République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia

Faculté des Sciences Exactes
Départements d'Informatique
Ecole Doctorale Réseaux et Systèmes Distribués

Mémoire de Magistère
En Informatique

Option
Réseaux et Systèmes Distribués

Thème

Approche Data Mining pour la Détection d'Intrusions

Présenté par

M^elle^ Lynda SELLAMI

Devant le jury composé de :

Président :	**Youcef KHELFAOUI**	**MC**	**Université de Bejaia**
Rapporteur :	**Djamal LOUANI**	**Professeur**	**Université Reims, France**
Examinateurs :	**Smail ADJABI**	**MC**	**Université de Bejaia**
	Adbelouahab MOUSSAOUI	**MC**	**Université de Sétif**

Remerciements

Louage à DIEU, maître des mondes qui m'a dirigé dans la bonne voie et pour avoir veillé à l'accomplissement de ce modeste travail.

Je remercie mon directeur de thèse Monsieur Louani Djamal, professeur à Paris 6 d'avoir accepté de m'encadrer, pour son appui et ses conseils.

Je tiens mes remerciements les plus sincères à Monsieur Khelfaoui Youcef, vice-doyen chargé de la poste graduation, pour ses conseils, son aide, son appui et pour l'aide précieuse qu'il ma apporté, je le remercie tout particulièrement pour sa patience et son sens de la diplomatie qu'il a su habilement distiller dans les moments difficiles.

Je remercie également Monsieur Kerkar, doyen de la faculté des sciences exactes pour ses conseils pour son appui, son soutien.

Je remercie Monsieur Belmehdi et Madame Timridjine, Professeurs à l'université de Bejaia pour leurs aides, leurs recommandations.

J'exprime ma profonde gratitude à Monsieur Kamel Tari, Responsable de la formation doctorale en informatique au département d'informatique de Bejaia pour ses précieux conseils et pour son soutien moral et technique.

Je remercie également tous les enseignants qui ont participé à ma formation, grâce à eux je suis arrivée à ce stade.

Je remercie les membres du jury qui m'ont fait l'honneur d'accepter de juger ce modeste travail.

Je remercie également tous les membres de l’équipe RESYD avec lesquels j’ai passé trois années riches et agréables, que le personnel du département d’informatique trouve ici l’expression de ma reconnaissance.

Que toutes les personnes qui ont contribué de prés ou de loin à l’achèvement de ce présent travail soient remerciées.

Enfin, ces remerciements ne seraient pas complets sans une pensée toute particulièrement à ma famille et mes amis proches qui savent toujours être là quand il le faut : Vous êtes merveilleux et je vous adore !

En témoignage de gratitude et de sincère affection.

Dédicace

Je dédie ce modeste travail à :

Ma très chère mère Rouha Aicha qui a toujours été ma lumière

Mon père pour ses encouragements.

Ma grand mère et à la mémoire de mes grands parents

Mon frère Khaled

Mes sœurs : Sabrina, Tounes, Sawsen, Maya et Asma

Mes oncles : Abdelhamid, Mustapha, Farid et Rachid

A mon beau frère Faicel (Soufiane) et sa famille Merabtine

Mes tantes, cousins et cousines

Aux familles ROUHA et SELLAMI

Tous mes amis

Tous ceux qui me sont chers.

Tous ceux qui possèdent un esprit saint

Tous les martyrs d'Algérie

Sommaire

Liste des tableaux

Liste des figures

Introduction Générale

Au début, les réseaux et les systèmes informatiques étaient isolés les uns des autres. La ***complétude*** de ces derniers et leurs relations rend leurs interconnexions indispensables. Ainsi le bon fonctionnement et la croissance des entreprises nécessitent l'utilisation des réseaux et systèmes informatiques.
Avec l'évolution de l'informatique, l'interconnexion des réseaux informatiques devient un point crucial pour le développement.

L'Internet d'aujourd'hui désigne le plus grand réseau informatique de la planète. Il regroupe une multitude de réseaux régionaux, gouvernementaux et commerciaux. L'Internet doit sa réussite à son aspect universel et ouvert et cela grâce aux multiples services qu'il fournie (messagerie électronique, commerce électronique, …). Donc, tous les réseaux informatiques font partie de l'Internet. L'ouverture à l'Internet et sa simplicité d'utilisation rendent les réseaux informatiques à la merci de tous les autres usagers du réseau (Internet). Cette ouverture crée alors un grand problème de sécurité pour le réseau dû à des utilisateurs mal intentionnés. La sécurité du réseau est alors une question cruciale. De ce fait, les organisations ont tout intérêt à se protéger contre les attaques sur leurs réseaux ou sur les hôtes des réseaux.

À l'heure actuelle, un grand nombre de systèmes sont attaqués. Pour cela, ces systèmes se doivent être protégés contre les anomalies provenant soit d'une attitude intentionnellement malveillante ou d'une faiblesse, rendant le système vulnérable. La mise en place d'un système de protection tel qu'un pare-feux devient indispensable, cependant, ces derniers sont insuffisants car ils se basent sur la surveillance de l'intrusion ou de l'attaque. Par contre une nouvelle technique se basant sur la surveillance de l'intrus ou de l'attaquant lui-même est élaborée pour mieux protéger et gérer l'intrusion et cela en accompagnant ces systèmes de protection par des outils de protections tel que les ***systèmes de détection d'intrusions*** (***IDS***). Ces dernières donnent la possibilité d'appliquer rapidement de nouvelles politiques de sécurité dans les réseaux afin de détecter et réagir le plus vite possible aux attaques survenant dans le réseau.

La détection d'intrusions consiste à découvrir ou à identifier l'utilisation d'un système informatique à d'autres fins que celles prévues. C'est une technique à multiple facette, difficile à cerner lorsqu'on ne les manipule pas. Mais, la plupart des travaux effectués dans ce domaine restent difficiles à comparer ou incomparable.

Le but d'un système de détection d'intrusions est d'extraire et de classifier des données pertinentes d'un large volume de données diverses. Ces données sont collectées et enregistrées sur un grand nombre d'événements de sécurité, grâce à un audit de sécurité.

Les systèmes de détection d'intrusions se basent sur la centralisation rendant le système à protéger vulnérable et surcharge la base de données. La solution à ce problème est d'introduire les systèmes multi-agent et cela grâce à leurs aspects distribués.
La technologie des agents aura un rôle de plus en plus important à jouer dans les télécommunications grâce à leurs propriétés notamment, d'autonomie, auto configuration, d'intelligence et / ou de mobilité. Ainsi, elle donne un autre souffle à la sécurité (qui est la détection d'intrusions) et donne de la force à la détection d'intrusion.

Le volume et la diversité des données collectées sont dû à l'augmentation du nombre d'attaques, la taille des journaux d'audit manipulés, ainsi qu'au manque d'un standard reconnu des systèmes de détection d'intrusions qui utilisent chacun leurs techniques et données appropriés. Pour remédier à tous cela, une des techniques les plus estimées à l'heure actuelle est celle du ***Data Mining***.

Le Data Mining (ou Fouille de données en français), est la réponse à la croissance des moyens informatiques et de calcul qui induit au stockage, au traitement et à l'analyse d'ensembles de données très volumineux.

Le Data Mining est un ensemble de techniques d'exploration de données permettant d'extraire d'une collection de données des connaissances sous forme de modèles de description afin de ***décrire*** le comportement actuel et / ou de ***prédire*** le comportement futur des données. Donc, le but du Data Mining est de ***trouver*** des schémas intéressants selon des critères fixés au départ, et ***extraire*** de ces données un maximum de connaissances utiles concernant l'application traitée.

Organisation du mémoire

Dans la première partie de ce mémoire, nous présentons une revue de la littérature sur deux sujets liés à notre travail :

Chapitre 1 : Généralités sur les Systèmes de Détection d'Intrusions

Ce chapitre sera consacré à la détection d'intrusions dans les réseaux. Nous dressons un état de l'art des approches globales applicables à la détection d'intrusions, l'approche comportementale et l'approche par scénarios. L'analyse des outils employés en détection d'intrusions nous permettra d'avoir une vue plus détaillée des moyens disponibles pour sécuriser un réseau.

La détection d'intrusions ne vient pas concurrencer les mécanismes de sécurités traditionnelles mais, au contraire, les compléter. Dans ce chapitre nous allons également présenter les principes mis en œuvre par les systèmes de détection d'intrusions pour atteindre leurs buts. Le but d'un système de détection d'intrusions est d'extraire et de classifier des données pertinentes d'un large volume de données diverses.

Chapitre 2 : Introduction au Data Mining

Notre problématique nous amène à nous pencher sur des considérations relevant du Data Mining. Dans ce chapitre nous allons exposer les principes et les méthodes du Data Mining qui sont utilisés pour extraire des connaissances d'un ensemble de données.

La seconde partie de ce mémoire, correspond aux chapitres 3 et 4, représente respectivement:

Chapitre 3 : Utilisation des technique d'extraction de connaissance pour la Détection d'Intrusions

Dans ce chapitre nous présentant les approches d'extraction de données (connaissances) pour la détection d'intrusions.

Chapitre 4: Détection d'Intrusions par Classification et Règles d'Association basé sur les Agents (CARIDA)

Pour couvrir les perspectives d'utilisation des deux approches que nous avons défini plus hauts, nous avons choisi de mettre en œuvre un système de détection d'intrusions basé sur l'approche Data Mining utilisant les agents mobiles et qui est intitulé "**Classification and Association Rule in Intrusion Detection with Agent (CARIDA)** "

Conclusion

Dans la conclusion, nous faisons le bilan de ce mémoire, et nous montrons comment notre objectif a été atteint. Ensuite nous exposons les perspectives et les travaux que nous envisageons à approfondir

Nous avons aussi introduit, en **annexe**, le concept d'agent mobile ainsi que la plate-forme Aglet.

Chapitre I Généralité sur les Systèmes de Détection d'Intrusions

I.1 Introduction

L'utilisation des réseaux et systèmes informatiques comme outils, devient nécessaire au bon fonctionnement et à la croissance des entreprises. L'interconnexion de ces réseaux devient indispensable après qu'ils soient isolés les uns des autres. Cette interconnexion induit à l'augmentation des utilisateurs et des points d'accès à ces réseaux.

La multitude d'utilisation de ces réseaux par des personnes (utilisateurs) connus ou non, les transforment en cibles d'attaques potentielles, car ces utilisateurs peuvent exploiter les vulnérabilités des réseaux et des systèmes pour accéder aux informations qu'ils contiennent ou pour porter atteinte à leurs bons fonctionnements.

La sécurité de ces réseaux visés par des attaquants est un enjeu d'une importance capitale. Pour cela de nombreux outils et moyens sont de nos jours disponibles comme solution qu'elle soit matérielle (firewalls) ou logiciels comme les logiciels d'audits.

L'évolution continue des attaques ainsi que l'apparition de nouvelles attaques et la nécessité de pouvoir appliquer rapidement de nouvelles politiques de sécurité dans un réseau afin de détecter et réagir le plus vite possible aux attaques survenant dans le réseau, ont conduit à l'apparition des ***Systèmes de Détection d'Intrusions*** (***IDS***) comme réponse à tous ces problèmes.

Dans cette partie, nous allons nous intéresser aux Systèmes de Détections d'Intrusions, à leur importance, aux approches de la détection d'intrusions et aux principes de ces derniers ainsi qu'à leurs limites.

I.2 Historique

Le concept de Système de Détection d'Intrusions a été introduit par **James Anderson** en **1980** dans son rapport ***Computer security threat Monitoring and Surveillance*** [5]. Mais le sujet n'a pas eu beaucoup de succès. Il a fallu attendre la publication d'un modèle de détection

d'intrusions par **Denning** en **1987** pour marquer réellement le démarrage du domaine [17], [8]

I.3 Intrusion

Une intrusion est tout ensemble d'actions visant à compromettre la ***confidentialité*** (pour chaque information, on défini l'ensemble d'utilisateurs autorisés à y accéder), l'***intégrité*** (les données reçues sont bel et bien celles qui ont été envoyées) des données et la ***disponibilité*** (les services du système doivent être opérationnels et accessibles selon des modalités définies) des services ou des ressources. Les intrusions peuvent être connues ou inconnues (anomalies).

I.4 Détection d'intrusions

La Détection d'Intrusions est la capacité pour un système informatique de déterminer automatiquement, à partir d'événements relevant de la sécurité, qu'une violation de sécurité se produit ou s'est produite dans le passé [40]. Ceci nécessite un audit de sécurité pour la collection et l'enregistrement d'un grand nombre d'événements de sécurité.

Il n'est pas envisageable de faire cette détection manuellement car la recherche d'actions suspectes se fait dans d'immenses volumes de données. Il a fallu donc trouver des méthodes et développer des outils pour analyser automatiquement les traces d'audit.

I.5 Systèmes de Détections d'Intrusions

Un système de détection d'intrusions est un outil (mécanisme) de détection d'intrusions qui implique une surveillance permanente des actions effectuées sur le système afin de repérer des activités anormales ou suspectes et permettre ainsi d'avoir une démarche de prévention sur les comportements hostiles dirigés contre le réseau.

Les systèmes de détections d'intrusions sont classés selon les principes présentés dans [14], [37], [38] :

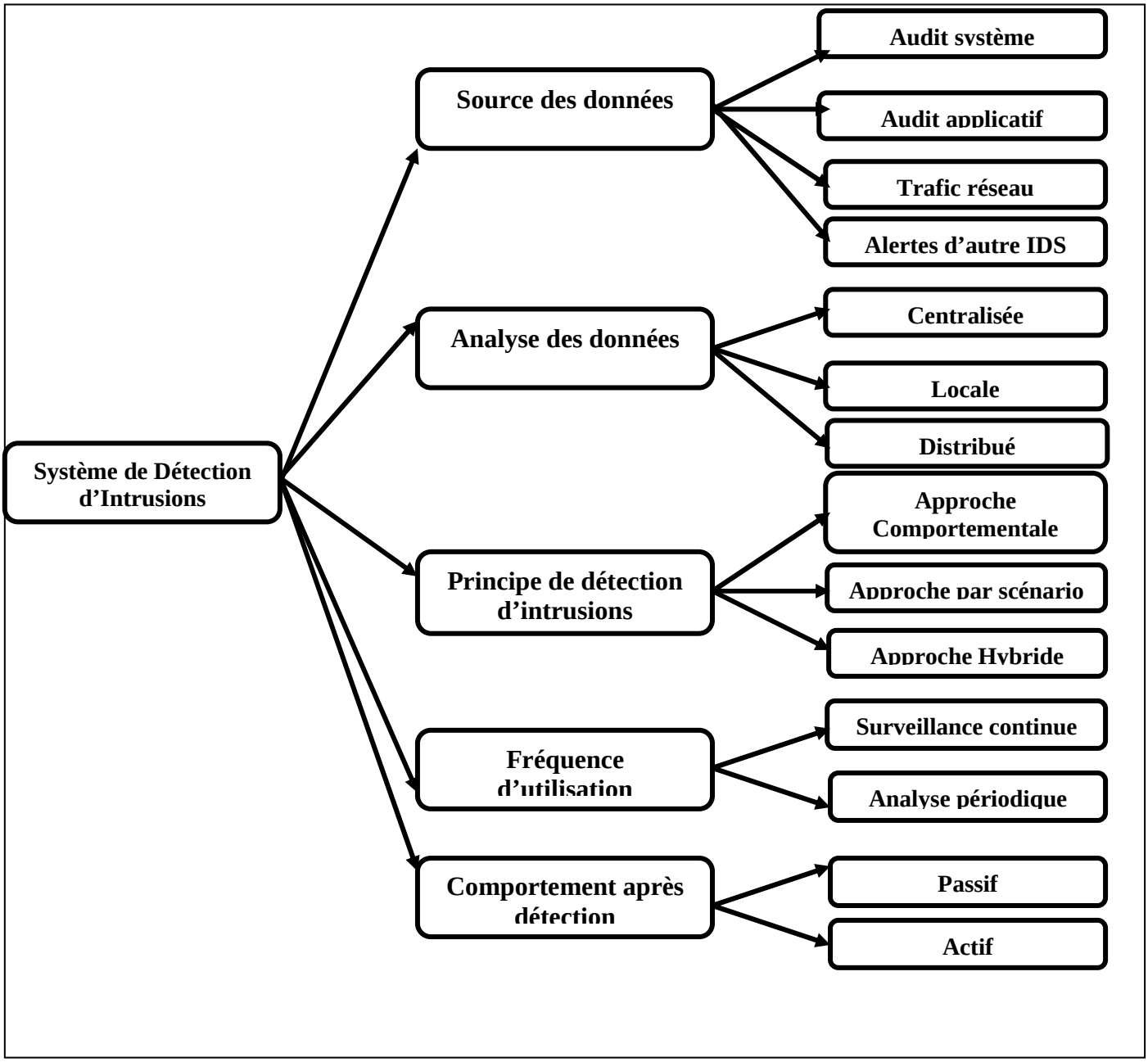

Figure 1 : Classification des systèmes de détection d'intrusions

1.5.1 Sources des données à analyser

Les sources des données constituent un point crucial dans la détection d'intrusions grâce aux informations qu'elles fournissent pour analyser le système pour d'éventuelles intrusions.

Il existe quatre sources de données qui sont : l'audit système, l'audit applicatif, le trafic réseau et les alertes d'autres IDSs.

1. **Audit Système :** Permet d'enregistrer les actions effectuées sur le système en exploitant l'outil d'administration système. Cette source de données est très pertinente, du fait que la totalité des attaques provoquent une interaction avec le système.

2. **Audit Applicatif :** C'est une source d'information de haut niveau, qui représente les services Web tel que le FTP (File Transfert Protocol) et le HTTP (Hyper Text Transfert Protocol).

3. **Trafic réseau** : C'est les paquets récupérés du réseau.

4. **Alertes d'autres IDSs :** ou source d'information basée IDS, c'est des alertes remontées par des analyseurs provenant d'un IDS. Chaque alerte synthétise déjà un ou plusieurs événements intéressants du point de vue de la sécurité. La corrélation de ces alertes conduit parfois à la détection d'une intrusion complexe de plus haut niveau.

I.5.2 Analyse de données du système

Les types d'analyse de données pour la détection qui ont été présentées à ce jour sont l'analyse ***Centralisée, Locale*** et l'analyse ***Distribuée.***

- ✓ **Analyse centralisée :** Le but de cette architecture est de faciliter la corrélation entre événements (alertes) puisqu'on dispose que d'une vision globale. Donc, cette analyse consiste à centraliser les alertes pour les analyser au sein d'une seule machine.

- ✓ **Analyse locale :** L'analyse se fait localement, c'est-à-dire sur chaque hôte séparément des autres. Donc, on lance l'analyse sur un hôte seulement, ce qui fait perdre du temps. De plus, il n'y a pas de surveillance entière du réseau en même temps.

- ✓ **Analyse distribuée :** L'analyse se fait de manière distribuée sur le réseau. L'analyse est lancée sur toutes les machines au même temps. Ceci fait gagner du temps dans l'analyse, permet la surveillance entière du réseau et évite donc d'avoir des attaques au moment de l'analyse.

I.5.3 Principe de détection d'intrusions

Les Systèmes de Détection d'Intrusions (IDSs) sont divisés en deux catégories, suivant l'approche utilisée ; approche par scenario, approche comportementale et la coopération entre les deux approche [9] :

1 Approche par scénario

Elle se base sur des attaques connues préalablement. Cela nécessite une ***connaissance à priori des attaques à détecter***. Chaque attaque connue produit une trace spécifique dans les enregistrements d'audit. Lorsque la trace d'une attaque connue est détectée, une alarme est déclenchée. L'approche par scénario utilise plusieurs techniques:

- ✓ **Les techniques à base de règles** : (les systèmes experts). La base de connaissances des systèmes experts contient des règles concernant non seulement les scénarios d'attaque que l'on veut détecter sur le système cible, mais aussi les vulnérabilités connues de ce système. La construction de cette base repose entièrement sur l'expérience et le savoir faire de l'officier de sécurité
- ✓ **Les approches fondées sur la signature des attaques** : Essaient de représenter les attaques sous forme de signatures. Les données courantes sont alors comparées à ces signatures. S'il y a des similitudes, une alarme est déclenchée.
- ✓ **Les algorithmes génétiques :** Utilisés afin d'optimiser la recherche de scénarios d'attaque dans des fichiers d'audit. Leur utilisation permet d'éviter une recherche exhaustive en sélectionnant les meilleurs éléments d'une population (ici les scénarios d'attaque).

On peut distinguer **deux inconvénients** à cette approche : l'incapacité à ne détecter que des attaques connues, par ailleurs il faut remettre à jour la base de scénarios d'attaque le plus souvent.

2 Approche comportementale

L'approche comportementale a été proposée en 1980 par Anderson [5]. L'idée maîtresse de cette approche est de prendre en compte le comportement d'un utilisateur pour détecter tout comportement anormal, que l'on considère alors comme une tentative possible d'attaquer le système.

Cette approche se base donc sur un comportement ou un ***profil normal,*** qui est représenté par des cas de références. Cela nécessite une ***phase d'apprentissage***. Cette approche vise soit à la ***modélisation*** ou à la ***prédiction*** du comportement normale. [8].

Lorsqu'une déviation trop importante vis-à-vis le profil normal est observée, une alarme est émise. Pour modéliser le profil normal, les approches comportementales utilisent :

- ✓ **Modèle statistique** : Proposé par Denning [17] et qui consistent à utiliser des mesures statistiques pour modéliser un profil de comportement et détecter ensuite des comportements intrusifs. Chacune de ces mesures est associée à un seuil ou à un intervalle de valeurs, dans lequel une activité est considérée comme normale. Tout dépassement de seuil ou situation de valeurs à l'extérieur des bornes de l'intervalle indique une activité anormale. L'outil NIDES utilise cette méthode [29].

