode peut améliorer les taux de détection et de détection erronée des systèmes de détection d'intrusion.

Mots-clés : Clustering, K-means, Gradient Descent, Global Optimum, Simulation Annealing, Anomaly based IDS, Security Policies, Cloud Storage, Load Balancing

Abstract

Cloud computing is one of the new and very promising trends in Computer technology, it is the subject of much research and continuous progress is being made. Its main objectives are to provide various services to end users, such as the infrastructure, the platform or the software with a reasonable cost.

To achieve these goals, some issues should be addressed, mainly the use of available resources in an efficient way in order to improve overall performance, while taking into account the security and availability of the cloud. Hence, among the aspects which are the most studied by researchers we have : Load Balancing, intrusion detection, Cloud Storage and Firewall policies in the cloud computing, especially for the wide and distributed cloud systems dealing with many customers and with high of data and applications.

In this thesis, we provide a type 2 security policy deployment algorithm that overcomes the failures collected in the ubiquitous algorithm 'Greedy-2-phase'. In addition, we discuss the topic of load balancing and Storage in the Cloud Computing, so we present a semi-centralized and multi-clustered architecture. This suggested approach mainly guarantees a better overall performance with an efficient load balancing, continuous availability and a safety aspect better honed. We are also approaching the Anomaly IDS drawbacks and we propose an underlying clustering algorithm, which can optimally handle the data clustering. This algorithm combines two methods of clustering and optimization, namely the K-means and Simulated Annealing, in order to obtain an optimal overall classification for the profile to be learned and, consequently, to avoid a blockage at a local optimal solution. K-Means in this work is used in its semi-supervised version to decrease the times number it is applied and thus keep our work useful in a real-time context. The developed algorithm has produced satisfactory results when applied to the NSL-KDD dataset, the tests prove that this method can improve the detection rate and false alarm in intrusion detection systems.

Keywords : Clustering, K-means, Gradient Descent, Global Optimum, Simulation Annealing, Anomaly based IDS, Security Policies, Cloud Storage, Load Balancing

Table des matières

Liste des acronymes

ACL	*Access Control List*
ADSL	*Asymmetric Digital Subscriber Line*
AES	*Advanced Encryption Standard*
ARP	*Address Resolution Protocol*
CBF	*Confidence Based Filtering*
CKMSA	*Combined K-Means and Simulated Annealing*
CPU	*Central Processing Unit*
CSA	*Cloud Security Alliance*
CSP	*Cloud Service Provider*
DaaS	*Data as a Service*
DaaS	*Desktop as a Service*
DAC	*Discretionary Access Control*
DDoS	*Distributed Deny of Service*
DDT	*Domain Definition Table*
DHCP	*Dynamic Host Configuration Protocol*
dIDS	*Distributed Intrusion Detection System*
DIT	*Domain Interaction Table*
DLT	*Divisible Load Theory*
DMZ	*Demilitarized Zone*
DNS	*Domain Name System*
DoS	*Deny of Service*
DTE	*Domain and Type Enforcement*
DTEL	*Domain and Type Enforcement Language*
DVFS	*Dynamic voltage and frequency scaling*
EC2	*Elastic Compute Cloud*
FAI	*Fournisseur d'Accès Internet*
FN	*False Negative*
FP	*False Positive*
FTP	*File Transfer Protocol*
GCE	*Google Compute Engine*
GNU	*Gnu is Not Unix*
HCF	*Hop-Count Filtering*
HPU	*Hewlett Packard Unix*
IaaS	*Infrastructure as a Service*

ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IMAP	*Interactive Message Access Protocol*
IT	*Information Technology*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
IPsec	*Internet Protocol Security*
ISO	*International of Standardization Organisation*
ITSEC	*Information Technology Security Evaluation Criteria*
KDD	*Knowledge Discovery Dataset*
KVM	*Kernel-based Virtual Machine*
LAN	*Local Area Network*
LLB	*Local Load Balancer*
MAC	*Mandatory Access Control*
MICS	*Multiplexed Information and Computing Service*
MLB	*Main Load Balancer*
MLS	*Multi-Level Security*
MTAM	*Monotonic Typed Access Matrix*
MV	*Machine Virtuelle*
NDP	*Non-deterministic Polynomial*
NIST	*National Institute of Standards and Technology*
PaaS	*Platform as a Service*
PKI	*Public Key Infrastructure*
QoS	*Quality of Service*
R2L	*Remote to Local*
RAM	*Random Access Memory*
RBAC	*Role-Based Access Control*
RFC	*Request For Comment*
RPH	*Random Port Hopping*
RSA	*Ronald Rivest, Adi Shamir et Leonard Adleman*
RTT	*Round Trip Time*
SA	*Simulated Annealing*
SaaS	*Software as a Service*
SECaaS	*Security as a Service*
SGBD	*Système de Gestion des Bases de Données*
SLA	*Service Level Agreements*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
SSPM	*Sandhu ?s Schematic Protection Model*
TAM	*Typed Access Matrix*
TCP	*Transmission Control Protocol*
TCSEC	*Trusted Computer System Evaluation Criteria*
TLS	*Transport Layer Security*
TN	*True Negative*

TP	*True Positive*
TTL	*Time To Live*
U2R	*User to Remote*
UDP	*User Datagram Protocol*
Vlan	*Virtual Local Area Network*
VPN	*Virtual Private Network*
VPS	*Virtual Private Server*
XAPI	*Xen Application Programming Interface*
XCP	*Xen Cloud Platform*
XSS	*Cross-Site Scripting*
WLC	*WorkLoad Control*
WSRF	*Web Services Resource Framework*

Liste des figures

Liste des tableaux

Introduction générale

Depuis l'invention du Web, le Cloud Computing s'impose et se présente comme la plus remarquable technologie IT en proposant des services depuis Internet à des pris très considérables. Son architecture robuste et bien flexible se proclame capable d'assurer les interactions, le stockage et le calcul dans les applications à venir. Cette architecture peut être utilisée pour les entreprises, l'informatique scientifique et plusieurs autres applications exigeantes en terme de ressources et de disponibilité.

La diversité des services fournis par l'infrastructure cloud augmente leur vulnérabilité aux incidents et attaques de sécurité. En fait, les principes de sécurité tels que la confidentialité, l'authentification et la disponibilité doivent être respectés lors de la conception et du déploiement d'une solution Cloud.

Les principaux objectifs du Cloud Computing sont de fournir différents services aux utilisateurs, tels que l'infrastructure, la plate-forme ou le logiciel. Pour atteindre ces objectifs, certaines questions doivent être abordées, principalement en utilisant les ressources disponibles de manière efficace afin d'améliorer la performance globale, tout en tenant compte de la sécurité et des côtés de disponibilité du nuage. Par conséquent, l'un des aspects les plus étudiés par les chercheurs est l'équilibrage de charge et le stockage dans le cloud computing, en particulier pour les grands systèmes de cloud distribués qui traitent de nombreux clients et de grandes quantités de données qui transitent en permanence en amont et en avale du cloud.

En plus, il est vrai que les fournisseurs de Cloud déploient les technologies de pointe, mais, chacune peut comporter des vulnérabilités qui peuvent nuire à ses services proposés. Chose qui nous ramène à la détection des intrusions et en particulier aux systèmes de détection qui se basent sur le comportement du système surveillé (Anomaly based intrusion detection system). Ces IDS sont réputés d'être très vulnérables aux alertes erronées et présentent en même temps un taux de détection très faible lorsque l'apprentissage est effectué sur des données mal classifiées. En outre, les pare-feus reconnus comme des périphériques de filtrage sont aussi vulnérables lors d'une phase critique de leurs configuration. En effet, le déploiement des politiques de sécurité dans ces derniers doit être sûr et correct. Ce qui est loin d'être assuré par les algorithmes existants.

La problématique

Comme tout système informatique très sollicité et accessible via Internet, le Cloud Computing doit assurer un taux de disponibilité parfait et d'être entouré d'une couche de sécurité hautement respectable pour garder ses clients.

Les quatre problématiques qu'on traite dans cette thèse sont la confidentialité des données à stocker dans le Cloud ainsi que leur disponibilité et la détection des intrusions qui menacent le Cloud en plus de la configuration des pare-feus à distance.

Objectifs de ce travail

A travers ce travail, nous essayons de déployer une architecture semi-centralisée et multi-cluster pour un équilibrage de charge efficient et un stockage confidentiel, ainsi qu'un IDS comportemental qui classifie les événements de sécurité pour surveiller le Cloud d'une manière très efficace. Nos apports également s'étendent pour fournir un algorithme sûr et rapide pour le déploiement des politiques de sécurité dans les pare-feus. Ces propositions répondent aux objectifs suivants :

1. Avoir une architecture hautement disponible grâce à l'équilibrage de charge ;
2. Avoir un système performant et capable d'offrir une meilleure tolérance aux pannes ;
3. Avoir un système qui peut être déployé dans différents environnements Cloud, en particulier les plus exigeants en termes de ressources ;
4. Avoir un outil de détection qui soit distribué et collaboratif ;
5. Avoir une architecture Cloud bien sécurisée.

La contribution brièvement

Parmi les points les plus sensibles de chaque Cloud [28][38] nous avons la disponibilité de ses services ainsi que la sécurité des données qu'il stocke. Dans ce travail, nous proposons deux solutions pour augmenter la disponibilité du Cloud et améliorer sa sécurité. La première solution consiste en une proposition d'une architecture semi-centralisée et multi-cluster. La deuxième prodigue une autre architecture confidentielle basée sur le chiffrement. Nos deux suivants apports se résument en deux algorithmes aspirant à fournir un déploiement et un profilage correct et sûr pour les pare-feus et les IDS respectivement.

Première contribution : Déploiement automatique des politiques de sécurité au sein des Firewalls

Dans ce travail [72], nous analysons l'algorithme "Greedy-two-phase Deployment" fourni en [64] et on démontre que celui-ci s'avère défaillant et non sûr. Nous prodiguons un algorithme amélioré, formalisé pour assurer un déploiement correct des politiques de sécurité des pare-feus. Notre contribution est éditée selon le langage d'édition type II largement utilisé dans les nouveaux Firewalls.

Deuxième contribution : Architecture Cloud basée sur l'équilibrage de charges

La deuxième contribution [36] consiste en une proposition d'une architecture d'équilibrage de charge sécurisée pour le Cloud Computing. Cette architecture se base sur l'utilisation de plusieurs clusters d'une manière hiérarchisée. Les composants de cette architecture sont :

- Main load balancer (MLB) : est le noeud coordinateur dans cette architecture. Il reçoit directement les requêtes à partir des clients, emploie un algorithme pour associer ces requêtes avec leurs origines et envoie par la suite la requête au cluster le plus adéquat pour son traitement.
- Resources clusters : est un ensemble de noeuds (ce qu'on désigne par ressources) qui contient un LLB (local load balancer). C'est ce dernier élément qui reçoit les requêtes dirigées par le MLB. Il a à choisir le noeud le plus approprié et libre pour le traitement de chaque requête reçue.
- Authentication element : permet d'avoir des flux de données sécurisés entre les deux éléments précédents.

Pour ne pas tomber dans un point de défaillance unique à cause du MLB, un algorithme est proposé pour le choix d'un nouveau MLB parmi les LLB qui se trouvent dans l'architecture.

Troisième contribution : Application des algorithmes de chiffrement pour sécuriser les données dans le Cloud Storage

Une application des algorithmes de chiffrement pour sécuriser les données dans le Cloud Computing a été proposé par [29]. Ce travail consiste en plusieurs contributions à la sécurité du Cloud Computing. On y propose une approche améliorée pour l'implémentation de la politique de sécurité des pare-feu permettant de sécuriser l'accès au Datacenter d'un Cloud. Par la suite, les algorithmes AES et RSA sont combinés pour chiffrer les données stockés dans le Cloud.

Quatrième contribution : Optimisation du profiling au sein d'un IDS comportemental à l'aide des algorithmes K-means et Simulated Annealing

La quatrième contribution [29] consiste à une combinaison de deux méthodes de classification pour optimiser la détection d'anomalies grâce aux IDS comportementaux. Les méthodes de classification sont l'algorithme K-means et l'algorithme Simulated Annealing. Le premier est utilisé pour classifier les événements de sécurité stockés dans la base de données pour les séparer selon leurs types. Les événements qui n'appartiennent à aucune classe existante seraient affectées à une nouvelle classe. Par la suite, le deuxième algorithme va nous permettre de créer des classes plus optimales afin de pouvoir éviter autant que possible les faux négatifs.

Parcours de la thèse

Une nouvelle approche pour la détection des intrusions

Les premiers pas dans le parcours de ma thèse étaient une documentation sur les travaux en sécurité informatique élaborés au sein de mon laboratoire d'accueil [55][54]. Ces travaux de recherche proposent une nouvelle approche de détection d'intrusion qui se base sur 3 niveaux de sécurité. Cette approche tente de sécuriser les réseaux informatiques de toute menace qu'elle soit externe ou interne. Ce premier travail a été concrétisé en un article de synthèse [35] ; Dans le premier niveau, un IDS réseau sera placé à l'entrée du système informatique. Cela permettra de détecter les attaques provenant depuis Internet (les attaques externes). Dans le deuxième niveau, une politique de sécurité est mise en oeuvre. Elle consiste en une activation des mécanismes de VLANs et d'ACLs. L'objectif de cette politique est la sécurisation du réseau informatique des attaques internes. Quant au troisième niveau, il consiste en une corrélation entre le contrôle d'accès physique avec le contrôle d'accès logique dans le but de, par exemple, interdire l'usurpation d'identité quand un employé n'est pas opérationnel. Le digramme 1 résume l'algorithme de la politique de sécurité à trois niveaux.

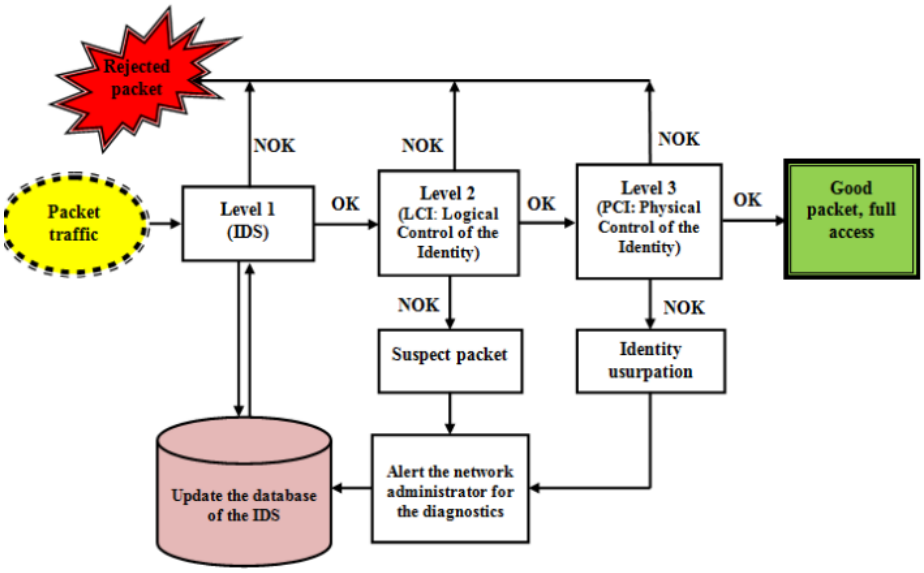

FIGURE 1 – Politique de sécurité à trois niveaux [35]

Une politique de déploiement correct et sûr pour les pare-feus

Les stratégies de pare-feu peuvent contenir plusieurs milliers de règles en raison de la grande taille et de la structure complexe réseaux. La taille et la complexité de ces politiques nécessitent des outils automatisés fournissant un outil convivial pour spécifier, configurer et déployer en toute sécurité une stratégie cible. Dans cet autre travail [33][34], ils ont démontré montrons que les approches naïves de déploiement peuvent facilement créer un trou de sécurité temporaire en permettant le trafic illégal et ainsi interrompre le service en rejetant le trafic légal pendant le déploiement. Quelques contributions à l'exactitude des politiques de sécurité de déploiements dans un pare-feu concrétisées par un algorithme formalisé par le langage type I ont été détaillées. L' algorithme peut être utilisé même pour le déploiement de politiques dont la taille est très importante.

Organisation du mémoire

De part les chapitres introduction générale et conclusion générale, ce mémoire se compose de quatre grands chapitres.

Dans le premier chapitre, nous présentons brièvement les concepts fondamentaux des éléments qui ont été fréquentés au cours de la thèse. Nous y présentons le Cloud, la sécurité informatique et l'équilibrage de charge.

Dans le deuxième chapitre, nous présentons le concept des politiques de sécurité avec comme exemples les politiques de contrôle d'accès. Nous décrivons ensuite notre algorithme de déploiement automatique de politiques de sécurité.

Dans le troisième chapitre, nous présentons notre deuxième contribution. Cette dernière qui consiste en une proposition d'une architecture de Cloud semi-centralisée et multi-cluster destinée au load balancing. Ce chapitre inclut également une autre structure aspirant à protéger la vie privée des utilisateurs du Cloud.

Dans le quatrième chapitre, nous présentons notre dernière contribution. Celle là consiste en une proposition d'un système de détection d'intrusion comportemental. Ce système se base sur deux méthodes de clustering et d'optimisation, à savoir, le K-means et Simulated Annealing.

Chapitre

1 La sécurité et le Cloud Computing

Sommaire

1.1 Introduction

Ce chapitre englobe des généralités qui concernent notre axe de travail. Nous évoquons la sécurité des réseaux informatiques puisque l'environnement Cloud que nous traitons est techniquement un réseau virtuel (interconnexion de plusieurs MVs). On présente brièvement les attaques réseau et mécanismes de défense associés. Nous introduisons également le Cloud Computing. On souligne ses avantages et ses inconvénients, on présente brièvement les solutions propriétaires et open source du Cloud et surtout, on présente la technologie qui a ajouté au Cloud de valeureuses caractéristiques qu'est la virtualisation. A la fin de ce chapitre, nous présentons le concept de la répartition de charges vu son rôle majeur dans notre deuxième contribution.

1.2 Présentation du Cloud

1.2.1 Les bases du Cloud

1.2.1.1 Qu'est-ce que le Cloud Computing

Le cloud computing est en train de devenir l'un des prochains mots à la mode de l'industrie informatique : les utilisateurs déplacent leurs données et applications vers le Cloud distant, puis y accèdent d'une manière simple et omniprésente. Ceci est à nouveau cas d'utilisation de traitement central. Un scénario similaire s'est produit il y a environ 50 ans : un serveur informatique à partage de temps desservait plusieurs utilisateurs. Jusqu'à il y a 20 ans, lorsque les ordinateurs personnels sont arrivés chez nous, les données et les programmes étaient pour la plupart situés dans des ressources locales. Certes, le paradigme Cloud computing n'est pas une récurrence de l'histoire. Il y a 50 ans, nous devions adopter les serveurs à partage de temps en raison des ressources informatiques limitées. De nos jours, l'informatique en nuage est en vogue en raison de la nécessité de construire des infrastructures informatiques complexes. Les utilisateurs doivent gérer diverses installations de logiciels, la configuration et les mises à jour. Les ressources informatiques et autres matériels sont susceptibles très bientôt d'être obsolètes. Par conséquent, l'externalisation des plates-formes informatiques est une solution intelligente permettant aux utilisateurs de gérer des infrastructures informatiques complexes. Au stade actuel, l'informatique en nuage évolue encore et il n'existe pas de définition largement acceptée. Sur la base de notre expérience, nous proposons une définition précoce du Cloud Computing : Un Cloud computing est un ensemble de services réseau, fournissant sur demande des infrastructures informatiques évolutives, QoS garanties, personnalisées et peu chères, accessibles d'une manière simple et omniprésente. [32][51].

1.2.1.2 Caractéristiques du Cloud

Bien que l'exploitation des ressources informatiques à distance n'est pas nouvelle, le Cloud se montre plus mature en apportant les caractéristiques suivantes :

- *Accès à la demande* : les ressources de calcul sont disponibles aux clients à n'importe quel moment via le réseau ;
- *Accès universel* : le Cloud assure la disponibilité de ses ressources malgré la diversité des terminaux et des réseaux d'accès que les clients peuvent utiliser pour accéder aux services Cloud ;
- *Mutualisation des ressources* : Le fournisseur du Cloud met à disposition ses ressources de calcul de façon mutualisée entre plusieurs clients. Un Cloud qui repose sur ce mécanisme il suit le modèle "multi-tenant" ;
- *Élasticité des ressources* : Le Cloud offre une adaptabilité rapide qui permet de bien accompagner les besoins de ses clients même au cas où ces besoins sont soudains ;
- *Facturation à l'usage* : Le client paye ne paye que la location des ressources de calcul qu'il réserve. Ceci est l'un des facteurs majeurs sur lesquels joue le fournisseur du Cloud puisqu'en général ce coût est remarquablement minime par rapport au déploiement d'un système informatique et sa maintenance.

1.2.1.3 Les offres de services

Le modèle de service est une exploitation commerciale de ressources distantes au lieu des machines locales. Les clients ne payent pas les licences d'utilisation des logiciels, mais des abonnements. Ce modèle de service présente un niveau d'abstraction du contrôle des ressources de calcul qu'un utilisateur du Cloud peut avoir. Ce dernier offre ses services de façon très souple et sur demande ; cela depuis l'utilisation simple d'une application tournante sur le Cloud jusqu'à la gestion de son propre serveur.

En général, nous avons trois modèles de base [32] :

- *SaaS (Software as a Service)* : Ce modèle d'exploitation concerne les applications. Le client peut utiliser le nombre d'applications en sa convenance. Cependant, il n'a aucun contrôle sur les ressources de calcul utilisées et même sur les applications qui y sont déployées, si ce n'est que pour quelques personnalisations sur la configuration des applications utilisées ;
- *PaaS (Platefrom as a Service)* : Dans ce modèle, c'est toute une plate-forme qui est mise à disposition aux clients. Le fournisseur maintient la plate-forme et les ressources de calcul utilisées, quant aux clients, ils peuvent développer et déployer leurs propres applications. L'avantage major avec ce modèle est le fait que le client ne sera plus cloisonné aux applications SaaS. Par contre, il peut déployer des applications spécifiques à lui ou même en cours de développement ;
- *IaaS (Infrastructure as a Service* : Dans ce modèle de services, l'infrastructure de calcul, du logiciel et des équipements réseau sont mises à disposition des clients. Son objectif est d'éviter l'achat et la gestion des composants matériels et logiciels. Ce modèle est destiné aux entreprises qui voudraient gérer elles-même le middleware des serveurs. Un abonnement à un Cloud de ce type permet à l'entreprise client de disposer de toute une infrastructure informatique.

Avant le Cloud, l'entreprise était responsable de tous les modules de gestion d'un système informatique. La figure 1.1 illustre une comparaison entre les différents modèles ainsi que le niveau de contrôle du client Cloud sur les ressources de calcul louées. La figure 1.1 illustre une comparaison des niveaux de contrôle d'une entreprise des ressources de calcul en tant que client SaaS, PaaS, IaaS et non cliente. D'autres modèles de services qu'on n'a pas défini ici mais qu'on peut présenter comme des sous-modèles de ceux en haut nous avons :

- *DaaS (Data as a Service)* : est un service Cloud pour le stockage de données ;
- *DaaS (Desktop as a Service)* : est la proposition d'une infrastructure de bureau virtuel (vdi) à la clientèle par un fournisseur Cloud ;
- *SECaaS (Security as a Service)* : est une dérivée du SaaS qui vise à procurer les clients des services de sécurité externe. Ceci inclut en général l'authentification, anti-virus, détection d'intrusion ...

1.2.1.4 Les schémas de déploiement

Avant de choisir le modèle de service convenant, l'entreprise doit décider de la structure de Cloud la plus adaptée à ses besoins. En général, le Cloud dispose des modèles de

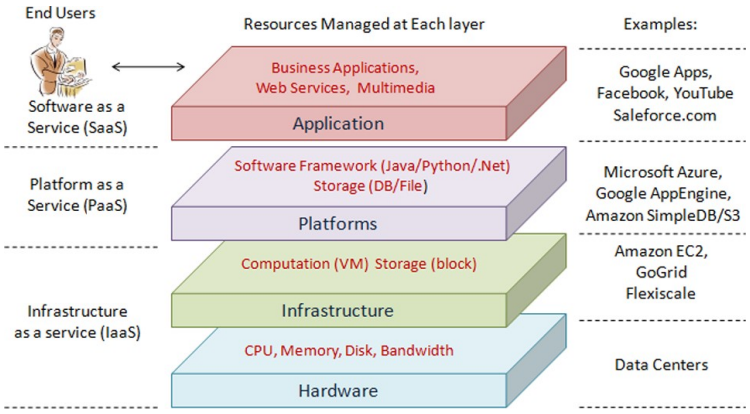

FIGURE 1.1 – Couches de services

déploiement suivants [28][32][51] :

- *Cloud publique* : L'infrastructure de ce Cloud est mise à disposition au publique via le réseau Internet. C'est une structure déployée et gérée par un fournisseur de Cloud et ses ressources sont partagées par plusieurs entreprises clientes ;
- *Cloud privé* : En contrepartie, on parle de Cloud privé lorsque l'infrastructure est mise à disposition d'une seule organisation. Ce Cloud peut être déployé dans le système informatique de l'organisation ou bien à l'extérieur, et maintenu par elle-même ou par une partie tierce. Son grand avantage par rapport au Cloud publique est le fait qu'il donne à l'entreprise le contrôle total sur l'infrastructure ;
- *Cloud communautaire* : Ce Cloud prend des caractéristiques des deux premiers Clouds. Ses ressources de calcul sont à disposition d'un ensemble d'entreprises qui ont des considérations de confidentialité communes. Ces entreprises pourront utiliser une infrastructure de Cloud de manière privée. L'infrastructure peut être déployée et maintenue par la communauté elle-même ou par une entreprise tierce ;
- *Cloud hybride* : Comme son nom l'indique, un Cloud hybride est la combinaison entre au minimum deux des modèles cités dessus. Une organisation peut se procurer des services classiques sur un Cloud publique et bénéficier du faible coût, déployer un Cloud privé en bénéficiant du contrôle total de l'infrastructure et se procurer quelques services sur un Cloud communautaire.

1.2.1.5 Motivations pour le Cloud

Tous service Internet qui nous permet de stoker des données, de les consulter et peut-être de les modifier est une illustration parfaite du Cloud. Prenons en exemple la messagerie électronique. En effet, Gmail, Yahoo, Hotmail, ... sont des services accessibles par Internet qui nous permettent de consulter des e-mails et de gérer un espace de données virtuel où seront mises par exemple les pièces jointes. Dropbox et GoogleDrive sont aussi des exemples du Cloud de type DaaS (Data as a Service).

1.2.1.6 Que faut-il améliorer ?

SaaS, PaaS et IaaS sont des modèles de services spécialement destinés aux entreprises. Comme toute technologie informatique, ces modèles apportent des avantages et des incon-

vénients. Parmi les avantages de ces modèles en comparaison avec un système informatique local nous avons :

— *Réduction des coûts* : il est très tentant d'éliminer les charges qui suivent le déploiement, la maintenance et la mise à niveau d'une infrastructure informatique par rapport au coût d'un abonnement au Cloud (surtout pour les courts et moyens termes) ;
— *Evolutivité* : plus besoin de faire des études avant l'extension du système informatique. Un ou plusieurs abonnements, et le Cloud s'en chargera même si c'est soudain ;
— *Accessibilité* : un terminal muni d'un navigateur Web et une connexion Internet fiable suffisent pour se connecter au Cloud depuis n'importe quel emplacement ;
— *Résilience* : en cas de sinistre sur les locaux, l'entreprise cliente peut facilement redevenir opérationnelle dès l'acquisition des terminaux et de la connexion Internet. Chose qui est très facile par rapport à tout reconstruire même en présence des sauvegardes ;
— Gain d'efficacité : les entreprises peuvent bénéficier de ressources de calcul très puissantes ;
— Meilleure réactivité : les fournisseurs de Cloud déploient les meilleures technologies informatiques qui permettent une utilisation très flexible des ressources ;
— Moins de ressources gaspillées : les entreprises n'auront pas à déployer des ressources qui ne seront pas exploitées avec efficience.

Parmi les inconvénients nous avons :

— *Sécurité* : cet inconvénient est la préoccupation major de chaque entreprise qui voudrait utiliser le Cloud. Il n'est pas facile de confier des processus métier et/ou des données sensibles à une partie tierce [28] ;
— *Interruptions de service* : il faut surveiller le fournisseur du Cloud et examiner le contrat de niveau de service (SLA) avant d'y externaliser une application cruciale ;
— *Conformité avec l'emplacement des données* : les fournisseurs de Cloud hébergent les données un peu partout dans le monde entier. Il faut avoir une bonne idée des lois du pays où débarqueront les données et voir si elles sont conformes à l'entreprise cliente ;
— *Mobilité des données* : en choisissant un fournisseur de Cloud, il faut toujours prévoir le changement du fournisseur pour une quelconque raison. Justement, mobiliser les données depuis un fournisseur vers un autre n'est pas une démarche toujours faisable ;
— *Intégration* : l'adoption de plusieurs applications SaaS par exemple peut engendrer un souci d'intégration ;
— *L'externalisation, c'est moins de maîtrise* : ce sera toujours gênant de passer au Cloud des données confidentielles.

1.2.2 Plateformes du Cloud

Le Cloud computing vit une progression remarquable que ce soit dans le domaine commercial ou dans le domaine de la recherche. Beaucoup de solutions propriétaires et libres sont en course pour proposer le meilleur service.