- ✓ **Systèmes experts** : Proposé par Debar [15]. La différence majeure entre un système expert et un modèle statistique est que ce dernier utilise des formules statistiques pour identifier des comportements dans les données d'audit alors que le système expert utilise un ensemble de règles pour représenter ces comportements.

- ✓ **Des réseaux de neurones :** Un réseau de neurones est constitué de plusieurs éléments de traitement simples appelés unités et qui interagissent en utilisant des connections pondérées. Le réseau constitue le profil normal d'un utilisateur. Ainsi, après chaque commande utilisée par cet utilisateur, le réseau essaie de prédire la commande suivante, en tenant compte des n commandes antérieures. Si la commande réelle dévie de celle prédite, alors une alarme est déclenchée.

L'approche comportementale permet de détecter des attaques inconnues auparavant ainsi que les abus de privilèges des utilisateurs légitimes du système. **Ce pendant**, le comportement de référence ne sont jamais parfait, nous sommes exposé à des risques de fausses alarmes (faux positif). De **plus**, si des attaques ont été commises durant la phase d'apprentissage, elles seront considérées comme normales (risque de faux négatifs).

3 Coopération entre les différentes approches

Afin de remédier aux inconvénients de chacune des techniques (approches), certains systèmes utilisent une combinaison de l'approche comportementale et de l'approche par scénarios. Donc, faire collaborer différentes approches afin de combiner les forces de

chacune tout en éliminant leurs faiblesses. Ainsi, un compte administrateur aura un profil qui lui permet d'accéder à certains fichiers sensibles, mais il peut être utile de vérifier que des attaques connues ne sont pas utilisées contre ces fichiers [8].

I.5.4 Fréquence d'utilisation

Il existe deux façons de mesurer, d'évaluer et d'estimer la fréquence d'utilisations des systèmes de détection d'intrusions, *la surveillance* ***continue*** et *l'analyse* ***périodique***.

1. La ***surveillance continue*** *revient à* de surveiller et d'analyser le système de manière permanente, sans arrêt et en temps réel (observation dans le temps du système).

2. ***L'analyse périodique*** est la surveiller et l'analyser le système de manière périodique, c'est-à-dire, à chaque période de temps qui est choisie par l'administrateur (observation périodique dans le temps).

I.5.5 Comportement après détection

Une autre façon de classer les systèmes de détection d'intrusions, consiste à voir quelles sont leurs réactions lorsqu'une attaque est détectée. Certains se contentent de déclencher une alarme (réponse ***passive***), alors que d'autres prennent des mesures correctives (réponse ***active***).

La plupart des IDSs n'apportent qu'une réponse ***passive*** à l'intrusion. Lorsqu'une attaque est détectée, ils génèrent une alarme en direction de l'administrateur système par e-mail, message dans une console, voire même par beeper. C'est à l'administrateur de prendre les mesures qui s'imposent.

Si le système est plus sophistiqué (et surtout plus récent), il peut prendre automatiquement des mesures pour empêcher ou stopper l'attaque en cours (réponse ***active***). Par exemple, il coupera les connexions suspectes ou même pour une attaque distante reconfigurera le pare-feu pour qu'il refuse tout ce qui vient du site incriminé. Il pourra également prévenir l'administrateur ; l'outil ***RealSecure*** utilise cette approche [8].

I.6 Les limites actuelles de la détection d'intrusions

Dans le tableau suivant nous exposons les avantages et incontinents des deux approches de détection d'intrusions à savoir l'approche par scénario et l'approche comportementale :

Approche	Avantage	Inconvénient
Comportementale	Détection d'intrusion inconnue possible.	Choix délicat des mesures à retenir pour un système cible donné. Pour un utilisateur au comportement erratique, toute activité est "normale". En cas de profonde modification de l'environnement du système cible, déclenchement d'un flot ininterrompu d'alarmes (faux positifs). Utilisateur pouvant changer lentement de comportement dans le but d'habituer le système à un comportement intrusif (faux négatifs).
Par scénarios	Prise en compte des comportements exacts des attaquants potentiels.	Base de règles délicates à construire. Seules les attaques contenues dans la base sont détectées.

Tab 1 : Approche Comportementale ou Approche par Scénarios

La détection de nouvelles attaques, et précisément la détection d'attaques inconnues, pose un problème pour les IDSs.

Les systèmes fondés sur l'approche par scénario ne détectent que des attaques connues, d'où la nécessité de la mise à jour de leur base de connaissance comme les ***Antivirus.***

Par contre, l'approche comportementale permet de détecter des attaques inconnues. Elles sont identifiées à partir du moment où une déviation vis-à-vis du comportement normale de référence est observée.

Au-delà du problème de la détection de nouvelles attaques, tous les IDSs existants sont soumis aux problèmes de l'***extraction*** et la ***classification*** de données pertinentes d'un grand

volume de données diverses [22]. Ceci est dû à l'augmentation du nombre d'attaques ainsi que la taille des journaux d'audits manipulés (qui contiennent de plus en plus d'événement), augmentation liée à celle de la taille et des débits des réseaux et leurs utilisation; ainsi qu'à l'éventail d'IDS utilisant chacun leurs techniques et données propres par manque de standard reconnu [12].

Les IDSs actuels sont trop ***fermés,*** ce qui ***limite*** les possibilités de comparaisons des performances d'une part, et les possibilités de coopération d'autre part. Pourtant, diverses initiatives tendent à résoudre ce problème. Ainsi le groupe de travail ***Common Intrusion Detection Framework*** (CIDF) vise à définir un standard d'interopérabilité entre outils.

Au-delà de ces deux limites il y a plus grave, les IDSs actuels peuvent être mis en ***défaut***, soit parce qu'ils sont incapables de détecter certains types d'attaque, soit parce qu'ils sont eux-mêmes attaquables [34] :

I.6.1 Attaques non détectables

Comme déjà mentionné, certains systèmes récents permettent de prendre automatiquement des contre-mesures. Un attaquant suffisamment doué peut effectuer une attaque qui aura l'air de provenir d'une machine du réseau interne. L'outil coupera alors les connexions avec la machine incriminée, ce qui constitue un ***déni de service***.
Un autre type d'attaque met en défaut tous les outils actuels : plusieurs personnes effectuent une attaque distribuée conjointe à la cadence d'une action par intervalle de temps (toutes les quelques heures). La distribution et la lenteur de l'attaque la fait passer inaperçue. [34]
Les outils basés réseau qui surveillent en temps réel le trafic sont incapables de suivre les débits des réseaux frame relay ou ATM.

I.6.2 Attaque des outils eux-mêmes

Les outils de détection d'intrusions peuvent eux-mêmes être la cible d'attaques les rendant inopérants sur un ou plusieurs aspects. L'attaque réelle passera ensuite inaperçue. Chacun des composants peut être attaqué : le module qui fournit les données à analyser (système d'audit ou autres), le module d'archivage, l'éventuel module de contre-mesures et le module d'analyse [34] :

- ✓ Si on réussit à empêcher l'arrivée des données en entrée, la détection d'intrusions s'arrête, bien sûr.

- ✓ Si le dispositif d'archivage peut être compromis, alors on ne peut assurer, ni l'enregistrement effectif des détails d'une attaque, ni son intégrité.
- ✓ Si le module de contre-mesures est mis hors service, alors l'attaque peut continuer puisque l'outil est incapable de réagir. Il n'y a que l'administrateur qui puisse agir après notification de l'intrusion.
- ✓ Le module d'analyse peut être mis à bout de ressources. En déterminant ce qui demande le plus de ressources à l'outil, un attaquant peut le surcharger avec des activités inutiles. Une autre manière de faire, consiste à forcer l'outil à allouer toute la mémoire dont il dispose pour analyser des actions sans intérêt. Pour continuer à tourner, il sera amené à libérer de la mémoire en cessant, par exemple, la surveillance de connections restées inactives depuis longtemps.

Toute attaque sur l'une d'entre elles restera non détectée. Enfin, si on réussit à faire consommer à l'outil tout son espace mémoire pour des activités sans importance, il ne pourra plus stocker les événements intéressants. Il n'y aura donc pas de traces pour une intrusion ultérieure.

I.7 Conclusion

Dans cette partie, nous avons montré que la détection d'intrusions dans les réseaux ne vient pas concurrencer les mécanismes de sécurité traditionnels bien, au contraire, les compléter. Même si on ne peut pas atteindre la sécurité absolue, on voudrait au moins pouvoir détecter l'intrusion afin d'y remédier. Nous avons également présenté les principes mis en œuvre par les systèmes de détection d'intrusions pour atteindre leur but. Finalement, nous avons vu que de nombreux problèmes restent à résoudre avant que la détection d'intrusions soit fiable.

Cette technologie n'est pas encore arrivée à maturité et les outils existants ne sont pas toujours à la hauteur des besoins. Certaines approches théoriques doivent encore être validées dans la pratique. De nouvelles approches demandent encore à être approfondies, et vu le volume important de données d'audit à extraire et à classifier, une des techniques les plus estimées à l'heur actuelle est celle du ***Data Mining***, que nous allons étaler dans la partie suivante.

Chapitre II Introduction au Data Mining

II.1 Introduction

La croissance des moyens informatiques et de calcul induit à de nouvelles possibilités pour le stockage, le traitement et à l'analyse d'ensembles de données très volumineux. Cette évolution, ainsi que la popularité de nouvelles techniques algorithmiques et outils graphiques, conduisent au développement et à la commercialisation des solutions logicielles intégrant un sous-ensemble de méthodes à base d'heuristiques, d'apprentissages et de statistiques connues sous la terminologie de ***Data Mining***. [7]

Dans cette partie, nous allons exposer le principe de la fouille de données, ses taxinomies de méthodes, ses domaines d'applications et ses types d'apprentissages.

II.2 Définition

Le Data Mining traduit généralement en Fouille de données est un *ensemble de techniques d'exploration de données* permettant d'*extraire* d'une base de données des *connaissances* sous la forme de *modèles de description* afin de *décrire le comportement actuel* et/ou de *prédire le comportement futur du système représenté par ces données*.

Le Data Mining se situe à la croisée des bases de données, l'intelligence Artificielle, statistique et l'analyse de données :

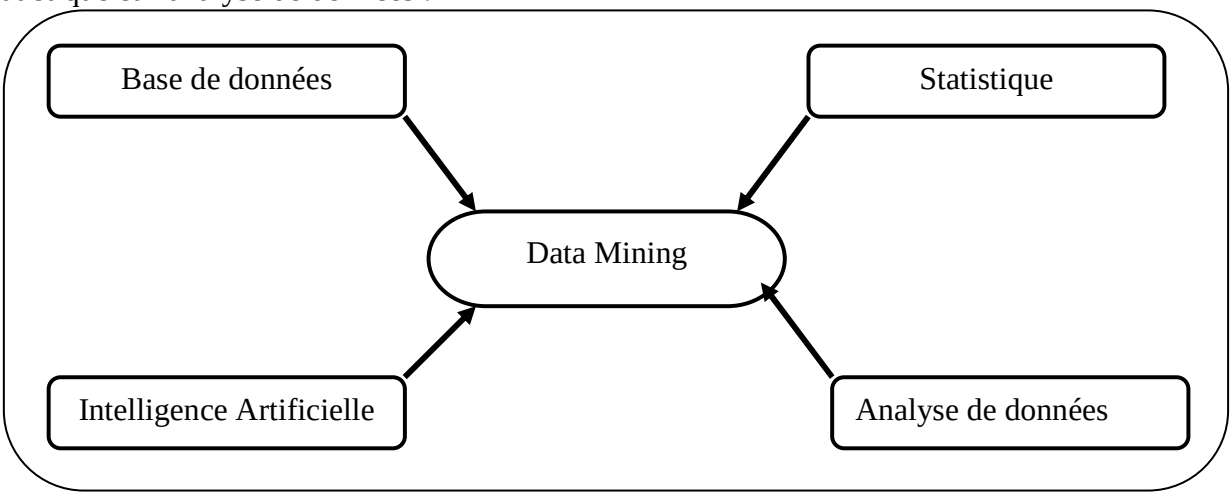

Figure 2 : Techniques utilisées

Le Data Mining impose d'être capable d'utiliser de manière opérationnelle les résultats des analyses effectuées, souvent dans des délais très courts. Le processus d'analyse doit permettre à l'organisation une réactivité importante.
Les données traitées sont issues de différent systèmes de stockage en place dans l'organisation et sont ainsi *hétérogènes*, *multiples*, plus ou moins *structurées*

Le Data Mining se propose donc d'extraire des connaissances à partir de grands volumes de données qui peuvent être *stockées* de manière diverse, dans des bases de données, dans un (ou plusieurs) entrepôt de données (data warehouse), mais qui peuvent aussi être *récupérées* de sources riches plus ou moins structurées comme Internet (Web), ou encore en temps réel (retrait d'argent dans un distributeur de billets...).

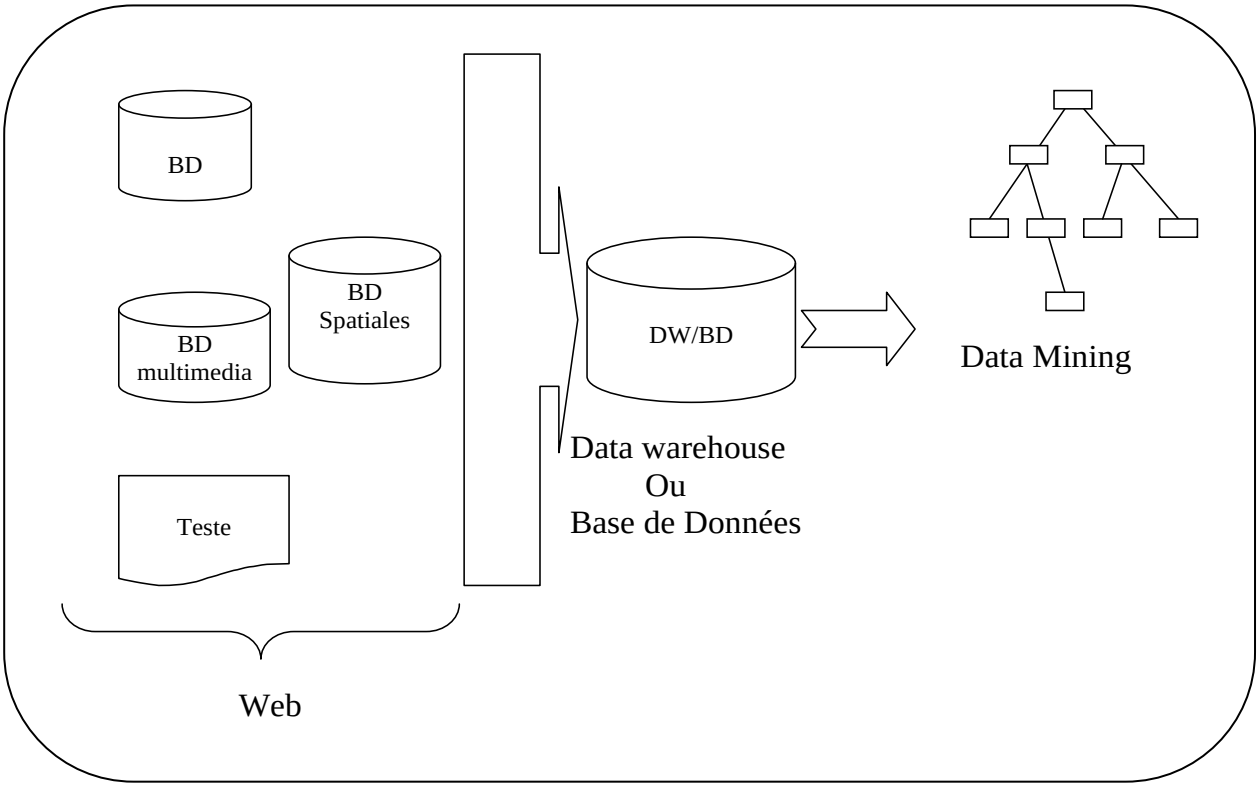

Figure 3 : Architecture du Data Mining

II.3 Principe et spécificité

II.3.1 Stratégie du Data Mining

Les statisticiens utilisent en générale des données expérimentales pour étudier un phénomène, en fixant certaines hypothèses. Par contre, le *Data Mining* utilise les données pour trouver toutes les hypothèses valables.

Le Data Mining tente alors de réaliser un arbitrage entre ***validité*** *scientifique*, ***interprétabilité*** des *résultats* et ***facilité*** *d'utilisation*, dans un environnement professionnel où le temps d'étude joue un rôle majeur et où les analystes ne sont pas toujours des statisticiens.

II.3.2 Méthodologie

Le Data Mining est un processus d'extraction de connaissances d'un large volume de données comportant les étapes suivantes : [42]

1. **Formaliser** ou identifier un problème que l'organisation cherche à résoudre en termes de données en cernant les objectifs.
2. **Accéder** aux données appropriées.
3. **Préparer** les données en vue des traitements et utilisations futures (collecter, nettoyer, enrichir, coder et normaliser les données).
4. **Modéliser** les données en leur appliquant des algorithmes d'analyse.
5. **Evaluer** et valider les connaissances ainsi extraites des analyses.
6. **Déployer** les analyses dans l'entreprise pour une utilisation effective.

Ce processus est *cyclique*, le Data Mining participe à une meilleure compréhension de l'activité de l'organisation, et à une rationalisation avancée du stockage de l'information et des données. Le Data Mining ne consiste pas en une succession d'études ad hoc mais doit bien permettre de capitaliser des connaissances acquises sous forme de connaissance explicites, donc de *structurer* les contenus nécessaires à l'ingénierie des connaissances. [42]

II.4 Taxinomie des méthodes

Afin de construire des modèles à partir des données, Le Data Mining se propose d'utiliser un ensemble *d'**algorithmes*** issus de disciplines scientifiques diverses (statistiques, intelligence artificielle, base de données). Donc, le but est de ***trouver*** des schémas « intéressants » (des patterns ou ***motifs de conception***) selon des critères fixés au départ, et ***extraire*** de ces données un maximum de connaissances utiles à l'entreprise.

Il existe trois typologies des méthodes qui sont :

II.4.1 Selon les objectifs

Selon des objectifs voulus, nous avons quatre types de méthodes :

1. ***Classification*** : Consiste à *examiner* les caractéristiques d'un objet et lui *attribuer* une classe (attribuer ou non un prêt à un client, établir un diagnostic,…). Donc, la classification consiste à apprendre à classer des objets dans des classes prédéfinies à partir d'exemples déjà classés. L'application de la classification construis un modèle à partir d'un ensemble d'apprentissage, où les éléments sont groupés dans des classes prédéfinies, et ensuite utilise ce modèle pour classifier automatiquement des nouveaux objets dans l'une des classes prédéfinies
2. ***Prédiction*** : Consiste à *prédire* la valeur future d'un attribut en fonction d'autres attributs (prédire la "qualité" d'un client en fonction de son revenu, …). Le but de la prédiction est d'attribuer à des objets une valeur pour une variable continue.
3. ***Association*** : Consiste à déterminer les attributs qui sont *corrélés*. Il s'agit de trouver des similarités ou des associations. L'application principale est "l'analyse du panier de la ménagère ", c'est à dire la recherche d'associations entre produits sur les tickets de caisse. Le but de la méthode est l'étude de ce que les clients achètent pour obtenir des informations sur les clients et pourquoi ils font certains achats. La méthode recherche les produits qui pourront être achetés ensemble. Les domaines applicables sont tous les secteurs pour lesquels il est intéressant de rechercher des groupements potentiels de produits.
4. ***Segmentation*** : Consiste à *former* des groupes homogènes à l'intérieur d'une population. Le clustering non supervisée consiste à chercher des groupes (ou classes) d'objets en se basant sur les caractéristiques de ces variables. Ces groupes doivent vérifier deux propriétés. D'une part, ils ne sont pas prédéfinis, mais découvert automatiquement au cours de l'opération. D'autres part, les clusters / groupes contiennent des objets ayant des caractéristiques similaires et séparent les objets ayant des caractéristiques différentes. On utilise souvent les algorithmes de types nués dynamiques (les centres mobiles, K-mens,. .) ou les cartes auto-organisatrices de Kohonen.

II.4.2 Selon le type d'apprentissage

Pour résoudre une problématique avec un processus de Data Mining, l'utilisation d'un grand nombre de méthodes et d'algorithmes différents s'impose. On peut distinguer trois grandes familles d'algorithmes :

1. **Les méthodes non-supervisées :** L'objectif de la méthode non supervisée est de trouver des relations entre les variables explicatives suffisamment significatives et permettant d'augmenter les connaissances du domaine étudié. Elles permettent de *travailler* sur un ensemble de données dans lequel aucune des données ou des variables à disposition n'a d'importance particulière par rapport aux autres, c'est-à-dire un ensemble de données dans lequel aucune variable n'est considérée individuellement comme la cible de l'objectif de l'analyse. On les utilise par exemple pour dégager d'un ensemble d'individus des groupes homogènes (typologie), pour construire des normes de comportements et donc des déviations par rapport à ces normes (détection de fraudes nouvelles ou inconnues à la carte bancaire, à l'assurance maladie...), pour réaliser de la compression d'informations (compression d'image).

   Parmi les techniques disponibles, nous avons :

   - Techniques à base de ***Réseau de neurones*** : réseau de Kohonen (SOM/TOM) (Carte Auto Adaptative), les réseaux Hebbienx...
   - Techniques utilisées classiquement dans le monde des statistiques : ***classification ascendante hiérarchique***, ***k-means*** et les ***nuées dynamiques*** (Recherche des plus proches voisins), les ***classifications mixtes*** (Birch...), les ***classifications relationnelle***s...
   - Les techniques dites de ***recherche d'associations :***elles sont à l'origine utilisées pour faire de l'analyse dite de panier d'achats ou de séquences, c'est-à-dire pour essayer de savoir parmi un ensemble d'achats effectués par un très grand nombre de clients et de produits possibles, quels sont les produits qui sont achetés simultanément (pour un supermarché par exemple ; elles sont également appliquées à des problèmes d'analyse de parcours de navigation de site web).: algorithmes apriori,GRI, Carma, méthode ARD...