1.2.2.1 Plateformes propriétaires

Les principaux acteurs propriétaires du Cloud nous avons :

- *Google Compute Engine (GCE)* : est l'IaaS de la plate-forme Cloud de Google. Comme tout humble IaaS, il permet aux utilisateurs de lancer des MVs sur le Cloud. Ces utilisateurs doivent s'authentifier en passant par le mécanisme *OAuth*[1]
- *Salesforce* : est un éditeur de logiciel américain. Il est précurseur du Cloud computing et en propose les services entre autres :
  - Services Cloud : est une plate-forme qui permet d'optimiser la gestion de la relation client d'une entreprise ;
  - Force : est une plateforme permettant de créer des applications d'entreprises ;
  - Heroku : est un PaaS ;
  - Analytics Cloud : permet aux entreprises d'analyser leurs données et de partager les résultats entre collaborateurs.
- *Citrix* : est une entreprise multinationale américaine qui propose des produits de virtualisation pour l'adoption de services Cloud. On retrouve parmi ses produits :
  - XenServer : est une plateforme de gestion de Cloud qui se base sur une solution libre de virtualisation qui s'appelle Xen ;
  - CloudGateway : est une plateforme qui met à disposition unifiée tous les services et utilisateurs d'une entreprise ;
  - CloudPortal : est un gestionnaire des services proposés par le fournisseur Cloud Citrix.
- *OVH* : est principalement un hébergeur de site Web français. Les services Cloud qu'il propose s'intitulent HubiC pour le stockage de fichiers et la création de son propre datacenter ;
- *Amazon Web Services* : est une collection de services informatiques proposés par le groupe américain Amazon. Nous en trouvons par exemple : Amazon Elastic Compute Cloud (EC2) qui fournit des MVs utilisant Xen [2], Amzon Virtual Private Cloud qui fournit un réseau virtuel privé ...
- *IBM SoftLayer* : Avant l'acquisition de SoftLayer par IBM en 2013, c'était déjà l'un des meilleurs fournisseurs de Cloud IaaS au monde.
- *Aruba* : est une entreprise italienne. Elle propose Aruba Cloud comme une IaaS très performante à des coûts intéressants ;
- *Microsoft Azure* : est la plateforme Cloud de Microsoft. Elle propose des services de type PaaS et IaaS ;
- *Cloudwatt (Orange Business Services)* : est une entreprise française fondée par Orange et Thales. Elle propose des solutions de stockage en ligne des données selon (DaaS) et la gestion des machines virtuelles, selon le modèle IaaS. Son architecture se base sur le Cloud open source Openstack ;
- *Ikoula* : est un hébergeur informatique indépendant qui détient son propre Data Center. Cette entreprise vise les petites entreprises avec la solution "Express hosting" proposant des paquetages de solutions disponibles rapidement sur Internet.

1. Ce n'est pas un protocole d'authentification, c'en est une délégation. Il permet par exemple à un site web d'utiliser l'API de sécurité d'un autre site web pour s'assurer de l'identité d'un utilisateur

2. hyperviseur libre de MVs

Quant aux entreprises exigeantes, Ikoula propose la solution “Ikoula Enterprise Services” qui apporte des solutions sur mesure et évolutives ;

— *Rackspace* : Cette entreprise propose des solutions de Cloud variées. Par exemple : “Cloud Sites” comme une PaaS, “Cloud Files” comme un Cloud de stockage de données.

1.2.2.2 Plateformes libres

Les principaux acteurs propriétaires du Cloud nous avons :

— ***Cloud Platform***

L’hyperviseur Xen [[7]] est une solution de virtualisation d’infrastructure. Il procure une couche d’abstraction entre le matériel et le système d’exploitation. Il est utilisé par plusieurs solutions (propriétaires et libres) telles que Amazon EC2, Nimbus et Eucalyptus. En 2010, Xen.org annonce la solution XCP. Cette solution est une plateforme de contrôle de Clouds. Il inclut Xen et la suite d’outils XAPI avec des fonctionnalités du genre :

— la capacité de gérer des pools de systèmes hôtes ;
— le soutien pour les dépôts de stockage avancés ;
— le soutien aux garanties de SLA.

La figure 1.2 illustre un pool de systèmes XCP responsables d’accueillir des machines virtuelles.

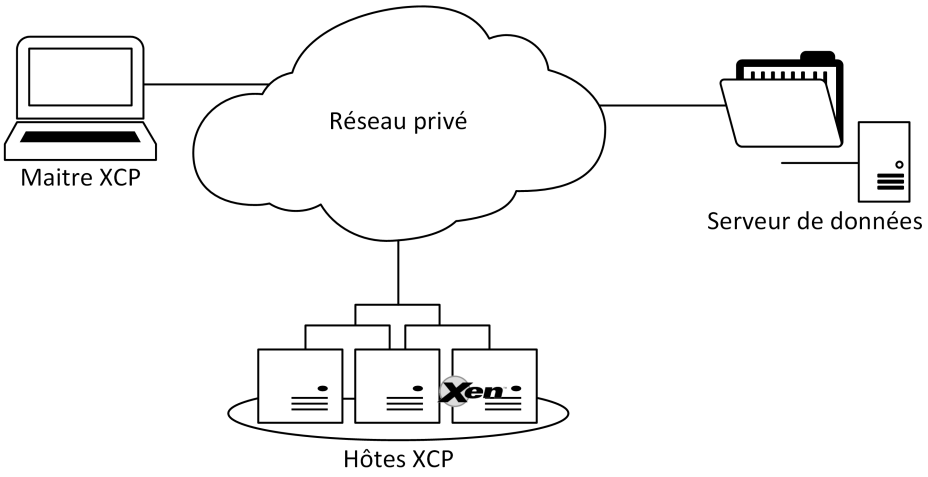

FIGURE 1.2 – Architecture XCP

— ***Nimbus***

Nimbus est un ensemble d’outils open source qui procure une solution Cloud de type IaaS. Sa vocation est d’évoluer l’infrastructure surtout pour le besoin de la science [39].

Nimbus se caractérise par les options suivantes :

— Des interfaces WSRF et EC2 ;
— L'implémentation de Xen ;
— Capacité de lancement de plusieurs machines virtuelles d'une façon flexible en assurant leurs configurations réseau ;
— Extension par plusieurs autres projets.

La figure 1.3 présente les composants de l'architecture de Nimbus.

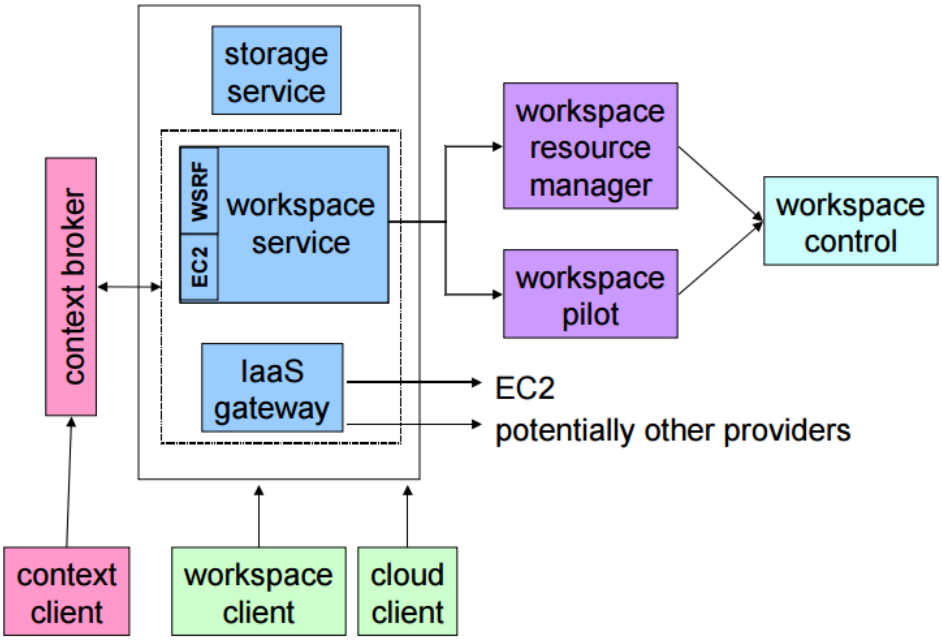

FIGURE 1.3 – Architecture Nimbus[[39]]

— ***OpenNebula***

OpenNebula est une plateforme libre, simple et flexible pour le déploiement de Clouds et de data centers virtualisés [[53]]. Il offre un large soutien pour les services de gestion tels que :

— Gestion d'utilisateurs : validation des utilisateurs suivant un mécanisme d'authentification interne ou externe (ssh, x509, ldap ou active directory) ;
— Virtualisation : support de plusieurs hyperviseurs (Xen, KVM et VMware) ;
— Monitoring : procuration d'un système de gestion propre à OpenNebula, comme il est possible d'utiliser un autre outil de gestion de data centers ;
— Networking : les réseaux virtuels peuvent être renforcés par 802.1q Vlans, ebtables, open vswitch ou VMware networking ;
— Stockage : plusieurs backends sont supportés tels que NFS, Lustre, ZFS, ... ;
— Bases de données : à part sqlite, mysql est aussi supporté.

— ***Eucalyptus***

Eucalyptus a été désigné pour qu'il soit assez facile à installer et non intrusif tant que possible [[66]]. Il est :

— compatible avec Amazon Web Service API ;
— capable d'y configurer plusieurs grappes de serveurs comme un seul Cloud ;

— possible de le déployer avec le gestionnaire de grappe de serveurs Rock Linux.

Eucaplyptus se compose de cinq éléments de haut-niveau. Chaque élément est implémenté comme un service Web.

La figure 1.4 représente ses composants.

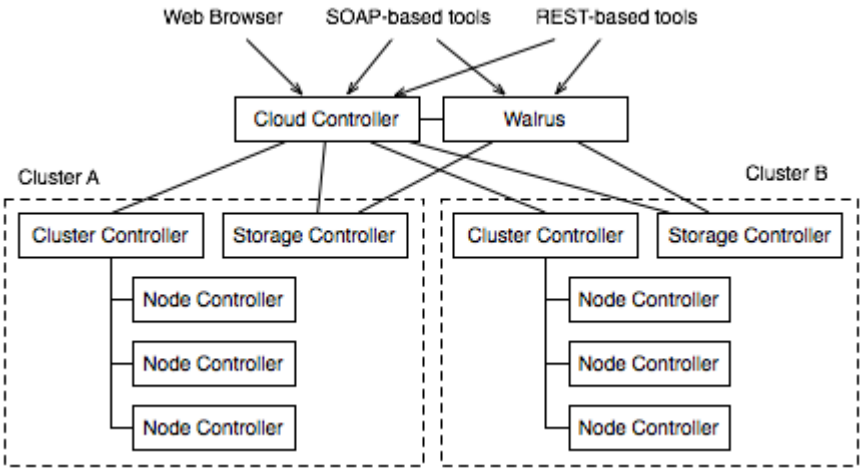

FIGURE 1.4 – L'architecture d'Eucalyptus[[66]]

— Cloud Controller : est un programme écrit en Java qui offre des interfaces compatibles avec EC2. Il s'occupe des requêtes et gère le Cloud en faisant un ordonnancement de haut niveau ;
— Cluster Controller : est un programme écrit en C. Il agit comme un front end pour un Cluster de machines et communique avec le Storage Controller ;
— Walrus : est un programme écrit en Java. Il offre un stockage persistant à toutes les machines virtuelles et eut être utilisé comme une simple solution HTTP de Storage as a Service ;
— Storage Controller : est un programme écrit en Java. Il communique avec le Cluster Controller et le Node Controller et gère des volumes de blocs Eucalyptus et des snapshots aux instances au sein de son spécifique cluster ;
— Node Controller : est un programme écrit en C. Il héberge les instances des machines virtuelles et gère le réseau virtuel.

— ***TPlatform***

TPlatform est une solution de Cloud qui procure une plateforme de développement pour les applications de web mining. Il est inspiré des technologies de Cloud de Google et agit comme une solution PaaS.

Le framework TPlatform se compose de trois niveaux [[52]]. La figure 1.5 représente les composantes de chaque niveau.

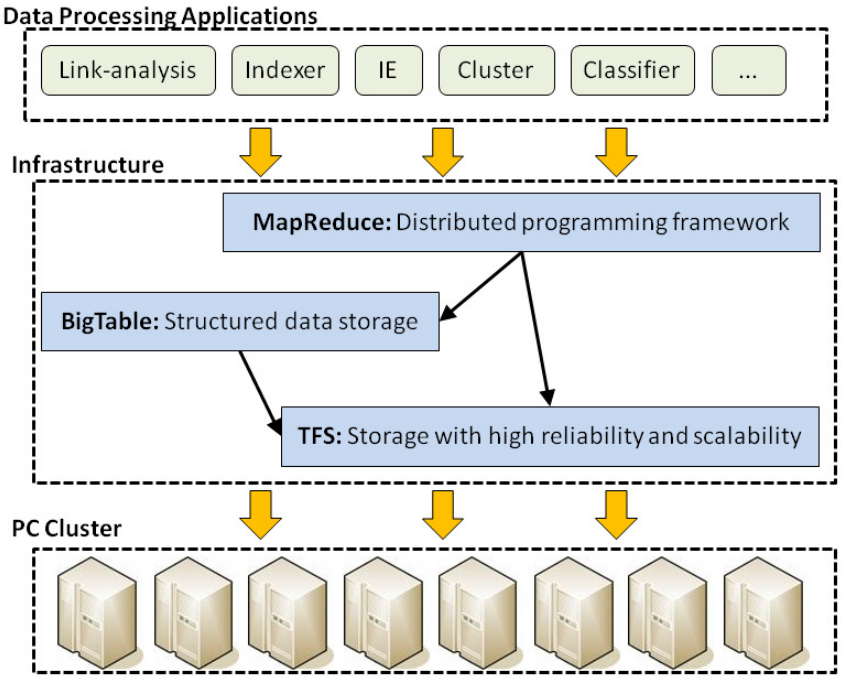

FIGURE 1.5 – Le framework TPlatform[52]

Le tableau suivant résume les solutions de Cloud libres qu'on a cité en haut.

XCP	IaaS	Un outil de maintenance automatique des Clouds	Xen
Nimbus	IaaS	Rend un cluster en un Cloud IaaS	Xen et KVM
OpenNebula	IaaS	3	Xen
Eucalyptus	IaaS	Architecture hiérarchique	Xen et KVM
TPlatform	PaaS	Spécialisé en applications web de text mining	4

TABLE 1.1 – Comparaison des solutions libres du Cloud

1.2.3 La virtualisation

1.2.3.1 Définition

La virtualisation est une séparation du système d'exploitation des ressources de l'ordinateur. Cela apporte un gain au niveau des applications qui tourneront dans la même machine physique ; cela va de la cohabitation d'applications non compatibles à la cohabitation de plusieurs systèmes d'exploitation.

Il existe beaucoup de types de virtualisation :

— Virtualisation d'applications : est l'encapsulation d'une application par rapport au système d'exploitation sur lequel elle est installée. Cela consiste en l'exécution des applications dans un environnement d'exécution donné ;

— Virtualisation de bureaux : est la dissociation de la machine physique de l'environnement de travail des utilisateurs. Ces derniers bénéficient d'un streaming de bureau des machines virtuelles se trouvant dans un serveur distant ;
— Virtualisation de stockage : permet de dissocier la gestion physique des disques vis-à-vis des serveurs de stockage qui l'utilisent. Plus simplement, elle permet de représenter les données de manière différente de leurs stockages en assurant des avantages de pointe tels que la restauration rapide des données et la réduction de besoin en matière d'alimentation et d'espace ;
— Virtualisation de plateformes : permet d'installer plusieurs environnements virtuels pour l'hébergement de plusieurs systèmes d'exploitation.

1.2.3.2 La virtualisation au service des anciennes architectures

Tout serveur qui bénéficie de ressource matérielle est un candidat idéal à une virtualisation réussie ; les services réseau (Web, Mail, DHCP, DNS, ...), les serveurs d'applications et les serveurs de bases de données peuvent tourner sur différents systèmes d'exploitation. Parmi les avantages dont l'administrateur va bénéficier nous avons :

— Temps de restauration rapide : la virtualisation diminue le temps de restauration remarquablement en gardant des copies complètes des MVs pour restaurer les services en cas de problèmes ;
— Pas de soucis à propos de l'âge de l'infrastructure : l'espérance de vie d'un matériel ne dépasse pas trop la durée de sa garantie. Avec la virtualisation et les mécanismes de migration à haut débit permettent de passer une MV d'une infrastructure à une autre en un clin d'oeil (un seul ping sera perdu techniquement) ;
— Pas de soucis si l'infrastructure n'est plus adaptée : avec une machine physique, ajouter ne serais-ce que de la RAM nécessite son arrêt total. Avec la virtualisation, appliquer une telle tâche ne nécessite que l'augmentation logique des capacités de la MV ;
— Plus de gaspillage : de par l'utilisation d'une infrastructure minime, la virtualisation permet de déployer des solutions de pointe telle que la haute disponibilité qui ne s'accompagne pas forcément de dépense sur du nouveau matériel.

1.2.3.3 Stratégies de virtualisation

Il existe plusieurs types de stratégies de virtualisation. Ces stratégies décrient le comportement du système d'exploitation invité vis-à-vis du matériel physique.

— ***Virtualisation classique*** : est le paradigme système hôte/système invité. Cela suppose une installation préalable d'un système d'exploitation directement sur la machine physique, puis une installation d'un logiciel de virtualisation (VMware player, VirtualBox, ...). Ce dernier crée des environnements virtualisés sur lesquels le système hôte sera garant des ressources dont bénéficiera un système invité. La figure 1.6 situe l'emplacement des machines virtuelles dans une machine physique par rapport aux autres applications qui y tournent.
— ***Hyperviseur*** : est un système minimaliste qui prend le contrôle du matériel et partage l'accès vers ce matériel avec les systèmes invités. Il peut s'assurer que chaque système invité n'accède qu'aux ressources matérielles autorisées sans per-

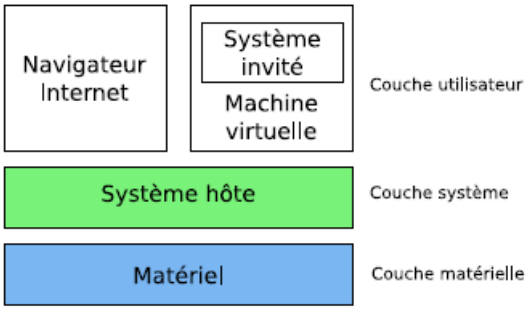

FIGURE 1.6 – L'architecture de virtualisation

turber les autres systèmes.

La figure 1.7 illustre l'emplacement de l'hyperviseur par rapport au matériel et aux systèmes invités. Parmi les hyperviseurs les plus célèbres, on retrouve Citrix

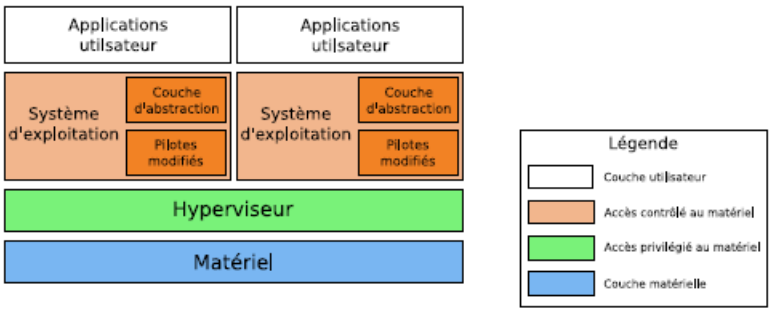

FIGURE 1.7 – L'hyperviseur

Xen, VMware ESXi, Microsoft HyperV.

— ***Emulation*** : consiste en une préparation d'un environnement virtuel qui assure tout ce dont un système d'exploitation invité ait besoin. Il se peut que ce dernier et le matériel sur lequel il est en train de s'exécuter ne soient pas compatibles. Le système invité ne saura même qu'il est entrain de s'exécuter sur une MV. Par exemple, on peut créer une MV dans un ordinateur non Sparc et y installer le système d'exploitation Solaris comme invité. Les logiciels d'émulation les plus célèbres sont QEMU et Bochs.

— ***A noyau partagé*** : consiste en un partage du noyau dont peut bénéficier des systèmes d'exploitation compatibles ; à savoir, Linux et Unix. Cette stratégie de virtualisation se base sur l'emprisonnement de processus (chroot [3]). L'idée est chrooté un ou plusieurs programmes de façon à ce qu'ils croient qu'ils sont entrain de s'exécuter sur un système à part. L'inconvénient majeur de cette stratégie de virtualisation est la compatibilité qui doit exister entre les noyaux des systèmes hôtes et leurs invités.

3. Une technique qui permet de modifier la racine d'un processus donné

1.2.3.4 Projection sur le Cloud

La virtualisation apporte des avantages valeureux qui peuvent intéresser n'importe quel responsable de système informatique et spécialement les fournisseurs du Cloud. Par exemple :

- Utilisation optimale des ressources d'un parc de machines ;
- Installation, déploiement et migration facile des machines virtuelles d'une machine physique à une autre ;
- Economie sur le matériel par mutualisation ;
- Installation, tests, développements, cassage et possibilité de recommencer sans casser le système d'exploitation hôte ;
- Sécurisation et/ou isolation d'un réseau ;
- Isolation des différents utilisateurs simultanés d'une même machine ;
- Allocation dynamique de la puissance de calcul en fonction des besoins de chaque application à un instant donné ;
- Diminution des risques liés au dimensionnement des serveurs lors de la définition de l'architecture d'une application, l'ajout de puissance (nouveau serveur etc,) étant alors transparent.

1.3 Sécurité des réseaux informatiques

1.3.1 Attaques réseaux

Il existe différents types de pirates informatiques. La manière de procéder et le contexte d'une attaque permettent de classifier le pirate. Un bon pirate informatique est celui qui ne lance aucune attaque à l'aveuglette. Un bon pirate informatique est celui qui a un objectif derrière sa volonté de pirater une cible, il consacre alors un maximum de temps pour rassembler le plus d'informations possibles à propos de sa cible, il cherche les vulnérabilités de son infrastructure, et il prépare une attaque qui très probablement exploitera une faille découverte au niveau de la cible.

1.3.1.1 Découverte de réseau

Découvrir l'ensemble des éléments d'un réseau informatique est l'objectif de tout respectueux pirate informatique. Il est impensable de lancer des attaques sur une machine sans avoir une idée sur les équipements qui l'entourent. Suivant, nous avons une description brève des plus principales de ces méthodes :

Cartographier un réseau Traceroute est un outil de diagnostic réseau qui permet de tracer le chemin que prend un paquet entre deux hôtes. L'outil s'appuie sur le protocole ICMP en manipulant à escient le champ TTL du paquet de test. Ce champ permet de ne pas garder infiniment un paquet dans le réseau informatique (surtout en cas d'erreur à propos du destinataire). Chaque routeur qui reçoit un paquet décrémente son champ TTL. Le routeur qui reçoit le paquet avec un TTL égal à 1, détruit le paquet au cas où le réseau de destination n'est pas connecté directement à ce routeur.

Pour tracer le chemin entre deux hôtes, l'outil Traceroute envoie une suite de paquets un après l'autre en incrémentant la valeur du TTL commençant par 1.

La figure 1.8 illustre le fonctionnement de l'outil traceroute.

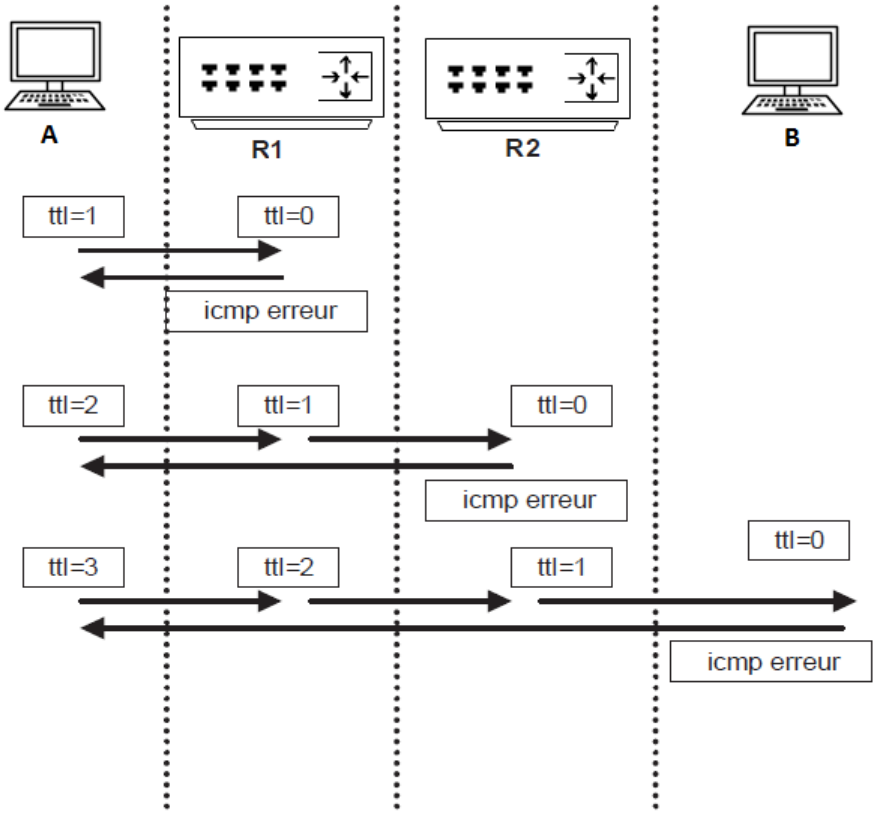

FIGURE 1.8 – Fonctionnement de Traceroute

Identifier les systèmes présents dans un réseau Avant de lancer des attaques, le pirate informatique a besoin d'avoir une idée sur l'ensemble des systèmes présents dans le réseau. Et sur chaque système, il a besoin d'identifier les applications qui y tournent. Pour chaque application découverte, le pirate cherche ses vulnérabilités et tente de les exploiter en lançant des attaques adéquates.

L'attaque qui permet d'identifier les systèmes s'appelle un balayage de systèmes. Il en existe différents types :

- Balayage ICMP : Envoyer un ping (icmp _request) à l'ensemble des machines susceptibles d'être présentes sur le réseau. A chaque fois qu'on reçoit une réponse au ping (icmp _response) cela fera une machine active dans le réseau.
  - Remarque : On peut tenter le ping sur chaque adresse IP comme on peut faire un *ping* directement sur une adresse de broadcast
- Balayage TCP : Envoyer un paquet SYN sur chaque port d'une machine distante afin de découvrir s'il y a une application qui y fait l'écoute.

— balayage semi-ouvert : Ce balayage ressemble au précédent, seulement il ne respecte pas le handshake pour ne pas donner à la cible la main pour journaliser l'événement.

Traverser un équipement de filtrage Lors d'une tentative de lancement des attaques telles que nous avons présenté en haut, il est très probable de tomber sur un équipement de filtrage comme première ligne de défense. Il existe pour cela plusieurs moyens qui permettent de dévier et de surpasser ces équipements tels que :

— Usurpation du port source : Par exemple, si l'entreprise a déployé un serveur de messagerie électronique, l'équipement de filtrage devra autoriser tout trafic associé à ce serveur avec comme port de destination le port 25. Le pirate informatique peut avoir la main sur tous les ports TCP du serveur s'il envoie un paquet avec 25 comme port source.

— Fragmentation des paquets : Cela consiste à fragmenter le paquet d'initiation d'une connexion TCP par exemple. Lorsque l'équipement de filtrage reçoit les fragments et qu'ils les contrôle séparément, il n'aura pas tous les éléments pour les bloquer. La cible s'occupera par la suite de leurs assemblage et accepter la demande de connexion.

1.3.1.2 Écoute du trafic réseau

L'écoute du trafic a essentiellement comme objectif la récupération des mots de passe qui transitent en clair sur le réseau. Cependant, le pirate peut se trouver dans un réseau qui fait parvenir le trafic vers tous les équipements connectés (grâce à un concentrateur), ou bien sur un réseau qui fait parvenir le trafic directement vers la destination (grâce à un commutateur) voire l'utilisation de VLan et cela le met dans un éventuel défi.

Concentrateur Avec ce type de périphérique de connexion, le pirate informatique n'a besoin que de configurer sa carte réseau pour qu'elle n'ignore pas un trafic qui ne lui est pas destiné. Le concentrateur envoie en diffusion tous ce qu'il reçoit depuis tous les ports qu'il possède. Le pirate par la suite pourra analyser le trafic à sa guise.

Commutateur Sur un réseau informatique connecté grâce à un commutateur, le trafic est dirigé directement vers la bonne destination. Le pirate qui voudrait tous recevoir devra commencer par attaquer le commutateur (MAC spoofing, MAC flooding ...) pour qu'il lui envoie une copie de tous ce qui passe par lui.

Vlans Si de plus le réseau est segmenté en Vlans, il faudra au pirate de configurer le port depuis lequel il est connecté en mode trunk. Cela lui permettra de recevoir tous le trafic transitant entre les Vlans.

1.3.1.3 Inférence avec une session réseau

A cause de l'implémentation naïve de la plupart des protocoles réseau [3][8][27], il est possible d'interférer avec une session réseau en profitant de cette naïveté.

Usurpation ARP Lorsqu'une machine essaye de communiquer avec une autre qui se trouve dans le même réseau qu'elle, elle a recours au protocole ARP. Ce dernier récupère l'adresse MAC de la machine destinataire et l'enregistre sur un tableau qu'on appelle le cache ARP. L'ARP spoofing consiste à envoyer des réponses ARP à une machine cible pour usurper une autre machine. Ainsi, à chaque fois que la cible voudrait communiquer avec la deuxième machine, elle le fera avec la machine usurpatrice.

Usurpation IP L'usurpation IP consiste à forger un paquet avec une adresse IP source falsifiée. Cela peut avoir comme objectif de surpasser un système d'authentification qui se base sur les adresses IP, comme il peut avoir comme objectif le lancement d'une attaque en modifiant son identité.

Homme du milieu L'attaque homme du milieu consiste à faire passer le trafic transitant entre deux machines par le biais d'une troisième qui est sous la prise d'un pirate sans que les intervenants ne le suspectent. Cela est possible par exemple en appliquant de l'usurpation ARP au niveau des deux machines.

La figure 1.9 représente cette manipulation.

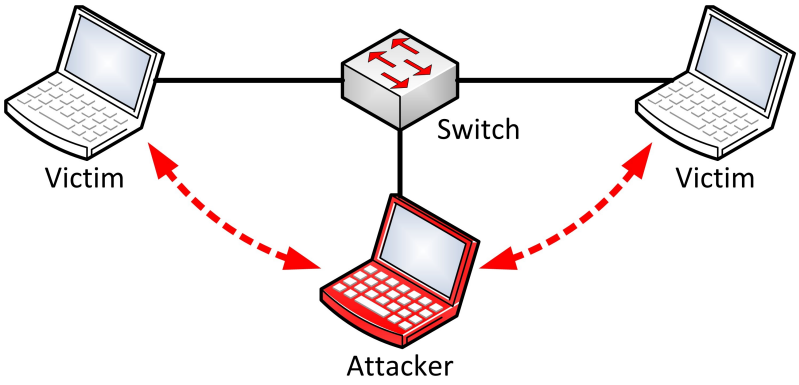

FIGURE 1.9 – L'attaque MITM

1.3.1.4 Déni de service d'un réseau

Les attaques de déni de service (DoS) ont pour objectif de rendre indisponible un service, une machine voire même tout un réseau informatique. Ces attaques sont imprévisibles et ont un impact défavorable sur les activités de l'institution cible. Parmi les techniques classiques qui peuvent causer un déni de service :

— Ping of death : Consiste à construire une requête ICMP (ping) dans un paquet d'une taille anormale et d'envoyer ses fragments à la cible. Une fois reçue, après l'assemblage, la cible peut avoir un comportement anormal.

— Win nuke : Consiste à construire un paquet avec un pointeur urgent dans son en-tête. Envoyer ce paquet au port TCP 139 (NetBIOS) à un destinataire peut lui bloquer sa machine et afficher un écran bleu. Les données qui n'ont pas été encore enregistrées seront alors perdues.

— Land (Local Area Network Denial) : Consiste à envoyer une demande de connexion TCP SYN dans un paquet IP falsifié. Le paquet contient dans les champs d'adresse IP source et destination l'adresse IP de la machine cible sur un port ouvert. Ce qui ramène la cible à répondre à elle même d'une façon continue.
— Teardrop : Consiste en l'insertion d'informations erronées à propos de la fragmentation d'un paquet IP. A la réception et lors du ré-assemblage, la cible se met dans un état d'instabilité.

Au cours de cette thèse, nous nous sommes intéressés au déni de service par inondation. Ce type est le plus classique et simple à élaborer. Il consiste à submerger la cible de requête qu'elle n'est capable de traiter. Le grand souci avec ce genre d'attaque est le fait que le trafic peut avoir l'air d'un trafic légitime. Chose qui le rend le type de DoS le plus ennuyeux à traiter.

1.3.1.5 DoS par inondation

Il existe plusieurs types d'inondations. nous traitons ceux qui suivent :

Inondation SYN Le protocole TCP/IP s'appuie sur un modèle appelé "le three way handshake" pour établir des connexions TCP. Ce modèle se dirige normalement de la façon décrite dans la figure 1.10. Le client commence par envoyer une demande de connexion

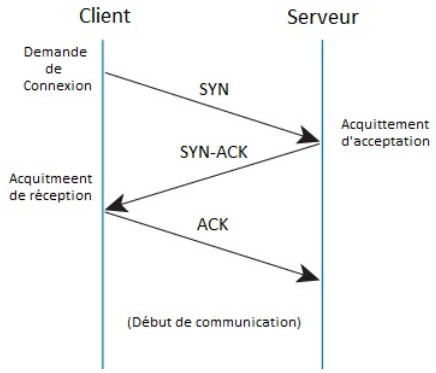

FIGURE 1.10 – Modèle d'une connexion TCP

(SYN). A la réception de cette demande le serveur réserve des ressources et envoie son acceptation sous forme d'acquittement (SYN-ACK). Le client à son tour doit accuser la réception de cette acceptation et s'attache à ces ressources réservées. Et puis, la communication peut s'établir suivant cette connexion.

Un client malveillant n'acquittera jamais la réponse du serveur et tentera de demander plus de connexions TCP plus vite que le "timeout" des semi-connexions. Cela occupera une partie de ressource du serveur jusqu'à l'épuisement.

Suivants, les techniques qui permettent de pallier à ce type d'attaques :
— La limitation du nombre de connexions pour la même source ;
— La libération des connexions semi-ouvertes après un délai raisonnable ;

— L'utilisation des SYN cookies [4]

Inondation UDP L'inondation UDP consiste en un envoi de multiples paquets UDP à une destination sur plusieurs de ses ports. Ne contenant pas de service sur ces ports, la cible se verra obligée de répondre par des messages ICMP "destination inaccessible". En multipliant les paquets UDP, la cible ne pourra faire que répondre à ces requêtes et sera indisponible pour les clients légitimes. En plus, si l'attaque est lancée entre deux machines, la bande passante sera consommée surtout par cette attaque vu que l'UDP est prioritaire par rapport au TCP dans les réseaux locaux.

Inondation ICMP L'inondation ICMP consiste en l'envoi de plusieurs ping à la cible. Cela nécessite que la bande passante de l'attaquant soit nettement importante par rapport à celle de la cible.

Une variante intéressante de cette attaque s'appelle le "smurf". Au lieu d'envoyer des ping vers la cible, avec l'attaque smurf on envoie un ping vers une adresse de broadcast en usurpant l'adresse IP. Cette adresse IP recevra les réponses de ping de la part de toutes les machines qui sont actives dans le réseau.

Pour contrer ce type de DoS, la cible peut :
— Utiliser un pare-feu pour qu'il arrête les echo ICMP anormalement nombreux.

1.3.1.6 Attaque DDoS

Le DDoS est une dérivée distribuée du déni de service. Il consiste en l'envoi d'attaques DoS depuis un ensemble de machines. Ces dernières peuvent être en collaboration pour une cause quelconque, comme elles peuvent être sous la prise d'une personne malveillante. Chose qui amplifie considérablement l'impact de l'attaque sur la cible.

La figure 1.11 représente l'architecture des machines qui peuvent être impliquées dans une attaque DDoS. Le pirate commence par s'emparer d'une première couche de machines dites "zombies". Chaque zombie prend le contrôle d'un ensemble de machines dites "botnet". Depuis chaque machine botnet, le pirate lance une attaque DoS simultanément vers une même cible. Ce qui en suit est une attaque DDoS.

1.3.2 Mécanismes de défense réseaux

1.3.2.1 Principes de sécurité

L'objectif de tout mécanisme de sécurité est d'assurer l'un ou une combinaison des principes de sécurité suivants [28] :
— La confidentialité : est le fait d'assurer que seules les personnes autorisées ont accès à l'information ;

4. Un mécanisme qui permet au serveur de ne pas allouer des ressources au profit du client tant qu'il n'a pas encore envoyé le dernier ACK du 3-way handshake

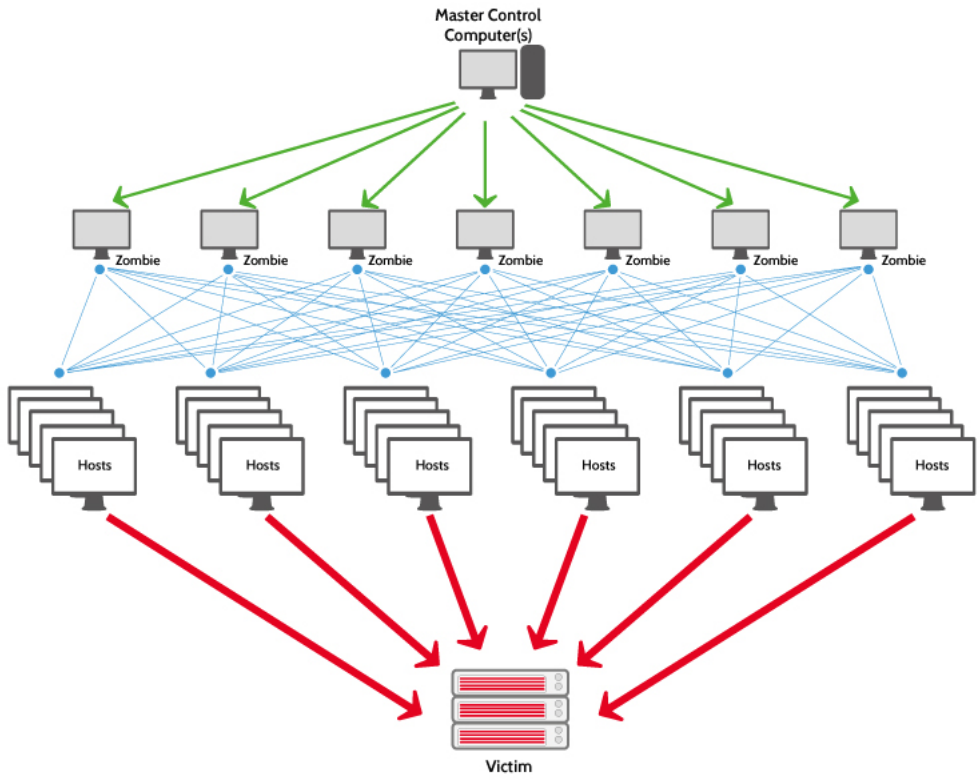

FIGURE 1.11 – Attaque DDoS

— L'intégrité : est le fait que l'information ne subisse pas de modification non autorisée, qu'elle soit accidentelle ou bien volontaire ;
— La disponibilité : est le fait de garantir qu'un service soit disponible aux utilisateurs illégitimes.

D'autres principes de sécurité s'ajoutent à ces trois principes essentiels. On en cite :

— La non-répudiation : Assurer qu'aucune des parties prenantes dans une transaction ne puisse renier le fait d'y avoir participé ;
— l'authentification : S'assurer de l'identité d'une machine distante qui essaye d'établir une communication ;
— L'anti-rejeu : Assurer la non-réutilisation d'une séquence d'un flux enregistré.

Dans le cadre de cette thèse, on se focalise sur les attaques qui réussissent le DoS et sa variante DDoS. Nous présentons alors dans la suite quelques techniques de sécurité qui permettent de lutter contre ce genre d'attaques :

— Blocage ICMP : Efficace contre les attaques inondation et le smurf ;
— Limitations UDP : Quelques services se basant sur UDP doivent être limités et d'autres doivent être désactivés telles que (echo, chargen, ...) ;
— Limitations de débit : Abandonner l'excès de paquets ICMP et segments SYN lorsque le réseau se trouve à une charge anormale ;
— Cookies SYN : Pour l'authentification des connexions entrantes.

Une configuration qui suit les bonnes pratiques est nécessaire pour une première couche de sécurité. Mais, pour une sécurité meilleure, nous avons des équipements qui permettent d'entraver les attaques.

1.3.2.2 Cryptage

Définition La cryptographie est l'étude et la pratique de techniques qui permet de sécuriser une information même en présence d'une personne non autorisée.

Objectifs du chiffrement Grâce aux techniques de chiffrement, on peut assurer des principes de sécurité tels que : la confidentialité, l'intégrité et l'authentification.

Types de chiffrement Nous avons trois types de chiffrement :

- *La cryptographie symétrique* : Elle repose sur l'utilisation d'une seule clé que ce soit pour le chiffrement que pour le déchiffrement. Elle se caractérise par sa rapidité, seulement, elle est entravée à cause de la difficulté de partage de la clé ;
- *La cryptographie asymétrique* : Elle consiste en l'utilisation d'une paire de clés par chaque intervenant. Une clé sera partagée publiquement et l'autre sera gardée en privé. A chaque fois qu'on voudrait communiquer d'une manière confidentielle avec quelqu'un, on chiffre l'information avec sa clé publique et on lui envoie l'information chiffrée. A la réception, le destinataire doit déchiffrer l'information avec sa propre clé privée. Puisqu'il est supposé qu'il soit le seul à avoir cette clé, on suppose que la confidentialité a été assurée. L'obstacle majeur de ce concept est la validité des clés publiques ;
- *La cryptographie hybride* : C'est une combinaison des deux autres types. Elle consiste en l'utilisation de la cryptographie asymétrique pour partager une clé symétrique, et assurer la confidentialité de la communication en utilisant cette clé symétrique.

1.3.2.3 Infrastructures PKI

Une PKI est un ensemble de mécanismes qui permet de vérifier, valider et gérer les clés publiques sous forme de certificats numériques. Elle se compose de trois éléments :

- Une autorité d'enregistrement pour vérifier les demandes des utilisateurs nouveaux ;
- Une autorité de certification pour la génération des certificats numériques, leurs signatures et leurs publications ;
- Un annuaire compatible avec les protocoles X.509 pour le stockage et la publication des certificats numériques [5].

Pour qu'une communication se passe d'une manière sécurisée, le serveur envoie au client son certificat numérique, le client vérifie la validité du certificat, l'identité du serveur, l'identité de l'infrastructure PKI et la signature de cette dernière. Normalement le client fait confiance à l'infrastructure PKI, il vérifie sa signature et si elle est correcte, il utilise la clé publique sans se poser des questions.

1.3.2.4 Réseau VPN

Un VPN est un mécanisme qui permet de créer un lien directe et sécurisé entre deux points sur un réseau publique. Il se base en général sur les mécanismes de chiffrement en

5. une entité qui contient la clé publique d'un intervenant accompagnée par des informations qui concernent l'intervenant lui-même et l'organisation qui lui a créé ses clés

employant des protocoles tels que SSL/TLS, SSH et IPsec :

— SSL/TLS : offre une très bonne solution de tunnelisation ;
— SSH : permet l'ouverture de sessions à distance et de manière sûre ;
— IPsec : permet de transporter des données chiffrées pour les réseaux IP.

Pour les entreprises, l'intérêt derrière le déploiement d'un serveur VPN est de permettre à des utilisateurs surnommés des nomades de pouvoir se connecter au réseau interne de l'entreprise de façon à faire croire aux équipements et ordinateurs du réseau interne que les nomades sont connectés depuis le réseau local. Pour les individuels, un serveur VPN peut jouer le rôle d'une passerelle et modifier les adresses IP sources des paquets qu'il transfère. Chose qui permettra de rendre plus difficile de localiser le vrai ordinateur émetteur.

1.3.2.5 Firewall

Définition Un pare-feu est une passerelle réseau qui filtre les paquets. Il permet de protéger un terminal ou un réseau contre les intrusions.

Politiques de sécurité Pour le déploiement d'un pare-feu, on peut lui définir une politique de filtrage parmi les deux suivantes :

— Autoriser les connexions décrites dans les règles et tout bloquer ;
— Interdire les connexions décrites dans les règles et tout autoriser.

Il est clair que la première politique est la plus sûre, seulement elle peut présenter une entrave pour les activités du système informatique.

Types de filtrage Nous avons trois types de filtrage :

— Le filtrage simple : Analyse l'en-tête de chaque paquet indépendamment des autres paquets ;
— Le filtrage dynamique : Analyse l'en-tête de chaque paquet en assurant un suivi de l'échange auquel il appartient ;
— Le filtrage applicatif : Filtre les communications application par application.

L'utilisation majeure de ce mécanisme de défense est la possibilité de déployer un réseau en respectant l'architecture DMZ.

Zone DMZ Une zone DMZ est une architecture qui sépare les serveurs accessibles par un réseau externe du réseau interne. Son objectif est de limiter la portée d'une attaque au cas où l'un ou plusieurs des services déployés au niveau de cette zone ne soient vulnérables à une quelconque attaque. La figure 1.12 présente les architectures possibles d'une zone DMZ.

— L'architecture de la gauche se base sur un seul pare-feu possédant trois interfaces réseau. Une interface lui permet de se connecter sur Internet, une deuxième de connecter la zone DMZ et la troisième pour connecter le reste du réseau. Ce type de séparation entre les serveurs publiques et le reste du réseau interne est conseillé pour les déploiements personnels et au niveau des petites entreprises.
— L'architecture de la droite utilise deux pare-feux. Elle est conseillée aux moyennes et grandes entreprises vu que son utilisation apporte plus d'avantages. Le deuxième pare-feu (le plus proche du réseau interne) représente une deuxième couche de

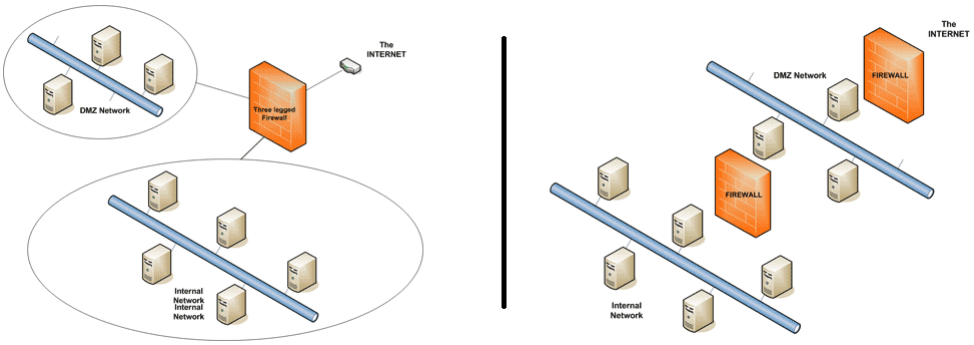

FIGURE 1.12 – Les architectures DMZ

sécurité au cas où le premier soit surpassé. Chose dont on ne profitera pas au déploiement de l'autre architecture.

1.3.2.6 *Systèmes de détection d'intrusion*

Définition Un système de détection d'intrusion récolte des informations à partir d'un ordinateur (ses fichiers de journalisation) ou d'un réseau (en écoutant le trafic) et les analyse pour identifier des violations possibles de la politique de sécurité.

Comment détecter une intrusion ? En général, il y a deux approches de détection :

- Détection par scénarios : Identifie les événements qui font part d'une attaque connue. Cela est possible grâce à une base de données qui contient des scénarios d'attaques connues et un algorithme de comparaison. Elle est facile à implémenter, fait moins de *faux négatif*[6] mais en aucun cas elle ne sera capable de détecter les attaques *Zero day*[7] ;
- Détection par anomalie : Identifie les événements qui ne font pas part du cours normal d'un système informatique. Cela est possible grâce à l'établissement d'un profil "idéal" des activités du système informatique. Par la suite, l'IDS ne fait qu'alerter chaque activité n'appartenant pas au profil normal. Cette approche est difficile à implémenter, génère trop de *faux positif*[8] mais très efficace face aux attaques zero day.

Types et emplacement d'IDS ? Le type d'IDS relève le type d'événement de sécurité que l'IDS va devoir analyser. Nous en avons trois :

- Network-IDS : Cet IDS analyse les paquets. Souvent on le met derrière un pare-feu pour n'analyser que les intrusions qui ont réussi à le surpasser ;
- Host-IDS : Celui là analyse les fichiers de journalisation d'un système d'exploitation et décide de la légitimité des événements enregistrés. Il est comparable à un logiciel antivirus et souvent on le déploie sur des terminaux ;
- Hybrid-IDS : Comme tout système hybride, celui-là englobe les caractéristiques des deux systèmes précédents.

6. Absence d'alertes de détection à propos d'événements malveillants
7. Des attaques non connues
8. Des alertes de détection à propos d'événements non malveillants

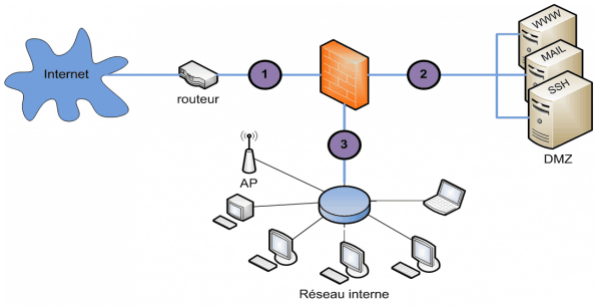

FIGURE 1.13 – Positionnement d'un IDS dans un réseau local

Emplacement de l'IDS Un IDS peut jouer un rôle différent d'un emplacement à un autre dans un réseau informatique. La figure 1.13 présente les emplacements possibles d'un IDS. Cette figure représente un réseau local déployé en se basant sur une architecture DMZ.

- L'emplacement 1 : en mettant l'IDS dans cet emplacement, il sera en mesure d'analyser le trafic provenant d'Internet. Au niveau du routeur, il se peut que l'administrateur y soit activé des règles de filtrage. Dans ce cas, l'IDS aura comme rôle, la détection des intrusions qui ont pu surpasser le routeur. Le problème ici est que l'IDS fera face à tout le trafic d'emblée ;
- L'emplacement : L'IDS dans cet emplacement sera en mesure de détecter les intrusions qui ont pu surpasser le pare-feu. L'avantage réside dans le fait que l'IDS aura moins de surcharge vu que le pare-feu aura bloqué une bonne part du trafic non désirable. Toutefois, dans cet emplacement, l'IDS sera neutre face au trafic malveillant qui peut circuler depuis Internet vers le réseau interne ;
- L'emplacement 3 : La plupart des attaques proviennent de l'intérieur. Cet emplacement permettra à l'IDS d'y avoir l'oeil.

Les IDS distribués A l'encontre de la détection, un IDS peut être infaillible face à une attaque distribuée. Pour remédier à cela, une nouvelle méthode de détection a vu le jour ; la détection distribuée. Cela consiste en un déploiement de plusieurs sondes dans le même réseau. L'objectif ici est d'analyser des événements provenant de plusieurs emplacements. Cela implique un langage commun entre les IDSs. En plus, plusieurs architectures sont proposées qui permettent de spécifier le niveau de la décision. Les architectures proposées sont :

- L'architecture centralisée : plusieurs sondes sont déployées. L'analyse et la détection se font dans un seul emplacement ;
- L'architecture hiérarchique : les sondes sont regroupées par ensembles, l'analyse et la détection se font dans un seul emplacement au niveau de chaque ensemble ;
- L'architecture distribuée : l'analyse et la détection se font au niveau de ces emplacements. Une communication s'établit pour les événements qui méritent d'être partagés.

1.3.2.7 Pots de miel

Définition Un pot de miel est un leurre qui permet d'attirer les pirates informatiques afin de les neutraliser.

Fonctionnement Dans un système informatique, un pot de miel n'entre dans aucune activité de l'organisation qui l'a déployé, ainsi, toute activité ou tout trafic survenu au pot de miel est suspecte d'appartenir à un comportement malicieux.

Types On distingue deux types de pots de miel :

— Les pots de miel à faible interaction : essayent d'enregistrer un maximum d'informations en offrant un minimum de privilèges aux attaquants ;

— Les pots de miel à forte interaction : à l'encontre du précédent, ce type est plus intentionnellement plus vulnérable. Cela est risqué, mais il permet d'aller jusqu'au bout avec les attaquants.

1.4 La répartition des charges

La répartition des charges (ou le load balancing) est la capacité d'une application à distribuer la charge de traitement sur plusieurs systèmes séparés afin d'augmenter les performances en traitement de requêtes. Ce concept parait simple, mais derrière, on tombe sur des facteurs qui accentuent sa complexité. La répartition de charge peut être comparé à la gestion d'équipes.

1.4.1 Les concepts de la répartition des charges

1.4.2 Définition

La répartition des charges est la capacité de pouvoir transférer n'importe quelle portion de traitement d'une requête à partir d'un système vers un autre système indépendant. Elle partage également la charge que peut avoir un seul système entre plusieurs sous-systèmes. Cela réduit remarquablement la charge du système destinataire, chose qui lui permet de pouvoir répondre à plus de requêtes.

1.4.3 Les ressources impliquées

Les ressources d'un ordinateur qui sont principalement impliquées pour le traitement des requêtes sont bien évidemment :

— Le processeur : tout action faite par l'ordinateur a besoin d'un temps accordé pour qu'elle soit traité par le processeur. Cela veut dire que le processeur est un composant clé qui indique le niveau de performance des ordinateurs. Sans une capacité de traitement de processeur satisfaisante, le serveur ne pourra plus répondre à plus de requêtes. Par exemple, un serveur Web qui traite des requêtes qui nécessitent l'interrogation d'une base de données influence la charge du réseau remarquablement en haussant considérablement le nombre de requêtes. La répartition des charges

ici peut se faire en séparant le serveur Web de la base de données. Ainsi, ces deux processus gourmands en terme de CPU ne seront plus en concurrence.

— La mémoire vive : ce composant de l'ordinateur est l'emplacement où vivent les applications en cours d'exécution. La mémoire vive permet aux applications et leurs données d'être plus proches du processeur pendant un temps limité. La mémoire vive comporte une taille maximale et peut comporter plusieurs applications gourmandes en concurrence dans le même ordinateur, ceci peut amener à son dysfonctionnement.

— Le réseau : le temps de réponse d'un serveur dépend principalement de la bande passante et du débit dont il bénéficie. Recevoir toutes les requêtes des différents clients sur la même interface réseau peut surcharger le réseau et conduire à ralentir l'arrivée de quelques requêtes voir leur perte.

— Le stockage : l'utilisation de l'espace libre d'un disque de stockage est cruciale. Si un serveur manque d'espace libre, tout type d'erreur peut arriver. Les données de la mémoire virtuelle peuvent être écrasées, ceci conduit à une exécution improbable des applications.

1.4.4 La répartition des charges en pratique

Le coeur de la répartition des charges est savoir gérer avec efficience les ressources impliquées dans l'exécution des applications. Voici les principales méthodes pour réussir la répartition des charges :

1. Partager les applications sur plusieurs serveurs (comme la séparation du serveur Web du serveur de base de données).
2. Ajouter des composants spécialisés aux applications pour équilibrer le traitement nécessaire pour chaque fonction d'application.
3. Déployer plusieurs serveurs pour un partage collectif de la charge. Cela augmentera remarquablement la quantité de puissance de traitement.

1.4.5 Recommandations

Comme pour n'importe quelle forme de modification d'un système, il existe de bonnes pratiques qu'un responsable doit respecter afin de déployer une solution avec la répartition de charges.

1. Comprendre son système : ce principe peut paraître simple, cependant il doit être fait de la manière la plus profonde possible. Pour rester simple, la compréhension de son système est le fait d'avoir une bonne idée des tâches qu'il assure pour les activités de l'organisme.
2. Planification : il existe plusieurs recommandations pour assurer la planification de la répartition des charges, cependant, ces recommandations restent limitées vu que les systèmes diffèrent d'une organisation à une autre. Le conseil ultime pour une bonne planification est d'avoir réussi le premier principe (Comprendre son système) et d'être proactif.

3. Administration et test : il faut savoir que les réglages de performances ne sont pas adaptés à toutes les situations. Il est alors primordial de garder un oeil sur le système et suivre ses développements.

1.4.6 La répartition de charge en pratique

La répartition de charge part toujours du même concept (distribuer des tâches à plusieurs entités), mais, elle peut toucher un ou plusieurs aspects des systèmes informatiques.

1.4.6.1 La répartition des charges d'un site Web

L'idée de répartir les charges pour un serveur Web est très simple. Elle consiste en une distribution des différentes requêtes Web destinées au serveur sur plusieurs machines. Cela est pratique non seulement pour la répartition des charges mais aussi pour une tolérance de pannes. Au cas où l'un des serveurs tombe en panne, les autres sauront maintenir le service.

Le déploiement de cette solution pour ce cas là dépend des attentes de l'administrateur. Sinon, le déploiement le plus simple qu'on peut avoir contiendra un serveur de répartition de charge muni de deux interfaces réseaux, une connectée sur Internet et l'autre connectée avec deux serveurs Web qui assurent le même service.

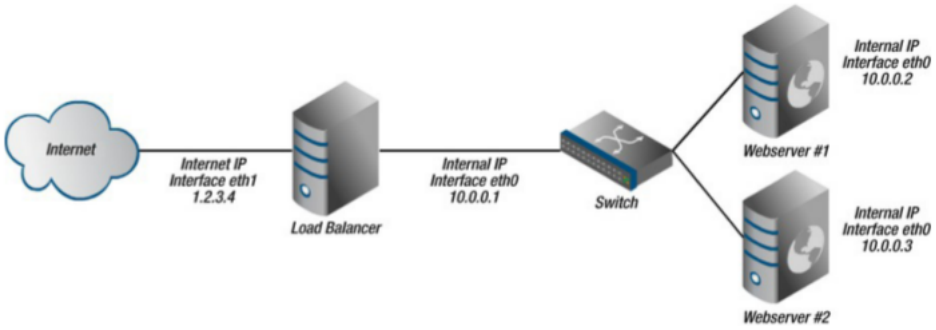

FIGURE 1.14 – Déploiement de base d'un serveur Web

Architecture de base La figure 1.14 représente une architecture de base pour le déploiement d'un serveur Web qui adopte la technique de répartition de charge. La maquette contient un répartiteur de charge (Load balancer) qui joue le rôle d'un intermédiaire entre les clients Web et l'ensemble de serveurs Web qui travaillent pour un même service. Le répartiteur de charge comporte deux interfaces réseaux, l'une lui permet d'être connecté avec les clients Web et l'autre lui permet d'être connecté avec l'ensemble de serveurs (le réseau virtuel).

Pour la distribution des requêtes, ce n'est pas aussi simple qu'on peut penser. La métrique n'est pas toujours quel est le serveur le moins surchargé pour qu'on lui transfert une requête donnée. Par exemple, on peut déployer plusieurs types de serveur Web qui chacun

est meilleur pour répondre à des types précis de requêtes ; on peut déployer Apache pour répondre aux requêtes dynamiques et nginx pour les pages statiques.

1.4.6.2 La répartition des charges d'une base de données

La répartition des charges au niveau d'un SGBD (Système de Gestion de Bases de Données) consiste à en déployer plusieurs sur un même réseau. Une donnée enregistrée au niveau de l'un de ces SGBD devra-être répliquée au niveau des autres SGBD, créant ainsi, un réseau de base de données. De cette façon là, en plus du gain qui concerne la répartition de charge (une requête est envoyée vers le SGBD le moins surchargé) si l'un de ces SGBD tombe en panne, les autres assureront toujours le service.

1.4.6.3 La répartition des charges d'un Cloud

Gestion des ressources virtuelles : Depuis l'angle de vue des hyperviseurs, la gestion des ressources pour une machine virtuelle peut s'avérer une tâche qui risque d'atténuer les performances d'un serveur en cas de modification.

— Pour le réseau, la performance consiste purement à assurer assez de débit dont le serveur aura besoin pour le transfert de données.

— Pour le stockage des données, il existe deux métriques : combien peut-on augmenter la capacité de stockage d'une machine virtuelle et combien peut-on hausser la capacité de stockage du hôte en entier. Si un serveur donné manque de capacité, ce n'est qu'une question d'un paramétrage simple pour en ajouter, mais s'il faut augmenter la taille de l'hôte en entier, là il sera délicat de le faire car ce sera un "upgrade" physique. Le problème avec les upgrade physique est très agaçant, car il faut redémarrer la machine hôte pour qu'il prenne effet, et cela aura comme conséquence le redémarrage de toutes les machines virtuelles hébergées. Pour contourner cela, il faut penser à un stockage sur un périphérique réseau, ce qui signifie que l'augmentation de capacité de stockage concerne une autre machine qui se trouve derrière la carte réseau du hôte.

1.5 Conclusion

Le Cloud représente un domaine de recherche bien séduisant. Le fait de proposer des services accessibles depuis Internet est un défi de sécurité challengeant. Après avoir introduit le Cloud et les éléments marquants de la solution que nous proposons, le chapitre suivant discute les problèmes de sécurité du Cloud et les techniques de détection qui y sont déployées.

Chapitre

2 Déploiement automatique des politiques de sécurité au sein des Firewalls

Sommaire

2.1 Introduction

Une politique de sécurité définit un ensemble des propriétés de sécurité, chaque propriété représentant un ensemble de conditions que le système doit respecter pour rester dans un état considéré comme sûr. Une définition incorrecte ou l'application partielle d'une politique peut entraîner le système dans un état non-sûr, autorisant le vol d'informations ou de ressources, la modification d'informations ou la destruction du système. Dans ce chapitre, nous donnons en premier lieu, une définition générale des propriétés de sécurité, d'une politique de sécurité et des mécanismes utilisés pour l'application d'une politique, et en deuxième lieu, on va traiter les outils et les mécanismes automatiques fournissant un environnement convivial pour spécifier, configurer et déployer en sûreté ces politiques au sein de pare feu. Nous allons donc, proposer des stratégies de déploiement pour les deux importantes catégories d'édition de politiques. Ensuite, nous fournissons un algorithme [72] correct, efficace et sûr pour le déploiement de politiques de sécurité.