2. **Les méthodes supervisées :** Dans le cadre de l'approche supervisée, les données sont constituées d'un ensemble de caractéristiques, appelées variables exogènes(ou variable explicative), décrivant chaque individu. Chaque individu possède une caractéristique particulière appelée variable endogène (variable à expliquer ou variable cible). L'objectif

de la fouille de données supervisée est de trouver des relations entre les variables exogènes permettant d'expliquer et/ou de prévoir le comportement de la variable endogène.

La fouille de données supervisée se décompose en deux étapes : apprentissage d'un modèle sur une partie des données et validation du modèle sur l'autre partie des données. Les données sont donc divisées en deux ensembles : l'ensemble d'apprentissage et l'ensemble de test.

Par exemple, on utilisera ce type de méthode lorsque l'on cherchera à comprendre pourquoi un individu a acheté un produit plutôt qu'un autre, pourquoi un individu a répondu favorablement à une opération de marketing direct, pourquoi un individu a contracté une maladie particulière.

Parmi les techniques disponibles, nous avons :

- Techniques à base ***d'arbres de décision***: CART, CHAID, ECHAID, QUEST, C5, C4.5, les forêts d'arbres...
- Techniques statistiques de ***régressions linéaires*** et ***non linéaires*** au sens large : Régression linéaire, Régression linéaire multiple, Régression logistique binaire ou multinomiale, Probit binaire, multinomial ou ordonné, Tobit, Cauchit, modèle Gamma, binomial négatif, log-log, Analyse discriminante linéaire ou quadratique, régression de cox, modèle linéaire généralisé, régression PLS, régressions non paramétrique, équations structurelles.
- Techniques à base de ***Réseau de neurones*** : Perceptron mono ou multicouches avec ou sans rétropropagation des erreurs, les réseaux à fonction radiale de base...
- Techniques à base d'***algorithme génétique***.
- Techniques à base *d'**Inférence bayésienne*** (Réseau bayésien).
- Le ***Raisonnement par cas***
- Le ***Filtrage collaboratif***

3. **Les méthodes de réduction de données :** Elles permettent de réduire un ensemble de données volumineux à un ensemble de taille plus réduite, épuré de ce que l'on considérera comme de l'information non pertinente ou non signifiante, comme du bruit. Elles sont ainsi très souvent, mais pas systématiquement, utilisées en amont des techniques supervisées ou non supervisées. Elles sont notamment très complémentaires des techniques non supervisées classiquement utilisées dans le domaine de statistiques.

- Techniques ***d'Analyse factorielle*** : Analyse en composantes principales, analyse factorielle des correspondances, analyse des correspondances multiples, analyses factorielles (maximum de vraisemblance, moindres carrés non pondérés, avec ou sans rotation orthogonale ou oblique)...
- Techniques de ***positionnement*** : positionnement multidimensionnel...

II.4.3 Selon les types de modèles obtenus

1. ***Modèles prédictifs*** : Utilisent les données avec des résultats connus pour développer des modèles permettant de prédire les valeurs d'autres données (modèle permettant de prédire les clients qui ne rembourseront pas leur crédit, classification, prédiction). Ils s'intéressent à la recherche de structures ou modèles destinés à la prévision de certaines quantités.
2. ***Modèles descriptifs*** : Proposent des descriptions des données pour aider à la prise de décision. Les modèles descriptifs aident à la construction de modèles prédictifs (segmentation, Association). Ils s'intéressent aux individus ainsi qu'aux groupes homogènes qu'ils peuvent former.

II.5 Domaines d'application

Parmi les domaines d'application du Data Mining, on trouve, le domaine d'assurance pour l'analyse des clients à haut risque, le service financiers pour l'attribution d'un crédit bancaire, la décision de fraude et le marketing ciblé, comme nous avons le domaine de grande distributions dans le profit des consommateurs, constitution des rayons, ainsi que le domaine de médecine dans l'aide au diagnostique.

Les principales applications de Data Mining pour le commerce permettent d'en savoir plus sur les clients et sur leurs besoins, de comprendre les éléments générateurs des ventes et les facteurs qui les affectent aussi, de concevoir des stratégies marketing et de mettre au point les futurs indicateurs de l'activité.

II.6 Techniques du Data Mining

II.6.1 Classification ascendante hiérarchique (CAH)

Il s'agit de regrouper tous les individus en une seul classe à la racine, en commençant par le bas (les deux plus proches) et en construisant progressivement un arbre ou *dendrogramme*. L'utilisateur de cette méthode doit faire un choix supplémentaire : comment définir la distance entre deux groupes connaissant celles de tous les couples d'individus avec ces deux groupes. Car à chaque étape ou regroupement, on calcul la distance entre un individu et un groupe ou la distance entre deux groupes [23].

Le nombre de classes est déterminé *a posteriori*, à la vue du dendrogramme ou d'un *graphique* représentant la décroissance de la hauteur de chaque saut, ou écart de distance, opéré à chaque regroupement.

La classification ascendante hiérarchique emploie l'algorithme suivant [23] :

1. Initialement, on dispose de n individus qui constituent les éléments à classer.
2. Construire la matrice de distances entre les n éléments.
3. Tant que le nombre d'éléments est supérieur à 1
   (a) Identifier les deux éléments les plus proches et les agréger en un nouvel élément au sens d'un critère d'agrégation d ;
   (b) Mettre à jour la matrice des distances en calculant les distances entre le nouvel élément et les éléments restants ;

La structure générale de cet algorithme est simple puisqu'il s'agit d'une boucle qui s'exécute *jusqu'à* ce qu'il ne reste plus qu'un seul élément regroupant tous les objets.
L'algorithme aboutit à une hiérarchie indicée de partitions, se présentant sous la forme d'arbres appelés dendrogrammes et contenant (n − 1) partitions (pour un ensemble de n objets).

II.6.2 Réallocation dynamique

Dans ce cas, par un tirage aléatoire, on initialise un k centre de classe, où k est fixé à *priori*, tous les individus sont affectés à la classe dont le centre est le plus proche au sens de la distance choisie (en principe, euclidienne pour cette méthode). Dans une deuxième étape, l'algorithme calcule des barycentres de ces classes qui deviennent les nouveaux centres. Le procédé (affectation de chaque individu à un centre, détermination des centres) est itéré jusqu'à convergence vers un minimum (local) ou un nombre d'itérations maximum fixé.

II.6.3 Classification mixte

La CAH nécessite impérativement la construction d'un tableau de dimension (n × n) et son stockage en mémoire ; le nombre maximum d'individus traités peut s'en trouver limité. Ce n'est pas le cas dans l'algorithme de réallocation, d'où l'intérêt possible d'une approche mixte pour, à la fois, classer de grands volumes de données et sélectionner le nombre de classes par CAH.

II.6.4 Réseaux de neurones

De façon formelle, un Réseau de Neurone (RN) est une fonction mathématique à laquelle sont associés des variables, un résultat et des paramètres ajustables (poids). A partir d'un ensemble de données (mesures, résultats de calcul, indices économiques ou financiers, etc.) on peut ajuster un RN en choisissant convenablement ses paramètres [27].

La *propriété fondamentale* des Réseaux de Neurones est *l'émergence* d'un comportement globale intelligent à partir de comportements de base élémentaires en plus de leurs *capacités d'apprentissage*, fait d'eux une représentation mathématique très avantageuse pour la modélisation statique et dynamique des processus.

L'idée principale des réseaux de neurones est la suivante :
On se donne une unité simple, un *neurone*, qui est capable de réaliser quelques calculs élémentaires. On relie ensuite entre elles un nombre important de ces unités et on essaye de déterminer la puissance de calcul du *réseau* ainsi obtenu. Il est important de noter que ces neurones manipulent des données **numériques** et non pas symboliques [13].

1 Neurone formel

Un neurone est une unité de calcul élémentaire dont le modèle est issu de certains principes de fonctionnement du neurone biologique. Sa première modélisation date des années quarante. Elle a été présentée par Mac Culloch et Pitts dans [13]. S'inspirant de leurs travaux sur les neurones biologiques, ils ont proposé le modèle suivant :

Un neurone formel fait une somme pondérée des potentiels d'actions qui lui parviennent, puis s'active suivant une valeur de cette addition pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien [13].

Ce neurone formel est caractérisé par :

- $(x_i)i=1...n$ les entrées du neurone formel ;
- S sa sortie ;
- ß son seuil (ou biais) ;
- W_i les paramètres de pondération ou poids synaptiques associé à la ième entrée ;
- f la fonction de seuillage :

$$f(x) = \begin{cases} 1 \text{ si } x > ß \\ 0 \text{ sinon} \end{cases}$$

Le fonctionnement du neurone formel est alors :

$$S = f(\sum_{i=1}^{n} W_i x_i)$$

Un neurone *réalise* simplement une fonction non linéaire, paramétrée de ses variables d'entrée. L'intérêt des neurones réside dans les propriétés qui résultent de leurs associations aux réseaux, c'est-à-dire de la composition des fonctions non linéaires réalisées par chacun des neurones. On distingue deux types de réseaux de neurones : les réseaux *non bouclés* (réalise une ou plusieurs fonctions de ses entrées, par composition des fonctions réalisées par chacun des neurones) et les réseaux *bouclés* (le graphe des connexions peut contenir des cycles) [18].

2 Schémas d'un réseau de neurone

Un réseau de neurones est ainsi constitué de cellules (ou neurones), connectée entre elles par des liaisons affectées de poids. Ces liaisons permettent à chaque cellule de disposer d'un canal pour envoyer et recevoir des signaux en provenance d'autres cellules du réseau. Chacune de ces connexions reçoit un poids (une pondération), qui détermine sont impact sur les cellules qu'elle connecte. Chaque cellule dispose ainsi d'une entrée, qui lui permet de recevoir de l'information d'autres cellules, mais aussi de ce que l'on appelle une *fonction d'activation*, qui est dans les cas les plus simples, une simple identité du résultat obtenu par l'entrée et une sortie. Ainsi, pour un réseau de neurones avec N cellules dans la première couche, notées W(1),...,W(N) , et N poids affectées aux liaisons et notés x(1),...,x(N) l'entrée

d'une cellule de la seconde couche sera généralement une somme pondérée des valeur de sortie des neurones précédents :

$$X = w(1)*x(1) + w(2)*x(2) + w(3)*x(3) + ... + w(N)*x(N)$$

Pour obtenir la valeur de sortie Y du neurone concerné, nous pouvons utiliser une fonction d'activation identité du type :

$$Y = d*X$$

Mais le choix d'une fonction d'activation se révèle être un élément constitutif important des réseaux de neurones. Ainsi, l'identité n'est pas toujours suffisante, bien au contraire, et le plus souvent des fonctions non linéaires et plus évoluées seront nécessaires.

L'exemple le plus simple de réseau de neurones est souvent donné par le perceptron multicouche (qui est un cas particulier de réseau de neurones). Dans un perceptron, plusieurs couches contenant des neurones sont connectées entre elles de l'entrée vers la sortie. Afin d'illustre un peu ces propos, la figure suivante représente le schéma type d'un perceptron à trois couches [18]:

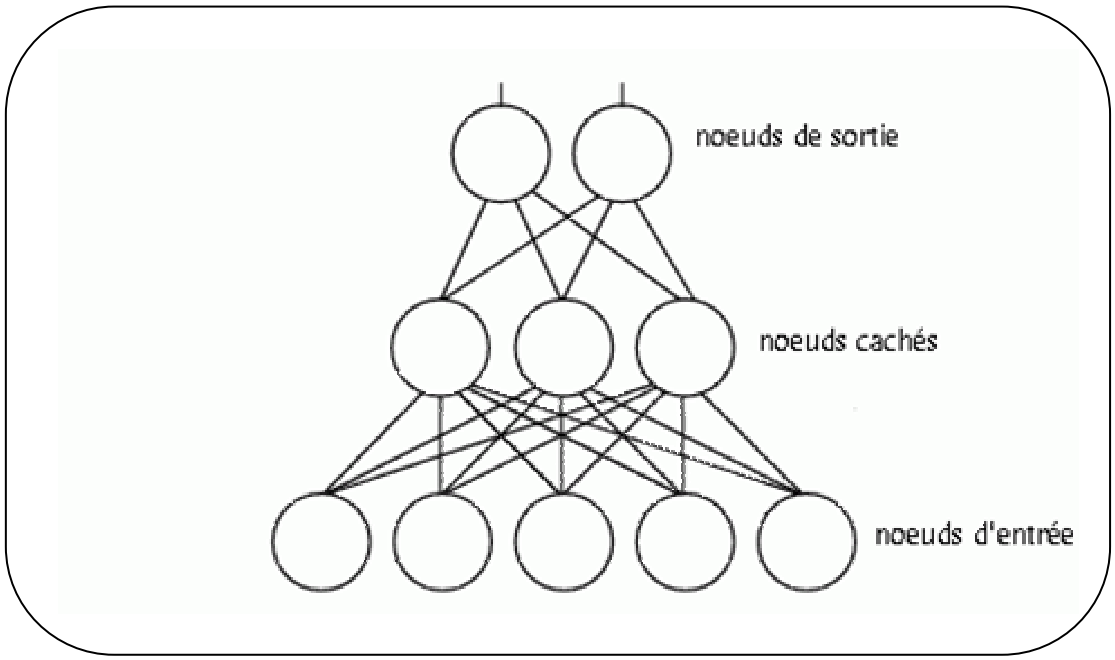

Figure 4 : Schémas d'un perceptron

- **Les nœuds d'entrée**

La première couche est appelée couche d'entrée. Elle recevra les données source que l'on veut utiliser pour l'analyse. Sa taille est donc directement déterminée par le nombre de variables d'entrées.

- **Les nœuds cachés**

La seconde couche est une couche *cachée*, en ce sens qu'elle n'a qu'une utilité intrinsèque pour le réseau de neurones et n'a pas de contact direct avec l'*extérieur*. Les fonctions d'activations sont en général non linéaires sur cette couche mais il n'y a pas de règle à respecter. Le choix de sa taille n'est pas implicite et doit être ajusté. En général, on peut commencer par une taille moyenne des couches d'entrée et de sortie mais ce n'est pas toujours le meilleur choix. Il sera souvent préférable pour obtenir de bons résultats, d'essayer le plus de tailles possibles.

- **Les nœuds de sortie**

La troisième couche est appelée couche de sortie. Elle donne le résultat obtenu après compilation par le réseau des données entrées dans la première couche. Sa taille est directement déterminée par le nombre de variables qu'on veut en sortie.

3 Les étapes de la conception d'un réseau de neurone

Pour construire un réseau de neurones, la première chose à faire est de bien choisir ses échantillons de données d'apprentissage, de tests et validation. Ce n'est qu'ensuite que le choix du type de réseau interviendra. Afin de clarifier un peu les idées, voici chronologiquement les quartes grandes étapes qui doivent guider la création d'un réseau de neurones :

- **Choix et préparation des échantillons**

Le processus d'élaboration d'un réseau de neurones commence toujours par le choix et la préparation des échantillons de données. Comme dans les cas d'analyse de données, cette étape est cruciale et va aider le concepteur à déterminer le type de réseau le plus approprié pour résoudre sont problème. La façon dont se présente l'échantillon conditionne : le type de réseau, le nombre de cellules d'entrée, le nombre de cellules de sortie et la façon dont il faudra mener l'apprentissage, les tests et la validation.

- **Elaboration de la structure du réseau**

La structure du réseau dépend étroitement du type des échantillons. Il faut d'abord choisir le type de réseau : un perceptron standard, un réseau de Hopfield, un réseau à décalage temporel (TDNN), un réseau de Kohonen, un ARTMAP etc... Dans le cas du

perceptron par exemple, il faudra aussi choisir le nombre de neurones dans la couche cachée. Plusieurs méthodes existent et on peut par exemple prendre une moyenne du nombre de neurones d'entrée et de sortie, mais rien de vaut de tester toutes les possibilités et de choisir celle qui offre les meilleurs résultats.

- **Apprentissage**

L'apprentissage consiste tout d'abord à calculer les pondérations optimales des différentes liaisons, en utilisant un échantillon. La méthode la plus utilisée est la rétropropagation : on entre des valeurs des cellules d'entrée et en fonction de l'erreur obtenue en sortie (le delta), on corrige les poids accordés aux pondérations. C'est un cycle qui est répété jusqu'à ce que la courbe d'erreurs du réseau ne soit croissante (il faut bien prendre garde ne pas sur-entrainer un réseau de neurones qui deviendra alors moins performant). Il existe d'autres méthodes d'apprentissage telles que le quickprop par exemple.

- **Validation et Tests**

Alors que les tests concernent la vérification des performances d'un réseau de neurones hors échantillon et sa capacité de généralisation, la validation est parfois utilisée lors de l'apprentissage. Une fois le réseau calculé, il faut toujours procéder à des tests afin de vérifier que notre réseau réagit correctement. Il y a plusieurs méthodes pour effectuer une validation : la cross-validation, le bootstrapping... mais pour les tests, dans le cas général, une partie de l'échantillon est simplement écarté de l'échantillon d'apprentissage et conservé pour les tests hors échantillon. On peut par exemple utiliser 60% de l'échantillon pour l'apprentissage, 20% pour la validation et 20% pour les tests. Dans les cas de petits échantillons, on ne peut pas toujours utiliser une telle distinction, simplement parce qu'il n'est pas toujours possible d'avoir suffisamment de données dans chacun des groupes ainsi crées. On a alors parfois recours à des procédures comme la cross-validation pour établir la structure optimale du réseau.

4 Principe d'un réseau de neurone

L'unité de calcul combine des entrées réelles $x_1,\ldots,x_n$ en une sortie réelle y_i. Les entrées n'ont pas toutes la même importance et à chaque entrée x_i est associée un poids C_i, l'unité calcule d'abord l'activité d'entrée. En générale, pour le neurone formel, l'activité en entrée est mesurée par la somme pondérée des entrées $\sum_{j=1\text{-}n} W_{ij}\, x_j$

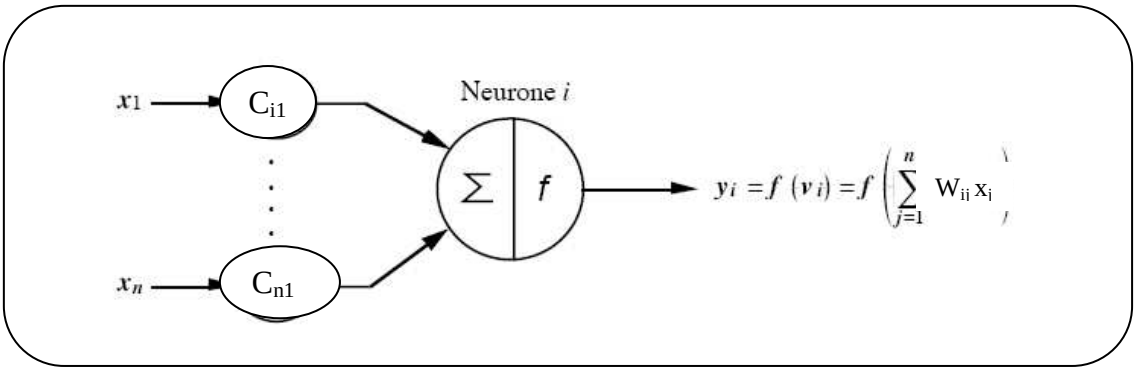

Figure 5 : Un réseau formel

Les Réseaux de Neurones possèdent une propriété fondamentale qui les distingue des techniques classiques de traitement des données : ce sont des *approximateurs* universels parcimonieux. Cela signifie que pour obtenir un modèle non linéaire de précision donnée, un Réseau de Neurone a besoin de moins de paramètres ajustables que les méthodes de régression classiques (par exemple la régression polynomiale). Or le nombre de données nécessaires pour ajuster le modèle est directement lié au nombre de ses paramètres [27].

5 Types d'apprentissage

1 - Le mode supervisé

Dans ce type d'apprentissage, le réseau s'adapte par comparaison entre le résultat qu'il a calculé, en fonction des entrées fournies, et la réponse attendue en sortie. Ainsi, le réseau va se modifier jusqu'a ce qu'il trouve la bonne sortie, c'est-à-dire celle attendue, correspondant à une entrée donnée.

2 - Le renforcement

Certains auteurs classe le renforcement dans la catégorie des modes supervisés. Dans cette approche le réseau doit apprendre la corrélation entrée/sortie via une estimation de son erreur, c'est-à-dire du rapport échec/succès. Le réseau va donc tendre à maximiser un index de performance qui lui est fourni, appelé signal de renforcement. Le système étant capable ici, de savoir si la réponse qu'il fournit est correcte ou non, mais il ne connait pas la bonne réponse.

3 - Le mode non-supervisé (ou auto-organisationnel)

Dans ce cas, l'apprentissage est basé sur des probabilités. Le réseau va se modifier en fonction des régularités statistiques de l'entrée et établir des catégories, en attribuant et en optimisant une valeur de qualité, aux catégories reconnues.

4 - Le mode hybride

Le mode hybride reprend en fait les deux autres approches, puisque une partie des poids va être déterminée par apprentissage supervisé et l'autre partie par apprentissage non-supervisé.