2.2 Politiques de sécurité

2.2.1 Définition des propriétés d'une politique de sécurité

La définition et l'application d'une politique de sécurité représente le coeur de la sécurité d'un système d'information. En effet, la sécurité des systèmes d'information repose sur trois propriétés fondamentales : la confidentialité, l'intégrité et la disponibilité. L'interprétation de ces trois aspects varie suivant le contexte dans lequel elles sont utilisées. Cette représentation est liée aux besoins des utilisateurs, des services et des lois en vigueur. La définition et l'application de ces propriétés font partie intégrante des critères d'évaluation de la sécurité, tant au niveau international dans le TCSEC [2], qu'au niveau européen dans ITSEC [25]. Plusieurs définitions de ces propriétés existent dans [2], [25], [11], nous proposons dans cette section une synthèse de ces propriétés.

2.2.1.1 Confidentialité

La confidentialité repose sur la prévention des accès non autorisés à une information. La nécessité de confidentialité est apparue suite à l'intégration des systèmes d'informations critiques, telles que les organisations gouvernementales ou les industries, dans des zones sensibles. La confidentialité est ainsi définie par l'Organisation Internationale de Normalisation (ISO) comme "le fait de s'assurer que l'information est seulement accessible qu'aux entités dont l'accès est autorisé". Plus précisément, la propriété de confidentialité peut être définie comme suit : Soit I de l'information et soit X un ensemble d'entités non autorisées à accéder à I. La propriété de confidentialité de X envers I est respectée si aucun membre de X ne peut obtenir de l'information de I. La propriété de confidentialité implique que l'information ne doit pas être accessible par certaines entités, mais doit être accessible par d'autres. Les membres de l'ensemble X sont généralement définis de manière implicite. Par exemple, lorsque l'on parle d'un document confidentiel, cela signifie que seulement certaines entités connues ont accès à ce document. Les autres, qui par définition ne doivent pas avoir accès à ce document, font partie de l'ensemble X.

2.2.1.2 Intégrité

D'une manière générale, l'intégrité désigne l'état de données qui, lors de leur traitement, de leur conservation ou de leur transmission, ne subissent aucune altération ou destruction volontaire ou accidentelle, et conservent un format permettant leur utilisation. Dans le cas d'une ressource, l'intégrité signifie que la ressource 'fonctionne' correctement, c'est à dire qu'elle respecte sa spécification. La propriété d'intégrité des données vise à prévenir toute modification non autorisée d'une information. La garantie de la fidélité des informations vis à vis de leur conteneur est connue sous le nom d'intégrité des données. La garantie des informations en rapport avec la création ou les propriétaires est connue sous le nom d'intégrité de l'origine, plus communément appelée authenticité. Plus précisément, la propriété d'intégrité peut être définie comme suit : Soit X un ensemble d'entités et soit I de l'information ou une ressource. Alors la propriété d'intégrité de X envers I est respectée si aucun membre de X ne peut modifier I. Tout comme la propriété de confidentialité, les membres de X sont généralement définis de manière implicite. Ainsi, les utilisateurs ayant le droit de modifier une information ou une ressource sont définis de manière explicite

(propriétaires de cette information, etc.). Les autres utilisateurs forment alors l'ensemble X.

2.2.1.3 Disponibilité

La disponibilité se réfère à la possibilité d'utiliser l'information ou une ressource désirée. Cette propriété est à mettre en parallèle avec la fiabilité car le fait d'avoir un système qui n'est plus disponible est un système défaillant. Dans le cadre de la sécurité, la propriété de disponibilité se réfère au cas où un individu peut délibérément interdire l'accès à certaines informations ou ressources d'un système. Plus précisément, la propriété de disponibilité peut être définie comme suit : Soit X un ensemble d'entités et soit I une ressource. Alors la propriété de disponibilité de X envers I est respectée si tous les membres de X ont accès à I.

2.2.1.4 Principe et Propriétés dérivées

La confidentialité, l'intégrité et la disponibilité sont des concepts fondamentaux de la sécurité. En se basant sur ces définitions, plusieurs cas particuliers, qui ne considèrent qu'un sous-ensemble de conditions d'intégrité, de confidentialité ou de disponibilité, peuvent ainsi être définis. Dans cette partie, nous donnons une définition de quatre de ces cas particuliers : 1) le confinement de processus, 2) le principe du moindre privilège, 3) la séparation de privilège et 4) la non-interférence. Ces propriétés peuvent ainsi être classées dans une des trois classes précédentes en fonction des privilèges qui sont contrôlés. Confinement de processus Lampson [42] caractérise le problème du confinement par : Définition (Problème du confinement) : Le problème du confinement concerne la prévention de la divulgation, par un service (ou un processus), d'information considéré comme confidentiel par les utilisateurs de ce service. Lampson définit alors les caractéristiques nécessaires à un processus pour qu'il ne puisse pas divulguer de l'information. Une de ces caractéristiques est liée au fait qu'un processus observable ne doit pas stoker de l'information pour la réutiliser ultérieurement. En effet, si un processus stocke de l'information liée à un utilisateur A et qu'un autre utilisateur B peut observer ce processus, il y a alors un risque que B puisse obtenir cette information. Un processus qui ne stocke pas d'information ne peut donc pas divulguer cette information. A l'extrême, si un processus peut être observé, il ne doit pas non plus effectuer d'opérations. En effet, dans certains cas, un analyste peut reconstituer le flux des événements (ou l'état d'un processus) et en déduire des informations sur les entrées de ce processus. En conclusion, un processus qui ne peut pas être observé et qui ne peut pas communiquer avec d'autres processus ne peut pas divulguer de l'information. Cette propriété est alors appelée l'isolation totale. En pratique, l'isolation totale est difficilement applicable sur un système. Les processus confinés partagent généralement des ressources telles que le processeur, le réseau ou le disque avec d'autres processus non confinés. Les processus non confinés peuvent alors divulguer de l'information provenant de ces ressources partagées. De plus, deux processus peuvent s'échanger de l'information de manière indirecte (sans communication explicite), via des canaux cachés. Définition (Canal caché) : Un canal caché est un canal de communication possible mais qui n'était pas prévu, lors de la conception du système, comme canal de communication. Supposons, par exemple, qu'un processus p veut transmettre de l'infor-

mation à un processus q et que ces processus ne peuvent pas communiquer directement. Si ces deux processus partagent une même ressource, par exemple un système de fichiers, alors p peut transmettre de l'information à q via un canal caché. Par exemple, p peut créer un fichier 0 ou 1 pour représenter les bit 0 ou 1 et end pour indiquer la fin de la communication. Ainsi à chaque création d'un fichier, q obtient un bit d'information et supprime le fichier pour obtenir le bit suivant jusqu'à la fin du message symbolisé par end. De ce fait q obtient de l'information de p sans communication directe par seule observation du comportement de p. Définition (Confinement de processus) : Soit p un processus, x et y deux entités. La propriété de confinement d'un processus p est garantie si p ne peut pas transférer de l'information de x à une autre entité y. Finalement, comme l'indique Lampson, la notion de confinement de processus est transitive. Si un processus q est considéré comme confiné contre la divulgation d'information, alors si ce second processus invoque un autre processus q, ce processus doit aussi être confiné sinon il peut divulguer de l'information provenant de p. De part la difficulté d'implanter l'isolation totale, le problème des canaux cachés et la difficulté pour avoir la transitivité, le confinement est une propriété difficile à appliquer sur un système réel. De ce fait, le confinement est généralement appliqué à un sous-ensemble du système, par exemple les processus privilégiés. On parle alors de confinement partiel du système. Moindre privilège : Le principe du moindre privilège [59] établit qu'une entité ne devrait jamais avoir plus de privilèges que ceux requis pour compléter sa tâche. l'application de ce principe vise à minimaliser les privilèges associés à chaque entité ou processus du système. Ce principe est lié au confinement de processus. En effet, pour être appliqué, il requiert que les processus soient confinés dans le plus petit domaine d'exécution possible. Ce principe peut ainsi être défini comme suit : Définition (Moindre privilège) : Soit x une entité, P un ensemble de privilèges assignés à x et T un ensemble de tâches attribuées à x. Alors, le principe du moindre privilège sur x pour les tâches T est respecté si tous les privilèges de P sont nécessaires pour réaliser T. Cette propriété implique qu'une entité qui n'a pas besoin d'un droit d'accès ne doit pas avoir la possibilité d'obtenir ce droit. De plus, si une action requiert l'augmentation des privilèges d'une entité, ces droits supplémentaires doivent être supprimés lorsque cette action est terminée. Par exemple, si une entité a le droit d'écrire ou d'ajouter de l'information dans un fichier et qu'une tâche donnée ne nécessite pas la modification mais seulement l'ajout de données, cette tâche devra être exécutée avec le privilège d'ajout et non d'écriture. Le privilège d'écriture ne devra être accordé que si une modification des données existantes est nécessaire. Séparation de privilèges : Une première définition du principe de la séparation de privilèges implique qu'un système ne doit pas permettre l'acquisition de privilèges via un contrôle basé sur une unique condition. Par exemple, sous GNU/Linux, l'acquisition des droits administrateurs nécessite de : 1) connaître le mot de passe administrateur, 2) appartenir au groupe wheel. Une seconde définition de ce principe, donnée dans [18] (page 187), sous entend que : 'Une règle élémentaire de séparation de privilèges peut être que toute personne ayant le droit de créer ou modifier des objets n'ait pas le droit de les exécuter'. De même dans [60] : 'La séparation de privilèges implique que différentes opérations réalisées sur un même objet doivent être effectuées par des utilisateurs différents'. d'une manière générale, ce principe peut être défini par : Définition (Séparation de privilège) : Soit o un objet, P un ensemble de privilèges associés à o et X un ensemble

d'entités. Alors, le principe de séparation de privilège implique que les privilèges P sur o soient distribués sur l'ensemble des utilisateurs X. Cette définition sous-entend que l'ensemble des privilèges d'un système doit être distribué sur l'ensemble des utilisateurs. Par exemple, une règle, généralement appliquée aux services système (ou démons système), implique qu'un service ne doit pas pouvoir créer/modifier un fichier puis l'exécuter. Ce principe, qui peut être étendu au cas des utilisateurs système, nécessite donc que les services systèmes ne possèdent pas le privilège de modification et d'exécution sur un même objet. Ce principe permet d'endiguer des attaques visant la génération d'un script par un service du système (ayant les droits administrateur) en vue de son exécution. Dans le cas des utilisateurs, ceci implique que les utilisateurs qui créent des objets soient différents de ceux qui les exécutent. Non-interférence : Le principe de la non-interférence [24] implique que deux ou plusieurs processus puissent s'exécuter simultanément sans interférer, c'est-à-dire sans modifier la vision des données ou le comportement des autres processus. d'une manière générale, la non-interférence peut être définie par : Définition (Non-interférence) : Soit X un ensemble d'entités et soit I un ensemble de données. Un ensemble d'entités X n'interfère pas avec un ensemble de données I si les valeurs de I sont indépendantes des actions effectuées par X. Dans le cas d'un système, si nous considérons deux groupes de processus : les processus de haut niveau (processus système) et de bas niveau (utilisateur). La non-interférence peut correspondre à : l'exécution des processus utilisateurs n'a pas d'incidence sur l'exécution des processus système. Cela permet d'endiguer certaines attaques des utilisateurs sur les services système.

2.2.2 Politique de sécurité

Une politique de sécurité spécifie ce que l'on entend par 'sécurisé', pour un système ou un ensemble de systèmes, et peut être définie de manière informelle (langage naturel) ou formelle (par des relations mathématiques). Une politique de sécurité considère les différentes propriétés de sécurité, définies dans la section précédente, relatives au système à protéger. En ce qui concerne la confidentialité, une politique identifie les différents états du système permettant l'accès à des données non-autorisées. Il ne faut alors pas seulement considérer le vol d'information direct, mais aussi une suite de transferts d'informations qui, par transitivité, peut conduire à un flux d'informations. Dans le cas de l'intégrité, une politique de sécurité identifie quels sont les moyens autorisés pour modifier de l'information, mais aussi quelles entités ont l'autorisation de modifier ces informations. Dans le cas de la disponibilité, une politique de sécurité définit quels services doivent être fournis. Des paramètres supplémentaires peuvent aussi être définis, comme un intervalle de temps pendant lequel un service doit être accessible, ou un temps minimal de réponse. Ce type de propriété a un lien étroit avec la qualité de service. La déclaration d'une politique de sécurité doit permettre de définir formellement les propriétés de sécurité désirées sur un système. Si l'on doit prouver que le système est sûr, la définition de la politique doit pouvoir permettre de prouver que les différentes propriétés définies et leurs implantations sont correctes. Si une preuve formelle est impossible les propriétés de sécurité peuvent être garanties par des tests. En pratique, peu de politiques de sécurité définissent formellement les propriétés. La politique définit alors l'ensemble des actions légales entre entités. Le problème de vérification de ces règles reste en général entier, au regard de la

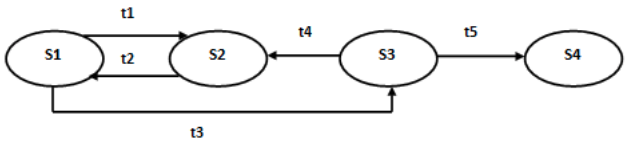

FIGURE 2.1 – Exemple d'automate à états finis représentant les états d'un système

confidentialité mais surtout vis à vis de l'intégrité.

2.2.2.1 Définition générale

Considérons un système d'information comme un automate à états finis avec un ensemble de fonctions de transition changeant l'état du système. Dans cette représentation, nous pouvons définir une politique de sécurité par : Définition (Politique de sécurité) : Une politique de sécurité est une déclaration qui partitionne les états d'un système en un ensemble d'états autorisés (ou sûrs) et un ensemble d'états non autorisés (ou non-sûrs). Une politique de sécurité fixe donc le contexte dans lequel on peut définir un système 'sûr'. Définition (Système sûr) : Un système sûr est un système qui, partant d'un état sûr, ne pourra jamais entrer dans un état non-sûr. Considérons l'automate présent dans la figure 2.1 contenant quatre états et cinq transitions représentant un système. Prenons l'exemple d'une politique de sécurité qui partitionne ce système en un ensemble d'états autorisés A = s1, s2 et un ensemble d'états non-autorisés NA = s3, s4. Ce système est considéré comme non-sûr car partant d'un état sûr, il existe une transition amenant le système dans un état non-sûr. Néanmoins, si l'arc entre s1 et s3 est supprimé, le système sera considéré comme sûr. En effet, dans ce cas, il n'est plus possible d'atteindre un état non-sûr à partir d'un état sûr. Le passage d'un état sûr à un état non-sûr est alors appelé une brèche de sécurité : Définition (Brèche de sécurité) : Une brèche de sécurité apparaît lorsqu'un système entre dans un état non-autorisé.

2.2.2.2 Garantie d'une politique de sécurité

La garantie et la définition d'une politique de sécurité sont deux aspects différents. La garantie dépend du mécanisme de sécurité. Définition (Mécanisme de sécurité) : Un mécanisme de sécurité est une entité ou une procédure dont l'objectif est de respecter chaque propriété de la politique ou d'en détecter les violations. Un tel mécanisme peut appliquer de manière active l'ensemble des propriétés de politique de sécurité ou passivement en détecter les violations. Ainsi, un mécanisme de contrôle d'accès garantit une propriété de sécurité alors qu'un système de détection d'intrusions en détectera les violations.

2.2.3 Contrôle d'accès

Comme nous l'avons vu dans la section 2.2.1, la sécurité des systèmes d'information repose sur trois propriétés fondamentales : la confidentialité, l'intégrité et la disponibilité. Le contrôle d'accès a pour objectif de garantir deux de ces trois propriétés fondamentales (confidentialité et intégrité) via la définition et l'implantation d'une politique sécurité régissant les accès : une politique de contrôle d'accès. L'exécution d'un système d'exploita-

tion peut être vue comme un ensemble de sujets et d'objets interagissant ensemble dans le but d'effectuer des actions qui modifieront (ou non) l'état du système. Ces entités peuvent être séparées en deux ensembles : celui des sujets, qui sont les entités actives (processus), et celui des objets, qui correspondent aux entités passives (ressources, fichiers, sockets). Une opération effectuée par un sujet sur un objet peut alors être représentée par un triplet (sujet, objet, type d'accès). Une politique de contrôle d'accès est composée d'un ensemble d'opérations permises. Un modèle de contrôle d'accès classique peut être modélisé par :

- Un ensemble de sujets : Un sujet est une entité active du système (un processus) ;
- Un ensemble d'objets : Un objet est une entité passive, un conteneur d'informations, sur lequel un sujet peut effectuer des actions (les fichiers, sockets de communication, périphériques matériels) ;
- Un ensemble de permissions : Une permission représente une action autorisée entre un sujet et un objet (par exemple, lecture, écriture, exécution, etc.), ou entre deux sujets (envoi de signal ou de message inter-processus) ;
- Un ensemble de commandes de modification : Ces commandes permettent de modifier les ensembles précédents (création et destruction de sujets, d'objets et de privilèges).

Les commandes de modification permettent de faire évoluer ces politiques de sécurité pour, par exemple, prendre en compte l'intégration de nouveaux utilisateurs dans le système d'exploitation. Cette évolution doit être contrôlée pour prévenir les abus de la politique ou les configurations indésirables favorisant les intrusions. C'est pourquoi, chaque mécanisme de contrôle d'accès définit des règles de modification plus ou moins restrictives suivant le but recherché, et nécessitant éventuellement certains privilèges comme un niveau administrateur. Les parties suivantes présentent les deux premières familles de modèles de contrôle d'accès. On en distingue principalement trois :

- Contrôle d'accès discrétionnaire : La caractéristique principale du DAC ou Discretionary Access Control est le fait que ce sont les utilisateurs qui attribuent les permissions sur les ressources qu'ils possèdent. C'est le type de mécanisme utilisé dans les systèmes d'exploitation traditionnels. En effet, étant donné que l'attribution des droits est faite par les utilisateurs et non par les administrateurs, il se révèle très léger en termes de sécurité.
- Contrôle d'accès obligatoire : Contrairement au DAC, le MAC ou Mandatory Access Control délègue l'attribution des permissions à une entité tierce, typiquement un administrateur externe de la politique de sécurité. Ainsi, les utilisateurs du système ne peuvent pas intervenir dans l'attribution des permissions d'accès, même s'ils disposent de droits d'administration dans le système d'exploitation.
- Contrôle d'accès basé sur les rôles : Le modèle RBAC pour Role-Based Access Control a pour but de simplifier l'administration des droits d'accès des utilisateurs individuels en fournissant un niveau d'indirection supplémentaire. Plutôt que de donner directement des permissions aux utilisateurs, on définit différents rôles possibles pour l'utilisation du système d'exploitation, avec des droits d'accès associés. Ensuite, chaque utilisateur a accès à un ensemble de rôles suivant son activité, et donc à un sous-ensemble de permissions correspondant.

2.2.3.1 Contrôle d'accès discrétionnaire

Le contrôle d'accès discrétionnaire (DAC) est actuellement le modèle implanté dans la majorité des systèmes d'exploitation. Son nom vient du fait que la définition des droits d'accès à une ressource du système est réalisée à la discrétion de son propriétaire. Typiquement, les droits d'accès sur un fichier sont positionnés par l'utilisateur déclaré comme propriétaire de ce fichier. Par exemple, sous Unix, le propriétaire d'un fichier gère les droits de lecture, d'écriture et d'exécution de ses fichiers pour lui-même, les membres d'un groupe et tous les autres utilisateurs du système.

1. Modèle de Lampson : Une manière naturelle pour représenter un modèle de contrôle d'accès est sous forme de matrice de contrôle d'accès. Dans cette représentation, chaque ligne représente un sujet, chaque colonne représente un objet ou un sujet et chaque élément correspond à un ensemble de privilèges. Le premier modèle de ce type fût proposé par Lampson en 1971 [26]. Lampson propose ainsi de placer, dans une matrice A, l'ensemble D des domaines de protection (qui représentent des contextes d'exécution pour les programmes, i.e. les sujets) sur les lignes, et en colonnes l'ensemble X des objets (incluant les domaines). Il pose ensuite deux définitions : Définition (Capabilities Lists) : Étant donné un domaine dD, la liste des capacités (capabilities) pour le domaine d est l'ensemble des couples (o, A[d, o]), X. Définition (Access Control Lists) : Étant donné un objet o X, la liste de contrôle d'accès (ACL) pour l'objet o est l'ensemble des couples (d, A[d, o]), D. La définition (Capabilities Lists) lie un domaine d avec l'ensemble des permissions qu'il possède sur chaque objet o. Une liste des capacités représente l'ensemble des actions permises pour un sujet sur le système. La définition (Access Control Lists) lie un objet o avec l'ensemble des permissions accordé à chaque domaine d. Une liste de contrôle d'accès contient ainsi l'ensemble des permissions sur un objet défini pour chaque sujet du système. A noter que les notions de capability et d'ACL avaient déjà été définies de façon générale dans [41]. Dans [26], l'auteur raffine ces définitions en les liants à la notion de matrice de contrôle d'accès. Dans ce modèle la modification de la politique de sécurité, via l'application des règles de modification, nécessite la modification de la matrice de contrôle d'accès. Par exemple, la prise en compte d'un nouvel utilisateur passe par l'ajout d'une ligne de la matrice. Par conséquent, la mise à jour d'une telle politique est quelque peu fastidieuse. Le modèle de Lampson a été progressivement amélioré pour donner naissance à d'autres modèles tels que le modèle HRU [30].
2. Modèle HRU : Établi par Harrison et al. en 1976, le modèle HRU est traditionnellement utilisé pour décrire les politiques de contrôle d'accès discrétionnaires. Dans ce modèle, une matrice P représente l'ensemble des droits d'accès des sujets sur des objets. Ainsi chaque sujet possède des droits d'accès sur certains objets. Mais dans ce modèle, les sujets sont eux-mêmes des objets. Ainsi chaque sujet possède des droits de modification de la matrice de contrôle d'accès afin de créer ou de détruire des sujets ou des objets (ajout ou suppression de colonne). Mais un sujet a aussi la possibilité de modifier les droits que possède un autre sujet sur un objet. HRU propose de modéliser la protection dans les systèmes d'exploitation comme suit :

— Une matrice de contrôle d'accès P ;
— L'ensemble des sujets S et l'ensemble des objets O modélisés par l'ensemble des entiers de 1 à k ;
— L'ensemble des droits génériques R tels que : possession, lecture, écriture, exécution ;
— Un ensemble fini C de commandes c1, . . . , cn, représente l'ensemble des opérations fournies par le système d'exploitation (création de fichier, modification des droits...) ;
— Un ensemble d'actions élémentaires E : enter et delete pour l'ajout et la suppression de droits, create subject et create object pour la création de nouveaux sujets et objets et enfin destroy subject et destroy object pour la destruction de sujets et objets.

Du point de vue de la protection, les commandes de l'ensemble C ont toutes la même forme : elles prennent en paramètres une liste de sujets et objets, et suivant la présence de certains droits dans la matrice P, elles effectuent des actions élémentaires sur le système. La configuration du système de protection est représentée par le triplet (S, O, P). Afin d'étudier le problème de la sûreté d'un système de protection, HRU s'intéresse au transfert de privilège (droit), qui se produit lorsqu'une commande insère un droit particulier r, dans une case de la matrice P où il était précédemment absent. Ce problème de sûreté peut être défini comme : 'étant donné une configuration initiale de la politique de sécurité, un système est considéré sûr (safe) pour un droit r si aucune des commandes de ce système ne provoque le transfert du droit r'. Les auteurs du modèle HRU ont ainsi prouvé que :

— Dans le cas d'un système de protection mono-opérationnel, i.e. dans lequel toutes les commandes ne contiennent qu'une seule action élémentaire, le problème de sûreté est décidable.
— Dans le cas général, le problème de la sûreté d'un système de protection est indécidable.

Même si le problème de protection du modèle HRU à mono-opération est décidable et que ce système est facilement manipulable, il reste trop simple pour couvrir des politiques de sécurité réelles. Par exemple, la seule commande de création de fichiers d'un système d'exploitation de type UNIX se compose déjà de deux actions : création d'un nouvel objet, et positionnement des droits sur celui-ci. En outre, les auteurs mentionnent le fait que la taille du problème est réduite à une taille polynomiale dès lors que les actions de création de sujet ou d'objet sont retirées du système de protection.

3. Modèle TAM : Le modèle TAM (Typed Access Matrix), introduit par [65], propose une extension du modèle HRU intégrant la notion de typage fort. Cette notion, étendant les travaux plus anciens sur SPM (Sandhu's Schematic Protection Model) par [62], se traduit par l'attachement de 'types de sécurité' immuables à tous les sujets et objets du système d'exploitation. Cette approche apporte au modèle HRU la notion de type. Un ensemble fini T de type de sécurité et un ensemble d'opérations élémentaires, permettant la gestion de ces types, est alors ajouté à

la modélisation de HRU vue précédemment. En outre, cet ensemble est fini : la création ou la suppression de nouveaux types n'est pas possible. De plus, lorsqu'un objet est créé, son type est défini et n'est jamais modifié. Les opérations autorisées sur la matrice de contrôle d'accès dépendent alors des types de sujets et d'objets concernés. Sandhu démontre ensuite que le problème de sûreté dans le cas de la version monotone (qui ne possède pas de requête de destruction ou de suppression) de ce modèle est décidable. Ce nouveau modèle, MTAM (Monotonic Typed Access Matrix), est obtenu en ôtant les opérations de suppression du modèle TAM. Toutefois la complexité de ce problème reste NP. C'est pourquoi, Sandhu définit le modèle MTAM ternaire, dans lequel toutes les commandes ont au maximum trois arguments. Au prix d'une perte d'expressivité, le problème de sûreté voit sa complexité ramenée à un degré polynomial. Une version dite augmentée de TAM, appelée ATAM [5], a ensuite été proposée afin de fournir un moyen simple de détecter l'absence de droit dans une matrice de contrôle d'accès. L'objectif de cette démarche étant de pouvoir facilement modéliser la séparation de privilèges, celle-ci préconisant l'intervention de plusieurs utilisateurs pour mener à bien une tâche.

4. Discussion : Le modèle de contrôle d'accès couramment utilisé sur les systèmes d'exploitation actuels est le Discretionary Access Control. Le DAC délègue l'accord des permissions d'accès à la discrétion des propriétaires des ressources du système. En pratique, ce modèle de contrôle d'accès a clairement montré ses limites. En effet, les attaques possibles contre les systèmes d'exploitation visent à obtenir un accès de niveau super-utilisateur par abus de services système afin de gagner de nouveaux privilèges (en anglais, privilege escalation). Lorsqu'une telle attaque est réussie, l'attaquant obtient des pouvoirs qui outrepassent le DAC et donnent un accès complet à l'ensemble des ressources du système d'information. De plus, diverses études [2], [61], [44] ont établi la faiblesse des modèles DAC. En effet, le contrôle d'accès discrétionnaire repose sur la capacité des utilisateurs à définir correctement les permissions sur les fichiers dont ils sont propriétaires. Toute erreur peut mener à une défaillance de sécurité. Par exemple si lorsque le fichier qui contient les mots de passe (/etc/shadow sous GNU/Linux) venait à être autorisé en écriture pour tous les utilisateurs, ceci pourrait modifier le mot de passe du super-utilisateur et obtenir ses droits. Les auteurs du modèle HRU [30] établissent que le problème de sûreté d'un système de protection, c'est-à-dire l'assurance qu'un droit d'accès ne sera jamais accordé à un utilisateur donné, est un problème indécidable. Dans le cas du modèle MTAM [65], qui introduit la notion de type de sécurité, le problème est décidable. L'auteur fournit une preuve avec une complexité NP dans le cas général, polynomiale dans le cas ternaire. Cependant le modèle MTAM ôte les opérations de suppression de droits, sujets et objets (propriété de monotonie) pour obtenir ce résultat. Or, ceci n'est pas envisageable dans le cadre de l'utilisation normale d'un système d'exploitation, où la suppression d'utilisateurs, d'applications, est courante. L'introduction de la notion de rôle par [61] se justifie par le fait que les modèles DAC ne répondent pas aux exigences de sécurité des systèmes d'exploitation, même non militaires. Enfin, les critères d'évaluation de la sécurité des systèmes [2], [25] établissent une nette différence

de niveau de sécurité entre un système implantant seulement DAC, et un système implantant également le modèle MAC. Il ressort donc de l'analyse des modèles DAC que ceux-ci n'offrent pas de réelle garantie quant à la confidentialité et l'intégrité. Pour que les propriétés puissent être garanties, il est nécessaire d'utiliser un modèle MAC permettant de restreindre les droits accordés au super-utilisateur et éviter que les utilisateurs modifient les permissions de leurs fichiers de façon erronée.

2.2.3.2 Contrôle d'accès mandataire

L'objectif du contrôle d'accès mandataire (MAC) est d'imposer donc une politique non modifiable par les utilisateurs finaux, dans le but de garantir la sûreté du système. La politique définit l'ensemble des interactions valides entre sujets et objets. Pour contrôler les accès entre sujets et objets, Anderson propose d'utiliser un Moniteur de Référence (Reference Monitor). Cette matrice étant fixe, il devient alors possible de garantir que des droits d'accès considérés dangereux ne pourront être donnés ou obtenus par erreur. Ce concept de Moniteur de Référence est au coeur de la définition du Contrôle d'accès mandataire. Ce terme, qui désignait initialement le modèle défini par Bell&LaPadula, a vu sa signification évoluer et représente aujourd'hui tout mécanisme qui place la gestion de l'attribution des permissions d'accès hors d'atteinte des utilisateurs concernés par ces permissions. Les différents modèles, présentés dans cette section, ont tous pour objectif de spécifier une politique de contrôle d'accès.

1. Modèle Bell et LaPadula : Le modèle Bell-LaPadula, établi par [21], plus couramment appelé BLP, est issu des besoins en confidentialité des données du monde militaire. Il a ainsi pour objectif de prévenir toute divulgation d'informations. Ce modèle repose sur le modèle HRU et exclut toute création ou destruction de sujets ou d'objets. Ce modèle introduit la notion de label associé à chaque sujet et objet du système. Un label représente un niveau de sécurité et contient deux types d'identifiants de sécurité :
   — Un identifiant hiérarchique. Cet identifiant représente le niveau de classification pour les objets (i.e. son niveau de sensibilité) ; et le niveau d'habilitation pour les sujets, qui sont typiquement : non classifié, confidentiel, secret, top secret. Ces niveaux de classification sont classés sous forme hiérarchique.
   — Des identifiants de catégories. Ces différentes catégories d'informations correspondent aux différentes organisations manipulant les données, par exemple militaire, privé, public. Ces identifiants sont indépendants de la hiérarchie de confidentialité.

   En plus du contrôle effectué à l'aide de la matrice de contrôle d'accès classique, ce modèle repose sur le respect de deux nouvelles règles :
   — ss-property ou simple security property : lorsqu'un sujet demande un accès en lecture sur un objet, son niveau d'habilitation doit être supérieur ou égal à celui de l'objet. Cette règle assure la confidentialité de l'information.
   — *-property ou star-property : seuls les transferts d'informations depuis des objets de classification inférieure vers des objets de classification supérieure sont autorisés. Cette règle assure donc la prévention contre la divulgation d'infor-

mations.

Le système est donc modélisé de la façon suivante : un ensemble de sujets S, un ensemble d'objets O, une matrice d'accès M et une fonction f : SO 1 . . . n retournant l'identifiant hiérarchique d'un sujet ou d'un objet. On dispose également d'un ensemble de permissions d'accès A=e, r, a, w classées suivant leur capacité d'observation (lecture) et d'altération (écriture) de l'information :

e : ni observation ni altération (execute) ;

r : observation sans altération (read) ;

a : altération sans observation (append) ;

w : observation et altération (write).

Les règles précédentes, qui doivent être vérifiées par le mécanisme de contrôle d'accès, peuvent donc s'écrire :

— r

— r

La vérification de la propriété * nécessite le contrôle de tous les flux d'informations entre sujets et objets possibles sur le système. Lors de l'implantation de ce modèle, l'existence de canaux cachés, dans le système d'exploitation cible, peut ainsi entraîner des problèmes de flux d'informations non contrôlables. Afin de prévenir ce problème, une version plus restrictive de la politique BLP utilise les règles suivantes :

— No Read Up : Lorsqu'un sujet demande un accès en lecture sur un objet, son niveau d'habilitation doit être supérieur ou égal à celui de l'objet.

— No Write Down : Lorsqu'un sujet demande un accès en écriture sur un objet, son niveau doit être inférieur ou égal à celui de l'objet.

Ce qui se traduit par :

— r

— a

— w

L'article de Bell et LaPadula décrit l'intégration de ce modèle dans MULTICS, on le trouve également dans certaines versions de Solaris, HPUX ou autres systèmes UNIX. Généralement, ce modèle n'est pas désigné sous le nom BLP, mais plutôt sous l'acronyme MLS (Multi-Level Security pour modèle de sécurité multi-niveaux) comme dans l'Unix System V/MLS. Cependant, l'implantation de ce modèle, sans aucune adaptation à l'environnement utilisé, peut être difficile. L'attribution des labels à certains sujets ou objets peut poser des problèmes, c'est par exemple le cas du dossier /tmp dans lequel n'importe quel processus est supposé pouvoir créer des fichiers. Certaines propriétés ont donc été ajoutées au modèle. Par exemple, le niveau de classification d'un objet passe à un niveau supérieur (celui du sujet) lorsqu'il est accédé en écriture par un sujet de niveau supérieur. Notons que dans le modèle initial, il est normalement interdit de modifier un fichier de niveau inférieur. L'effet néfaste lié à cet aménagement est que les objets ont tendance à être "tirés vers le haut", et donc se trouver sur le même niveau (le plus élevé) après un certain temps. On doit attirer l'attention sur le fait que le modèle BLP ne traite que la confidentialité et comme nous le verrons par la suite il ne garantit pas la confidentialité telle que définie dans [25].

2. Modèle Biba : Alors que le modèle BLP vu précédemment ne répond qu'à des besoins de confidentialité, le modèle dit 'Biba' [10] vise à garantir l'intégrité. Il s'agit en réalité d'un modèle similaire à BLP qualifié de modèle dual à BLP. A chaque sujet et objet est associé un niveau d'intégrité qui correspond respectivement au 'pouvoir d'accès' et au niveau d'intégrité (du sujet qui l'a créé). Les objectifs de sécurité de cette politique visent à :
   — Interdire toute propagation d'information d'un objet vers un autre objet de niveau d'intégrité inférieur ;
   — Interdire à tout sujet de modifier des objets possédant un niveau d'intégrité supérieur.

   Ainsi, les règles relatives à la matrice de contrôle d'accès n'autorisent la modification du contenu d'un objet qu'aux sujets possédant un niveau d'intégrité suffisant. De plus, la communication entre sujets est prise en compte via la notion 'd'invocation de s2 sur s1' représentant un flux d'informations unidirectionnel de s1 vers s2. L'ensemble de permissions d'accès A se voit alors enrichi d'un élément i correspondant à l'invocation. Deux règles permettent alors d'assurer l'intégrité :
   — No Read Down : Un sujet n'est pas autorisé à accéder en lecture à un objet d'intégrité strictement inférieure, car cela pourrait corrompre sa propre intégrité ;
   — No Write Up : Un sujet n'est pas autorisé à altérer le contenu d'un objet d'intégrité strictement supérieure.

   Ces règles pouvant être écrites sous la forme :

   $r \in M[s, o] \Rightarrow f(s) \leq f(o)$
   $w \in M[s, o] \Rightarrow f(s) \geq f(o)$
   $i \in M[s_1, s_2] \Rightarrow f(s_1) \leq f(s_2)$

   Ces règles signifient que :

   (a) Pour qu'un sujet s ait accès en lecture à un objet o, son niveau d'intégrité f(s) doit être inférieur ou égal au niveau d'intégrité f(o) de l'objet ;

   (b) Pour qu'un sujet s ait accès en écriture à un objet o, son niveau d'intégrité doit être supérieur ou égal au niveau d'intégrité de l'objet ;

   (c) Pour qu'un sujet s1 puisse invoquer un sujet s2, son niveau d'intégrité doit être supérieur ou égal au niveau d'intégrité du sujet invoqué.

   Ces règles évitent à tout moment que ne se produise un transfert d'informations d'un niveau d'intégrité bas vers un niveau d'intégrité haut, ce qui signifierait une compromission de l'intégrité du niveau haut. La troisième règle découle du fait que si tous les canaux de flux d'informations apparaissent sous forme d'objets, alors une invocation de s2 par s1 s'apparente à une écriture par s1 dans un certain objet o suivie d'une lecture de o par s2. A l'instar du modèle BLP, le modèle Biba est difficile à utiliser tel quel dans un système d'exploitation. Ici l'idée est de migrer un sujet vers un niveau d'intégrité inférieur lorsqu'il accède à un objet de niveau inférieur. L'effet néfaste est que l'ensemble des sujets va rapidement se trouver en bas de l'échelle des niveaux d'intégrité. Dés lors, l'intérêt du modèle Biba est fortement remis en cause, puisqu'il n'y a plus de différence entre les sujets.

Finalement, le modèle Biba n'est qu'un modèle d'intégrité. Comme nous le verrons ensuite, il ne traite ni la propriété d'intégrité ni la confidentialité.

3. Modèle DTE : Le modèle Domain and Type Enforcement (DTE) [67] est un modèle de contrôle d'accès de haut niveau. Disponible depuis des années dans certains systèmes d'exploitation commerciaux [6], il se rapproche de l'utilisation du 'typage fort' disponible dans le modèle TAM et constitue une plate-forme sur laquelle peuvent être implantées des politiques de contrôle d'accès telles que Bell & La-Padula ou Biba. Concrètement, dans un système d'exploitation, les politiques de sécurité définies via le modèle DTE visent à :
   — Restreindre les ressources accessibles par un programme, notamment les programmes privilégiés (s'exécutant sous le compte root), suivant le principe de moindre privilège décrit dans [14] ;
   — Contrôler quels programmes ont accès aux ressources 'sensibles', et empêcher l'accès par tout autre programme à ces ressources (confinement d'application).

   Le modèle DTE reprend le principe de matrice de contrôle d'accès, mais la notion de sujet et objet est remplacée ici par la notion de domaine et de type. Ainsi chaque objet (fichier, message, mémoire partagée, socket, etc.) du système possède un type, et chaque processus s'exécute dans un domaine précis, dont découlent ses droits d'accès. Le modèle DTE définit ainsi trois tables :
   — La table de typage, qui assigne les types aux objets du système ;
   — La table de définition des domaines (DDT), qui spécifie les droits d'accès (lecture, écriture, exécution, ajout, suppression) de chaque domaine sur les différents types ;
   — La table d'interaction des domaines (DIT), qui définit les droits d'accès entre domaines (création, destruction, envoi de signal).

   La définition des trois tables en DTEL est effectuée par un administrateur, et les utilisateurs ne peuvent y déroger. C'est pourquoi DTE est un modèle de contrôle d'accès mandataire (MAC). Notons finalement que la définition de ces tables pour un système complet est une tâche difficile et fastidieuse qui nécessite des connaissances précises sur le fonctionnement de chaque application et une modélisation de bas niveau du système. Lorsqu'un nouveau processus est créé, il hérite automatiquement du domaine de son père. La DDT définit également les "points d'entrée" des domaines, c'est à dire quels sont les fichiers qui, une fois exécutés, vont créer un processus qui aura un domaine spécifique pouvant être différent du processus l'ayant invoqué. Dans la définition de ce modèle, il n'existe aucun autre moyen pour un processus d'opérer un changement automatique de domaine (une transition). La figure 2.2 est un exemple de politique DTE pour le serveur Web apache. Cette politique définit trois domaines distincts : un domaine administrateur_web_d consacré à l'administration de la configuration d'apache ; un domaine développeur_web_d destiné à l'écriture des pages Web ; et un domaine apache_d consacré à l'exécution du serveur. Cette politique permet de garantir d'information des fichiers de configuration et des pages Web en instaurant une séparation de privilèges. Ainsi chaque domaine est destiné à une tâche qui lui est propre, et ils ne peuvent interférer entre eux. En conséquence, même si apache s'exécute sous le

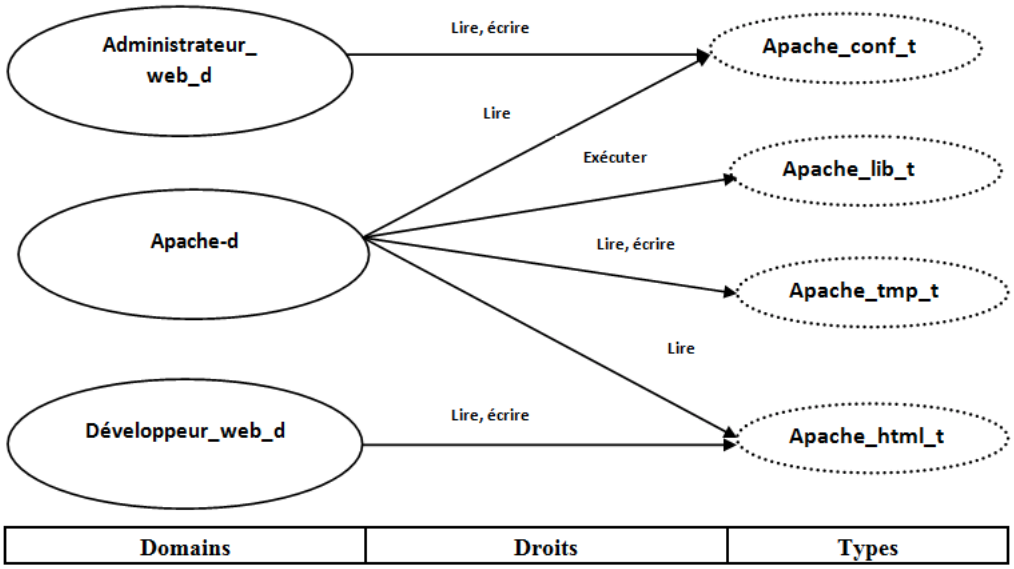

FIGURE 2.2 – Configuration DTE pour apache

compte administrateur root, il n'a accès qu'à ses fichiers de configuration, ses librairies et aux pages Web. De même seul apache a accès à son fichier de configuration, prévenant ainsi le système contre les attaques internes visant d'information de ce fichier. De plus, ce type de confinement permettra d'endiguer certaines attaques sur le processus apache ayant pour objectif d'obtenir les privilèges administrateur. Par exemple une attaque de type Buffer Overflow sur apache peut entraîner l'exécution d'un shellcode afin d'ouvrir un terminal sera bloquée par le mécanisme de contrôle d'accès. Notons que ce n'est pas l'attaque (le Buffer Overflow) qui sera détectée où bloquée mais la nécessité de droits supplémentaires, non nécessaires à apache et donc non présents dans la politique, afin de réussir la deuxième partie de l'attaque (ouverture d'un terminal). Mais ce confinement ne permettra pas d'empêcher une attaque ne nécessitant pas plus de droits que ceux accordés à un processus. Ainsi un vol d'information, par exemple dans une base de données ou un fichier accessible par apache, ne pourra être empêché.

4. Discussion : Les modèles Mandatory Access Control (MAC) ont été envisagés afin de répondre aux faiblesses des modèles DAC. On distingue deux orientations dans les modèles de type MAC. Une première orientation [21], [10], [19], [12] répond à des problématiques spécifiques, par exemple la nécessité de disposer d'une habilitation adéquate pour la lecture de documents classifiés dans BLP. Cependant, les systèmes d'exploitation relèvent de problématiques plus complexes. Une seconde orientation est le modèle DTE [67]. Plutôt que de considérer une propriété spécifique, DTE permet la spécification de politiques. Le problème de cette technique est qu'elle ne permet pas de spécifier les propriétés requises. Par ailleurs, nous ne traitons pas les modèles à base de rôle (RBAC pour Role-Based Access Control) car ils n'apportent rien en terme de propriété de sécurité. Ces modèles simplifient uniquement l'administration. Les modèles MAC fournissent des propriétés que l'on peut prouver. Par exemple, BLP repose sur deux lois qui empêchent la transmis-

sion directe d'information [21]. En revanche, les propriétés qu'ils permettent de garantir ne sont pas suffisamment larges. Il s'agit soit de garantir la confidentialité, soit l'intégrité. De plus, nous verrons que ces modèles ne traitent pas les séquences et donc ne garantissent pas les propriétés définies dans [25].

2.3 Déploiement automatique des Politiques de sécurité au sein des Firewalls

2.3.1 Préambule

Actuellement, les politiques de pare-feu (en anglais firewall) peuvent contenir de milliers de règles et ce à cause de la taille énorme et la structure complexe des réseaux modernes. De ce fait, ces politiques nécessitent des outils automatiques fournissant un environnement convivial pour spécifier, configurer et déployer en sûreté une politique cible. Beaucoup de travaux de recherche ont traité de la spécification des politiques, la détection des conflits et le problème d'optimisation, mais très peu de travaux se sont intéressés au déploiement de politiques. Dans ce chapitre, nous allons proposer des stratégies de déploiement pour les deux importantes catégories d'édition de politiques. Ensuite, nous fournissons deux algorithmes corrects, efficaces et sûrs pour le déploiement de politiques de sécurité.

2.3.2 Pare-feu et politique de sécurité

Un pare-feu est un dispositif de sécurité de périmètre qui filtre les paquets qui traversent à travers les frontières d'un réseau sécurisé. La décision de filtrage est basée sur une politique de pare-feu définie par l'administrateur du réseau. Une politique de pare-feu est une liste ordonnée de règles. Une règle r de pare-feu définit une action, généralement accepter ou rejeter, pour l'ensemble des paquets correspondant à ses critères. La majorité des pare-feux filtre le trafic en fonction de la sémantique de la première correspondance, lorsqu' un paquet p arrive, il est comparé avec des règles de haut en bas jusqu'à ce qu'une règle de correspondance est trouvée et le processus est répété pour le paquet suivant. Toutes les politiques ont une règle invisible par défaut « match-all " à la fin. Par conséquent, quand un paquet ne correspond pas à une règle de la politique, l'action par défaut est appliquée. Dans la plupart des pare-feux, la règle par défaut est « deny-all ", mais la règle par défaut « permit-all " est également possible. La majorité des pare-feux ne permettent pas de règles identiques dans la même politique. Par conséquent, nous supposons que cette restriction ne permet pas la duplication des règles dans une politique. Une règle est un ensemble de champs, chaque champ peut avoir une valeur atomique ou une plage de valeurs. Il est possible d'utiliser n'importe quel champ de l'entête IP, UDP et TCP. Toutefois, les cinq champs suivants sont les plus couramment utilisés : le type de protocole, l'adresse IP source, le port source, l'adresse IP de destination et le port destination. Tout le champ de l'en-tête de paquet peut être utilisé pour le processus de correspondance. Cependant, les cinq champs qu'on a cités auparavant sont les plus couramment utilisés. Dans un paquet, chacun de ces champs a une valeur atomique. Si tous les champs d'un paquet p coïncident avec les champs correspondants d'une règle r, alors p est accepté ou rejeté selon la décision de r. Si p ne correspond pas à aucune règle en matière de politique,

a. permit TCP 210.168.1.1 12.3.4.0/24 53 b. deny IP 101.1.1.0/24 any c. permit UDP 172.10.0.0/16 any 123 d. deny IP 101.1.2.0/24 76.54.32.1 e. permit IP 101.0.0.0/8 any	a. permit TCP 210.168.1.1 12.3.4.0/24 53 f. deny IP 101.1.1.1 any c. permit UDP 172.10.0.0/16 any 123 g. permit IP 101.0.0.0/16 any h. permit IP 101.1.0.0/16 any
Politique α	Politique β

TABLE 2.1 – Deux politiques de pare-feu

alors la valeur par défaut « match-all " de la règle est appliquée. Le tableau 4 donne un exemple de deux politiques de pare-feu basées sur le langage des pare-feux PIX.

2.3.3 Déploiement des politiques de sécurité

Le déploiement de la politique est le processus par lequel les commandes d'édition de politique sont émises sur les pare-feux afin que la politique cible devient la politique en cours d'exécution. Un administrateur réseau ou un outil de gestion ajoute des commandes au sein de pare-feu pour transformer la politique cible T en une politique en cours d'exécution R. L'ensemble des commandes qu'un pare-feu prend en charge est appelé son langage d'édition de politique. Généralement, un pare-feu utilise un sous-ensemble des commandes d'édition suivant [74] :

(app r) : ajoute la règle r à la fin de la politique R.
(del r) : supprime la règle r de la politique R.
(del i) : supprime la règle de la position i de la politique R.
(ins i r) : insère la règle r à la position i.
(mov i j) : déplace la ième règle vers la jème position dans R.

Les langages d'édition de politique peuvent être classés en deux catégories : Type I et Type II.

2.3.3.1 Déploiement de type I

Le langage d'édition type I prend en charge deux commandes uniquement, ajouter et supprimer (append and delete). La commande (app r) ajoute une règle r à la fin de la politique R qui est en cours d'exécution, sauf si r est déjà dans la politique R, dans ce cas, la commande échoue. La commande (del r) supprime la règle r de la politique R, si elle est présente. Le langage d'édition type I peut transformer toute politique en cours d'exécution en une politique cible [74], il est donc complet. La plupart des anciens pare-feux et certains qui sont récents, tels que FWSM 2.x [17] et JUNOSe 7.x [31], seuls supportent le langage d'édition type I. L'algorithme de base utilisé dans ce type de déploiement est le suivant :

```
Algorithm 1 Scanning Deployment
ScanningDeployment (I, T) {
 /* An algorithm using only app and del */
 /* to transform policy I into policy T */
 S ← empty stack
 H← empty hash table
 /* Phase 1: add rules */
 i ← 1
 for t ← 1 to SizeOf(T) do
 while i ≤ SizeOf(I) and I[i]<>T[t] do
 /* I[i] needs to be deleted */
 S.PUSH (I[i])
 H.ADD(I[i])
 i ← i + 1
 if i > SizeOf(I) then
 if H.Contains(T[t]) then
 H.Remove(T[t])
 IssueCommand( del T[t])
 IssueCommand( app T[t])
 /* Phase 2: clean up */
 for j ← SizeOf(I) down to i do
 IssueCommand( del I[j])
 while not S.IsEmpty() do
 r ← S.POP()
 if H.Contains(r) then
IssueCommand( del r)
 }
```

FIGURE 2.3 – L'algorithme 'Scanning Deployment'

2.3.3.2 *Déploiement de type II*

Le langage d'édition type II autorise la modification aléatoire de la politique de pare-feu. Il prend en charge trois opérations : la commande (ins i r) insère la règle r comme la ième règle dans la politique en cours d'exécution R, sauf si la règle r est déjà présente ; la commande (del i) supprime la ième règle de R et enfin la commande (mov i j) déplace la ième règle à la jème position dans R. Le langage d'édition type II peut transformer toute politique en cours d'exécution en une politique cible sans accepter des paquets illégaux ou rejeter paquets légaux [74], par conséquent, il est à la fois complet et sûr. Il est évident que pour un ensemble donné de politiques initiales et cibles, un déploiement de Type II utilise normalement moins de commandes d'édition que son équivalent utilisant le déploiement de type I. SunScreen 3,1 Lite [22] et Enterasys Matrix X [23] sont des exemples de pare-feux utilisant le langage d'édition type II. L'algorithme de base utilisé dans ce type de déploiement est le suivant :

```
Algorithm 2 Greedy 2-Phase Deployment
 TwoPhaseDeployment (I, T) {
 /* algorithm to calculate a safe type II deployment */
 /* to transform firewall policy I into T */
 /* Phase 1: insert and move */
 inserts ← 0
 for t ← 1 to SizeOf(T) do
 if T[t] ∉ I then
 IssueCommand(ins t T[t])
 inserts ← inserts + 1
 else
 IssueCommand( mov IndexOf(T[t] , I) + inserts t)
 /* Phase 2: backward delete */
 for i ← SizeOf(I) down to 1 do
 if I[i] ∉ T then
 IssueCommand( del i + inserts)
 }
```

FIGURE 2.4 – L'algorithme 'Greedy 2-Phase Deployment'

2.3.4 Défaillances de l'algorithme "Greedy-2-Phase Deployment"

Dans [64], deux algorithmes de déploiement de type II sont proposés. Le premier algorithme est un déploiement gourmand en deux phases appelé Greedy-2-Phase-Deployment (figure 2.4), tandis que le second algorithme est un algorithme plus efficace appelé SANITIZEIT. Dans cet article, nous nous intéressons au premier algorithme. Il est revendiqué dans [64] que Greedy-2-Phase-Deployment est correct et sûr. Cependant, on peut montrer qu'il n'est pas correct même pour des déploiements très simples. Considérons l'application de Greedy-2-Phase-Deployment à I et T donnée dans la figure 2.5 et la figure 2.6.

Case 1:

I	T		R
a	c		a
b	b	–TWOPHASEDEPLOYMENT–	a
c	a		c
			b

Proof:

1- t=1 ; indexof(T(t)=c,I)=3 ; mov(3,1) ; R0= c-a-b
2- t=2 ; indexof(T(t)=b,I)=2 ; mov(2,2) ; R1= c-a-a-b
3- t=3 ; indexof(T(t)=a,I)=1 ; mov(1,3) ; R2= a-a-c-b

FIGURE 2.5 – l'exécution de l'algorithme " Greedy-2-Phase-Deployment " pour le 1er cas

Case 2:

I = A-M-C-L-K-E
T = L-C-E-M-B-D-F-K
R = L-E-A-M-B-D-F-C-K

Proof:

1- t=1 ; indexof(T(t)=L,I)=4 ; mov(4,1) ; R0= L-A-M-C-K-E
2- t=2 ; indexof(T(t)=C,I)=3 ; mov(3,2) ; R1=L-M-A-C-K-E
3- t=3 ; indexof(T(t)=E,I)=6 ; mov(6,3) ; R2= L-M-E-A-C-K
4- t=4 ; indexof(T(t)=M,I)=2 ; mov(2,4) ; R3= L-E-A-M-C-K
5- t=5 ; T(5)=B ; ins (5,B) ;inserts=1 ; R4= L-E-A-M-B-C-K
6- t=6 ; T(6)=D ; ins (6,D) ;inserts=2 ; R5= L-E-A-M-B-D-C-K
7- t=7 ; T(7)=F ; ins (7,F) ;inserts=3 ; R6= L-E-A-M-B-D-F-C-K
8- t=8 ; indexof(T(t)=K,I)=8 ; mov(5+3=8,8) ; R7= L-E-A-M-B-D-F-C-K

FIGURE 2.6 – l'exécution de l'algorithme " Greedy-2-Phase-Deployment " pour le 2ème cas

Nous pouvons clairement observer que l'ordre des règles n'est pas respecté, celui-ci étant très primordial. Ainsi, on peut conclure que le déploiement ne répond pas au critère de sûreté. Lorsqu'on déplace une règle vers une position supérieure, çà entraîne un décalage des positions des autres règles. On obtient à la fin un résultat différent de celui de la politique cible T. Le déploiement n'est donc pas correct et ne correspond pas aux caractéristiques déjà mentionnées pour un déploiement efficient.

2.3.5 Contributions

Malheureusement, l'algorithme " Greedy 2-Phase Deployment " ne donne pas des résultats souhaitables. C'est pour cela que nous avons proposé un nouvel algorithme [72] appelé " Enhanced_Greedy_2_Phase_Deployment " qui est correct et qui nous a permis de déployer une politique cible d'une façon correcte.

Algorithm : ENHANCED-Greedy-2-Phase Deployment

```
1. ENHANCEDTwoPhaseDeployment (I, T) {
2. /* algorithm to calculate a safe type II depl
3. /* to transform firewall policy I into T */
4.
5. /* Phase 1: insert and move */
7. for t←1 to SizeOf(T) do
8. if T[t] ∉ I then
9. IssueCommand(ins t T[t])
11. else
12. IssueCommand( mov IndexOf(T[t] , I) t)
```

```
13.
14. /* Phase 2: backward delete */
15. for i←SizeOf(I) down to 1 do
16. if I[i] ∉T then
17. IssueCommand( del i )
18. }.
19.  /* Phase 3: Duplicate Finder */
20. for i←1 down to SizeOf(T)  do
21.  for j←i+1 down to SizeOf(T)  do
22.    if (T[i] eq T[j])and i <> j then
23.              del j
```

FIGURE 2.7 – l'algorithme " Enhanced-Greedy-2-Phase-Deployment "

Le nouvel algorithme inclut les modifications suivantes :

1. Suppression de la variable " inserts " : Initialement, elle aurait servi dans le positionnement des éléments à l'intérieur de " I ". Ce problème ne se pose pas puisque le " I " est évolutif et donc il prend en charge les nouveaux éléments ajoutés. Ceci se confirme ainsi par le fonctionnement des fonctions " indexof " et " del " qui travaillent sur des " I " intermédiaires.
2. L'introduction d'un algorithme de vérification de doublons qui permet de trouver et supprimer les règles répétées à l'instar de l'algorithme de type 1 " Scanning Deployment ". Cet apport valide le principe d'unicité des règles dans une politique de sécurité.

Les deux premiers exemples peuvent être réutilisés pour démontrer la véracité de la modification deux car ils présentent des insertions à la fin et puisque ce sont les derniers opérations, l'algorithme devrait normalement mener à un bon positionnement.

Case 1:
I = a-b-c
T = c-b-a
R = c-b-a

Proof:

1- t=1 ; indexof(T(t)=c,I)=3 ; mov(3,1) ; R0= c-a-b
2- t=2 ; indexof(T(t)=b,R0)=2 ; move(3,2) ; R1= c-b-a
3- t=3 ; indexof(T(t)=a,R1)=3 ; move(3,3) ; R2= c-b-a-a
4- Phase 3 : Duplicate Finder ; R3=c-b-a

FIGURE 2.8 – l'exécution de l'algorithme " Enhanced-Greedy-2-Phase-Deployment " pour le 1er cas

Case 2:

I = A-M-C-L-K-E
T = L-C-E-M-B-D-F-K
R = L-C-E-M-B-D-F-K

Proof:

1- t=1 ; indexof(T(t)=L,I)=4 ; mov(4,1) ; R0= L-A-M-C-K-E
2- t=2 ; indexof(T(t)=C,R0)=4 ; mov(4,2) ; R1=L-C-A-M-K-E
3- t=3 ; indexof(T(t)=E,R1)=8 ; mov(8,3) ; R2= L-C-E-A-M-K
4- t=4 ; indexof(T(t)=M,R2)=8 ; mov(5,4) ; R3= L-C-E-M-A-K
5- t=5 ; T(t)=B ins(B,5) ; R4= L-C-E-M-B-A-K
6- t=6; T(t)=D ins(D,6) ; R5= L-C-E-M-B-D-A-K
7- t=7; T(t)=F ins(F,7) ; R6= L-C-E-M-B-D-F-A-K
8- t=8; indexof(T(t)=K,R6)=8 ; mov(8,8) ; R7= L-C-E-M-B-D-F-A-K
9- Phase 2 : Backward Delete ; R8= L-C-E-M-B-D-F-K

FIGURE 2.9 – l'exécution de l'algorithme " Enhanced-Greedy-2-Phase-Deployment " pour le 2ème cas

Lorsque nous appliquons l'algorithme "Enhanced_greedy_2_phase_Deployement", il est clair que le résultat donné est correct. L'insertion et la modification se feront toujours dans l'ordre exact car même la modification de la position d'une règle sous-entend son insertion après dans la position "t" qui reflète la bonne position recherchée dans la cible.

Pour évaluer les performances de l'algorithme "Enhanced_greedy_2_phase_Deployement" par rapport aux anciens algorithmes, nous l'avons implémenté en JAVA et pour cela, nous avons utilisé quatre politiques de pare-feu dont les tailles sont 2000, 5000, 10000, et 25000 règles. Nous avons effectué trois tests différents pour chaque politique. En effet, les résultats de ce test sont représentés dans le tableau ci-dessous.

Taille / Test	Policy 1 (2000)		Policy 2 (5000)		Policy 3 (10.000)		Policy 4 (25.000)	
	EG2PD	SanitizeIT	EG2PD	SanitizeIT	EG2PD	SanitizeIT	EG2PD	SanitizeIT
Test 1	.0100	.012	.02507	.023	.02846	.044	.06825	.242
Test 2	.015	.012	.01338	.028	.02630	.049	.06653	.283
Test 3	.00732	.038	.02011	.049	.02097	.070	.03267	.325

Tableau 2.2 Résultats de l'expérience en secondes

Pour avoir une visibilité meilleure, nous avons affiché les résultats sous forme de graphique dans la figure ci-dessous.

Il s'est avéré que notre solution est plus performante par rapport à 'SanitizeIT' surtout pour les politiques de grande taille.