Règles d'apprentissage

1 - Règle de correction d'erreurs

Cette règle s'inscrit dans le paradigme d'apprentissage supervisé, c'est à dire dans le cas où l'on fournit au réseau une entrée et la sortie correspondante. Si on considère **y** comme étant la sortie calculée par le réseau, et **d** la sortie désirée, le principe de cette règle est d'utiliser l'erreur **(d-y)**, afin de modifier les connexions et de diminuer ainsi l'erreur globale du système. Le réseau va donc s'adapter jusqu'à ce que y soit égal à d. Ce Principe est notamment utilisé dans le modèle du perceptron simple.

2 - Apprentissage de Boltzmann

Les réseaux de Boltzmann sont des réseaux symétriques récurrents. Ils possèdent deux sous-groupes de cellules, le premier étant relié à l'environnement (cellules dites visibles) et le second ne l'étant pas (cellules dites cachées). Cette règle d'apprentissage est de type stochastique (qui relève partiellement du hazard) et elle consiste à ajuster les poids des connexions, de telle sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée.

3 - Règles de Hebb

Cette règle est basée sur des données biologiques, modélise le fait que si des neurones, de part et d'autre d'une synapse, sont activés de façon synchrone et répétée, la force de la connexion synaptique est croissante. Il est à noter que l'apprentissage est localisé, c'est-à-dire que la modification d'un poids synaptique wij ne dépend que de l'activation d'un neurone i et d'un autre neurone j.

4 - Règle d'apprentissage par compétitions

La particularité de cette règle, c'est qu'ici l'apprentissage ne concerne qu'un seul neurone. Le principe de cet apprentissage est de regrouper les données en catégories. Les patrons similaires vont donc être rangés dans une même classe, en se basant sur les corrélations des données, et seront représentés par un seul neurone, on parle de « winner-take-all ».

Dans un réseau à compétition simple, chaque neurone de sortie est connecté aux neurones

de la couche d'entrée, aux autres cellules de la couche de sortie (connexions inhibitrices) et à elle-même (connexion excitatrice). La sortie va donc dépendre de la compétition entre les connexions inhibitrices et excitatrices.

II.6.5 Algorithmes génétiques

Un algorithme génétique est un *algorithme stochastique itératif* qui opère sur des ensembles de points *codés*, à partir d'une *population initiale*, et qui est construite à l'aide de trois opérateurs : *croisement*, *mutation*, *sélection* qui sont la base de l'algorithme génétique [54]. Dans la littérature on parle d'opérateurs de ***reproduction***. Le principe de l'algorithme génétique est basé sur trois phases :

1. **La genèse** qui est l'initialisation aléatoire d'individus pour la population de la première génération.
2. **La reproduction** qui est l'évolution des individus de la génération courante vers la suivante dont nous avons [48]:
    a. La ***sélection*** des individus reproducteurs. c'est la première opération dans un algorithme génétique. Au cours de cette opération l'algorithme ***sélectionne*** les éléments pertinents qui optimisent mieux la fonction.
    b. Le ***croisement*** génétique de ces individus pour la création de nouveaux individus. Il permet de *générer* deux chromosomes nouveaux "*enfants*" à partir de deux chromosomes sélectionnés "*parents*"
    c. La ***mutation*** de certains individus pour que le pool génétique ne s'affaiblisse pas. Elle réalise l'inversion d'un ou plusieurs gènes d'un chromosome [48]
    d. L'***évaluation*** des individus par le calcul de leur fitness.
3. **Recueil du meilleur individu** qui est la recherche de l'individu le plus adapté selon les critères souhaités.

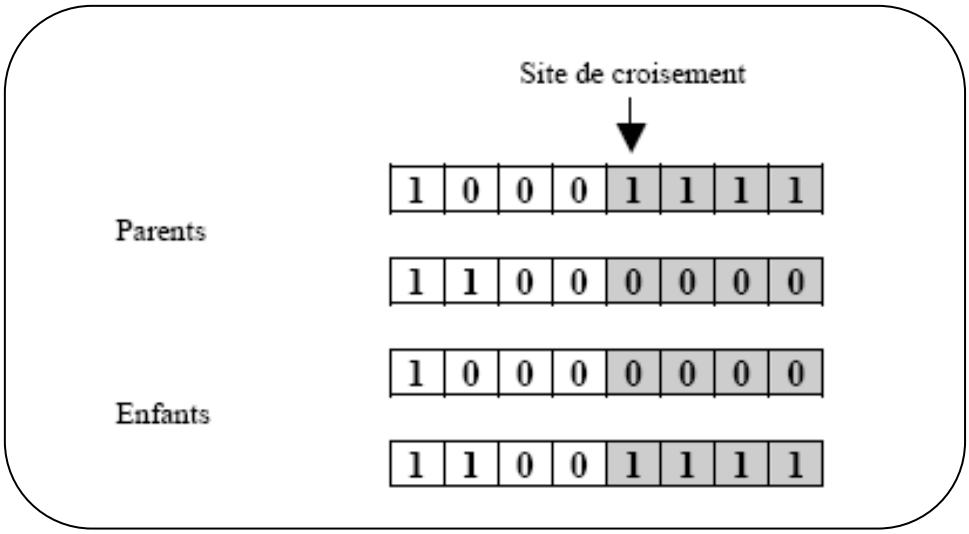

Figure : Opération de Croisement

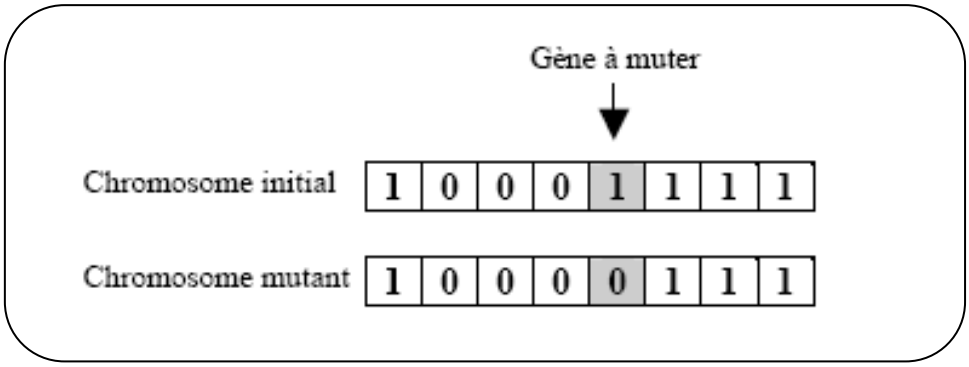

Figure 6 : Opération de Mutation

Un algorithme génétique de base passe par les opérations suivantes : [19]

Génération aléatoire de la population initiale

Calcul de la fonction sélective

Répéter

Sélection

Croisement

Mutation

Calcul de la fonction sélective

***Jusqu'à** satisfaction du critère d'arrêt*

L'application de l'Algorithme Génétique à la résolution d'un problème nécessite le *codage* des solutions potentielles à ce problème en des chaînes finies de bits afin de constituer les chromosomes, de *trouver* une fonction sélective permettant une bonne discrimination entre les chromosomes et de *définir* les opérateurs génétiques qui seront utilisés [35].

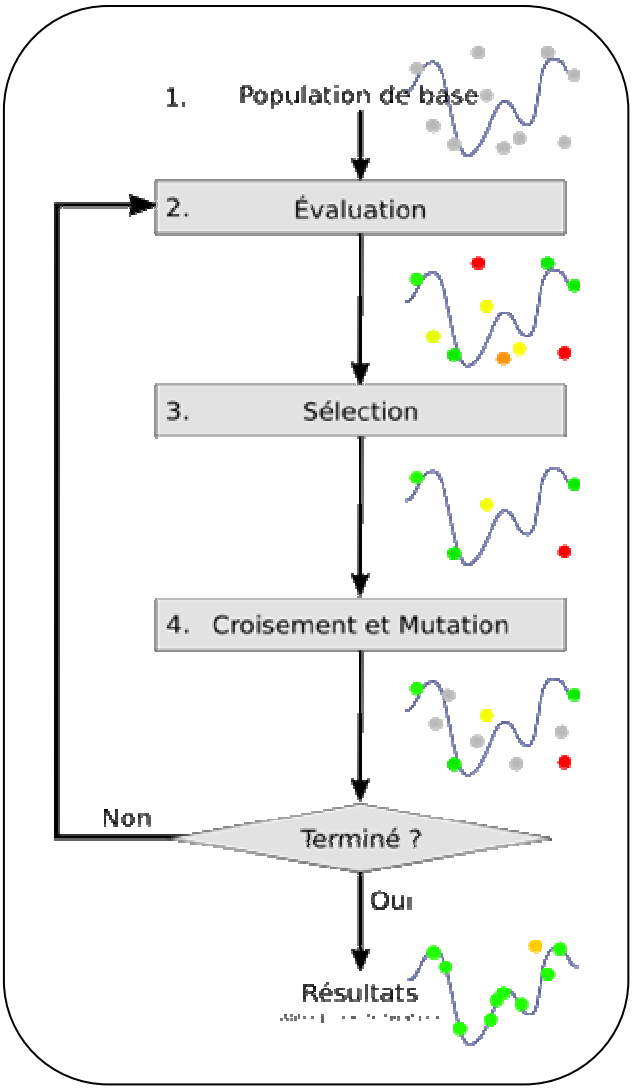

Figure 7 : Opération d'un Algorithme Génétique

Principes généraux

La population initiale est définie comme un ensemble de N individus. Chaque individu est représenté par un ou plusieurs chromosomes. Chaque chromosome comprend un nombre de gènes égal aux variables à optimiser et dont la valeur est fournie par le tableau d'entrée.

1 Codage des variables

Le codage le plus utilisé est binaire car on peut facilement coder et décoder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes, des chaînes de caractères. Chaque gène est donc représenté par une chaîne de bits.

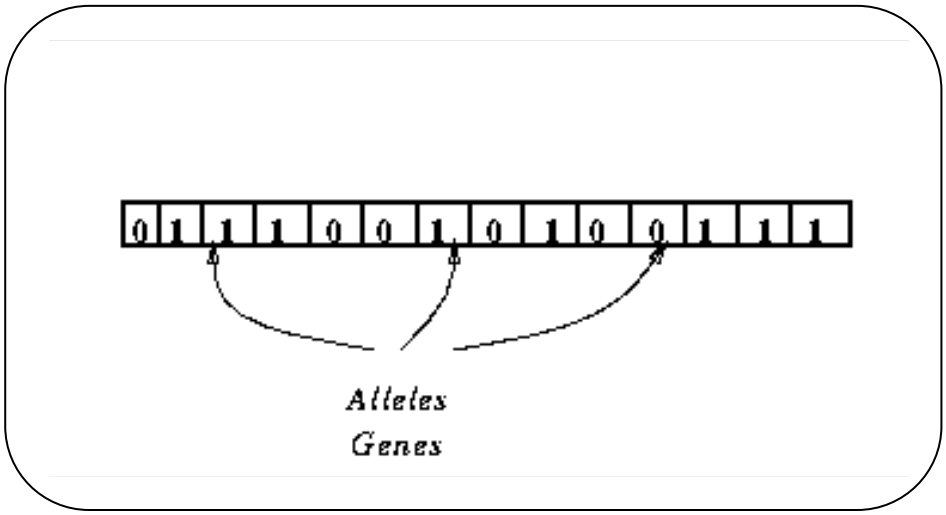

Figure 8 : Représentation du gène

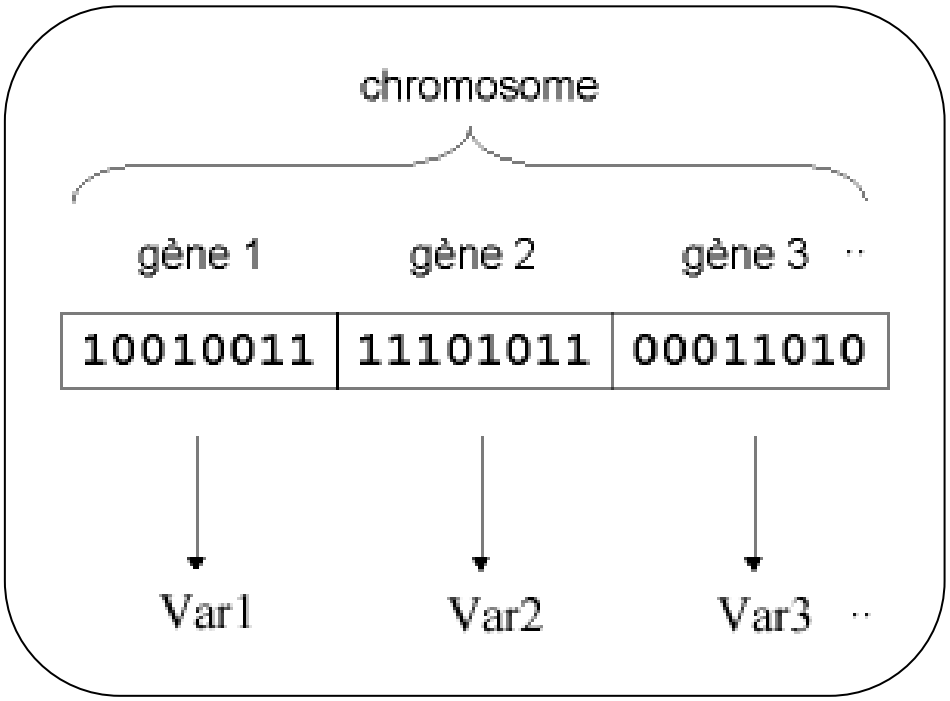

Figure 9 : Chromosome

2 Création d'une population initiale

La génération est en général aléatoire. Mais peut être également dirigée.

3 Définition de la fonction fitness

La fitness mesure, la qualité de l'individu exprimée sous forme d'un nombre ou d'un vecteur.

On dit qu'un individu i est meilleur que l'individu j quand i est plus proche de la solution que j.

4 Evaluation de tous les individus de la population initiale

Les individus sont classés suivant la valeur de leur fitness.

5 Sélection des parents

1 La méthode sélectif : La probabilité de reproduction d'un individu est proportionnelle à sa fitness relative

2 La méthode aléatoire

6 Génération de nouveaux individus.

Pour assurer sa reproduction, on fait appel à des opérateurs génétiques.

1 Mutation : Cet opérateur substitue un ou plusieurs bits d'un individu aléatoirement par une nouvelle valeur (0 ou 1).

2 Croisement (Crossover) : Cet opérateur échange des parties de bits d'un individu avec les parties correspondantes d'un autre individu.

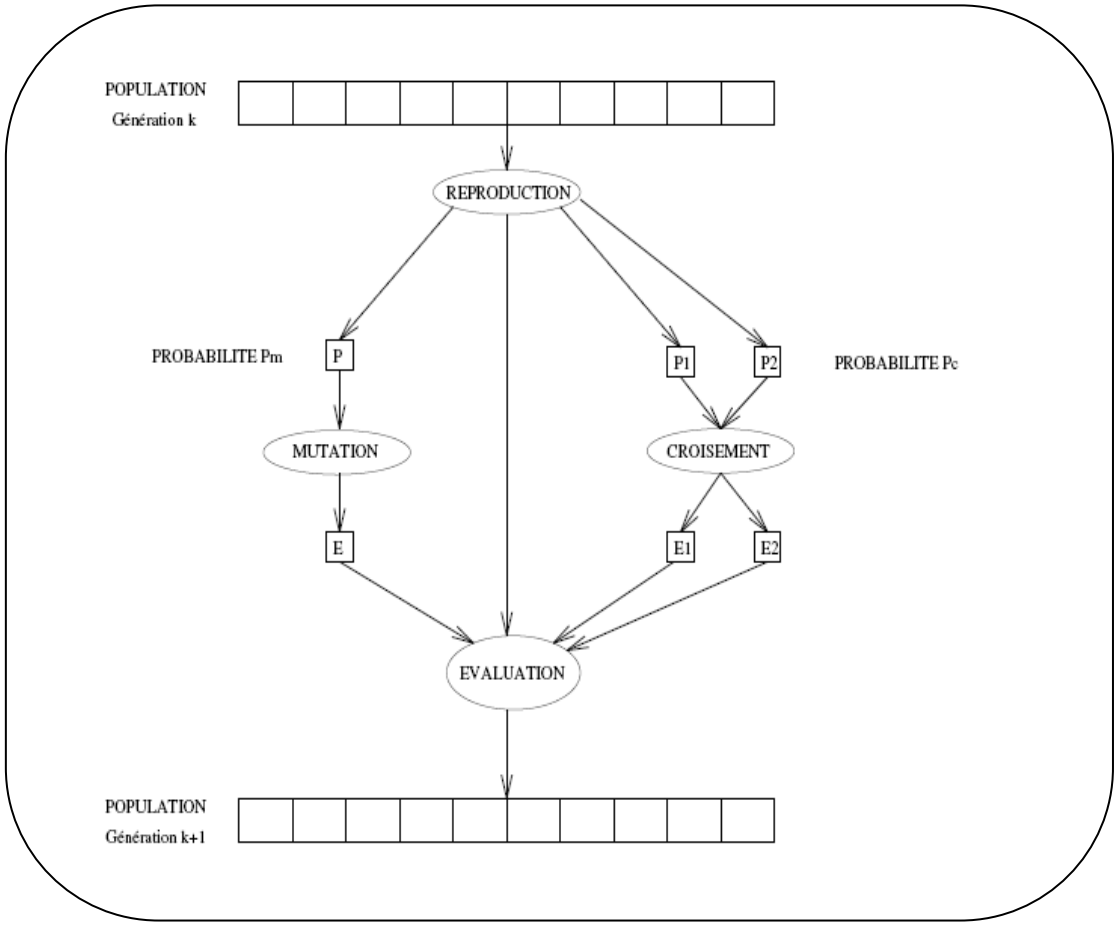

Figure 10 : Principe générale d'un Algorithme Génétique

Le principe général du fonctionnement d'un algorithme génétique est représenté comme suit : On commence par *générer* une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération k+1, les opérations suivantes sont répétées pour tous les éléments de la population k : [2]

1-Des couples de parents P1 et P2 sont *sélectionnés* en fonction de leurs adaptations.

2-L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0,6) et *génère* des couples d'enfants C1 et C2.

3-D'autres éléments P sont *sélectionnés* en fonction de leur adaptation. L'opérateur de *mutation* leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et génère des individus mutés $P^{'}$.

4-Le niveau d'adaptation des enfants (C1, C2) et des individus mutés P' sont ensuite *évalués* avant insertion dans la nouvelle population.

5-Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

II.6.6 Arbres de décision

Un **arbre de décision** est un outil d'aide à la décision et à l'exploration de données. Il permet de modéliser simplement, graphiquement et rapidement un phénomène mesuré plus ou moins complexe.

Un arbre de décision est une représentation graphique d'une procédure de classification. Les nœuds internes de l'arbre sont des tests sur les champs ou attributs, les feuilles sont les classes. Lorsque les tests sont binaires, le fils gauche correspond à une réponse positive au test et le fils droit à une réponse négative [47].

1 Construction d'un arbre de décision

Un arbre de décision permet de classer des objets en sous-classes par divisions hiérarchiques, dans lequel un nœud représente une sous-classe du nœud parent et un arc représente un prédicat de placement des objets de la classe parente dans la sous-classe [23].

Pour construire un arbre de décision, on se base sur la division récursive de l'échantillon d'apprentissage en plusieurs sous-ensembles (dans le cas d'un arbre n-aire) ou en deux sous-ensembles (dans le cas d'un arbre binaire), de telle sorte que les nœuds descendants soient plus homogènes que les nœuds parents et qu'ils soient les plus différents possibles entre eux vis-à-vis de la variable à expliquer [23].

Les nœuds internes qui engendrent des nœuds descendants immédiats sont des tests sur les variables explicatives. Quand l'arbre est binaire, le fils gauche correspond à une réponse positive au test et le fils droit à une réponse négative. Les nœuds terminaux ou feuilles dénotent une décision ou bien une branche spécifiant un test sur une valeur d'un attribut. Le nombre de descendants de chaque nœud dépend des résultats du test effectué à ce niveau [47]. Les différentes phases de construction de l'arbre sont [23] :

1. Établir pour chaque nœud l'ensemble des divisions admissibles. : Une division est dite admissible si aucun des deux nœuds descendants qui en découlent n'est vide. Le critère de division repose sur la définition d'une fonction d'hétérogénéité ou de désordre.
2. Définir un critère permettant de sélectionner la "meilleure" division d'un nœud parmi toutes celles admissibles pour les différentes variables.
3. Définir une règle permettant de décider qu'un nœud est terminal (feuille).
4. Affecter chaque feuille à l'une des classes (cas de la discrimination) ou à une valeur de la variable à expliquer (cas de la régression).
5. Estimer le taux d'erreur associé à l'arbre.
6. Élaguer l'arbre de décision obtenu : Il est possible de poursuivre la croissance de l'arbre jusqu'à obtention d'un arbre complet A_{max} d'erreur réelle mesurée sur l'ensemble d'apprentissage la plus petite possible. Pour éviter le phénomène de sur-apprentissage, on applique une procédure d'élagage qui est basée sur la suppression successive des branches les moins informatives en terme de discrimination entre les classes ou en terme d'explication de la variable a expliquer [23]. Ainsi, on obtient un arbre plus petit avec un meilleur pouvoir de généralisation.

2 Critères d'arrêt dans les arbres de décision

Le nœud courant est terminal si :

1. Il n'y a plus d'attributs disponibles, c'est-à-dire que sur le chemin menant de la racine au nœud courant tous les tests disponibles ont été utilisés.
2. Tous les exemples de l'échantillon courant sont dans une même classe.
3. La proportion d'observations d'une classe est supérieure à un seuil prédéfini.
4. Il n'existe pas de test ayant au moins k éléments sur deux branches. L'objectif de ce critère est d'éviter une croissance trop grande de l'arbre par l'exploration de branches comprenant trop peu d'exemples.
5. **Critère de la variance résiduelle minimale** : Lorsque la variable Y est continue, le critère de sélection de la meilleur division d'un nœud est fondé sur la variance de Y

dans les segments descendants. Cette variance doit être plus faible que la variance de Y dans le nœud parent.

Pour une division δ d'un nœud t par une variable X_j, la variance résiduelle de t est :

$$Var(\delta, t) = \frac{n_g}{n_t} Var(t_g) + \frac{n_d}{n_t} Var(t_d)$$

où n_g, n_d, n_t sont respectivement les effectifs des segments t_g, t_d et t. Et $Var(t_g)$ et $Var(t_d)$ sont les variances de Y à l'intérieur des segments t_g et t_d.

La meilleure division d^*_j réalisée par la variable X_j est définie par :

$$Var(d_j^*, t) = \min_{\delta \in d_j} \{Var(\delta, t)\}$$

où d_j est l'ensemble des divisions réalisées par X_j

La meilleure division d^* est alors donnée par :

$$Var(d^*, t) = \min_{1 \leq j \leq p} \{Var(d_j^*, t)\}$$

Pour un nouvel individu i, on définit une règle d'affectation en le faisant descendre dans l'arbre. La valeur affectée à Y_i sera la moyenne obtenue dans le segment et l'écart-type sera celui du segment.

Erreur de Prévision : À chaque segment terminal t de l'arbre A, l'erreur associée est :

$$R_t = \frac{n_t}{n} S_t^2$$

Où n est l'effectif total, n_t est le nombre d'individus dans le segment t et

$$S_t^2 = \frac{1}{n_t} \sum_i (y_i - \overline{y}_t)^2$$

avec y_t la moyenne de Y sur le segment t.

L'erreur de prévision associée à l'arbre A est :

$$E(A) = \sum_{t \in A} R_t$$

6. **Critère de la pureté maximale** : Soit Y une variable nominale à K modalités. La sélection d'une division doit être telle que les segments descendants soient plus purs que le nœud parent. On associe à chaque segment t une mesure de l'impureté définie par :

$$I(t) = \sum_{r=1}^{k} \sum_{s=1}^{k} P(r \backslash t) P(s \backslash t) \quad avec\, r \neq s$$

Où P(r\t) et P(s\t) sont les proportions d'individus contenus respectivement dans les classes C_r et C_s dans le segment t.

Un segment est pur s'il ne contient que des individus d'une seule classe, dans ce cas :

$$I(t) = 0$$

Chaque division dj du nœud t par la variable X_j entraîne une réduction de l'impureté qui s'exprime par : $\triangle_j = I(t) - P_g I(t_g) - P_d I(t_d)$

Où P_g et P_d sont les proportions d'indice du nœud t respectivement dans les segments t_g et t_d.

Par conséquent, pour chaque variable X_j, la meilleure division d^*_j est telle que la réduction de l'impureté $\triangle_j$ soit maximale :

$$\Delta_j^* = \max_{d_j}\{\Delta_j\}$$

Sur l'ensemble des p variables, la division du nœud t est effectuée à l'aide de la variable :

$$\Delta^* = \max_{1\leq j\leq p}\{\Delta_j^*\}$$

Règle d'affectation : Un nouvel individu qui descend dans l'arbre arrive dans un segment terminal et sera affecté à la classe correspondante.

Erreur de classement : A tout segment terminal t de l'arbre A associé à une classe C_s correspond une erreur de classement de la forme :

$$R(s\backslash t) = \sum_{r\neq s} P(r\backslash t)$$

Où P(r\t) est la proportion d'individus du segment t affectés à la classe C_s et qui appartient à la classe C_r :

$$P(r\backslash t) = \frac{n_r(t)}{n_t}$$

L'erreur de classement associée à l'arbre A est :

$$E(A) = \sum_{t\in A} \frac{n_t}{n} R(s\backslash t)$$

3 Les algorithmes

1- CART: Classification And Regression Tree

L'arbre CART a été défini en 1984 par L. Breiman, J. H. Friedman, R. A. Olshen et C. J. Stone (Université de Berkeley et Stanford) [23]. On le trouve notamment dans les logiciels Entreprise Miner (SAS), SPAD (Cisia), Intelligent Miner (IBM), Darwin (Thinking Machines) et CART (Salford System). CART peut servir à la prédiction (la variable à expliquer est continue) comme à la classification (la variable à expliquer est

nominale). Pour l'élagage, on effectue un parcours ascendant de l'arbre construit. Pour décider si un sous arbre peut être élagué, on compare l'erreur réelle estimée de l'arbre courant avec l'arbre élagué. L'estimation de l'erreur réelle est mesurée sur un ensemble test ou par validation croisée.

2- C5.0

Développé en 1998 par J. Ross Quinlan, C5.0 est une version améliorée des deus algorithmes ID3 (Inductive Decision Tree) [23] et C4.5, développés également par Quinlan. Il est implémenté dans Clementine (SPSS). Il est aussi commercialisé sur les plates-formes Windows sous le nom de See5, le nom de C5.0 étant réservé aux plates formes Unix.

Pendant la construction de l'arbre, C5.0 cherche à maximiser le gain d'information réalisée. La fonction qui mesure ce degré de mélange (et donc le gain) est la fonction entropie. Tout comme CART, il est adapté pour tout type de variables. Par contre, il n'est pas binaire, c'est à dire qu'au niveau d'un nœud père, le traitement des variables nominales donne naissance à un nœud fils par modalité.

3- CHAID

L'arbre CHAID, conçu par J. A. Hartigan, descend directement du premier arbre de décision : AID (Automatic Interaction Detection) qui est développé en 1964 par Morgan et Sonquist. CHAID est implémenté notamment dans les logiciels Entreprise Miner (SAS), Knowledge SEEKER (Angoss) et Answer Tree (SPSS). Comme critère de sélection des variables, CHAID utilise le test du chi-2. Il est binaire comme CART, mais il ne permet pas l'optimisation automatique de l'arbre par élagage. CHAID détecte l'interaction entre variables dans un jeu de données. En utilisant cette technique on peut établir des relations de dépendance entre variables.

II.6.7 Règles d’associations

Il s’agit de trouver les associations ou les relations intéressantes entre les éléments d’un ensemble de données. Ces associations sont exprimées sous forme d’une suite de règles simple et facilement compréhensible par les utilisateurs. L’application principale est « l’analyse du panier de la ménagère », c'est-à-dire la recherche d’associations entre produits sur les tickets de caisse. Le but de la méthode est la recherche des produits qui pourront être achetés ensemble.

Les règles d'associations sont introduites par Agrawal et al [1] au début des années 90 pour exprimer simplement des tendances implicatives entre les attributs d'une table relationnelle.

Une règle d'association est de la forme : Si Condition alors résultat. Dans la pratique, on se limite généralement à des règles où la condition est une conjonction d'éléments et le résultat est constitué d'un seul élément. Par exemple, une règle à trois éléments sera de la forme : Si X et Y alors Z ; règle dont la sémantique peut être énoncée [41] : Si les élément X et Y apparaissent simultanément alors l'élément Z apparaît [21.]

1 Description de la méthode

L'algorithme des règles d'associations est orienté vers l'analyse des données transactionnelles, également connu sous l'appellation « données des tickets de caisse ». Il sert principalement aux prédictions extrêmement performantes dans des applications de Data Mining de ventes croisées. Cet algorithme opère en termes d'itemsets (ensembles d'éléments).

Il prend les enregistrements de transaction brute et crée une structure de données sophistiquée pour assurer le suivi des nombres d'éléments (produits) dans l'ensemble de données. L'algorithme crée des groupes d'éléments (les itemsets) et collecte des comptages statistiques les concernant.
L'un des paramètres les plus importants d'un modèle est un seuil permettant d'exclure les éléments et les itemsets non populaires. Ce paramètre est appelé le support minimum.

Le résultat de cet algorithme est la collection d'itemsets et de règles résultant des données. Chaque règle est fournie avec un score appelé score lift et une certaine valeur de support supérieure ou égale au support minimum. Le score lift mesure le degré de pertinence de la règle dans sa prédiction de l'élément cible. Une fois que l'algorithme trouve les règles intéressantes, elles sont employées pour obtenir des recommandations de produits.

Les règles d'associations sont traditionnellement liées au secteur de la distribution car leur principale application est l'analyse du panier de la ménagerie qui consiste en la recherche d'association entre produits sur les tickets de caisse.

Pour choisir une règle d'association, il nous faut définir les quantités numériques qui vont servir à valider l'intérêt d'une telle règle :

1- Le *support* d'une règle est la fréquence d'apparition simultanée des articles qui apparaissent dans la condition et dans le résultat dans la liste d'achat donnée en entrée, soit :

$$\text{Support} = \text{Freq (condition et résultat)} = \frac{d}{N}$$

Où d est le nombre d'achat où les articles des parties condition et résultat apparaissent et N le nombre total d'achat.

2- La confiance est le rapport entre le nombre d'achat où tous les articles figurant dans la règle apparaissent et le nombre d'achats où les articles de la partie condition apparaissent, soit :

$$\text{Confiance} = \frac{\text{Freq (condition et résultat)}}{\text{Freq(condition)}} = \frac{d}{c}$$

Où c est le nombre d'achat où les articles de la partie condition apparaissent.

2 Définition formelle

Formellement, le problème de recherche des règles d'association, tel qu'il est présenté dans [1], est défini de la façon suivante :

Soit $I= \{i_1, i_2, \ldots, i_m\}$ un ensemble d'attributs, appelé ***items***. Un ensemble d'items X de cardinalité k ($X \subseteq I$ avec $|X| = k$) est appelé un **k-*itemsets*** (ou ***itemset***). Soit D un ensemble d'itemset, où chaque itemset $t \in D$ est appelé ***transaction***. A Chaque transaction t est associée un identificateur unique, appelée *TID*. Une transaction t contient X est un ensemble d'items, si $X \subseteq t$. Donc, une règle d'association est une implication de la forme : $X \Rightarrow Y$ [σ ; φ] tels que X est l'antécédent de la règle et Y son conséquent :

$X \subseteq I, Y \subseteq I, X \cap Y = \phi$ (c'est-à-dire, X et Y sont des ensembles disjointes d'items).

Une règle d'association $X \Rightarrow Y$ [σ ; φ] traduit le fait que si les items de X sont présents dans une transaction *t*, alors les items de Y le sont avec une certaine probabilité. La qualité d'une règle d'association est évaluée par deux critères : le support σ et la confiance φ. Plus ces deux critères sont très élevé plus la règle d'association est très intéressante.

- Le **support** σ exprime la fiabilité ou l'utilité de la règle $X \Rightarrow Y$ [σ ; φ] . C'est une mesure indiquant le pourcentage de transactions $t \in D$ qui vérifient une règle

d'association. En désignant par $|X|$ le nombre de transactions comportant l'ensemble *X*, $|X \cup Y|$ le nombre de transactions comportant en même temps *X* et Y, et N le nombre total de transactions dans la base *D*. Le support d'une règle d'association peut être défini ainsi :

$$\text{support}\,(X \Rightarrow Y[\sigma;\varphi]) = prob(X \cup Y) = \frac{|X \cup Y|}{N}$$

- La **confiance** φ mesure la précision ou la validité de la règle $X \Rightarrow Y\ [\sigma\ ;\varphi\ ]$. C'est une mesure indiquant le pourcentage de transactions $t \in D$ qui vérifient le conséquent d'une règle d'association parmi celle qui vérifient l'antécédent. On utilisant la notation introduite ci-dessus, la confiance d'une règle d'association est donnée comme suit :

$$\text{confiance}\,(X \Rightarrow Y[\sigma;\varphi]) = prob(X / Y) = \frac{|X \cup Y|}{|X|}$$

Ainsi, à partir d'une base de transactions *D*, le problème de recherche des règles d'association consiste à extraire toutes les règles d'association dont le support et la confiance sont supérieurs à des seuils (*minsupp* et *minconf*) spécifiés par les utilisateurs ou les experts de domaine.

Exemple explicatif :

Soit l'exemple suivant, dans lequel nous supposons avoir prédéfini une classification des articles comme suit :

	Produit A	Produit B	Produit C	Produit D	Produit E
Achat 1	X			X	
Achat 1	X	X	X		
Achat 1	X				X
Achat 1	X			X	X
Achat 1		X		X	

Tab 2 : Liste des achats

A partir de ces données, si on cherche des associations entre deux produits, on construit le tableau suivant :

	Produit A	Produit B	Produit C	Produit D	Produit E
Produit A	4	1	1	2	1
Produit B	1	2	1	1	0
Produit C	1	1	1	0	0
Produit D	2	1	0	3	1
Produit E	1	0	0	1	2

Tab 3 : Tableau de co-occurrence

Ce tableau permet de déterminer avec quelle fréquence deux produits se rencontrent dans un même achat.

Il nous faut préciser comment extraire les règles. Pour cela, Considérons les règles :

1- si A alors B

2- si A alors D

3- si D alors A

La règle 1 a un support de 20% et les règles 2 et 3 ont un support de 40%. Considérons les règles 2 et 3, c'est-à-dire les produit A et D. D apparaît dans trois achats et A apparaît deux fois. Par contre, A apparaît quatre fois et, lorsqu'il apparaît, D n'apparaît que deux fois. On définit alors la confiance d'une règle, la règle 3 a une confiance de 67% et la règle 2 a une confiance de 50%. On préfère donc la règle 3 : si D alors A.

II.6.8 Régression logistique

La régression logistique est une technique prédictive. Le modèle de régression logistique permet d'estimer la force de l'association entre une variable qualitative à deux classes (dichotomique) appelée variable *dépendante* et des variables qui peuvent être qualitatives ou quantitatives appelées variables *explicatives* ou indépendantes. La variable dépendante est la survenue ou non de l'événement étudié et les variables explicatives sont des facteurs susceptibles d'influencer la survenue de l'événement (facteurs d'exposition ou facteurs de confusion).

La régression logistique peut être univariée mais son intérêt réside dans son utilisation multivariée puisqu'elle permet d'estimer la force de l'association entre la variable dépendante et chacune des variables explicatives, tout en tenant compte de l'effet simultané de l'ensemble

des autres variables explicatives intégrées dans le modèle. L'association ainsi estimée est dite « ajustée » sur l'ensemble des autres facteurs.

Même si des adaptations permettent de l'appliquer à certains cas particuliers, le modèle de régression logistique requiert, en principe, certaines conditions : indépendance des différentes observations entre elles, normalité de la distribution des variables quantitatives intégrées dans le modèle, et linéarité de la relation entre chacune de ces variables quantitatives et la variable dépendante.

Objectif

L'objectif du modèle de régression logistique est de chercher à estimer les probabilités :

$$\Pi(X) = P(Y = 1|X) \text{ ou } 1 - \Pi(X) = P(Y = 0|X)$$

par l'observation conjointe des variables explicatives $(X_1, X_2, \ldots, X_p) = X$.

L'idée est alors de chercher un modèle linéaire de la forme :

$F(\Pi_i(X)) = X_i\alpha$

avec F une fonction réelle monotone opérant de [0,1] dans R

Il existe de nombreuses fonctions qui sont conformes à cette description. On retrouve fréquemment les trois fonctions suivantes :

• probit : F est alors la fonction inverse de la fonction de répartition d'une loi normale N(0, 1).

• log-log avec F définie par : $F(\Pi(X)) = \ln[-\ln(1 - \Pi(X))]$

• logit est définie par :

$$F(\Pi(X)) = \ln \frac{\Pi(X)}{1 - \Pi(X)}$$

Le rapport $\Pi(X)/(1 - \Pi(X))$ est appelé l'odds. Il exprime une côte ou une chance de voir se réaliser la modalité "1" plutôt que la modalité "0" de la variable Y. La régression logistique s'interprète donc comme la recherche d'une modélisation linéaire du "log odds".

MÉTHODE

La réalisation pratique d'un modèle de régression logistique comporte plusieurs étapes :

1. La qualité d'une régression logistique repose, avant tout, sur le **choix des variables explicatives** que l'on est susceptible d'intégrer au modèle. Ce choix est fondé sur la pertinence et sur la connaissance de facteurs de confusion avérés ou supposés. C'est pourquoi, une recherche bibliographique approfondie est, au préalable, obligatoire.

2. Il est nécessaire ensuite **d'étudier chacune de ces variables** : analyse de la distribution des variables qualitatives selon leurs différentes modalités et, s'il y a lieu, regroupement de ces dernières ; étude de l'existence d'une relation linéaire entre chacune des variables quantitatives explicatives et la variable dépendante. Si, pour une variable, cette condition n'est pas vérifiée, on procédera à la transformation de celle-ci en une variable ordinale en créant des classes dont le choix repose sur des critères statistiques.

3. On procède ensuite à l'analyse des liaisons entre chacune des variables explicatives et la variable dépendante : on réalise une **analyse univariée** ; les *odds-ratios* calculés sont bruts. Deux catégories de variables explicatives pourront être intégrées dans un **modèle de départ** : celles pour lesquelles l'association avec la variable dépendante est suffisamment forte sans toutefois être trop stricte afin de ne pas omettre d'éventuels facteurs de confusion (p-value inférieure ou égale à 0,20, et non pas 0,05, seuil habituellement retenu) et celles qui ont un intérêt avéré en dehors de tout critère d'association (elles sont rares : ce sont des variables dites « forcées »).

4. **Plusieurs stratégies sont possibles** pour parvenir à un **modèle final** qui devra porter le maximum d'informations tout en ayant un nombre limité de variables afin de faciliter l'interprétation : les plus employées sont les procédures dites « pas à pas descendantes ou pas à pas ascendantes ». La déclinaison des modèles permettra de rechercher les phénomènes d'interaction ou de confusion qu'il faudra prendre en compte lors de l'interprétation. Certaines variables seront impérativement conservées dans le modèle : la variable explicative d'intérêt principal et les facteurs de confusion.

5. En fin d'analyse, **plusieurs modèles finaux** peuvent s'avérer satisfaisants sur un plan statistique. Parmi ceux-ci, on retiendra le modèle le plus adéquat avec le phénomène constaté: des tests d'adéquation permettent de guider le statisticien.

Estimation des coefficients du modèle

Pour l'estimation des coefficients α_j du modèle de régression logistique, la méthode généralement utilisée est celle du maximum de vraisemblance. On peut décrire cette méthode pour n observations (Y_i,X_i) (où $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ sont indépendantes et les Y_i sont des variables de Bernoulli) comme suit.

La vraisemblance $\xi(\alpha, Y_i)$ pour une observation s'écrit :

$$\xi(\alpha, Yi) = \Pi(X_i)^{Yi}[1 - \Pi(X_i)]^{1-Yi}$$

Comme on admet que les observations sont indépendantes, la vraisemblance de l'échantillon selon le modèle est :

$$\xi(\alpha, Y) = \prod_{i=1}^{n} \xi(\alpha, Y_i) = \prod_{i=1}^{n} \Pi(X_i)^{Y_i}[1 - \Pi(X_i)]^{1-Y_i}$$

Le critère du maximum de vraisemblance détermine la valeur de α qui rend maximale le logarithme de cette vraisemblance (log-likelihood).

Les solutions $\widehat{\alpha_j}$ de ces équations sont les estimateurs du maximum de vraisemblance de α_j

En général, elles sont calculées à l'aide de méthodes de calcul numérique telle que la procédure de Newton-Raphson .

Les estimations sont souvent associées à des tests d'hypothèses du type :

$(H_0) : \alpha_j = 0$

L'hypothèse (H_0) exprime la non-influence d'une variable ou d'une modalité X_j sur la variable Y. c'est à dire, X_j n'est pas utile pour expliquer la probabilité conditionnelle de succès

$P(Y = 1|X)$.

Algorithmes de sélection

Lorsque le nombre de variables explicatives p est grand, il n'est pas raisonnable de penser explorer les 2^p modèles possibles afin de sélectionner le "meilleur" au sens d'un critère.

Différentes stratégies sont donc proposées. Parmi elles, on peut citer :

Backward (méthode descendante) : cette méthode s'intéresse au modèle concevant toutes les variables explicatives. Puis, élimine, progressivement les variables les moins significatives (pour un certain seuil désiré).

Forward (méthode ascendante) : Cette méthode considère d'abord le modèle sans aucune variable explicative. Puis, ajoute, progressivement, les variables les plus significatives.

Stepwise : Cette méthode est une combinaison des deux méthodes précédentes (Backward et Forward). En effet, à chaque étape, elle ajoute une variable, réexamine toutes les variables introduites dans le modèle et élimine d'éventuelles variables non significatives. Ainsi, une variable qui était considérée comme la plus significative lors d'une étape donnée, peut devenir non significatives à une étape ultérieure, en raison de ses corrélations avec les nouvelles variables introduites.

II.6.9 Analyse factorielle discriminante

L'analyse factorielle discriminante (AFD) est une méthode permettant de modéliser l'appartenance à un groupe d'individus en fonction des valeurs prises par plusieurs variables,

puis de déterminer le groupe le plus probable pour un individu, connaissant uniquement les valeurs des variables qui le caractérisent.

L'analyse factorielle discriminante (AFD) est une méthode descriptive et explicative, apparentée à l'analyse en composantes principales (ACP), s'appliquant à des données quantitatives sur lesquelles est déjà définie une typologie ou partition.
Par exemple des indicateurs associés à des clients d'une banque classés comme "bons payeurs", "mauvais payeurs" ou "ayant fait faillite".

Pouvant être considérée comme une ACP du nuage des k centres de gravité pour une métrique particulière, l'AFD ne peut obtenir plus de k-1 directions discriminantes, et donne par ailleurs lieu aux diverses représentations et indices.

Le but de l'AFD est de réduire le nombre de dimensions des données, en recherchant celles suivant lesquelles les classes se séparent le mieux. Les directions factorielles discriminantes successives sont déterminées, tandis que des graphiques factoriels plans permettent de visualiser les individus ou les variables. Divers indicateurs et tests sont également calculés, qui permettent de juger de l'intérêt et de la pertinence des résultats obtenus.