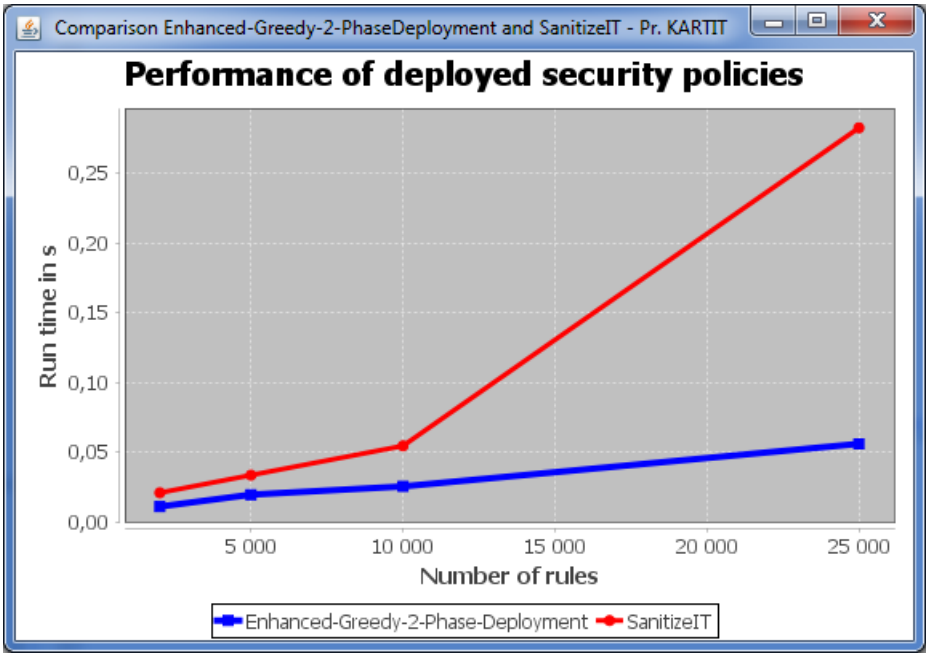

FIGURE 2.10 – Comparaison de l'algorithme Enhanced_Greedy_2_Phase_Deployment et SanitizeIT

2.4 Conclusion

La sécurité de déploiement des politiques de pare-feu est un nouveau domaine de la recherche. Dans ce chapitre, nous avons montré que les approches récentes [49] pour le déploiement de stratégies de pare-feu contiennent des erreurs critiques. En effet, ces approches peuvent présenter des failles de sécurité temporaires qui autorisent le trafic illégal et / ou interrompent des services de réseau en bloquant le trafic légal lors d'un déploiement. Nous avons proposé une formalisation pour le déploiement sûr et nous avons utilisé cette formalisation comme une base pour fournir un algorithme [72] sûr et efficace pour le langage d'édition type 2.

Chapitre

3

Load balancing et stockage sur le Cloud

Sommaire

3.1 Introduction

Dans le chapitre courant, nous allons introduire le concept de l'équilibrage de charge, présenter quelques travaux de recherche qui consistent en son utilisation dans le Cloud Computing et on va finir par présenter notre première contribution [36] qui fait usage également dudit concept pour le profit du Cloud Computing.

3.2 Équilibrage de charge (load balancing)

L'équilibrage de charge représente le fait de distribuer la charge entre plusieurs ressources dans un système distribué ou parallèle afin d'égaliser les charges de travail de manière efficace et d'améliorer le temps d'exécution d'une tâche [48], [50]. En fait, il évite une situation où certains noeuds sont fortement chargés tandis que d'autres noeuds ne font aucun travail. L'équilibrage de charge garantit que tous les processeurs du système ou de chaque noeud fournissent des efforts du travail, approximativement, égaux à tout instant. Ce concept ramène d'abord à décomposer le calcul global en tâches et assigner les tâches aux noeuds [69]. Les étapes de décomposition et d'affectation sont ainsi souvent

appelés partitionnement. L'objectif d'optimisation par le partitionnement est d'équilibrer la charge entre les noeuds et de minimiser les besoins de communication entre eux. L'exécution d'une tâche dans cet environnement distribué nécessite l'association entre processus et noeuds. Le nombre de ressources générées par l'étape de séparation peut ne pas être égal au nombre total de noeuds. Ainsi, un noeud peut être ralenti ou chargé avec de multiples emplois.

Comme l'a démontré avant et sur la base [4], [45], les objectifs d'équilibrage de charge sont :

- L'amélioration des performances de manière significative
- La redistribution des tâches sur les noeuds dans le cas où les derniers souffrent de surcharge, dysfonctionnement ou une panne
- Le maintien de la stabilité du système.
- Modularité et flexibilité : le système distribué, dans lequel l'algorithme est mis en oeuvre peut changer de taille ou de la topologie. Donc, l'algorithme doit être évolutif et suffisamment souple pour autoriser de telles modifications à être exécutées facilement.
- Priorité : priorisation des ressources ou des tâches qui doit être fait au préalable par l'algorithme lui-même pour un meilleur service aux tâches importantes ou prioritaires en dépit de la disponibilité d'un service égal pour toutes les tâches, indépendamment de leur origine.

3.2.1 Équilibrage de charge et le Cloud Computing

L'indice fondamental du Cloud Computing est de fournir des actifs tels que les machines virtuelles comme des services à la demande. L'affectation efficace de la machine virtuelle à la demande s'effectue à l'aide des algorithmes d'équilibrage de charge dans le Cloud Computing [40]. L'algorithme d'équilibrage de charge joue un rôle important lors de la détermination de la MV à attribuer à l'occasion de la demande de l'utilisateur.

Les fournisseurs du Cloud se basent sur les services d'équilibrage de charge automatique [56], qui permettent aux clients d'accroître le nombre de processeurs, mémoires ou disques durs pour leurs ressources. Ce service est implicite et dépend des exigences professionnelles des clients. Donc, le Load balancing fournit deux bases importantes, principalement à promouvoir la disponibilité des ressources et secondairement de maintenir une performance globale, ainsi qu'une économie d'énergie en cas de sous charge.

Un système d'équilibrage de charge est dit parfait s'il peut contourner la surcharge ou la sous charge de n'importe quel noeud. La sélection de l'algorithme d'équilibrage de charge n'est pas facile, car elle implique des restrictions supplémentaires tels que la sécurité, la fiabilité, le débit, etc. Par conséquent, l'objectif principal d'un algorithme d'équilibrage de charge dans un environnement de Cloud Computing est d'améliorer le temps de réponse en simplifiant l'interaction entre les noeuds, en choisissant la nature des tâches à transférer et en sélectionnant les noeuds possibles qui pourraient supporter la tâche ou le processus et de le déplacer en cas de défaillance de son noeud hôte.

3.2.2 Métriques du Load Balancing

L'objectif de l'équilibrage de charge est de répartir efficacement la charge du travail entre les ressources disponibles, afin de maximiser le rendement de ces ressources et de minimiser le temps de réponse et de traitement pour les demandes des clients. Il est fait de manière à rendre l'utilisation des ressources efficace, en évitant d'avoir certains noeuds surchargés tandis que d'autres ne le sont pas [58].

Pour évaluer la qualité d'une technique d'équilibrage de charge, architecture ou un système, de nombreuses métriques peuvent être utilisées. Certaines d'entre elles doivent être maximisées tandis que d'autres doivent être réduites au minimum, afin de disposer d'un système d'équilibrage de charge efficace. Les plus courantes sont décrites par Dash, M et al dans [16], comme suit :

- Le débit : est utilisé pour calculer le nombre de tâches dont l'exécution est terminée. Il doit être élevé afin d'améliorer la performance du système.
- Le coût général impliqué en mettant en oeuvre un algorithme d'équilibrage de charge : Il est composé des frais généraux en raison du mouvement de tâches, inter-processeur et de la communication interprocessus. Cela devrait être réduit au minimum afin que la technique d'équilibrage de charge puisse fonctionner efficacement.
- La tolérance aux pannes : c'est à dire la capacité d'un algorithme pour effectuer l'équilibrage de la charge uniforme en dépit de noeud arbitraire ou défaillance de la liaison. L'équilibrage de charge devrait être une bonne technique à tolérance de pannes.
- La durée de migration : est la durée pour migrer les tâches ou les ressources d'un noeud à un autre. Il devrait être minimisé afin d'améliorer les performances du système.
- Temps de réponse : est le temps nécessaire pour obtenir une réponse à partir d'un algorithme d'équilibrage de charge dans un système distribué. Cette métrique doit être minimisée.
- Utilisation des ressources : elle sert à vérifier l'utilisation des ressources et doit être optimisé pour un équilibrage de charge efficace.
- La performance : est utilisée pour vérifier l'efficacité du système et doit être améliorée à un coût raisonnable, par exemple, de réduire les temps de réponse des tâches en gardant les délais acceptables.

3.2.3 Algorithmes du Load Balancing

Afin de parvenir à une répartition équitable des ressources entre les tâches exigeantes et d'avoir une certaine performance dans l'ensemble du système, le Load balancing a recours à différents algorithmes d'ordonnancement. Puisque la recherche dans ce domaine est dans ses débuts, aucune technique ne peut être considérée comme la meilleure. Par conséquent, le choix de l'algorithme à utiliser dépend de nombreux facteurs, en particulier la taille du système Cloud, de la nature des demandes, la quantité des ressources disponibles ...

Ces algorithmes peuvent être classés de différentes façons. Ainsi, ils peuvent être classés en deux catégories : statique ou dynamique. Un algorithme d'équilibrage de charge statique ne tient pas compte de l'état antérieur ou le comportement d'un noeud lors de la répartition de la charge. D'autre part, l'algorithme d'équilibrage de charge dynamique vérifie l'état précédent d'un noeud lors de la répartition de la charge. L'algorithme d'équilibrage de charge dynamique est appliqué soit comme distribué ou non distribué.

Le Load balancing peut fonctionner de deux manières : coopératif et non coopératif. En mode coopération : les noeuds fonctionnent simultanément afin d'atteindre l'objectif commun pour optimiser le temps de réponse global. En mode non-coopératif : les tâches sont exécutées de manière indépendante dans le but d'améliorer le temps de réponse des tâches locales.

Les algorithmes les plus connus sont répertoriés par Mathew, T et al [64] dans le tableau 3.1. Comme le montre ce tableau, chaque algorithme a ses propres caractéristiques, les avantages et les inconvénients. Ainsi, un algorithme est choisi selon le contexte pour effectuer l'équilibrage de charge, tels que le type des demandes, si elles sont toutes de même nature ou non, ...

Méthodes d'ordonnancement	**Métriques considérées**	**Avantages**	**Inconvénients**
Premier arrivé premier servi	Temps d'arrivée	Simple en implementation	Ne considère pas d'autres critères pour l'ordonnancement
Round Robin	Temps d'arrivée, Temps quantique	Moins de complexité et la charge est équilibrée plus équitablement	La préemption est nécessaire
Opportuniste	Equilibrage de charge	Une meilleure utilisation de ressources	Durée totale d'exécution peu optimisée
Minimum Execution Time Algorithm	Temps d'exécution prévu	Choisit la machine la plus rapide pour l'ordonnancement	Charge non équilibrée
Minimum Completion Time Algorithm	Temps prévu pour l'achèvement, Equilibrage de charge	L'équilibrage de charge est pris en considération	La sélection de la meilleure ressource n'est pas l'ordre du jour
Min-Min, Max-Min	Global d'exécution, temps prévu pour l'achèvement	Une meilleure Durée globale d'exécution en comparaison avec d'autres algorithmes	Equilibrage de charge très pauvre et le facteur qualité de service non considéré
Algorithmes génétiques	Durée globale d'exécution, Efficience, Optimisation, Performance	Une meilleure efficience et performance en terme de durée globale d'exécution	Complexité avec un temps partiel d'exécution très long
Simulated Annealing	Durée globale d'exécution, Optimization	Trouve plus de solutions optimales dans un espace de solutions plus large, Meilleure durée globale d'exécution	Les facteurs qualité de service et hétérogénéité d'environnement peuvent être considérés
Algorithme de commutation	Durée globale d'exécution, Equilibrage de charge, Performance	Planification opérée selon la charge du système, meilleure durée globale d'exécution	Coût et temps d'exécution partiel sont dépendants de la charge

K-percent Best	Durée globale d'exécution, Performance	Choisit la meilleure machine pour l'ordonnancement	La ressource est choisit en se basant seulement sur le temps d'achèvement
Heuristique de vote	Temps d'achèvement minimal, Fiabilité	Une meilleure durée d'exécution avec un équilibrage de charge plus optimal	La ressource est choisit en se basant seulement sur la valeur du vote
Bénéfice meilleure, Puissance, Equilibrage de charge	La consommation de l'énergie, Coût, Equilibrage de charge	La consommation de l'énergie est réduite ainsi que le coût qui décroît même si plus de serveurs sont utilisés	Les autres facteurs de qualité de service et le temps d'achèvement des tâches sont moins considérés
Une méthode efficiente pour l'énergie en utilisant DVFS	Consommation d'énergie, Durée globale d'exécution, Temps d'exécution	Economie d'énergie pour chaque charge dans le système entraînant une meilleure durée globale d'exécution	Le coût et la complexité de l'implémentation peuvent être optimisés dans le futur
DENS	Trafic en équilibrage de charge, Congestion, Consommation d'énergie	La charge de communication est considérée et la consolidation du travail est effectuée afin d'économiser l'énergie	Les applications avec utilisations intenses des données sont les seules considérées par cette méthode où le besoin de traitement est médiocre
e-STAB	Efficience en énergie, Reconnaissance de réseau, QoS, performance	L'équilibrage de charge ainsi que l'efficience en énergie sont réalisés en se basant sur la charge du traffic et la congestion ainsi les retards sont évités	Les facteurs de qualité de service peuvent être considérés dans une approche globale de performance

Ordonnancement des tâches & Provisonnement des serveurs	Consommation d'énergie, Temps de réponse par tâche, Deadline	L'énergie est réduite rejoignant ainsi les deadlines des tâches	La durée globale d'exécution et le coût sont moins considérés
Algorithme de coût minimal	Coût de traitement, durée globale d'exécution	Le coût des ressources et la performance de calcul sont considérés avant l'ordonnancement	L'environnement dynamique du Cloud et d'autres attributs de qualité de service ne sont pas considérés
Algorithmes d'ordonnancement des tâches basés sur la priorité	Priorité des tâches, Temps prévu pour l'achèvement	La priorité est considérée pour l'ordonnancement. Il est conçu en se basant sur des critères de décision multiples pour établir le modèle	La durée globale d'exécution, L'uniformité et la complexité de la méthode proposée peuvent être améliorés
Algorithmes d'ordonnancement des tâches basés sur l'équilibrage horizontal des tâches	Tolérance aux pannes, Equilibrage de charge, Temps de réponse, Utilisation des ressources, Coût, Temps d'exécution	Affectation probabiliste basée sur le coût. Ressource et tâche les plus probables sont choisies pour l'affectation	L'algorithme ne mentionne en aucun cas comment le temps d'achèvement sera affecté
Algorithme Min-Min basé sur les priorités d'utilisateur	Priorité , Temps globale d'exécution, Exploitation des ressources, Equilibrage de charge	La priorité est accordée à certains utilisateurs sans que cela affecte le temps d'achèvement	Le réordonnancement des tâches au service de l'équilibrage de charge accroîtra la complexité et la durée
Ordonnancement WLC	Equilibrage de charge, Efficience, Vitesse de traitement	Une stratégie dynamique d'affectation de tâche est proposée. L'hétérogénéité des tâches est considérée	Considère uniquement la fonctionnalité d'équilibrage de charge
Ordonnancement DLT basé sur le coût et les facteurs QoS	Equilibrage de charge, Durée globale d'exécution, QoS, Performance, Coût	Un modèle d'optimisation DLT est conçu afin d'obtenir une meilleure performance globale	Pannes des machines, surchauffe dû aux communications et charges dynamiques ne sont pas considérées

Algorithme Max-Min amélioré	Durée globale d'exécution, Equilibrage de charge, Temps moyen d'exécution	Améliore la durée globale d'exécution et l'équilibrage de charge malgré les nombreux changements de tâches ou de la vitesse des processeurs	Les paramètres considérés sont limités et uniquement théoriques

TABLE 3.1 – Caractéristiques des algorithmes du Load balancing

Les métriques et les algorithmes ci-dessus ont pour but de relever le défi qui est de réaliser un partage de charge parfait dans le Cloud Computing. Ainsi, pour ce faire, les objectifs suivants doivent être atteints :

- L'amélioration de la performance globale, qui est l'objectif principal de l'équilibrage de charge dans son ensemble.
- Eviter la famine des tâches traitées, ceci peut être réalisé en améliorant le rendement et en minimisant la réponse ainsi que les durées de migration.
- Fiabilité, qui peut être obtenue par une approche à tolérance de panne.
- Les problèmes de sécurité, qui sont indispensables car les flux de données et de communication doivent être protégés de toute forme d'activités indésirables.

Pour réaliser ces objectifs, tout en prenant en considération les défis associés, nous avons proposé une approche multi-cluster. La section suivante détaille cette approche et décrit comment on traite les difficultés énoncées.

3.2.4 Approche proposée

Avec la propagation de l'utilisation du Cloud Computing et la demande croissante des services fournis par l'intermédiaire de cette nouvelle technologie, de nombreuses techniques et approches sont proposées par les chercheurs pour faire face à cette demande. La plupart des travaux ont tendance à travailler sur des algorithmes d'équilibrage de charge pour améliorer les performances et allouer les ressources disponibles le plus efficacement possible. Pourtant, en raison de l'hétérogénéité et les disparités entre les environnements et architectures, toutes les techniques laissent certains problèmes non résolus. Notre objectif est de résoudre certains de ces problèmes. Les problèmes que nous avons traités sont : Un Load balancing efficace entre les différents noeuds exécute les tâches du client de manière sécurisée, ainsi que l'élimination du possible point unique de défaillance et de la préservation de l'utilisation continue de toutes les ressources disponibles, en effet, aucune ressource ne doit rester inutilisée en raison d'un quelconque incident, comme l'échec de l'équilibreur de charge central entre autre , qui peut conduire à l'arrêt partiel ou total du système (noeuds de ressources). Ainsi, la méthode proposée [36] offre l'avantage d'un équilibrage de charge efficace. Elle assure la disponibilité continue du service en dépit de pannes dans le load balancer central. Elle sécurise également les flux de données entre les noeuds du système. La description détaillée de l'architecture ci-dessus est la suivante [9] :

Les éléments d'équilibrage de charge sont (Figure 3.1) :

— L'équilibreur de charge principal (MLB : Main Load Balancer) : Il est le noeud central
— Les clusters de ressource
— Un élément d'authentification, afin de garantir des flux de données sécurisée des flux entre les deux éléments précédents.

Le MLB est considéré comme le coordinateur du système, reçoit les demandes d'exécution des tâches des clients, applique un algorithme pour affecter à chaque demande de client un cluster approprié. Il a sa propre table de clusters, chaque cluster doté d'un poids représentant sa capacité moyenne de traitement, ainsi le principal équilibreur de charge peut choisir le bon cluster pour lui transférer une demande de client et la charge est partagée entre tous les clusters. Aucune file d'attente n'est nécessaire à ce niveau, parce que chaque cluster possède sa propre file d'attente locale.

Chaque cluster se compose d'un groupe de noeuds (ressources), il a son propre équilibreur de charge locale (LLB), ce dernier reçoit des demandes de la MLB et traite l'équilibrage de charge à l'intérieur de son groupe local. Il a une file d'attente où les demandes à venir sont stockées jusqu'à ce qu'un noeud à l'intérieur du groupe soit disponible pour les traiter. Les équilibreurs de charge locaux utilisent un algorithme d'ordonnancement pour effectuer l'équilibrage de charge, ils gardent aussi une table de ressource qui est constamment mis à jour si un changement survient dans les noeuds du cluster correspondant (par exemple, en cas de défaillance d'un noeud, l'ajout d'un nouveau noeud ...). En fonction des changements dans ce tableau, le poids du cluster est mis à jour dans la table MLB. Le tableau d'équilibrage de charge local contient également le poids de chaque noeud, d'où la charge locale est assez équilibrée entre les éléments du cluster.

Pour assurer la disponibilité du système et l'élimination des points de défaillance uniques, une autre fonctionnalité proposée de cette architecture consiste à faire de chaque LLB un candidat MLB, il remplacera le MLB dans le cas où il vient d'échouer. Dans ce cas, le premier LLB qui détecte la défaillance du MLB, devient le nouveau MLB. Il informe les autres de ce changement, recueille des données sur les autres LLBs, distribue ses noeuds à travers les autres clusters et commence à fonctionner en tant que nouveau MLB en envoyant les éléments dans sa file d'attente aux autres LLBs. Cela garantit la disponibilité continue du système et l'utilisation efficace des ressources.

La gestion d'une couche d'authentification de la sécurité est nécessaire, de sorte que seuls les utilisateurs authentifiés peuvent envoyer des tâches par le biais de notre architecture d'équilibrage de charge. Le serveur d'authentification devrait aussi attribuer les priorités des tâches jugées urgentes.

3.2.5 Conclusion

Dans ce chapitre nous avons présenté la notion d'équilibrage de charge qui permet de distribuer la charge entre plusieurs ressources dans un système distribué ou parallèle afin

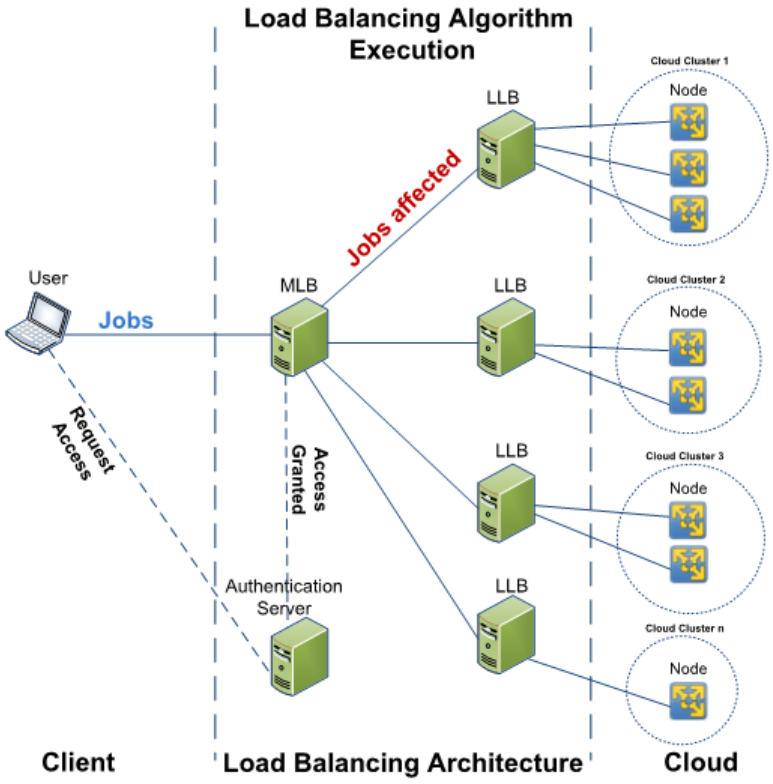

FIGURE 3.1 – Architecture Proposée pour le Load Balancing

d'égaliser les charges de travail de manière efficace et d'améliorer le temps d'exécution d'une tâche. Nous avons présenté aussi notre approche [36] permettant de réaliser un équilibrage de charge efficace en utilisant les clusters LLB et un MLB central.

3.3 Sécurisation des données dans le Cloud Storage

3.3.1 Introduction

Le Cloud Storage consiste à ne plus gérer ses données sur son propre dispositif (serveur, ordinateur, téléphone, . . .) mais à en corner la gestion à une entité tierce accessible par Internet. De nos jours, on entend souvent parler de l'avènement du Cloud avec les nouveaux produits technologiques qui sont connectés en permanence à Internet, comme les smartphones/tablettes, et avec des connexions réseaux de plus en plus performantes (3G/4G, ADSL/fibre/satellite . . .). Les utilisateurs peuvent ainsi accéder à leurs données de n'importe où et à n'importe quel moment. Ceci est d'autant plus intéressant lorsque le dispositif qu'il utilise dispose de peu de mémoire, comme par exemple un téléphone mobile. Bon nombre de systèmes proposent ce type de service. C'est le cas de DropBox, Amazon, google, HP, IBM, Microsoft, iCloud.

Dans ce chapitre, nous présentons notre contribution [37][73], basée sur les avantages des deux catégories de la Cryptographie : symétrique et asymétrique pour un stockage

sécurisé de données dans le Cloud.

3.3.2 Énoncé du problème et objectifs

La confidentialité des données des clients dans les environnements du Cloud Computing est la préoccupation essentielle, en raison du fait que les données des clients résident dans les serveurs distants répartis, gérés par des fournisseurs de Cloud potentiellement non fiables. Par conséquent, il y a des risques potentiels que les données confidentielles ou des renseignements personnels soient divulgués à des entités non autorisées. De toute évidence, afin de garantir la préservation de la vie privée, la confidentialité et l'intégrité deviennent essentiels, en veillant à ce que les données et le calcul soient gardés secrètes et inaltéré.

Les services de stockage de données dans le Cloud apportent de nombreux problèmes de conception complexes, considérablement en raison de la perte de contrôle physique. Ces défis ont une influence notable sur la sécurité des données et les performances des systèmes du Cloud. Autrement dit, les données du Cloud sont souvent soumises à un grand nombre de vecteurs d'attaque.

D'un côté, la préservation de la **confidentialité des données**, dans des environnements multi-tenants, devient plus difficile et conflictuelle. Ceci est largement dû au fait que les utilisateurs stockent leurs données dans des serveurs distants contrôlés par des fournisseurs de service (CSP) non fiables. Il est communément admis que le chiffrement des données, du côté client, est une bonne solution garantissant la confidentialité des données [62] et [63].

Ainsi, le client conserve les clés de déchiffrement, loin de la portée du fournisseur de Cloud. Néanmoins, cette approche suscite plusieurs problèmes de gestion de clés, tels que, le stockage et le maintien de la disponibilité des clés. En outre, la préservation de la confidentialité devient plus compliquée avec un partage de données dynamique entre un groupe d'utilisateurs. Premièrement, il faut un partage efficace des clés de déchiffrement entre les différents utilisateurs autorisés. Le défi consiste à définir un mécanisme efficace permettant la révocation d'un membre sans avoir besoin de mettre à jour tous les secrets des autres membres. Deuxièmement, les politiques de contrôle d'accès doivent être souples et permettent de distinguer des privilèges différents selon les utilisateurs pour accéder aux données. En effet, les données peuvent être partagées par différents utilisateurs ou groupes, et les utilisateurs peuvent appartenir à des différents groupes.

D'un autre côté, l'intégrité des données est considérée comme une préoccupation majeure, dans des environnements Cloud. Afin de renforcer la résilience, les fournisseurs de service ont généralement recours à la distribution des fragments de données selon des politiques de réplication. Néanmoins, afin de réduire les coûts d'exploitation et de libérer des capacités de stockage, les fournisseurs malhonnêtes pourraient volontairement négliger ces procédures de réplication, entraînant des erreurs irréversibles ou même des pertes de données.

Même lorsque les fournisseurs de Cloud mettent en oeuvre une politique de tolérance aux pannes, les clients n'ont pas les moyens techniques pour vérifier que leurs données ne sont pas vulnérables. Il pourrait y avoir des implémentations de vérification, à distance de l'intégrité et la possession des données externalisées, selon trois niveaux, comme suit :

- Entre un client et un CSP : un client devrait avoir un moyen efficace, lui permettant d'effectuer des vérifications périodiques d'intégrité à distance, sans garder les données en local. En outre, le client doit également détecter la violation du SLA, par rapport à la politique de stockage. La préoccupation de ce client est accentuée par ses capacités de stockage et de calcul limitées, et de la grande taille des données externalisées.
- Au sein du CSP : il est important pour un fournisseur de Cloud de vérifier l'intégrité des blocs de données stockés sur plusieurs noeuds de stockage, afin de réduire le risque de perte de données, dû à la défaillance du matériel.
- Entre deux CSPs : dans un environnement des Clouds imbriqués, où les données sont réparties sur de différentes infrastructures de Cloud, un CSP, à travers sa passerelle, doit périodiquement vérifier l'authenticité de blocs de données hébergées par une autre plate-forme de service.

Ces problèmes de sécurité sont d'autant plus importants, que les règlements à venir prévoient des mesures de protection plus sévères et rigides pour protéger efficacement les données à caractère personnel qui sont externalisées sur des serveurs distants.

3.3.3 Confidentialité des données stockées dans le Cloud

En confiant la tâche de stockage de données à une tierce partie, l'assurance de la confidentialité et la protection de la vie privée deviennent plus difficiles. La protection de la vie privée est une préoccupation majeure des utilisateurs du Cloud (figure 3.2), du fait que leurs données soient stockées dans des serveurs publiques distribués. Par conséquent, il y a un risque potentiel de divulgation des informations confidentielles (par exemple, les données financières, dossiers de santé, secret professionnel) ou des informations personnelles (par exemple, profil personnel). Quant à la confidentialité, elle implique que les données des clients doivent être gardées secrètes et ne doivent pas être divulguées à des personnes non autorisées, à savoir le fournisseur de service ou d'autres utilisateurs malveillants.

Dans cette section, nous nous concentrons sur la confidentialité des données, un enjeu important étant donné le besoin d'assurer un partage de données flexible au sein d'un groupe dynamique d'utilisateurs. Cet enjeu exige, par conséquence, un partage efficace des clés entre les membres du groupe.

Cependant, un des risques majeurs de ces technologies est la perte de confidentialité de leurs données puisqu'elles sont confiées à une entité tierce, qui leur est souvent inconnue.

L'informatique dématérialisée est divisée en deux catégories :

- Le calcul dématérialise ou le traitement des données est confié à l'entité tierce.
- Le stockage dématérialisé ou le stockage des données est assuré par cette même entité.

FIGURE 3.2 – Vue d'ensemble du stockage

Ces deux informatiques peuvent être compatibles entre elles, comme le propose, par exemple, le système Amazon Cloud Drive. Nous décidons cependant de nous focaliser sur le stockage dématérialisé et de concevoir une réalisation de celui-ci possédant certaines propriétés :

- Un utilisateur doit pouvoir partager ses données avec les utilisateurs en qui il a confiance.
- Un utilisateur non autorisé ne doit pas pouvoir accéder aux données, notamment le serveur qui est un utilisateur non autorisé.
- Un utilisateur stockant des données dans un serveur doit pouvoir gérer les droits d'accès comme il le souhaite.