Détails mathématiques

Il existe plusieurs présentations mathématiques équivalentes de l'AFD, on donne ici l'une d'entre elles :
Ayant N individus, répartis en k groupes, sur lesquels ont été relevées p variables quantitatives, on note :

- Y_{ih} , élément de R^p, est le h-ième individu du groupe i
- n_i est l'effectif du groupe i
- $y_{i.}$, élément de R^p, est le centre de gravité du groupe i
- $y_{..}$, élément de R^p, est le centre de gravité de l'ensemble

Les différents vecteurs sont reliés par des relations barycentriques.
On définit d'autre part les différentes matrices de variances-covariances :

$$T = \sum_{i\,h} (y_{ih} - y_{..})'.(y_{ih} - y_{..}) / N$$ totale

$$W_i = \sum_{h} (y_{ih} - y_{i.})'.(y_{ih} - y_{i.}) / n_{i.}$$ du groupe i

$$W = \sum_i n_{i.}.W_i / N$$ *intraclasse*

$$B = \sum_i n_{i.}.(y_{i.} - y_{..})'.(y_{i.} - y_{..}) / N$$ *interclasse*

et on a l'égalité matricielle :

T = W + B relation

On munit enfin l'espace R^p des observations de la métrique T^{-1}, dite *métrique de Mahalanobis*, pour laquelle le nuage total est *isotrope* (ou de même inertie dans toutes les directions).

L'inertie totale dans une direction de vecteur u, T^{-1}-unitaire, de R^p (noté en colonne) est en effet :

$$I_u = u'.T^{-1}.T.T^{-1}.u = 1$$

et se décompose en :

$$I_u = u'.T^{-1}.W.T^{-1}.u + u'.T^{-1}.B.T^{-1}.u$$

Ou, en notant a le vecteur T-unitaire $T^{-1}.u$:

$$I_u = a'.W.a + a'.B.a$$

Le pouvoir discriminant de la direction u sera d'autant meilleur que l'inertie interclasse a'.B.a est grande, ou ce qui revient au même que l'inertie intraclasse a'.W.a est faible.

On est donc conduit au problème classique d'optimisation sous contrainte :

$$\text{Max } a'.B.a \quad \text{avec} \quad a'.T.a = 1$$

qui a pour solution les vecteurs propres associés aux valeurs propres décroissantes successives λ de la matrice $T^{-1}.B$. Les a□ sont T-orthogonaux et les ul correspondants T^{-1}-orthogonaux.

II.6.10 Nuées dynamiques

La segmentation par nuées dynamiques (ou k-means) est une méthode de classification automatique qui a pour objectif de partitionner l'espace en k classes (k connu).

A partir d'une partition initiale, on améliore itérativement la partition de l'espace en minimisant la variance et en maximisant l'écart entre les classes.

La solution proposée par cet algorithme dépend de la partition initiale.

Principe

Les nuées dynamiques sont en fait une généralisation de l'algorithme des k-moyennes. On cherche à constituer une partition en K classes des données d'entrée. Chaque classe est

représentée par son **centre**, également appelé **noyau**, constitué du petit sous-ensemble de la classe qui minimise le critère de dissemblance.

Les deux fonctions de base sur lesquelles repose l'algorithme sont les suivantes :

1 La fonction de **réallocation**

Elle partitionne en affectant chaque individu du nuage *E* aux centres d'attractions que forment les noyaux. Elle est définie par l'équation :

$$\pi(X, A_j) = \frac{1}{n_j} \sum_{X' \in A_j} d(X, X'),$$

où n_j est le nombre d'éléments du noyau

2 La fonction de **recentrage**

Elle recalcule les nouveaux noyaux à partir des classes déjà formées. Elle est définie par l'équation :

$$v(A_j, P_j) = \frac{1}{N_j} \sum_{X \in P_j} \pi(X, A_j)$$

où $N_j$est le nombre d'éléments de la classe ou partition

Déroulement de l'algorithme

- Initialisation aléatoire des K premiers noyaux
- Affectation: Calcul de la classe de chaque point du nuage
- Mise à jour des centres (ou attributs des classes).
- Calcul des nouveaux centres de chaque classe (barycentre)
- Test de convergence : L'exécution de l'algorithme se termine lorsque le partitionnement n'évolue plus, c'est à dire lorsque le critère d'inertie intra-classe, défini par l'équation converge est atteint.

$$I_W = \sum_{j=1}^{K} \sum_{X \in P_j} d^2(X, G_j)$$

où G_j est le centre de gravité de la classe défini par l'équation

$$g_j = \frac{\sum_{a_i \in E} \mu_i a_i}{\sum_{a_i \in E} \mu_i}$$

Le résultat change selon le choix des conditions initiales (Monte-Carlo). Il faut donc exécuter plusieurs fois l'algorithme et comparer les résultats de manière à extraire les classes stables, c'est à dire à dégager ce qu'on appelle des formes fortes.

II.7 Conclusion

Le Data Mining est une discipline née en dehors de la statistique, dans la communauté des bases de données et de l'Intelligence Artificielle dans le but de valoriser les bases de données. Le Data Mining offre des perspectives nouvelles pour la statistique et répond au défi du traitement des giga bases de données. Il est la branche de la statistique exploratoire qui cherche à découvrir des structures inconnues et utiles.

Les deux principales fonctions du Data Mining sont la classification et la prédiction (ou prévision). Le Data Mining aide à donner une signification aux giga-octets de données brutes stockées dans les bases de données, en identifiant les modèles et règles présents dans les données ou qui en résultent. Les analystes se servent alors de cette connaissance pour effectuer des prédictions et des recommandations sur les données nouvelles ou futures.

Chapitre III Les Techniques d'extraction de connaissances pour la Détection d'Intrusions

III.1 Introduction

L'utilisation répandue de l'Internet et des réseaux informatiques expérimentés a apporté, avec tous ses avantages, une menace : celle des personnes qui utilisent des moyens illicites pour accéder, envahir et attaquer des ordinateurs (des réseaux).

La détection des tentatives d'attaques sur un réseau est une problématique très importante dans le domaine de la sécurité informatique. Les technologies classiques de protection des réseaux de type Firewall sont en effet inefficaces contre la plupart des attaques actuelles. Aussi sont apparus de nouveaux équipements réseaux pour prendre en compte ces carences, systèmes de détection d'intrusions dans des réseaux, dont le but est de détecter les tentatives d'attaque qu'un firewall ne peut pas bloquer.

Les systèmes de détection d'intrusion ont été développés par différents organismes et groupes de recherche. De nouveaux systèmes de détection d'intrusion basés sur l'exploration et l'analyse de données comportementales des utilisateurs ont fait leur apparition dans le domaine. [20], [52]

Dans cette partie, nous allons nous intéresser à ces approches qualifiées d'approches Data Mining pour la détection d'intrusions. Nous les exposons dans ce qui suit.

III.2 Limitations des protections de type firewall

Les firewalls utilisés dans les réseaux TCP/IP ont comme mode de fonctionnement habituel l'analyse des couches IP (pour déterminer quelles sont les machines impliquées dans la connexion) et TCP/UDP/ICMP (pour déterminer à quel service la connexion s'adresse). Ce genre d'approche, bien que nécessaire, se révèle inadéquate dans plusieurs cas [10]. Afin d'offrir un service à l'extérieur, par exemple un serveur web public, le firewall sera configuré pour accepter toutes les connections entrantes vers le port 80 d'une machine donnée. Tant que les trames seront à destination du serveur web sur le bon port, le firewall les laissera passer. Or de nombreux problèmes de sécurité peuvent exister sur ce serveur web (bug dans le code conduisant à des failles de sécurité, erreur de configuration, etc. . .). Un pirate pourra alors essayer d'exécuter des commandes en tentant d'exploiter certaines failles et erreurs de

configuration courantes ou essayer de récupérer des fichiers tels que le fichier contenant les mots de passe des utilisateurs. En effet, les firewalls usuels sont incapables d'analyser autre chose que les adresses IP des trames et le port auxquels s'adressent les paquets.

III.3 Systèmes de détection d'intrusions (IDS)

Pour pallier aux limitations inhérentes aux firewalls, il faut pousser plus loin l'analyse en regardant non seulement les couches 3 et 4 (IP et TCP/UDP/ICMP) mais aussi les couches supérieures. Cependant ceci est beaucoup plus difficile que d'analyser et filtrer les couches 3 et 4. Les IDSs sont des logiciels criés pour détecter les activités anormales au niveau réseau. Ils fonctionnent le plus souvent par signatures, sur le même principe que les anti-virus [53]. Les attaques connues sont répertoriées dans des bases de signatures et à chaque fois qu'une trame ressemble à une de ces attaques, une alarme est générée. Lorsqu'un nouvel exploit (tentative d'intrusion réussie) est détecté, une signature adaptée sera ajoutée à la base de signatures.

Une autre approche utilisée parfois conjointement avec la précédente est une approche statistique. Pendant une phase d'apprentissage (réglage des paramètres du modèle statistique). L'IDS apprend un profil type du réseau (nombre de paquets échangés, volume des flux, nombre de connections, etc.) et alarme l'administrateur lorsque le trafic dévie de ce profil.

Le problème principal que découvrent les administrateurs lorsqu'ils installent des IDSs est la quantité importante d'alertes générées. Sur un réseau de taille moyenne, plusieurs milliers d'alertes sont générées quotidiennement, rendant quasiment impossible l'exploitation des résultats. La conséquence est que l'administrateur est obligé de revoir sérieusement à la hausse son seuil de tolérance, ce qui va le conduire à passer à côté de plusieurs problèmes réels et permettre à un pirate de haut niveau de réussir une attaque suffisamment discrète pour ne pas être détectée. Ainsi le principal problème de l'IDS n'est pas de laisser passer certaines attaques (dans la pratique il en détecte la quasi totalité) mais de noyer l'administrateur sous un flot d'information. En effet, l'IDS n'est pas capable de juger de la pertinence, de la gravité et de la corrélation des attaques. Il génère tellement d'alertes qu'il va être très difficile de détecter les problèmes graves au milieu de toutes les alarmes.

III.4 Les systèmes de détection d'intrusions

Plusieurs approches à base d'extraction de données (connaissances) ont été proposées pour la détection d'intrusions. Nous pouvant les séparer en deux niveaux : ceux qui emploient des "signatures" pour détecter les attaques dont le comportement est connue et ceux qui emploient des modèles statistique et l'analyse de données comportementales pour faire le

travail. De nombreux outils disposent des deux types d'approches pour optimiser la probabilité de pouvoir identifier les attaques.

III.4.1 IDS basés sur la signature d'attaque (Approche par scénario)

1- Production Best Expert System Toolset (P-Best)

P-Best est proposé par Lindqvist et Porras, il est à base de règles, un système expert avec chaînage avant a été appliqué pour la détection d'intrusions basée sur la signature d'attaque.

L'idée principale du P-Best est d'indiquer les caractéristiques d'un comportement malveillant et puis vérifier le flux de données ou d'événements produits par le système, pour identifier une intrusion.

L'utilisation de système expert existant donne de bons résultats dans les environnements ouverts (OS). En général, une règle de production d'un système expert se compose d'une expression de prédicat sur un ensemble bien défini de faits, et induit la dérivation d'autres faits quand le prédicat est vrai (conclusion).

Dans le contexte de la détection d'intrusions, les faits sont généralement des événements du système avec l'ajout des prédicats (attributs) (par exemple "retourner le code" avec la valeur "mot de passe erroné"). Ces valeurs d'attributs (prédicats) peuvent être employées comme arguments dans les antécédents de règles.

P-Best a été développé à SRI et il a été déployé en premier lieu dans le système MIDAS ID au National Computer Security Center. Plus tard, P-Best a été choisi comme moteur d'inférence basé sur les règles de NIDS, un successeur de prototype d'IDES.

2- State Transaction Analyses Tools (STAT)

Le STAT est basé sur les techniques d'analyse de transition d'état. Une attaque sera modélisée par un automate augmenté (dont les états peuvent posséder des variables), l'automate possédant un état initial dit sécuritaire et peut aboutir, par suite de transitions correspondant à une attaque, vers un état dit compromis.

Les attaques sont définies à l'aide d'un langage de haut niveau appelé STATL [45]. STATL est un langage extensible qui permet de représenter les attaques avec un certain niveau d'abstraction en prenant compte de quelques propriétés clés de l'attaque. De cette façon, un seul scénario d'attaque peut représenter une classe entière d'attaques.

3- Unix State Transaction Analyses Tools (USTAT)

Une extension de STAT, appelée USTAT [28], applique la méthode STAT à un système Unix. Cette extension permet de spécifier les signatures d'attaque au niveau hôte.

Donc, USTAT [28] est un système en temps réel de détection d'intrusions pour UNIX, il a été développé dans le service d'informatique de l'université de la Californie, de Santa Barbara. Il a été développé par Porras qui l'a présenté dans [45] par STAT (Outil d'Analyse de Transition d'Etat).

4- Network-Based State Transaction Analyses Tools (NetSTAT)

Une seconde extension de STAT, appelée NetSTAT [51], permet de spécifier des attaques au niveau réseau.

NetSTAT [51] effectue la détection d'intrusion en temps réel dans les réseaux en prolongeant la technique d'analyse d'états de transitions. Le système opère sur des réseaux complexes, composés de plusieurs sous réseaux, en utilisant l'état des diagrammes de transition pour présenter des attaques de réseaux. Plusieurs avantages sont attribués à cet outil, y compris la capacité de déterminer automatiquement les données à collecter pour soutenir l'analyse de l'intrusion, ayant pour résultat une exécution légère et scalable du réseau.

L'approche NetSTAT exige que le réseau soit modélisé formellement suivant un modèle basé sur les hypergraphes. Cela permet de tenir compte de la topologie particulière d'un réseau donné pour spécifier correctement les attaques.

5- Multics Intrusion Detection and Alerting system (MIDAS)

MIDAS est construit autour du concept de détection d'intrusion heuristique. Les auteurs prennent exemple sur un administrateur humain en analysant comment il mènerait une analyse sur des journaux d'audit pour trouver des preuves d'intrusions. Il pourrait par exemple se dire que les intrusions de déroulent sans doute tard dans la nuit quand le système est sans surveillance.
MIDAS est un système expert à base de règles. Il utilise la Production Based Expert System Toolset (P-BEST).

III.4.2 IDS basés sur des méthodes statistique et l'extraction de données

1- IDES, NIDES

Ces deux systèmes présentés dans [36], [16] et [44] partagent un principe commun et ils sont construits avec un composant basé sur une signature traditionnelle qui coexiste avec une unité de profilage statistique. L'unité de profilage statistique trouve le comportement qui semble anormal en ce qui concerne un profil dans les données extraites. L'unité statistique entretient une base de connaissance des profils avec la description du comportement normal relatif à un ensemble de mesures choisies [30] [3].

L'idée est de décrire l'activité apurée par un vecteur de variables de détection d'intrusions et de comparer ce vecteur aux valeurs prévues stockées dans les profils. Si le vecteur apuré d'activité s'avère être suffisamment loin du comportement prévu, une anomalie est marquée. Ce vecteur, ou *statistique récapitulative d'essai* (dans la terminologie d'***IDES***) est formé de différentes mesures, telles que l'utilisation d'unité centrale de traitement et d'accès aux fichiers. Chaque mesure reflète le point auquel un type particulier de comportement est comparable au profil historique établi pour elle. La manière dont ceci est calculé est en associant chaque mesure à une variable aléatoire correspondante.

Dans [39], les auteurs proposent l'utilisation de plusieurs techniques de réduction et de choix de dispositif généralement employées dans des applications d'extraction de données pour réduire les conditions informatiques et de stockage des méthodes statistiques de détection d'intrusions telles que celle utilisée dans ***NIDES***. Ces techniques exploitent le fait que typiquement plusieurs paramètres comportementaux d'utilisateurs sont corrélés.

IDES (Intrusion Detection Expert System) repose sur l'hypothèse que le comportement d'un utilisateur reste à peu prés le même au cours du temps, et que la manière dont il se comporte peut être résumée en calculant diverses statistiques sur son comportement.
IDES construit ses profils par groupe d'utilisateur censés avoir un comportement proche et tente de corréler le comportement actuel d'un utilisateur avec son comportement passé et le comportement passé du groupe. Il observe trois types de sujets : les utilisateurs, les hôtes distants et les systèmes cible.

NIDES (Next-generation Intrusion Detection Expert System) est la continuation directe du projet IDES. Il est centralisé dans le sens où l'analyseur tourne sur un hôte spécifique,

et il collecte des données venant de divers hôtes à travers le réseau. La collecte d'audit est faite sur ces derniers en utilisant des sources d'audit variées.[4]

2- Haystack

Dans Haystack présenté dans [50], l'évaluation des mesures comparées au comportement historique a lieu à la fermeture d'une session, plutôt qu'en temps réel. Le Haystack compte seulement des mesures telles que la quantité d'I/O et de TEMPS- CPU.
La Haystack contient un ensemble de six types génériques d'intrusions d'ordinateur. Pour chaque type, elle associe un ensemble de poids (de 0 à 9) qui indiquent dans quel degré chaque mesure est liée à ce type d'intrusion. En analysant une session, chaque dispositif de session en dehors de la gamme prédéfinie de la normalité cause un poids pour que ce dispositif soit ajouté aux points de la session. Alors il est possible de calculer la distribution de probabilité des points d'intrusions d'ordinateur et si nécessaire, alerter l'administrateur de sécurité.

3- Java Agent For Metalearning (JAM)

JAM utilise une approche nécessitant le calcul d'un profil issu de données étiquetées durant une phase d'apprentissage. Un algorithme d'apprentissage est ensuite utilisé pour classifier des événements inconnus. Ce projet présente un caractère original car le système JAM est indépendant de l'algorithme d'apprentissage.

L'idée principale en JAM présentée dans [32], [31] et [33], est de produire des classificateurs en utilisant un algorithme d'apprentissage sur les ensembles de données issus de l'utilisation du système. Les résultats du classificateur, qui donnent un ensemble de règles de classification, sont employés pour identifier des anomalies et détecter des intrusions connues.

Spécifiquement, l'approche d'utilisation des classificateurs est testé sur deux ensembles de données : le premier concerne les attaques utilisant le sendmail, l'autre est relatif aux attaques en utilisant TCPdump.
Les données de Sendmail se composent de deux ensembles de traces, une classe de données normales et l'autre classe de données anormales. Les données de formation sont alimentées avec RIPPER [11] qui est un programme de règle d'apprentissage. Les règles de RIPPER classifient les données de formation dans les deux classes "normale" et "anormal".

Chaque trace est alors post-traitée en la comparant aux prévisions de RIPPER, afin de filtrer les fausses erreurs de prévision. Le raisonnement pour l'arrangement de post-traitement est qu'une intrusion réelle est caractérisée par une majorité d'ordres adjacents d'appels anormaux. Une deuxième expérience consiste en des règles de calcul de classification en utilisant seulement les traces normales. Dans ce cas-là, afin de détecter des intrusions, l'information de confiance liée aux règles produites est utilisée. Chaque trace donne des points selon qu'une trace soumise au classificateur au temps d'exécution viole une des règles produites.

L'expérience de TCPdump montre comment des classificateurs peuvent être induits à partir des données du trafic.

Le prétraitement est appliqué aux données brutes de TCPdump et alors le RIPPER est appliqué aux données. Les auteurs mentionnent l'utilisation d'un modèle de Méta détection qui décrit comment des classificateurs multiples peuvent être combinés afin d'exploiter l'évidence combinée des modèles de trafic multiples.

4- Automated Discovery of Concise Predictive Rules for Intrusion Detection

Le projet "Automated Discovery of Concise Predictive Rules for Intrusion détection" de Helmer & Al ([26], [24], [25]), qui utilise la technique suivante : RIPPER est lancé sur un jeu de données d'entraînement étiqueté (les attaques sont identifiées). Cet algorithme génère ensuite des règles de classification pour la classe comprenant le moins d'éléments.

Illustrons cela par un exemple. Soit V un vecteur d'événements inconnus à classifier. Si dans le jeu d'entraînement, il y a moins d'attaques que d'événements normaux, les règles générées ressemblent à :

Si caractéristique no1 de V > x
alors V est une attaque
Si (caractéristique no2 de V > y et caractéristique no5 de V <> 3)
alors V est une attaque
...
Sinon
V n'est pas une attaque.

Tab 4 : Un exemple de règles générées par RIPPER

5- RIPPER

A partir du jeu de données, des règles précises sont générées permettant de caractériser une activité, caractérisation normale ou anormale.

6- Intrusion Detection with Unlabeled Data Using Clustering

Le projet "Intrusion Detection with Unlabeled Data Using Clustering" de Leonid Portnoy [46] qui définit un framework permettant de détecter des attaques inconnues nécessitant uniquement un jeu de données non étiqueté durant la phase d'apprentissage.

Les données brutes non identifiées sont agrégées en clusters, puis le cluster contenant le plus d'éléments est étiqueté "normal", les autres clusters étant étiquetés "anormaux". Lorsqu'un événement inconnu est soumis au système, un algorithme détermine à quel cluster appartient l'événement avec des calculs de distance euclidienne.

7- Hyperview

Hyperview est un système avec deux comportements principaux. Le premier est un système expert qui observe les données d'audit à la recherche de signes d'intrusion connus, l'autre est un réseau de neurones qui apprend le comportement d'un utilisateur et déclenche une alerte lorsque les données d'audit proviennent du comportement appris.

III.5 Conclusion

Dans cette partie, nous avons présenté un état de l'art en matière de détection d'intrusions en utilisant l'approche Data Mining à partir des enregistrements d'audit de sécurité**.**

Le volume de données généré par les mécanismes d'audit des systèmes actuels est très important (de l'ordre du méga-octet par utilisateur et par heur). Il est donc indispensable d'offrir aux officiers de sécurité des méthodes et des outils leur permettant d'en extraire les informations utiles.

Deux grandes approches existent pour cela : l'approche comportementale et l'approche par scénario. Chacune d'entre elles présentant des avantages et des inconvénients, leur utilisation simultanée est nécessaire.

Les outils existant mettent en œuvre l'une ou l'autre de ces méthodes, voire les deux. Il est du ressort de l'officier de sécurité de choisir le ou les outils les plus appropriés non

seulement au système informatique, mais à la politique de sécurité et au niveau d'expertise des intrus potentiels. L'expérience et la compétence de l'officier de sécurité sont ici essentielles.

Nous avons présenté un aperçu des techniques d'extraction de diverses données qui ont été proposées pour la détection d'intrusions. Ainsi, nous avons exposé les techniques de l'exploitation de données qui facilite le processus de la détection d'intrusions et des manières dans lesquelles les diverses techniques ont été appliquées et évaluées.

Chapitre IV Détection d'Intrusions par Classification et Règles d'Association basé sur les Agents (CARIDA)

IV.1 Introduction

Après avoir exposé dans les chapitres précédents une étude des systèmes de détection d'intrusions, du Data Mining ainsi que les technique de détection d'intrusions à base d'extraction de données ; il a été constaté que les attaques contre les réseaux informatiques et leurs ressources sont en augmentation constante, et deviennent de plus en plus sophistiqués. Pour résoudre ces problèmes, plusieurs outils ont été développés. Parmi ces outils on trouve les systèmes de détection (et de réponse) d'intrusions. Etant donné le volume important de données d'audit à extraire, à analyser et à classifier, une des techniques les plus estimées à l'heure actuelle est celle du ***Data Mining***, qui se propose de transformer en connaissance, de grands volumes de données, Nous avons présenté un aperçu des techniques d'extraction de données qui ont été proposées pour la détection d'intrusions.

Nous allons entamer notre modèle qui tire profit des aspects positifs de ces IDSs et éviter les aspects négatifs. Avec l'émergence du paradigme d'agent mobile qui apporte de grandes modifications dans le sens positifs aux IDSs, en terme de décentralisation, mobilité, robustesse, casse de l'architecture hiérarchique qui est en sorte une grande faiblesse des IDSs, en ce sens où il est simple de contourner les inconvénients de ces derniers (IDSs).

Nous allons présenter dans ce chapitre notre approche d'un système de détections d'intrusion basé sur l'approche Data Mining **CARIDA** pour **Classification and Association Rule in Intrusion Detection with Agent** qui tire profit des avantages du paradigme d'agent mobile et qui est implémenté avec la plateforme Aglets qui est à son tour 100% implémentée en Java.

IV.2 Choix du domaine et objectif du système

Vu la complexité et la multiplicité des étiologies de l'information qui devient de plus en plus diffusée et distribuée dans de multiples objets et fonctionnalitées qui sont amenés à coopérer. De plus, la taille, la complexité et l'évolutivité croissantes de ces nouvelles applications informatiques font qu'une vision centralisée, rigide et passive (contrôlée explicitement par le programmeur) atteint ses limites. On est ainsi naturellement conduit à

chercher à donner plus d'autonomie et d'initiative aux différents modules logiciels. Le concept de système multi-agent propose un cadre de réponse à ces deux enjeux complémentaires (et à première vue contradictoires) : **autonomie** et **organisation**.

D'un autre coté les réseaux à grande échelle donnent accès à un grand nombre de sources d'informations distribuées et hétérogènes. La réalisation d'outils d'exploitation automatisés est cependant complexe, en particulier à cause des besoins d'évolution et d'adaptation dynamique aux sources d'information et au réseau. Notre objectif est d'évaluer l'intérêt des agents mobiles adaptables pour faciliter cette réalisation. Pour cela, nous développons un prototype de système de détection d'intrusions basé sur l'approche Data Mining en utilisant les agents mobiles. Nous montrons comment les propriétés de mobilité et d'adaptation des agents aident à mener à terme les problèmes d'attaques contre les réseaux informatiques ainsi à l'intrusion. C'est donc un prototype d'un système de détections d'intrusion intitulé **CARIDA (Classification and Association Rule in Intrusion Detection with Agent).**

IV.3 Intérêt d'utilisation du langage Java pour la programmation des Aglets

Puisque les Aglets sont des objets Java, qui est un bon langage de programmation orienté objet, on va citer quelques propriétés ou bien avantages de ce langage et ce q'il apporte pour la programmation des agents mobiles.

IV.3.1 Langage multi plateforme

Java était conçu pour opérer dans les environnements hétérogènes, cette façon permet à toutes les applications Java de s'exécuter dans n'importe quelle hôte dans le réseau, le compilateur Java génère un code intermédiaire (byte code), ce code sera exécuté sur un ordinateur où se présente la machine virtuelle Java (JVM).

IV.3.2 Exécution sécurisée

Java est prévu pour l'utiliser dans l'Internet, où le problème de sécurité se pose, alors Java offre des mécanismes pour assurer la sécurité du code, on peut citer les aspects de sécurité ci-après :

- Java ne permet pas la manipulation directe des variables mémoire (attributs d'un objet).
- Java ne permet pas l'accès direct aux variables et méthodes privées d'un objet.
- Java fournit un mécanisme de contrôle d'accès aux ressources locales d'un ordinateur par exemple l'ouverture d'un fichier, l'établissement d'une connexion à une base de donnée, l'établissement d'une connexion réseau ,etc.…

Cela et d'autres mécanismes de sécurité permettant à un hôte d'accueillir des agents mobiles d'une manière sécurisée.

IV.3.3 Chargement dynamique de classes

Ce mécanisme permet à la machine virtuelle Java de charger les classes au runtime, il offre un espace uniforme à chaque agent, et lui permet de s'exécuter d'une manière sécurisée et indépendamment d'autres agents, aussi ce mécanisme permet le chargement des classes via l'Internet

IV.3.4 Sérialisation d'objet

C'est la clé future des agents mobiles, dont le fait est de permettre la sérialisation et la déssérialisation d'objets, ce mécanisme consiste à coder l'état d'un objet dans un format standard (vecteur de bits) au niveau de l'émetteur, et de les décoder au niveau du récepteur, tous les objets accessibles par cet objet (graphe d'objets) seront sérialisés avec cet objet et restaurés au même temps.

IV.4 Domaines d'intérêts

Dans notre application, nous nous somme intéressé pour la partie détection d'intrusion aux points suivants :

- Analyse ***centralisée.***
- Approche ***comportementale*** pour étudier le comportement du système.
- Approche par ***scénario*** pour identifier les attaques connues et inconnues.
- ***Audit système*** pour la source des données a analyser.
- Analyse ***périodique*** de telle façon que l'analyse du système se fait à la demande (quand l'administrateur du réseau le décide).

Pour la partie Data Mining, nous nous somme intéressé aux points suivants :

- Les règles d'association.
- La classification pour classifier le type d'attaque qui se produit.

IV.5 Description du système

Le système (prototype) ***CARIDA*** s'exprime sur le principe d'un réseau où les agents se promènent. Le réseau constitue l'infrastructure de notre système, il se compose d'un ensemble de nœuds interconnectés, chaque nœud peut héberger un ou plusieurs agents et se caractérise par son état.

Les agents en promenade permanentent dans ce réseau se différent de types et de stratégies. Il y a des agents de détection d'intrusions (AgentIDS), des agents stationnaire (AgentStat) et des agents de calcul de profil normal (AgentTest).

IV.6 Les Composants

IV.6.1 Les nœuds

Les nœuds constituent l'emplacement où l'agent accomplit effectivement ses tâches, donc ils lui offrent un environnement uniforme d'exécution en plus d'un ensemble des outils lui facilitant la tâche.

Un nœud est constitué de :

- ***Un environnement d'exécution***

L'environnement d'exécution ou bien le serveur d'agent présente l'abstraction du système sur le quel se déroule le code de chaque agent mobile, cette entité a une relation intrinsèque avec le hôte qui l'accueillit, elle a pour rôle de cacher les détails hétérogènes des systèmes.

- ***Une structure des propriétés du nœud***

Cette structure de données est très utile pour l'agent mobile, elle lui fournit une vue globale sur les nœuds comme les identificateurs des agents locaux et le mappage entre les services disponibles et les agents stationnaires correspondants.

- ***Des agents stationnaires***

Les agents stationnaires sont les complémentaires des agents mobiles pour l'accomplissement de la tâche de détection (et de réponse) aux intrusions et de calcul du profil normal.

Notre stratégie de conception consiste à attribuer aux agents mobiles les opérations principales qui demandent que peu de données, et qui doivent être non centralisées pour renforcer la sécurité. Et d'attribuer aux agents stationnaires les opérations qui demandent une grande quantité de données et relativement beaucoup de temps. Ce choix est fait pour ne pas charger l'agent mobile et par conséquent ralentir ses déplacements au sein du réseau.

IV.6.2 Les interconnexions

Les interconnexions entre les nœuds sont le seul moyen d'établir un réseau d'ordinateurs. En réalité il n y a pas une liaison directe entre deux nœuds, ni une structure maillée. Mais il y a tout simplement une liste des adresses des voisins au niveau de chaque nœuds. Un agent qui veut se déplacer doit consulter cette liste pour obtenir l'adresse de sa destination.

✓ **AgentPrinc**

C'est l'agent d'interface graphique. Il lance l'AgentIDS et reçoit les informations de ce dernier.

✓ **AgentIDS**

Comme notre objectif est la détection d'intrusions. L'agent de détection d'intrusion AgentIDS est considéré comme l'entité principale dans notre conception.

Il peut être défini comme l'entité responsable pour surveiller le réseau virtuel et signaler tout ce qui apparaît anormale en se basant sur ses connaissances préalables et les acquis au fur et à mesure de son expérience et sa durée de vie.

Donc l'AgentIDS est en déplacement permanente au sein du réseau dés sa création. Il lance une opération d'analyse des données. Ces données sont obtenues par l'agent stationnaire. En suite deux cas seront prévus :

- Soit les données analysées sont normales.

- Soit les données analysées sont anormales. Dans ce cas l'AgentIDS affiche une fenêtre d'avertissement à l'administrateur.

L'analyse des données par l'agentIDS prend deux aspects:

- ***L'analyse du comportement*** : Dans ce cas l'agentIDS tente de découvrir des déviations qui dépassent les limites prédéfinies pour juger la présence d'une intrusion.

  La déviation est calculée par rapport au profil normal déjà calculé et enregistré.

- ***Analyse des abus*** : Dans ce cas l'agentIDS possède des connaissances préalables sur la nature de plusieurs attaques, ces connaissances sont représentées sous la forme d'une structure de données (graphe).

  L'agentIDS fait la comparaison de l'état du nœud avec les types d'attaque connus et alerte la présence d'une intrusion dans le cas ou il trouve une similitude.

Pour la détectons d'intrusion, nous avons opté pour l'analyse du comportement. Pour identifier les attaque connues et inconnues nous avons choisie l'analyse des abus..

Juste après son initialisation l'agentIDS commence ses déplacements, il s'appuie sur un module de déplacement pour déterminer la destination suivante.

Le module de déplacement se charge d'appliquer une stratégie dictée par l'administrateur pour ordonner la tournée de l'agentIDS, pour cela il existe deux techniques :

- ***Choix aléatoire*** : C'est le plus simple l'agentIDS choisi le nœud qu'il va visiter aléatoirement sans tenir compte d'aucune considération.

- ***Choix déterministe*** : Le choix dans ce cas est pris selon certains critères comme :

  - Le nœud le moins visité.

  - Le nœud le moins récemment visité.

  - Le nœud qui ne contient pas un nombre maximal d'agents

Vu la complexité du choix déterministe, nous avons utilisé la première technique.

En plus de son rôle de détection, l'agentIDS réalise une autre fonction qui consiste à collecter des informations statistiques qui auront par la suite une grande utilité notamment en ce qui concerne l'amélioration des performances du système et pour afficher à l'administrateur des informations sur le système ***CARIDSA.***

- ✓ **L'AagentStat**

AgentStat pour Agent Stationnaire c'est la partie responsable de la collection d'information pour l'AgentIDS et l'AgentTest.

- ✓ **L'AgentTest**

AgentSTest est la partie responsable du calcul du profil normal du réseau

IV.7 Principe du Système CARIDA

CARIDA emploie une combinaison d'extraction de règles d'associations et de classification pour détecter des attaques dans les fichiers d'audits système.
CARIDA est décomposé en deux sous systèmes, à savoir : ***AgentCalcul*** et ***AgentPrinc***.

IV.7.1 AgentCalcul

AgentCalcul est chargé du calcul du profil normal du système (réseaux).

Donc, tout d'abord, un profil normal est construit à partir des fichiers d'audits système enregistrés durant une période ne comportant pas d'attaques en construisant l'ensemble des attributs (informations) "normaux" .

AgentCalcul est un agent d'interface. Il lance l'agent mobile "AgentTest" qui se déplace d'un nœud à un autre. Au niveau de chaque nœud, l'AgentTest lance une opération d'analyse des données locales. Ces dernières sont obtenues par l'AgentStat. Un profil normal est construit au niveau de chaque nœud décrivant le comportement normal au sein de chaque nœud. Une fois, tous les sites (nœuds) visités, et le profil normal de chacun (nœud) crié, l'AgentTest assemble tous ces profils normaux au niveau de l'administrateur. Au niveau de l'administrateur, l'AgentTest construit le profil normal du réseau (système) décrivant son comportement normal. Se profil normal est affiché et enregistré au niveau de l'administrateur et de chaque nœud.

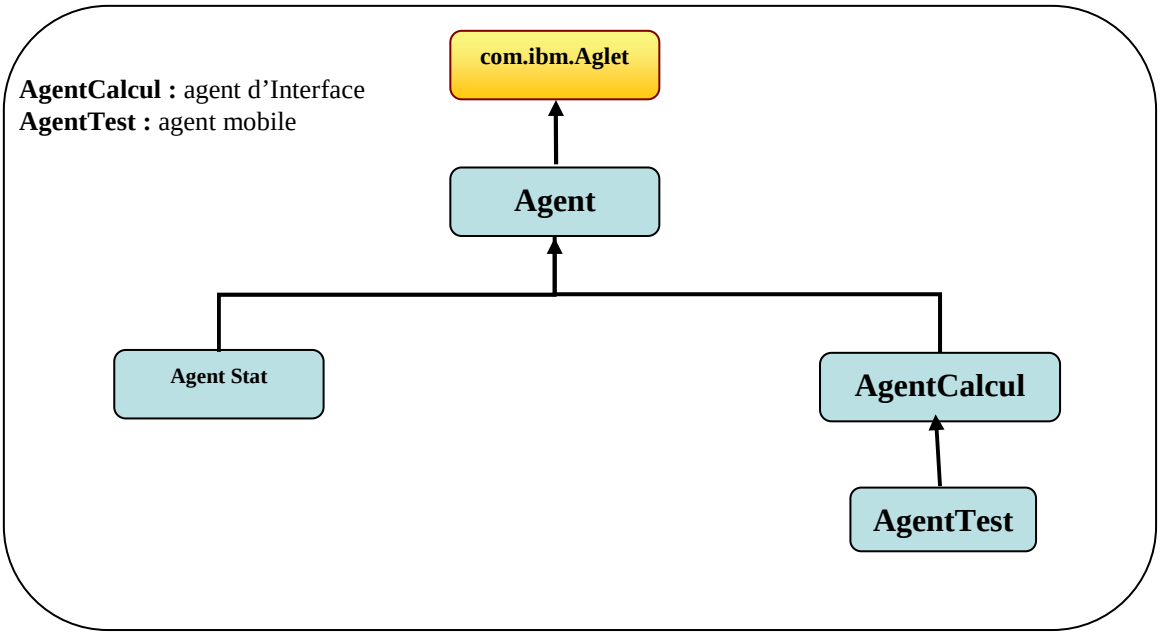

Figure 11 : Architecture du sous de système AgentCalcul

Algorithme du AgentCAlcul :

1- Initialiser l'Agent (AgentCalcul)
2- Introduire les sites (Nœuds) a visité
3- Se déplacer vers le site suivant
4- Au niveau du site (nœud) :
   a. Récupérer les informations de l'agent stationnaire (AgentStat)
   b. Analyser les informations collectées
   c. Construire le profil normal du site.
   d. Enregistrer le profil normal du site
   e. Se déplacer vers le site suivant, muni du profil normal enregistré
5- Répéter 4 jusqu'à avoir visité tous les nœuds (sites)
6- Au niveau du site de l'administrateur
   a. Analyser les profils normaux des sites visités.
   b. Construire le profil normal du système (réseau).
   c. Enregistrer le profil normal du réseau

IV.7.2 AgentPrinc

AgentPrinc est un agent d'interface, chargé de lancer l'agent mobile (AgentIDS) qui est l'agent de détection d'intrusions proprement dit, son rôle est de détecter là où se déroule une éventuelle intrusion.

L'AgentIDS utilise l'ensemble des attributs (informations) dans les dernières connexions et les comparent à ceux stockées dans le dépôt normal (profil normal) d'informations.

Des règles d'association sont utilisées pour assembler la connaissance nécessaire au sujet de la nature des données d'audits.

Donc, l'AgentPrinc lance l'AgenIDS qui est un agent mobile et qui se déplace d'un nœud à un autre. Au niveau de chaque nœud se trouve l'AgentStat qui est un agent stationnaire résidant au sein du nœud, son rôle est d'aider l'agent mobile (AgentIDS) pour accomplir ses tâches. Il collecte les informations et les transmet à l'AgentIDS.
Au niveau de l'administrateur et une fois tous les nœuds visités et toutes les informations collectées, la deuxième tâche de l'AgentIDS est l'analyse des données collectées en les comparant au profil normal construit préalablement par le premier algorithme.
Cette comparaison est effectuée par la dérivation d'un ensemble de règles d'association sous forme de ***SI X alors Y*** tel que X et Y sont des ensembles de valeurs d'attributs.
Dans notre cas, nous nous intéressons aux : ***Temps processus, DataGramme envoyés/s et au DataGramme reçu/s.***

Une fois la comparaison du comportement actuel au profil normal est effectuée, nous avons deux cas possible :

1- Soit le comportement du système est semblable (adéquat) à celui du profil normal.
2- Soit nous avons une déviation du comportement du système par rapport au profil normal, dans ce cas, l'administrateur est informé par un message d'alerte qu'une attaque s'est produite (se produit) ainsi de la provenance de l'attaque, et c'est à lui seul que revient d'agir aux attaques détectées.

Dans le cas d'une attaque, l'AgentIDS accompli sa tâche par une classification de l'attaque en attaques connues,et attaques inconnues.
Dans le cas où CARIDA n'a pas de connaissance préalable de l'attaque, ce dernier (CARIDA) le classe dans la catégorie d'attaque inconnue et enregistre sa trace. Grâce à cette dernière, que CARIDA développe la notion d'apprentissage, toutes les prochaines attaques similaires, sont classées dans la catégorie d'attaques connues.

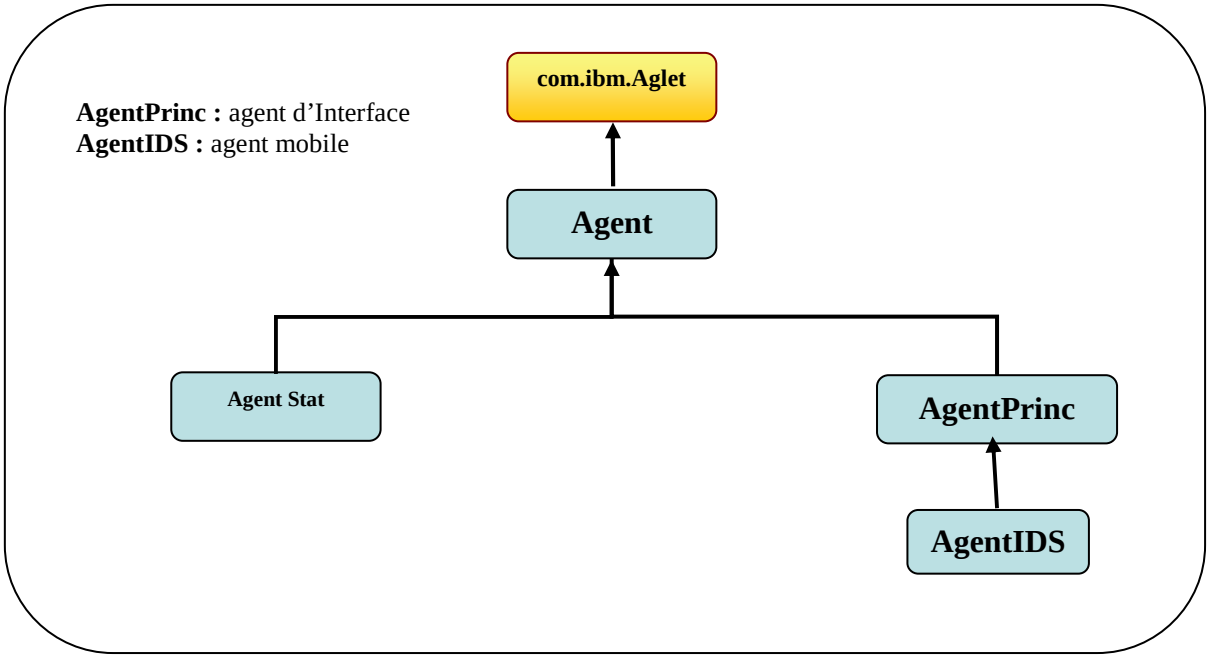

Figure 12 : Architecture du sous système AgenPrinc

Algorithme de l'AgentPrinc :

1- Introduire les sites (Nœuds) a visité
2- Se déplacer vers le site suivant
3- Au niveau du site (nœud) :
   a. Récupérer les informations de l'agent stationnaire (AgentStat)
   b. Se déplacer vers le site suivant, muni des informations
4- Répéter 3 jusqu'à avoir visité tous les nœuds (sites)
5- Au niveau du site de l'administrateur
   a. Analyser les données collectées
   b. Comparer les données collectées au profil normal
   c. Si le comportement du système est différent au profil normal alors :
      - Classification de l'attaque en : attaque connue ou attaque inconnue
      - enregistrer l'attaque
      - Informer l'administrateur de l'attaque, de sa classe et de sa provenance

IV.8 L'ensemble des agents du système

CARIDA utilise un ensemble d'agents qui sont :

IV.8.1 AgentStat

L'AgentStat est un agent stationnaire, chargé de collecter les informations sur les Entrées / Sorties pour chaque hôte en utilisant l'audit système. Il réside au niveau des hôtes du réseau. L'AgentStat continu à collecter des informations sur le hôte de manière permanente.