Les préoccupations de sécurité des données stockées dans le Cloud ont émergé pour devenir d'un intérêt croissant d'une grande importance dans la communauté de recherche en cryptographie et informatique appliquée. En effet, Les serveurs du Cloud agissent comme des boîtes noires distribuées du point de vue du client. Dans ce chapitre, nous donnons un aperçu sur les fondamentaux de cryptographie, nous étudions l'utilisation de mécanismes cryptographiques dans des environnements Cloud et ensuite nous allons proposer une architecture et modèle de sécurité des données permettant de répondre aux difficultés citées ci-dessous afin d'établir un service efficace et digne de confiance. L'architecture d'un service de stockage dans un Cloud s'appuie sur les entités suivantes, permettant à un client de stocker, récupérer et partager des données avec plusieurs utilisateurs :

- Fournisseur de service Cloud (CSP) : un CSP dispose de ressources importantes pour contrôler les serveurs de stockage. Il fournit également une infrastructure virtuelle pour héberger des services d'application. Ces services peuvent être utilisés par le client pour gérer ses données stockées dans les serveurs Cloud.
- Client (C) : un client est le propriétaire de données. Il utilise les ressources du fournisseur de service pour stocker, récupérer et partager des données avec plusieurs utilisateurs.
- Utilisateurs (U) : les utilisateurs sont en mesure d'accéder au contenu stocké dans le nuage, en fonction de leurs droits d'accès s'agissant des autorisations accordées par

le client, comme les droits de lecture et d'écriture. Ces droits d'accès permettent de déterminer plusieurs groupes d'utilisateurs.

3.3.4 La contribution du stockage sécurisé de données

Pour protéger les données stockées sur des serveurs publiques de Cloud, Après s'être authentifié avec succès avec le CSP, le client démarre le processus de stockage. Notre protocole repose sur le chiffrement hybride, un mécanisme de chiffrement qui présente deux niveaux : le chiffrement symétrique des données et le chiffrement asymétrique de la clé :

- Chiffrement symétrique des données : avant de stocker les données sur les serveurs du Cloud, le déposant chiffre le contenu des fichiers, en utilisant un algorithme symétrique. Par conséquent, les capacités de stockage sont conservées puisque le même contenu sera chiffré de la même manière.
- Chiffrement asymétrique de la clé : le déposant chiffre la clé de déchiffrement, en se basant sur un algorithme asymétrique, tout en utilisant la clé publique du serveur des clés. En effet, n'importe quel destinataire autorisé peut accéder au serveur des clés, afin de récupérer la clé symétrique de données chiffrées.

La figure 3.3 illustre les différents échanges entre le client et son fournisseur de Cloud. Le scénario de stockage se base sur les étapes suivantes :

- L'authentification au niveau du Cloud et au niveau du serveur des clés.
- Chiffrement du fichier à l'aide du système de chiffrement symétrique (AES dans notre cas).
- Envoyer le fichier chiffré au Cloud.
- Chiffrer la clé AES avec RSA et l'envoyer au serveur des clés.

Le scénario de récupération des données dans le Cloud se base sur les étapes suivantes :

- L'authentification au niveau du Cloud et au niveau du serveur des clés.
- Récupération de la clé du serveur et la déchiffrer.
- Récupération du fichier du Cloud et le déchiffrer à l'aide de la clé récupérée précédemment.

3.3.4.1 Algorithme proposé

Notre idée consiste à utiliser la cryptographie pour sécuriser les données dans le Cloud. En fait, notre proposition bénéficie des avantages des deux systèmes de chiffrements : symétrique et asymétrique, à savoir la rapidité du système de chiffrement symétrique et la sécurité de la communication des clés du système de chiffrement asymétrique.

Algorithme de stockage des données : Notre algorithme a deux phases. Dans la première phase, l'algorithme encrypte le texte en clair avec le chiffrement AES. Dans la deuxième phase, nous chiffrons la clé AES en utilisant l'algorithme RSA-1024.

Nous avons utilisé un ensemble de fonctions suivantes :

- NumberOfBlock(F) : Retourne le nombre de bloc dans le fichier F.
- ENCAES (B,K) : Chiffre le bloc B par le système de chiffrement AES avec la clé K.

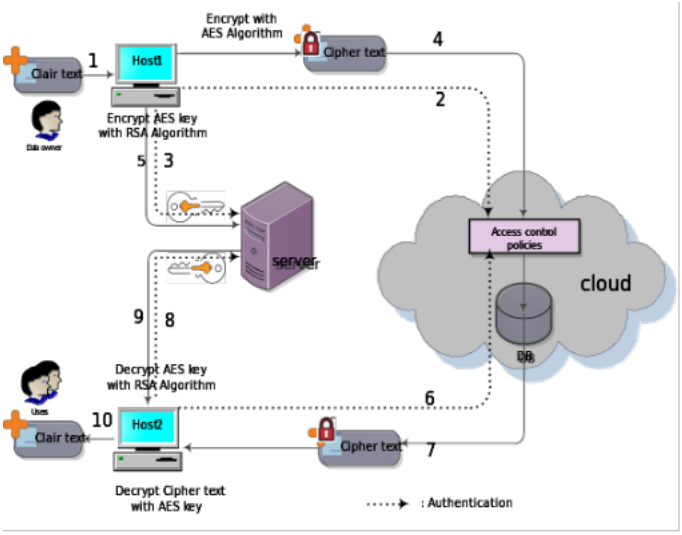

FIGURE 3.3 – Modèle Proposé de stockage de données dans le Cloud

— SendtoCloud(F') : Envoyer le fichier chiffré F dans le Cloud Storage.
— ENCRSA(k) : Chiffre la clé k en utilisant l'algorithme RSA.
— Saveinserver(K') : Sauvegarde la clé AES chiffrée K dans le serveur des clés.

Algorithm 1: FILEUPLOAD

```
1. Encryptfile (F) {
2. /* algorithm to encrypt file onto Cloud storage */
3. /* to transform Clair text in file F into Cipher text in file F' */
4.
5. /* Phase 1: Encrypt Clair text with AES Algorithm */
6.
7. For B ← 1 to numberOfBlock(F) do
8.    {
9.       B'=ENCAES(B,K)
10.   }
11.  sendtoCloud(F')
12. /* Phase 2: Encrypt AES key with RSA Algorithm */
13. For k ← 1 to SizeOf(K) do
14.   {
15.      k'=ENCRSA(k)
16.   }.
17.  Saveinserver(K')
18. }.
```

FIGURE 3.4 – Algorithme de stockage des données

Algorithme de récupération des données : Cet algorithme a également deux phases. Dans la première phase, l'algorithme déchiffre la clé AES en utilisant le RSA. Dans la deuxième phase, il déchiffre le texte chiffré en utilisant la clé AES récupérés à partir du serveur.

Notre algorithme utilise un ensemble de fonctions suivantes :
- NumberOfBlock(F) : Retourne le nombre de bloc dans le fichier F.
- DECRSA(k) : Déchiffre la clé k en utilisant l'algorithme RSA.
- DECAES(B,K) : Déchiffre le bloc B par le système AES avec la clé K.

Algorithm 2: FILEDOWNLOAD

```
1. Decryptfile (F') {
2. /* algorithm to decrypt file downloaded from Cloud storage */
3. /* to transform Cipher text in file F' into Clair text in file F */
4.
5. /* Phase 1: Decrypt AES Key with RSA Algorithm */
6.
7. For k' ← 1 to SizeOf(K') do
8.       {
9.           k=DECRSA(k')
10.       }
11.  return(K)
12.
13. /* Phase 2: Decrypt Cipher text with AES Algorithm */
14. For B' ← 1 to numberOfBlock(F') do
15.      {
16.          B=DECAES(B',K)
17.       }.
18.   return(F)
19. }.
```

FIGURE 3.5 – Algorithme de récupération des données

Implémentation et analyse des résultats : La mise en oeuvre de notre approche met en évidence le temps d'exécution en upload et en download des fichiers avec des tailles différentes et avec différentes tailles de la clé AES. Notre application est développée en Java 7. Le résultat ci-dessous (Tableau 3.2, Figure 3.6, Tableau 3.3 et Figure 3.7) est obtenu en utilisant un hp PC Compaq dc 5800 avec les spécifications suivantes : Intel (R) Core (TM) 2 Duo CPU E6550 @ 2,33 GHz (2 processeurs), avec 3072 Mo de RAM. Le temps de récupération d'un fichier est plus grand du temps de stockage, ceci est expliqué par l'addition du temps de récupération de la clé sur le serveur [3] [5] [7].

File size (kb)	128	256	512	1024	2048
Upload time (s)	0.2025	0.4051	0.8118	1.6201	3.2903
Donwload time (s)	0.4225	0.8452	1.6905	3.3802	6.7711

TABLE 3.2 – Temps d'upload par taille du fichier en upload et en download

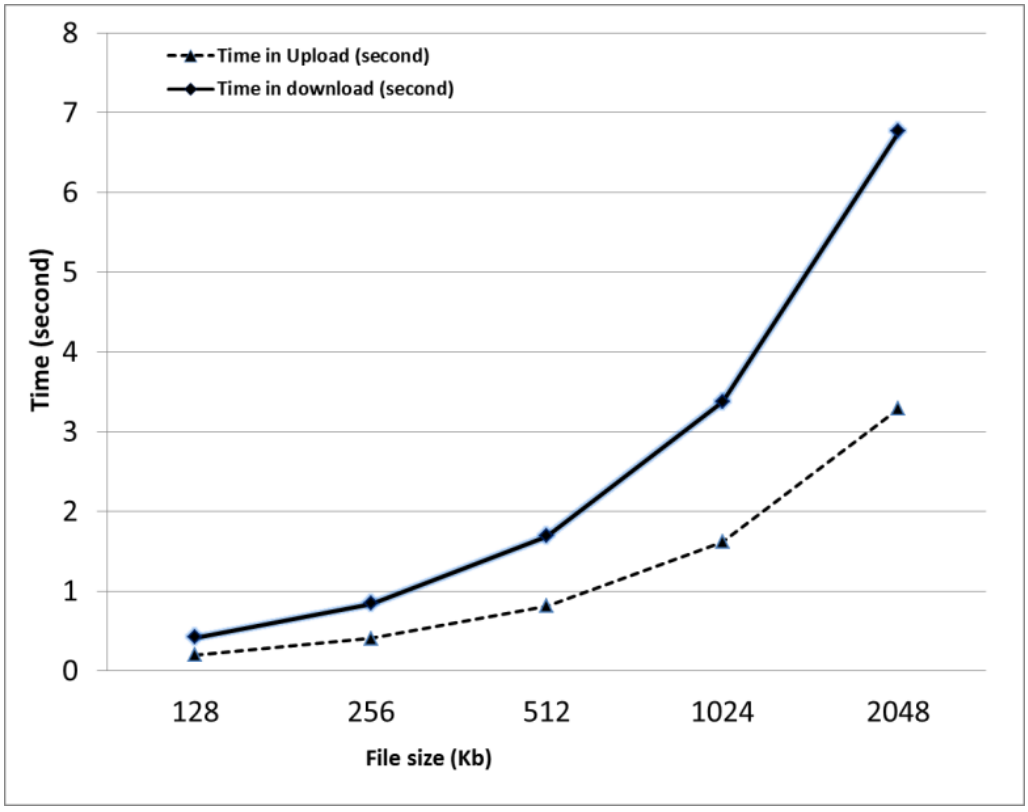

FIGURE 3.6 – Graphe du temps de l'exécution avec différentes tailles du fichier

Key size / File size (kb)	128	256	512	1024	2048
128	0.198	0.397	0.794	1.587	3.1744
192	0.224	0.448	0.896	1.792	3.584
256	0.25	0.499	0.998	1.997	3.9936

TABLE 3.3 – Temps d'exécution par taille du fichier et taille de la clé AES

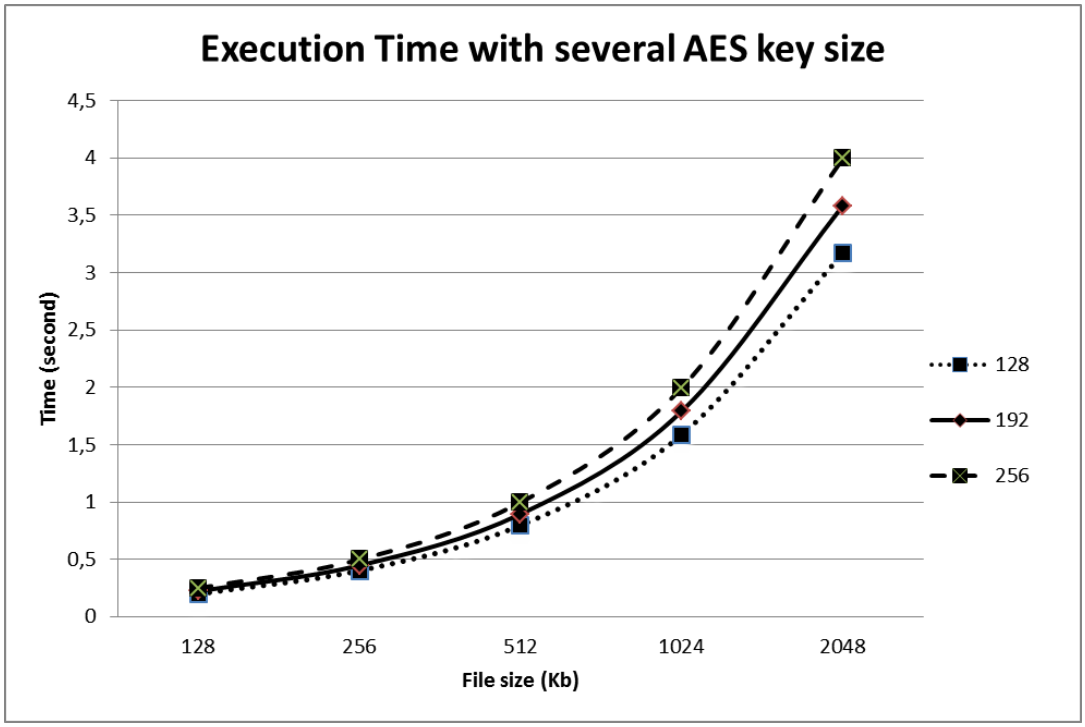

FIGURE 3.7 – Graphe du temps de l'exécution avec différentes tailles de la clé AES

La figure 3.7 et le tableau 3.3 montrent le temps d'exécution requis par différents fichiers de différentes tailles avec différentes tailles des clés AES pour le processus du chiffrement. Notre approche présente les avantages et les points forts suivants :

- Les données sont cryptées de bout en bout c'est-à-dire de la machine source jusqu'à la machine de destination et la clé de déchiffrement n'est pas communiqué dans le Cloud.
- Algorithme AES utilisé est un algorithme symétrique rapide sûr et est l'un des algorithmes de chiffrement les plus sûrs. Il n'a pas été brisé à ce jour. Cela signifie que notre algorithme est rapide dans les deux sens : upload et download.
- Possibilité de changer la clé symétrique fréquemment pour améliorer la sécurité.
- La clé AES utilisé pour le chiffrement des données est cryptée par un algorithme RSA-1024 robuste et n'a jamais été rompu.
- Le décryptage des données nécessite une double authentification. L'utilisateur doit disposer des droits d'accès au serveur des clés et au Cloud.

3.3.5 Conclusion

Nous avons présenté dans ce travail une contribution [37][73] pour pallier à deux besoins de sécurité dans des environnements de stockage en nuage, à savoir la confidentialité des données. Notre objectif consiste à définir de nouvelles méthodes pour améliorer la confidentialité des données dans des applications Cloud, tout en améliorant le partage dynamique entre les utilisateurs. En réponse à cet objectif, nous avons proposé une approche basée sur l'utilisation de la cryptographie hybride. Les résultats expérimentaux ont montré l'efficacité de notre solution en termes de sécurité et de performance. Nos perspectives de recherche consistent à définir de nouvelles méthodes pour améliorer l'intégrité des données dans des applications Cloud.

3.4 Conclusion

Grâce à ces deux contributions [36][73], la disponibilité et la sécurité des données stocké dans le Cloud sera renforcé. Nous allons présenter dans le chapitre suivant notre deuxième contribution qui consiste en un IDS comportemental.

Chapitre 4

CKMSA : Une détection d'intrusion comportementale basé sur les algorithmes d'optimisation K-means et Simulated Annealing

Sommaire

4.1 Introduction

Comme on l'a déjà abordé, les systèmes de détection d'intrusions comportementaux commencent par l'établissement d'un profil normal du système surveillé et par la suite la détection de toute variation de ce profil [71][13]. L'établissement d'un profil normal est une phase très sensible, si elle se déroule sur des données qui ne sont pas conformes, il sera très probable que toute détection par la suite soit un faux positif et bien sûr, il y aura un taux de faux négatifs bien élevé. D'où, le besoin pressant d'un algorithme de classification qui permet de classifier les données d'une manière bien optimale.

Dans cette section, nous présentons notre deuxième contribution qui consiste en une combinaison de deux méthodes de classification et d'optimisation : *K-means* et *Simulated Annealing* (qui est connu comme une méthode probabiliste [40][57]). Ce dernier en général, fonctionne en émulant le processus physique par lequel un solide est lentement refroidi de sorte que lorsque, finalement, sa structure est gelée, cela conduit à une configuration minimale d'énergie [9]. Pour nous, elle nous permettra d'aboutir à une classification optimale des données qui font l'objet des événements à classifier dans la phase de l'établissement d'un profil normal. Dans ce travail, c'est la variante semi-supervisé de la méthode K-means qui est utilisée. Cela permet de réduire le nombre de fois pour lesquels l'algorithme est utilisé et en conséquence permettre à notre travail d'être utilisé en temps réel.

Nous avons testé une application de notre contribution en utilisant les données de NSL-KDD [1]. Les résultats ont apporté une bonne satisfaction. Les tests nous permettent de dire que notre méthode améliore les taux de détection et de non détection des IDS.

4.2 Travaux connexes

La *Data Mining* est un ensemble de techniques et de méthodes qui permet de traiter les données en masse dans les domaines de statistiques, mathématiques et informatique. Ces techniques tentent d'extraire, identifier et valider de possibles patterns compréhensibles dans les données traitées. Parmi ces techniques, beaucoup ont été proposées et utilisées pour la détection des intrusions telles que : la *classification tree* et la *support vector machines* [43] [70], les algorithmes génétiques et les réseaux de neurones [47].

Quand ces techniques sont utilisées dans le contexte de la sécurité informatique, un niveau de sûreté important est additionné que ce soit contre les attaques externes ou bien les attaques internes ou encore les nouvelles attaques. Les techniques de classification sont les plus préférées puisqu'elles ne requièrent pas un labeling manuel des données d'apprentissage et le système n'a pas besoin de connaître les nouvelles attaques. Ces techniques sont très populaire dans la détection des intrusions et sont devenues l'objet de beaucoup de travaux de recherche. K-means est la méthode la plus connue pour l'analyse et la classification des données. D'après [20], K-means est un algorithme relativement rapide ; donc, il ne nécessite pas un grand temps de calcul. Selon la comparaison des méthodes K-means, Y-means et Fuzzy C-means [20], K-means est la méthode qui garantit le meilleur taux de détection des intrusions avec le taux le plus faible des faux positifs. Malgré les avantages de l'algorithme K-means, ce dernier reste bloqué dans un optimum local lors de la recherche d'une solution optimale. Raison pour laquelle, nous essayons de surpasser ce problème par une approche qui combine entre K-means et Simulated Annealing. Cela permettra d'aller au delà de l'optimum local pour aboutir à un optimum global.

4.3 K-means

K-means est un algorithme d'apprentissage non supervisé qui permet de résoudre les problèmes de la classification. Avant l'exécution de l'algorithme, on fixe un nombre K qui représente le nombre de clusters dans lesquels un ensemble de données seront classifiées. Alors, l'algorithme commence par la définition de K points depuis l'ensemble des données comme centroïdes initiaux, un par cluster. Par la suite, l'algorithme détermine pour chaque point le centroid le plus proche et attribue ce point au cluster correspondant définit. À la deuxième étape, K-means recalcule, pour chaque cluster, un nouveau centroïde définit comme la moyenne des distances entre le centroïde actuel et ses points associés. Une fois les nouveaux centroïdes sont calculés, l'algorithme replace tous les points de l'ensemble de données dans le cluster le plus proche correspondant au plus proche centroïde. Cette dernière étape est refaite assez de fois jusqu'à ce qu'il n'y est plus de changement au niveau de tous les centroïdes. Cela veut dire que tous les points de l'ensemble de données ne changent plus de cluster.

L'objectif de cet algorithme est de minimiser une fonction objective. Dans ce cas, une fonction d'erreur quadratique :

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

sachant que $||x_i^{(j)} - c_j||^2$ est une mesure de distance choisie entre un point de données $x_i^{(j)}$ et le centre du cluster (centroïde) c_j. Ça représente également un indicateur de distance entre les points et leurs centres de clusters respectives.

La méthode K-means se déroule comme suit [49] :

Input :

Un nombre de clusters K et une base de donnée $D = d_1, d_2, ...d_n$ contenant n objets de données.

Output :

Un ensemble de K clusters

Étapes :

— Choisir arbitrairement K objets depuis la base de données D comme centres initiaux de clusters.
— Répéter
— Calculer la distance entre chaque objet $d_i(1 <= i <= n)$ et tous les K centres de clusters $c_j(1 <= j <= k)$ et affecter l'objet d_i au clusters la plus proche.
— Pour chaque cluster j $(1 <= j <= k)$, recalculer le centre du cluster.
— Tant que les changements dans les centres des clusters sont détectés.

4.4 Simulated Annealing SA

La Simulated Annealing est une méthode métaheuristique inspirée du domaine de la physique, et plus précisément du processus de recuit dans le processus du métal. Celui-ci implique le chauffage et le refroidissement d'un métal de manière graduelle qui permet d'altérer ses propriétés physiques pour rendre fixe la structure métallique et de stabiliser ses nouvelles propriétés. Le processus commence par une température élevée, qui est progressivement refroidie de manière contrôlée jusqu'à ce que la température cible soit atteinte.

L'analogie qui conduit à l'approche SA est basée sur la simulation de ce processus de recuit dans le monde mathématique, plus précisément dans le domaine d'optimisation. Son but est de surmonter la limitation des algorithmes d'optimisation qui peuvent se coincer dans l'optima local, en permettant une exploration plus large de l'espace de recherche.

Ainsi, une variable T similaire à la température est utilisée, elle est initialement réglée à une valeur à minimiser T_0 élevée qui décroit selon une certaine vitesse jusqu'à la fin du processus SA.

1	Initializing :
	Initial solution S_i
	Initial and final temperature : T_i and T_f
	$T = T_i$
2	Temperatures cycle :
3	Metropolis cycle :
	Generating S_j from S_i
	$dif = J(S_j) - J(S_i)$
	If $dif < 0$ then
	$S_i = S_j$
	else if $e - dif > rnd(0, 1)$ then
	$S_i = S_j$
4	Metropolis condition :
	If thermal equilibrium is reached
	goto 5
	Else
	goto 3
5	Stop criterion :
	If the final temperature T_f is reached
	End
	Else
	Update T
	goto 2

TABLE 4.1 – SIMULATED ANNEALING ALGORITHM

En outre, on fixe une solution initiale S_0, une fonction objective f_0, un taux de décroissance pour la variation température p et la température finale T_f, ainsi que la limite de Markov B. Dans les températures élevées, la méthode accepte de meilleures solutions et autorise également de prendre des solutions moins bonnes avec une certaine probabilité, à ce stade la condition de Metropolis est utilisée pour évaluer les nouvelles solutions candidates. Comme la température diminue, les chances d'accepter des moins bonnes solutions se réduisent et nous avons tendance à accepter de meilleures solutions.

Ce comportement permet d'explorer plus l'espace de solutions et par conséquent d'éviter de se coincer dans un optimum local, donc SA a la capacité de trouver une solution très proche de l'optimum global.

L'algorithme de la table 4.1 comporte le SA dans sa version classique. Dans l'algorithme, on peut voir le cycle de températures entre les étapes 2 et 5. Dans ce cycle de température, les étapes 3 et 4 correspondent à l'algorithme de Metropolis [49].

4.5 Processus proposé de détection par anomalie basé sur K-means et SA

Dans cette section, nous présentons notre approche basée sur deux méthodes combinées, à savoir les K-Means et la Simulated Annealing. Ensuite, nous projetons l'algorithme sur un IDS comportemental.

4.5.1 CKMSA Statement

Dans l'approche proposée [29], nous commençons par appliquer l'algorithme K-means qui va effectuer le clustering de l'ensemble de données afin de parceller ses éléments sur les k clusters.

Au début, chaque cluster a un centroïde initialisé par un objet étiqueté du DataSet. K-means calcule alors la distance entre chaque objet et tous les centroïdes, puis place les objets dans le cluster approprié, correspondant au centroïde le plus proche de l'objet en question. Dans le reste des itérations du k-means, il recalcule les centroïdes avec la moyenne des dimensions numériques des objets et chaque fois que ces objets seront remplacés autour des nouveaux centroïdes.

L'algorithme s'arrête lorsque la fonction objective converge, c'est-à-dire que les centroïdes ne changent plus. Dans notre approche, lors de l'initialisation k représente le nombre de catégories (n) constituant le problème : $k = n + 1$; Une nouvelle catégorie est ajoutée pour attirer des objets anormaux par rapport aux catégories initiales. Elle sera initialisée par un centroïde qui symbolise le point le plus éloigné du centre fictif de toutes les données représentées par le centre des centroïdes initiaux.

Après son exécution, K-means engendre une solution potentiellement optimale localement en raison de sa caractéristique de descente de gradient. Afin de surmonter ce problème, nous utilisons SA pour améliorer la solution trouvée précédemment et donc pour assurer le saut à une autre configuration plus optimale.

La métaheuristique SA s'exécute comme décrit ci-dessous de l'étape 3 à l'étape 8. Voici la description des étapes de l'algorithme :

- **Étape 1** : Initialiser le nombre de clusters k par $n + 1$ où n est le nombre de catégories qui composent le problème : $k = n + 1$.
- **Étape 2** : Initialiser l'algorithme K-means par une solution de départ S_0 et la fonction objective f_0, $f_0 =$ somme des distances euclidiennes entre chaque centroïde et ses objets.
- **Étape 3** : Définir la plage de températures SA et sa vitesse de décrémentation (T_0, T_f et r) ainsi que B : le seuil au-dessus duquel la longueur de Markov d'une solution n'est plus acceptée (Longueur de Markov d'une solution : Le nombre de changements apportés à une solution S_i pour obtenir une solution S_{i+1}).
- **Étape 4** : Pour une température donnée T_i, effectuer un changement d'affectations de cluster pour une donnée choisie au hasard par SA, la solution résultante n'est pas acceptée si son nombre de changements C dépasse le seuil B.

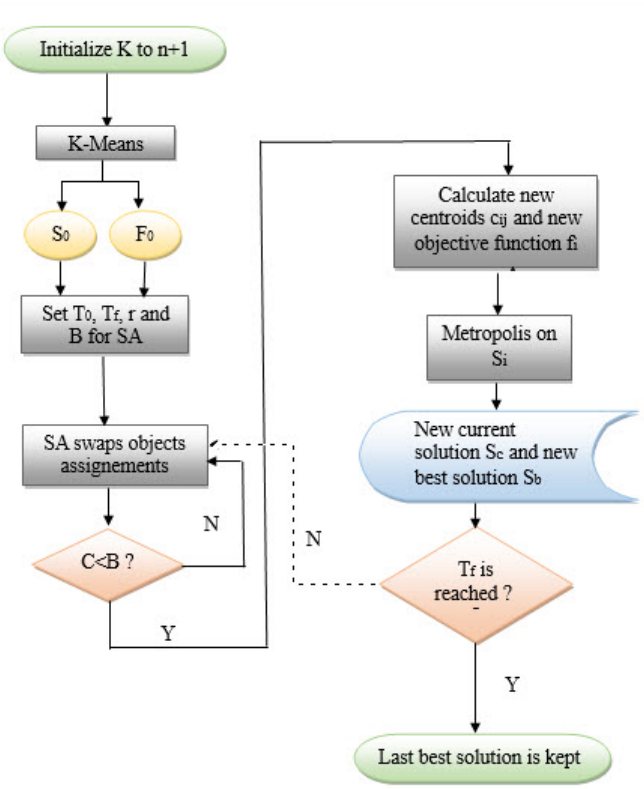

FIGURE 4.1 – L'organigramme de la CKMSA

— **Étape 5** : Déterminer les nouveaux centroïdes c_{ij} et recalculer la fonction objective f_i.
— **Étape 6** : Évaluer la nouvelle solution par la condition de Metropolis afin de la conserver comme la solution candidate, ou de l'accepter ou non avec une probabilité ($P = (exp^{\frac{(-(f_i - f_{i-1}))}{Ti}}$).
— **Étape 7** : Assigner la solution candidate comme la meilleure solution au cas où elle est plus optimale.
— **Étape 8** : La SA est réutilisée avec une nouvelle température (nous revenons à l'étape 4, $Ti + 1 = Ti - r$) et l'algorithme est appliqué de nouveau de l'étape 4.
— **Étape 9** : L'algorithme s'arrête à T_f et nous gardons la meilleure solution trouvée.

4.5.2 Projection sur les IDS comportementaux

Notre contribution [29], lorsqu'elle est projetée sur le contexte IDS, peut donner lieu à des catégories bien définies pour les données d'entrée, à savoir les objets normaux et les attaques. De plus, la $(K+1)^{\text{ème}}$ catégorie référencée dans notre approche aidera à détecter de nouvelles attaques, par conséquent le taux de détection IDS sera meilleur. En outre, le clustering effectué à l'aide de K-means suivie par SA conduira à une catégorisation plus optimale des objets, ce qui permettra d'éviter de possibles détections erronées.

Symbolic (3)	Protocol_type, service, flag
Discrete (6)	Land, logged_in, root_shell, su_attempted, is_host_login, is_guest_login
Continue (32)	Duration, src_bytes, dst_bytes, wrong_fragment, urgent, hot, num_failed_logins, num_compromised, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same srv rate di_ srv rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

TABLE 4.2 – Types de dimensions

4.6 Expérimentations et résultats

Afin d'évaluer la méthode suggérée [29], nous avons opté pour l'utilisation de la DataSet NSL-KDD. Cette dernière incarne l'évolution de la DataSet KDDCUP99 qui a été prouvée être engloutit de plusieurs pépins [68].

NSL-KDD a donc été créé pour surmonter ces problèmes, elle consiste en des enregistrements sélectionnés de l'entière KDD. Les avantages du DataSet NSD KDD par rapport au DataSet KDD original [1] sont les suivants : Tout d'abord, pas d'enregistrements en double dans la DataSet d'apprentissage, de sorte que le classificateur n'engendre aucun résultat faux. Deuxièmement, aucun enregistrement identique dans la DataSet de test qui a un meilleur taux de réduction et le nombre d'enregistrements sélectionnés de chaque groupe de niveau de difficulté est inversement proportionnel au pourcentage d'enregistrements dans la DataSet KDD d'origine.

Par conséquent, les taux de classification de différentes méthodes d'apprentissage varient dans un intervalle plus large, ce qui permet d'obtenir une estimation correcte des différentes techniques d'apprentissage. Troisièmement, le nombre d'enregistrements dans les DataSets d'apprentissage et de tests est rationnel, ce qui rend abordable l'exécution des expériences sur le DataSet complet sans la nécessité de sélectionner au hasard une partie moindre. Par conséquent, les résultats d'évaluation des différents travaux de recherche seront cohérents et comparables.

L'ensemble NSL KDD a un actif de 41 dimensions [46] résultant de chaque paquet en plus de chaque enregistrement étiqueté comme normal ou type d'attaque défini. Ces caractéristiques peuvent être continues, discrètes et symboliques comme le montre le tableau 4.6.

La dimension d'étiquette non mentionnée dans le tableau ci-dessus se situe dans 5 classes qui sont normales et 4 types d'attaques [15] Dos, Probe, R2L et U2R. En fait, la DataSet d'apprentissage comprend 22 attaques différentes parmi les 37 présentes dans la DataSet de test. Le tableau 4.3 montre les attaques appartenant à chaque classe avec une

Probe	Contourne un système de sécurité en collectant préalablement les données.	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint
DOS	Occupé à répondre à des requêtes illégitimes interdisant ainsi les utilisateurs authentiques.	Back, Land, Neptune, Pod, Smurf, Teardrop, Mail bomb, Processtable,Udpstorm,Apache2,Worm
R2L	Obtention d'un accès local à distance	Guess password, Ftp write, Imap, Phf, Multihop,Warezmaster, WarezClient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Accès à un compte d'utilisateur normal du système afin d'exploiter les vulnérabilités et gagner ainsi un accès root.	Bufferoverow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

TABLE 4.3 – Classes d'attaques NSL KDD

brève description.

4.6.1 Résultats et comparaison des performances

Dans notre méthodologie expérimentale, nous avons pris en compte que le $42^{\text{ème}}$ champ peut être généralisé comme Normal, DoS, Sondage, U2R et R2L. Les types d'attaques appartenant à chaque classe sont listés dans le tableau 4.3 et la répartition des données de test dans les DataTest d'apprentissage et de test est montrée à la figure 4.2 et la figure 4.3. Les performances de K-Means et de CKMSA (K-Means combiné à Simulated Annealing) sont comparées en fonction du Détection Rate, False Alarm et False Positive Rate [63] en utilisant les expressions suivantes :

- Accuracy = (TP + TN)/ (TP + TN + FP + FN)
- DetectionRate = (TP)/(TP + FN)
- FalseAlarm = (FP)/(FP + TN)

Sachant que :

- FN is False Negative
- TN is True Negative
- TP is True Positive
- FP is False Positive

Les tableaux 4.6.1 et 4.6.1 montrent les résultats de l'application des algorithmes K-means et CKMSA, regroupés par catégories de données. K-means est moins performant que CKMSA parce qu'il génère 18659 attaques qui étaient censées être normales. Il considère également un nombre assez important de données comme normal, alors qu'elles devraient être dans les catégories des attaques.

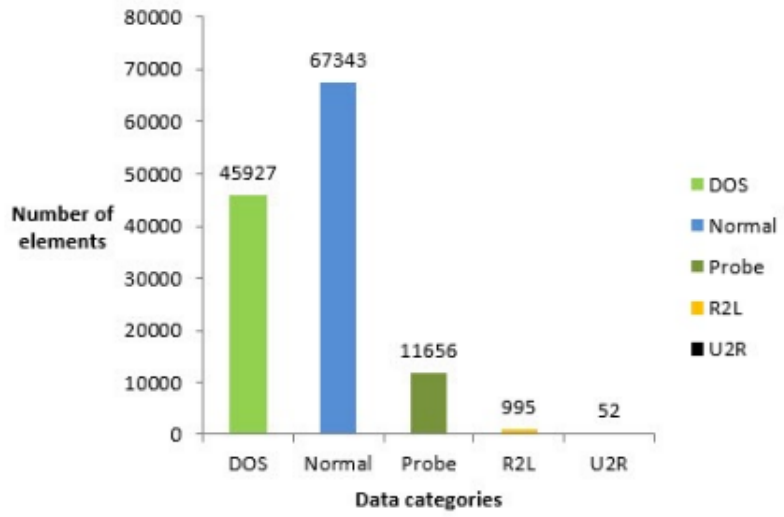

FIGURE 4.2 – Répartition des données d'apprentissage par catégorie d'attaque

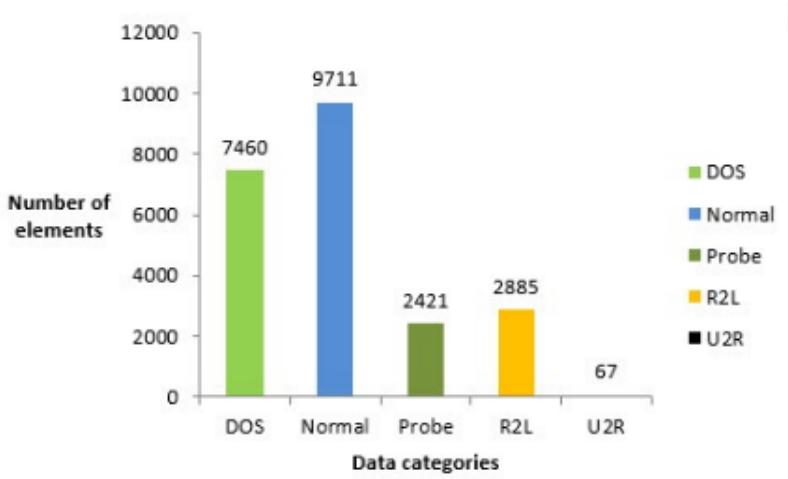

FIGURE 4.3 – Répartition des données de Test par catégorie d'attaque

Detected / Original	Normal	Probe	DoS	U2R	R2L
Normal	48684	3483	12341	14	342
Probe	8459	4702	4243	5	119
DoS	9816	3362	29118	9	56
U2R	38	23	36	17	10
R2L	167	53	161	459	6
New Category (K+1)	179	33	28	1	9

TABLE 4.4 – Résultats de détection des clusters normales et d'attaque à l'aide du DataSet d'apprentissage (KMEANS)

Detected / Original	Normal	Probe	DoS	U2R	R2L
Normal	63513	3357	8431	6	58
Probe	1189	5075	3937	3	97
DoS	2427	2927	33417	7	33
U2R	32	16	21	25	15
R2L	125	253	94	3	778
New Category (K+1)	57	28	27	8	14

TABLE 4.5 – Résultats de détection des clusters normales et d'attaque à l'aide du DataSet d'apprentissage (CKMSA)

Detected / Original	Normal	Probe	DoS	U2R	R2L
Normal	7771	584	2116	18	774
Probe	382	1330	981	9	257
DoS	1219	363	3460	16	483
U2R	29	24	18	18	13
R2L	263	105	857	6	1351
New Category (K+1)	47	15	28	0	7

TABLE 4.6 – Résultats de détection des clusters normales et d'attaque à l'aide du DataSet de Test (KMEANS)

CKMSA se présente également comme une solution plus efficace. En effet, appliquée à l'ensemble de données de Test, il donne de meilleurs résultats comme le montrent les tableaux 4.6.1 et 4.6.1. En outre, on peut remarquer que la nouvelle catégorie contient plus d'attaques dans le cas de K-means. Cependant, nous ne pouvons en déduire plus que ça car le pourcentage de fausses alarmes pour cet algorithme est significativement élevé.

Dans l'ensemble, l'approche proposée offre un meilleur taux de détection par rapport à K-means, comme le montre le tableau 4.6.1, avec 87% de taux d'exactitude et un taux de fausses alarmes inférieur à 6%. L'approche d'apprentissage hybride CKMSA s'avère ainsi plus efficace par rapport à l'approche précédente qui est synonyme de taux élevé de fausses alarmes. En conséquence, nous pouvons avoir plus confiance dans le dernier

Detected / Original	Normal	Probe	DoS	U2R	R2L
Normal	9291	439	1335	27	397
Probe	102	1493	258	3	254
DoS	168	328	5507	11	431
U2R	25	10	36	20	13
R2L	94	133	312	2	1865
New Category (K+1)	31	18	12	4	15

TABLE 4.7 – Résultats de détection des clusters normales et d'attaque à l'aide du DataSet de Test (CKMSA)

Dataset	Training		Testing	
Method	K-means	CKMSA	K-means	CKMSA
Accuracy	72,34%	87,55%	75,90%	88,39%
Detection Rate	72,40%	79,79%	72,79%	82,87%
False Alarm	27,71%	5,69%	19,98%	4,32%

TABLE 4.8 – Comparaison de K-Means et de CKMSA en utilisant les DataSets d'apprentissage et de Test

algorithme pour décider si un paquet appartient à la nouvelle catégorie d'attaques.

4.7 Conclusion

Le terrain concret des technologies de l'information d'aujourd'hui font de l'IDS une bonne solution pour surmonter les problèmes de sécurité liés à des données de plus en plus sensibles qui sont stockées et traitées dans des systèmes connectés. Cependant, des accidents de sécurité se produisent, les dommages économiques ruineux sont inéluctables. Nous avons proposé une heuristique efficace [29] pour optimiser le taux de détection dans les IDS comportementaux, cette approche regroupe le K-means semi-supervisé et le Simulated Annealing afin de contourner l'optimum local lors du clustering des données. Nous espérons dans le futur travailler à tester notre méthode sur réseau très dense et réel afin de tester la faisabilité, mais aussi l'acceptation d'attaques sophistiquées.

Conclusion générale et perspectives

Le Cloud Computing est une solution informatique très prometteuse. Vu que la sécurité informatique est un facteur principal, nous avons essayé dans cette thèse d'améliorer la disponibilité et la confidentialité du Cloud grâce à des solutions qui se basent sur les concepts d'équilibrage de charge et de chiffrement. Des algorithmes pour la détection des intrusions, en utilisant quelques méthodes de classification et d'optimisation, et pour le déploiement automatique des politiques de sécurité des pare-feus ont également fait l'objet de notre travail.

Comme première contribution [72] nous avons fait le tour des différents obstacles rencontrés par un administrateur afin qu'il déploie sûrement et correctement une politique de sécurité dans un pare-feus. Notre algorithme se révèle comme une solution juste et efficace en vue de les surmonter.

Comme deuxième [36] et troisième contribution, nous avons décrit certaines caractéristiques des performances du cloud computing, en explorant certaines de ses questions et défis critiques, à savoir la disponibilité et la confidentialité. Nous avons ensuite proposé une approche efficace [37][73] pour profiter davantage de l'équilibrage de charge, au moyen d'un schéma semi centralisé et multi-cluster. Cette méthode prometteuse est très susceptible d'entraîner une bonne performance globale et de fournir une meilleure tolérance aux pannes. Cette solution peut être déployée dans différents environnements cloud, en particulier les plus exigeants en termes de ressources. L'autre architecture s'avère aussi un bon appui pour cerner les problèmes de vie privée dans le Cloud.

D'un dernier point, aucun système informatique n'est idéal, il existerai toujours des vulnérabilités et des failles de sécurité qu'il faut savoir mitiger. La détection des intrusions comportementale existe justement pour cela, alors, notre quatrième contribution [29] consista en une heuristique efficace pour optimiser le taux de détection dans les IDS basées sur l'anomalie, dans cette approche nous avons combiné les K-Means semi-supervisés et le Simulated Annealing afin de contourner l'optimum local lors de la classification des données.

Apports de ce travail

Les contributions de cette thèse proposent une couche de sécurité intéressante pour le Cloud Computing :

- Augmenter les performances et la disponibilité du Cloud et de ses services ;
- Une architecture de Cloud qui peut supporter une quantité de données importante ;
- Une architecture semi-centralisée et multi-cluster ;
- Une détection des intrusions en réduisant les faux positifs ;
- Une classification optimale des événements de sécurité ;
- Une confidentialité mieux garantie pour les clients du Cloud Storage ;
- Un déploiement automatique et sans erreurs pour les politiques de sécurité de pare-feus.

Perspectives

Pour le travail à suivre, nous espérons pouvoir maîtriser l'architecture d'équilibrage de charge proposée. Elle reste un terrain fertile, avec beaucoup de possibilités à l'horizon vu le nombre de simulations et de cas qu'on peut étudier. Nous espérons alors pouvoir lancer différentes simulations pour analyser profondément son comportement.

Pour la validation de l'IDS proposé comme quatrième contribution, nous espérons pouvoir le mettre en oeuvre dans un système informatique réel contenant un réseau très dense afin de pouvoir le tester concrètement face à des attaques sophistiquées.

Concernant les politiques de sécurité de pare-feus, nous aspirons à ce que notre implémentation couvre en plus un paramétrage inter-réseau car l'actuelle solution est encore limitée dans un réseau local.

Enfin, pour le Cloud Storage, nous espérons considérer d'autres types de chiffrement tels le chiffrement homomorphique.

Bibliographie

[1] Accessed 26 November 2014. The nsl-kdd data set. 2014.

[2] DoD 5200.28-STD. *Trusted Computer System Evaluation Criteria*. Dod Computer Security Center, December 1985.

[3] Kartit A., Kamal Idrissi H., and Belkhouaf M. Z. Improved methods and principles for designing and analyzing security protocols. *International Journal of Network Security*, 18(3) :523 ?528, May 2016.

[4] Ali M. Alakeel. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Network Security (IJCSNS*, pages 153–160, 2010.

[5] Paul E. Ammann and Ravi S. Sandhu. The extended schematic protection model. *J. Comput. Secur.*, 1(3-4) :335–383, May 1992.

[6] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat. Practical domain and type enforcement for unix. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, SP '95, pages 66–, Washington, DC, USA, 1995. IEEE Computer Society.

[7] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5) :164–177, October 2003.

[8] Mohamed Belkhouraf, Hamza Kamal Idrissi, Ali Kartit, Mohamed El Marraki, Hassan Ouahmane, and Zaid Kartit. Improved methods for analyzing security protocols. In *La cinquième édition des Journées Nationales de la Sécurité*, 2015.

[9] D. Bertsimas and J. Tsitsiklis. Simulated annealing, statistical science. 8(1) :10–15, 1983.

[10] K. J. Biba. Integrity considerations for secure computer systems. Technical report, MITRE Corp., 04 1977.

[11] Matthew A. Bishop. *The Art and Science of Computer Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[12] D. F. C. Brewer and Michael J. Nash. The chinese wall security policy. In *IEEE Symposium on Security and Privacy*, pages 206–214. IEEE Computer Society, 1989.

[13] Carlos A. Catania and Carlos GarcíA Garino. Automatic network intrusion detection : Current techniques and open issues. *Comput. Electr. Eng.*, 38(5) :1062–1072, September 2012.

[14] National Computer Security Center. Integrity in automated information systems. 1991.

[15] H. S. Chae, B. O. Jo, S. H. Choi, and T. K. Park. Feature selection for intrusion detection using nsl-kdd. *Recent Advances in Computer Science*, 2013.

[16] Devyaniba Chudasama, Naimisha Trivedi, and Richa Sinha. Cost effective selection of data center by proximity-based routing policy for service brokering in cloud environment by, 2013.

[17] Cisco. Cisco security manager. 2001.

[18] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *IEEE Symposium on Security and Privacy*, pages 184–195. IEEE Computer Society, 1987.

[19] David D. Clark and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. *IEEE Symposium on Security and Privacy*, page 184, 1987.

[20] M. Dsilva and D. Vora. Comparative study of data mining techniques to enhance intrusion detection. *International Journal of Engineering Research and Applications (IJERA)*, 2013.

[21] November An Electronic, Len Lapadula, The Original, D. Elliott Bell, and Leonard J. Lapadula. Secure computer systems : Mathematical foundations, 1973.

[22] M. Englund and Sun BluePrints Online. Securing systems with host-based firewalls. 2001.

[23] Entrasys. Entrasys matrix x core router. 2001.

[24] Riccardo Focardi and Roberto Gorrieri. *Classification of Security Properties*, pages 331–396. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[25] Michael Gehrke, Andreas Pfitzmann, and Kai Rannenberg. Information technology security evaluation criteria (itsec) - a contribution to vulnerability ?, 1991.

[26] G. Scott Graham and Peter J. Denning. Protection : Principles and practice. In *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, AFIPS '72 (Spring), pages 417–429, New York, NY, USA, 1972. ACM.

[27] Kamal Idrissi H. and Kartit A. An application of methods and principles for designing and analyzing security protocols. *International Journal of Applied Engineering Research*, 10(23) :43692–43696, December 2015.

[28] Kamal Idrissi H., Kartit A., and El Marraki M. Foremost security apprehensions in cloud computing. *Journal of Theoretical and Applied Information Technology*, 59(3) :580 ?588, January 2014.

[29] Kamal Idrissi H., Kartit A., and Kartit Z. Ckmsa : An anomaly detection process based on k-means and simulated annealing algorithms. *International Review on Computers and Software*, 11(1) :42–48, January 2016.

[30] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8) :461–471, August 1976.

[31] Juniper. Juniper network and security manager. 2001.

[32] Hamza Kamal Idrissi, Ali Kartit, and Mohamed El Marraki. A taxonomy and survey of cloud computing. In *La troisième édition des Journées Nationales de la Sécurité*, 2013.

[33] A. Kartit and M. El Marraki. On the correctness of firewall policy deployment. *Journal of Theoretical and Applied Information Technology*, 19(1) :22–27, 2010.

[34] A. Kartit, A. Radi, M. El Marraki, and B. Regragui. Safe and correctness strategies for updating firewall policies. *International Journal of Computer Science and Network Security*, 11(3) :15–20, 2011.

[35] A. Kartit, A. Saidi, F. Bezzazi, M. El Marraki, and A. Radi. A new approach to intrusion detection system. *Journal of Theoretical and Applied Information Technology*, 36(2) :284–289, 2012.

[36] Ali Kartit, Hamza Kamal Idrissi, and Belkhouaf. A secured load balancing architecture for cloud computing based on multiple clusters. In *International Conference on Cloud Computing Technologies and Applications*, 2015.

[37] Zaid Kartit, Ali Azougaghe, Hamza Kamal Idrissi, Mohamed El Marraki, M. Hedabou, M. Belkasmi, and Ali Kartit. Applying encryption algorithm for data security in cloud storage. In *The International Symposium On Ubiquitous Networking*, 2015.

[38] Zaid Kartit, Hamza Kamal Idrissi, Ali Kartit, and Mohamed El Marraki. Network issues in cloud computing and countermeasures. In *La quatrième édition des Journées Nationales de la Sécurité*, 2014.

[39] Kate Keahey and Tim Freeman. Nimbus or an open source cloud platform or the best open source ec2 no money can buy. 2008.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598) :671–680, 1983.

[41] B. W. Lampson. Dynamic protection structures. In *Proceedings of the November 18-20, 1969, Fall Joint Computer Conference*, AFIPS '69 (Fall), pages 27–38, New York, NY, USA, 1969. ACM.

[42] Butler W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10) :613–615, October 1973.

[43] Wenke Lee and Salvatore J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 6–6, Berkeley, CA, USA, 1998. USENIX Association.

[44] Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, Ruth C. Taylor, S. Jeff Turner, and John F. Farrell. The inevitability of failure : The flawed assumption of security in modern computing environments. In *In Proceedings of the 21st National Information Systems Security Conference*, pages 303–314, 1998.

[45] Zoltán Ádám Mann. Allocation of virtual machines in cloud data centers&mdash ;a survey of problem models and optimization algorithms. *ACM Comput. Surv.*, 48(1) :11 :1–11 :34, August 2015.

[46] F. Martinez-Rios and J. Frausto-Solis. A simulated annealing algorithm for the satisfiability problem using dynamic markov chains with linear regression equilibrium. *International Journal of Computer Application*, 2014.

[47] Amuthan Prabakar Muniyandi, R. Rajeswari, and R. Rajaram. Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm. *Procedia Engineering*, 30(Complete) :174–182, 2012.

[48] Ajith Singh. N and M. Hemalatha. Article : An approach on semi-distributed load balancing algorithm for cloud computing system. *International Journal of Computer Applications*, 56(12) :5–10, October 2012. Full text available.

[49] S. Na, L. Xumin, and G. Yong. Research on k-means clustering algorithm : An improved k-means clustering algorithm. *Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI)*, 2010.

[50] Rourkela National Institute of Technology. Load balancing in cloud computing systems. 2011.

[51] NIST. The nist definition of cloud computing. 2011.

[52] Bo Peng, Bin Cui, and Xiaoming Li. Implementation issues of a cloud computing platform, 2009.

[53] OpenNebula Project. *OpenNebula 4.10 Design and Installation Guide*. 2015.

[54] A. Radi, A. Kartit, D. Aboutajdine, B. Regragui, M. El Marraki, and A. Ramrami. On the three level security policy comparaison between pca and svm and decision trees. *Journal of Theoretical and Applied Information Technology*, 35(1) :56–68, 2011.

[55] A. Radi, A. Kartit, B. Regragui, M. El Marraki, and D. Aboutajdine. An enhanced three levels security policy. *Journal of Theoretical and Applied Information Technology*, 24(1) :50–61, 2010.

[56] Soumya Ray. Execution analysis of load balancing algorithms in cloud computing environment. *Int. J. Cloud Comput. Serv. Archit.*, 2(5) :1–13, 2012.

[57] S. Revathi and A. Malathi. Network intrusion detection based on fuzzy logic. *Journal of Optimization Theory and Applications*, 45(1) :41–51, Jan 1985.

[58] Gupta1 Ruhi and Bhatia Rekha. An enhanced and secure approach of load balancing in cloud computing. *International Journal of Computer Sciences and Engineering*, 2(8) :112–116, 2014.

[59] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems, 1975.

[60] Ravi Sandhu. Separation of duties in computerized information systems. In *IN DATABASE SECURITY IV : STATUS AND PROSPECTS*, pages 179–189. North-Holland, 1990.

[61] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2) :38–47, February 1996.

[62] Ravinderpal Singh Sandhu. The schematic protection model : Its definition and analysis for acyclic attenuating schemes. *J. ACM*, 35(2) :404–432, April 1988.

[63] H. Saxena and D. V. Richariya. Intrusion detection system using k-means, pso with svm classifier. *A Survey, International Journal of Emerging Technology and Advanced Engineering*, 4(2), 2014.

[64] Sukhpal Singh and Inderveer Chana. A survey on resource scheduling in cloud computing : Issues and challenges. *J. Grid Comput.*, 14(2) :217–264, June 2016.

[65] Masakazu Soshi. *Safety Analysis of the Dynamic-Typed Access Matrix Model*, pages 106–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[66] Eucalyptus Systems. *Eucalyptus 4.1.2 Installation Guide.* 2015.

[67] Liuying Tang and Sihan Qing. *A Practical Alternative to Domain and Type Enforcement Integrity Formal Models*, pages 225–237. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[68] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, Piscataway, NJ, USA, 2009. IEEE Press.

[69] Chengzhong Xu and Francis C. Lau. *Load Balancing in Parallel Computers : Theory and Practice.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[70] Yanfang Ye, Tao Li, Donald Adjeroh, and S. Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Comput. Surv.*, 50(3) :41 :1–41 :40, June 2017.

[71] Yingbing Yu. A survey of anomaly intrusion detection techniques. *J. Comput. Sci. Coll.*, 28(1) :9–17, October 2012.

[72] Kartit Z., Kamal Idrissi H., Kartit A., and El Marraki M. Improvement of algorithm for updating firewall policies. *Journal of Theoretical and Applied Information Technology*, 66(1) :284 ?289, August 2014.

[73] Kartit Zaid, Azougaghe Ali, Kamal Idrissi Hamza, El Marraki Mohamed, Hedabou M., Belkasmi M., and Kartit Ali. Applying encryption algorithm for data security in cloud storage. In Jan Fagerberg, David C. Mowery, and Richard R. Nelson, editors, *Advances in Ubiquitous Networking*, chapter 12, page 141 ?154. 2016.

[74] C. C. Zhang, M. Winslett, and C. A. Gunter. On the safety and efficiency of firewall policy deployment. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 33–50, May 2007.

ANNEXES

Le code en Matlab de l'application 'CKMSA' est :

```
function [Y, I, J] = shake(X,dim)
% SHAKE - Randomize a matrix along a
% specific dimension
%   Y = SHAKE(X) randomizes the order
%of the elements in each column of the
%   2D matrix. For N-D matrices it randomizes
% along the first non-singleton
%   dimension.
%
%SHAKE(X,DIM) randomizes along the dimension DIM.
%
%   [Y, I, J] = SHAKE(X) returns indices so
% that Y = X(I) and X = Y(J).
%
%   Example:
%     A = [1 2 3 ; 4 5 6 ; 7 8 9 ; 10 11 12] ;
% see <SLM> on the FEX ...
%     Dim = 2 ;
%     B = shake(A,Dim)  % ->, e.g.
%       % 3      2      1
%       % 6      4      5
%       % 7      8      9
%       % 11     10     12%
%     C = sort(B,Dim) % -> equals A!
%
%     The function of SHAKE can be thought
% of as holding a matrix and shake
%     in a particular direction (dimension),
% so that elements are getting
%     shuffled within that direction only.
%
%   See also RAND, SORT, RANDPERM
%   and RANDSWAP on the File Exchange
```

```
% for Matlab R13
% version 4.1 (may 2008)
% (c) Jos van der Geest
% email: jos@jasen.nl

% History
% Created: dec 2005
% Revisions
% 1.1 : changed the meaning of the DIM.
%Now DIM==1 works along the rows, preserving
% columns, like in <sort>.
% 2.0 (aug 2006) : randomize along any dimension
% 2.1 (aug 2006) : output indices argument
% 3.0 (oct 2006) : new & easier algorithm
% 4.0 (dec 2006) : fixed major error in 3.0
% 4.1 (may 2008) : fixed error for scalar input

error(nargchk(1,2,nargin)) ;

if nargin==1,
    dim = min(find(size(X)>1)) ;
elseif (numel(dim) ~= 1) || (fix(dim) ~= dim) ||
                                                (dim < 1),
    error('Shake:DimensionError','Dimension
argument must be a positive integer scalar.') ;
end

% we are shaking the indices
I = reshape(1:numel(X),size(X)) ;

if numel(X) < 2 || dim > ndims(X) || size(X,dim) < 2,
    % in some cases, do nothing
else
    % put the dimension of interest first
    [I,ndim] = shiftdim(I,dim-1) ;
    sz = size(I) ;
    % reshape it into a 2D matrix
    % we'll randomize along rows
    I = reshape(I,sz(1),[]) ;
    % get new row indices
    [ri,ri] = sort(rand(size(I)),1) ;
    ci = repmat([1:size(I,2)],size(I,1),1) ;
    % but keep old column indices
```

```
    I = I(sub2ind(size(I),ri,ci)) ; % retrieve values
    % restore the size and dimensions
    I = shiftdim(reshape(I,sz),ndim) ;
end

% re-index
Y = X(I) ;

if nargout==3,
    J = zeros(size(X)) ;
    J(I) = 1:numel(J) ;
end

%# Distance euclidienne totale:DT
function DT = DistET(ClustIDXDist)
DT=sum(ClustIDXDist)

%# ClustIDXDist: Vecteur des distances entre
%#chaque point et son centroide
function ClustIDXDist = DistPC(D,ClustIDX)
ClustIDXDist = [1:size(ClustIDX)];
    for i=1:numel(ClustIDX)
        %#for i=1:4000
        j=ClustIDX(i);
       ClustIDXDist(i) = D(i,j);
    end
ClustIDXDist=ClustIDXDist(:)

%#C: centroides
%# distance  entre chaque point et tous les centroides
%#La première fonction dans un fichier .m
%#doit avoir le meme nom que ce fichier
function D = DistPTC(data,C)
[n,p] = size(data);
D = zeros(n,size(C,1));
clusts = 1:size(C,1);



for i = clusts
    D(:,i) = sum((data - C(repmat(i,n,1),:)).^2, 2);
end

function AP=acceptanceProbability(currentDT, newDT,
                                                    temperature)
```

```
        if (newDT < currentDT)
            AP= 1;
        end

        AP=exp((currentDT - newDT) / temperature);

%#ML: longueur de markov
%# data nÃ©cessaire pour calculer les centroides
%# candidats
function [candidateC,candidateClustIDX,nbc]=
generatecandidate(currentC,currentClustIDX,data,ML)
[candidateClustIDX,I,J]=shake(currentClustIDX);
nbc=nombredechangementdistincts(currentClustIDX,
candidateClustIDX);
if nbc>ML
[candidateC,candidateClustIDX]=generatecandidate(
currentC,currentClustIDX,data,ML);
else
datawithclusteridx=[data,candidateClustIDX];
for i=1:size(currentC,1)
data_filtered =
datawithclusteridx(datawithclusteridx(:, end) == i, :);
if size(data_filtered,1)~=1
candidateC(i,:)=mean(data_filtered);
else
candidateC(i,:)=data_filtered
end
end
candidateC(:,42)= [];
end
%# test [clusterstest,clustIDXtest,nbc]=
generatecandidate(clusters,clustIDX,data1plusdata,100)

function nbc=nombredechangementdistincts(
                        currentClustIDX,candidateClustIDX)
nbc=0;
for i=1:numel(currentClustIDX)
if currentClustIDX(i)~=candidateClustIDX(i)
nbc=nbc+1;
end
end
nbc=nbc/2;

%#best solution=best c(centroids+best clustIDX(
                configuration des points)
%#la meilleure valeur de la fonction objective
```

```
function [bestC,bestClustIDX,bestDT] =ksa(ti,tf,p,C,ClustIDX,data,ML)
currentC=C;
currentClustIDX=ClustIDX;
bestC=C;
bestClustIDX=ClustIDX;
bestDT=DistET(DistPC(DistPTC(data,bestC),
                                                bestClustIDX));
for o=ti:-p:tf

[candidateC,candidateClustIDX,nbc]=
                generatecandidate(currentC,
                                        currentClustIDX,data,ML);
candidateDT = DistET(DistPC(DistPTC(
                data,candidateC),candidateClustIDX));
currentDT = DistET(DistPC(DistPTC(data,currentC),currentClustIDX));
AP=acceptanceProbability(currentDT, candidateDT, o)

if AP > rand(1,1)
            currentC=candidateC;
                currentClustIDX=candidateClustIDX;

end
            currentDT = DistET(DistPC
        (DistPTC(data,
                currentC),currentClustIDX));
bestDT=DistET(DistPC(DistPTC(data,bestC),
                                        bestClustIDX));
             if currentDT < bestDT
                        bestC=currentC;
                        bestClustIDX=currentClustIDX;

             end
end
%#[bestC,bestClustIDX,bestDT]=
ksa(1000,1,1,clusters,clustIDX,data1plusdata,400)
```