IV.8.2 AgentTest

L'AgentTest est un agent mobile, il se déplace dans le réseau passant par tous les hôtes, il récupère les informations et construit le profil normal pendant une période ne comportant pas d'attaques.

IV.8.3 AgentPrinc

C'est un agent d'interface résidant chez l'administrateur, il lance l'AgentIDS qui récupère les informations transmises par l'AgentStat.

IV.8.4 AgentIDS

L'AgentIDS est un agent mobile, il se déplace d'un nœud à un autre. Au niveau de chaque nœud, il récupère les informations transmises par l'AgentStat.
Au niveau de l'administrateur et une fois tous les nœuds visités et toutes les informations collectées, la deuxième tâche de l'AgentIDS est l'analyse des données collectées en les comparant au profil normal construit préalablement par le premier algorithme.
Cette comparaison est effectuée par la dérivation d'un ensemble de règles d'association sous forme de ***SI X alors Y*** tel que X et Y sont des ensembles de valeurs d'attributs.

Il classifie l'attaque et informe l'administrateur en cas d'attaque qu'une intrusion s'est produite par une alarme ou un message d'avertissement, du type d'attaque détectée et de sa provenance.
Pour remédier à cette attaque, c'est à l'administrateur que revient le pouvoir d'agir et de répondre à l'attaque détectée.
En fin, il transmet les résultats à l'AgentPrinc.

IV.8.5 AgentIDS

L'AgentCalcul est un agent d'interface résidant au niveau de l'administrateur. Il lace l'AgentTest.

IV.9 La réalisation du système CARIDA

Pour l'implémentation de notre système nous avons opté pour le langage java, car il convient le mieux à la nature de notre système.

IV.9.1 Environnement de développement

1- Environnement matériel

Notre logiciel est implémenté sur un micro-ordinateur « PENTIUM IV » doté de 504 MO de mémoire vive (RAM) et d'un disque dur de capacité 80 GØ.

2- Environnement logiciel

Programmation sous WindowsXP Professional HP : WindowsXP offre une interface Homme/Machine conviviale et facile d'emploi, ainsi il offre à l'utilisateur une interface graphique multifenêtres et un gestionnaire multitâche, il donne à l'utilisateur l'impression de guider lui-même son programme.

IV.9.2 Le serveur TAHITI

Pour exécuter un agent on a besoin d'une plate-forme et d'un serveur. Nous avons opté pour la plate-forme Aglets qui implémente le serveur TAHITI.

Le serveur TAHITI permet entre autre de créer et de charger les Aglets, les dispatchers vers un nouveau contexte et de les retirer. Lors de la création d'un Aglet, les opérations suivantes sont effectuées :

- Charger le fichier class
- Instancier l'Aglet
- Etablir l'Aglet dans son contexte
- Invoquer la méthode onCreation()
- Invoquer la méthode run()

IV.9.3 La mise en oeuvre de l'application

Dans cette partie, nous allons suivre une exécution étapes par étapes pour voir les résultats de notre application.

IV.9.4 Lancer le serveur TAHITI

Dans la console de commande, on se met dans le répertoire où se trouve la plate-forme Aglets et on lance le serveur par :

1- La commande ***ant*** (ou ***ant install***)

2- La commande ***agletsd***.

On aura une boite de dialogue dans laquelle on doit saisir le nom et le mot de passe qui sont respectivement :***aglet_key*** et ***aglets***

Et enfin on aura le serveur TAHITI.

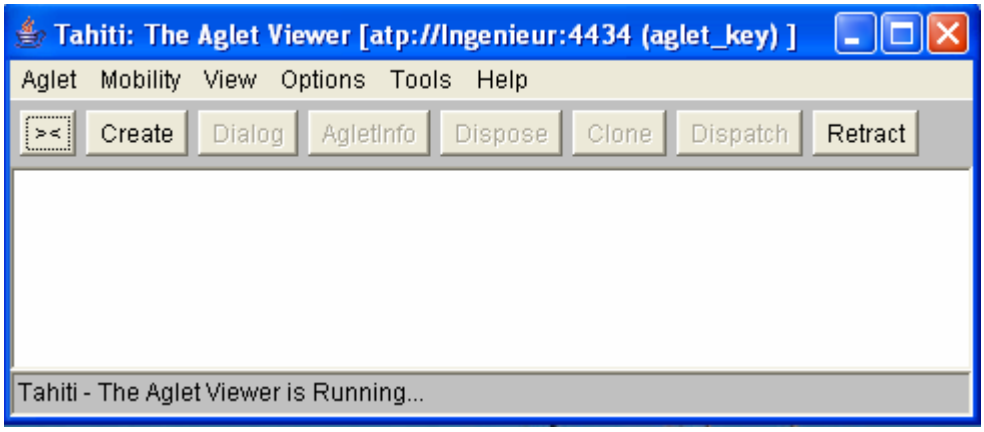

Figure 13 : Le serveur TAHITI

IV.9.5 Lancer l'application

Etant donné que nous avons deux sous système, nous allons procéder au lancement de chaque sous système.

Pour lancer une application à partir du serveur TAHITI, il faut appuyer sur ***Creat*** et saisir le nom de notre application:

IV.9.6 Lancement du Calcul de Profil

Nous saisissons le nom de notre application et nous aurons :

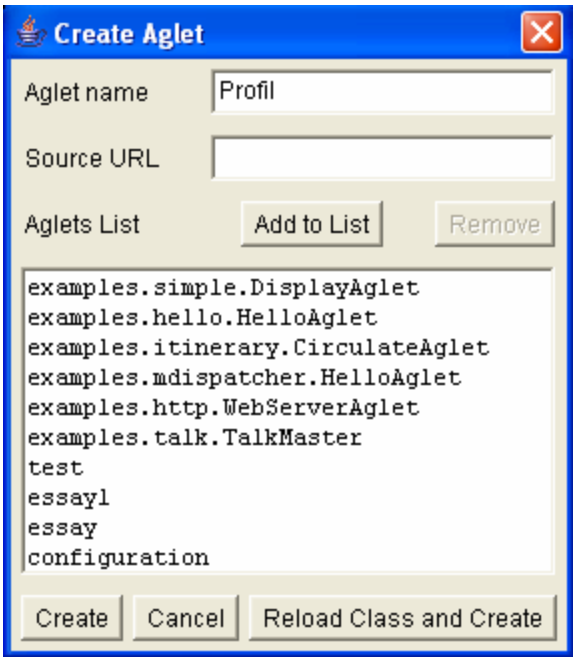

Figure 14 : Lancement du Calcul du Profil dans le serveur TAHITI

Après la saisie du nom de notre application nous avons une fenêtre qui s'affiche pour saisir le login et le mot de passe:

Figure 15 : Interface mot de passe

Après la saisie du mot de passe, une fenêtre s'affiche dans laquelle :

- ✓ L'administrateur saisi les adresses des nœuds (sites) qu'il veut visiter et qui constituent son réseau.

- ✓ L'agent d'interface AgentCalcul crée une instance de l'agent mobile AgentTest qui se déplace vers les nœuds du réseau et que l'administrateur avait saisis au par avant.
- ✓ Une fois au niveau des nœuds, l'AgentTest reçoit les informations (données) sur les nœuds de l'agent stationnaire AgentStat résidant au sein des nœuds et construit le profil normal du nœud en question.
- ✓ Au niveau de l'administrateur, l'AgentTest construit le profil normal du réseau et le transmet à l'AgentCalcul
- ✓ L'AgentCalcul affiche le profil normal, l'enregistre et affiche les sites (nœuds) visités.

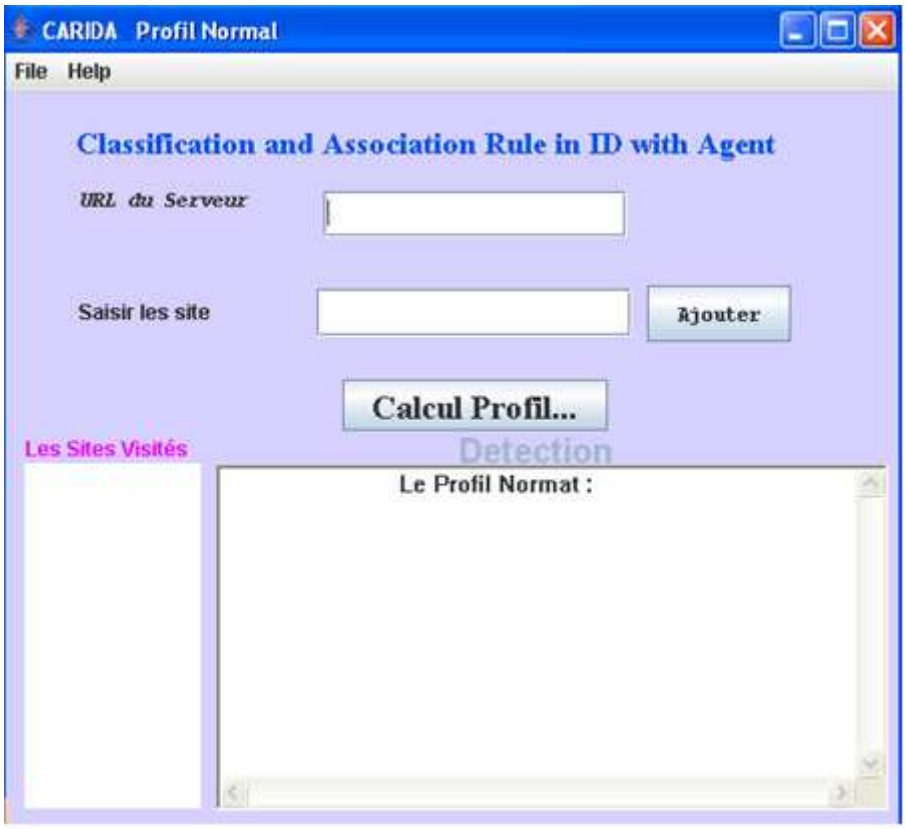

Figure 16 : Lancement du système Calcul du Profil Normal

IV.9.7 Lancement de l'agent de détection d'intrusions

Nous saisissons le nom de notre application et nous aurons :

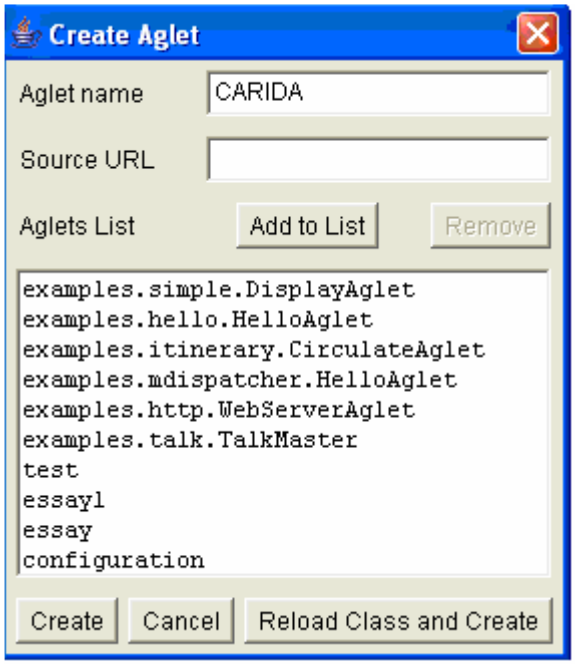

Figure 17 : Lancement de CARIDA dans le serveur TAHITI

Après la saisie du nom de notre application nous avons une fenêtre qui s'affiche pour saisir le login et le mot de passe.

Après la saisie du mot de passe, une fenêtre s'affiche dans laquelle :

- ✓ L'administrateur saisi les adresses des nœuds (sites) qu'il veut visiter et qui constituent son réseau.
- ✓ L'agent d'interface AgentPrincl crée une instance de l'agent mobile AgentIDS qui se déplace vers les nœuds du réseau et que l'administrateur avait déjà saisis
- ✓ Une fois sur les nœuds, l'AgentIDS reçoit les informations (données) sur les nœuds de l'agent stationnaire AgentStat résidant au sein des nœuds.
- ✓ L'AgentIDS munie des informations (données) collectées des nœuds visités, se déplace au niveau de l'administrateur où se trouve l'AgentPrinc les analyse et les compare au profil normal construit au par avant. En cas d'attaque l'agentIDS informe l'AgentPrinc par un message de l'attaque, de sa classe et de sa provenance.
- ✓ L'AgentPrinc affiche les informations, enregistre l'attaque.

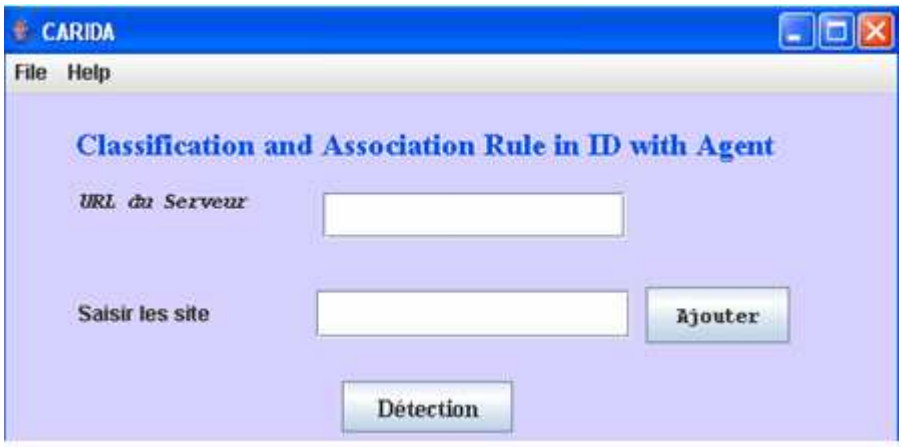

Figure 18 : Lancement du système CARIDA

En cas d'attaque, l'AgentPrinc affiche un message d'alerte, dans lequel il informe l'administrateur de la source d'attaque, son type et sa catégorie, du genre :

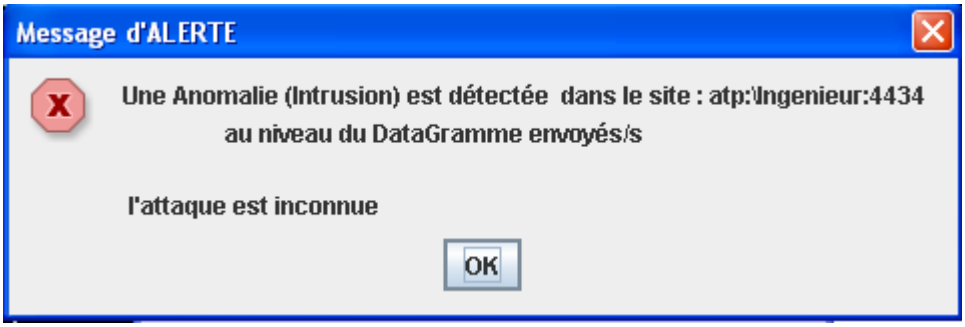

Figure 19 : Le message d'alerte d'une attaque.

IV.10 Discussion

Le but d'un système de détection d'intrusions est de déterminer qu'une violation de sécurité se produit eu s'est produite. Pour cela, nous avons développé et réalisé un système de détection d'intrusion que nous l'avons appelé ***CARIDA*** pour **Classification and Association Rule in Intrusion Detection with Agent.**

Etant donné le grands volume de données a extraire et a analyser nous a conduit à l'approche data mining qui est un ensemble de techniques d'extraction de connaissances valides et exploitables à partir de grands volume de données.

Pour assurer la propriété essentielle d'un IDS qui est la possibilité de détecter là où une éventuelle attaque se produit (ou s'est produite), nous avons testé notre système ***CARIDA*** en provoquant et simulant plusieurs types d'attaques.

CARIDA a identifie et détecté toutes les anomalies et dérivations vis-à-vis le profil normal de référence construit en premier lieux en se basant sur les profils normaux de tous les hôtes du réseau. Il donne de bons résultats, il identifie et classe toutes les nouvelles attaques détectées dans la catégorie d'attaques inconnues, et les attaques déjà rencontrées, ***CARIDA*** les classes dans la catégorie d'attaques connues.

CARIDA tire profit de l'approche comportementale pour la détection de toutes déviation du comportement actuel du réseau vis-à-vis le comportement normal de référence (détection d'intrusions). Pour la classification des intrusions en attaques connues ou inconnues, ***CARIDA*** utilise l'approche par scénario en basant sur les traces d'attaque enregistrées aux par avant.

CARIDA a une fréquence d'utilisation périodique ; au besoin de l'administrateur, ce dernier lance ***CARIDA.***

Le principe de détection de ***CARIDA*** consiste à lancer l'agent mobile de détection AgentIDS, qui se déplace dans le réseau, recevoir les données de l'agent stationnaire AgentStat et entame une analyse centralisée de ces dernière (données) au niveau de l'administrateur pour découvrir toutes les anomalies et attaques que le réseau subie soit de l'intérieur ou de l'extérieur. ***CARIDA*** permet entre autre, de déterminer la source de l'attaque son type et sa catégorie (connue ou inconnues).

L'analyse centralisée permet d'avoir une vision globale du réseau. Donc, de centraliser les alertes pour les analyser au sein d'une seule machine. Dans cette analyse, le réseau cour un risque d'attaques au moment de l'analyse, étant donné que les informations sont collectées du réseau ensuite analysées sur un seul hôte (celui de l'administrateur) et le réseau reste sans protection.

En plus de ce point, la fréquence d'utilisation périodique qui permet d'analyser le réseau à chaque période de temps choisie par l'administrateur laisse le réseau sans protection en dehors de cette période, ce qui pourrait mettre le réseau en menace (danger) d'attaques.

Ces deux points rendent le réseau vulnérable aux attaques.

Pour remédier à ces problèmes, nous entamons l'amélioration de notre système ***CARIDA.***

IV.11 Description du système amélioré

Nous gardons le système ***CARIDA*** comme il est. Donc, nous avons les deux sous systèmes (algorithmes) à savoir le calcul du profil normal de référence et la détection d'intrusions qui se fait de manière périodique au besoin de l'administrateur. Pour éviter des attaques lors de l'analyse et en dehors de la période d'analyse, nous doterons notre système ***CARIDA*** d'un algorithme de détection d'intrusions qui sera exécuté par l'agent stationnaire ***AgentCDD*** pour ***Continues Distributed Detection Agent*** qui se base sur l'analyse distribué des données et qui a une fréquence d'utilisation continue (temps réel).

IV.11.1 Domaine d'intérêt

Pour cette étape, nous nous somme intéressé pour la détection d'intrusion aux points suivant :

- Analyse ***distribuée*** qui convient le mieux à la technologie agent mobile.
- Approche ***comportementale*** pour étudier le comportement du système.
- Approche par ***scénario*** pour identifier les attaques connues et inconnues.
- ***Audit système*** pour la source des données a analyser.
- Analyse ***continue (temps réel)***

Pour la partie Data Mining, nous nous somme intéressé aux points suivants :

- Les règles d'association.
- La classification pour classifier le type d'attaque qui se produit

IV.11.2 Principe de AgentCDD

CARIDA continu à s'exécuter à chaque période de temps choisie par l'administrateur de la même façon sans changement.

L'***AgentCDD*** est un agent stationnaire de détections d'intrusion résidant au niveau des nœuds du réseau. Il est lancé au niveau de chaque nœud, son rôle est de détecter les intrusions qui se produisent sur le nœud en question.

Sur chaque nœud se trouve l'***AgentStat*** qui est agent stationnaire résidant au sein du nœud. Son rôle est d'aider l'***AgentCDD*** à accomplir ses tâches, il collecte les informations et les transmet à l'***AgentCDD***. L'***AgentCDD*** analyse les données dûment collectées en les

comparants au profil normal de référence construit préalablement par l'AgentCalcul du système ***CARIDA*** et enregistré au niveau de chaque nœud.

Cette comparaison est effectuée par la dérivation d'un ensemble de règles d'association sous forme de ***SI X alors Y*** tel que X et Y sont des ensembles de valeurs d'attributs.

Une fois la comparaison du comportement actuel au profil normal est effectuée, nous avons deux cas possible :

1- Soit le comportement du nœud est semblable (adéquat) à celui du profil normal.
2- Soit nous avons une déviation du comportement du nœud vis-à-vis profil normal de référence. Dans ce cas, l'administrateur est informé par un message d'alerte qu'une attaque s'est produite (se produit) au niveau du nœud en question.

Une classification de l'attaque est effectuée au niveau de l'administrateur ainsi que l'enregistrement de la trace de l'attaque dans le cas d'une attaque inconnue.

L'***AgentCDD*** continu l'analyse des données et la détection d'intrusions de manière permanente et à temps réel.

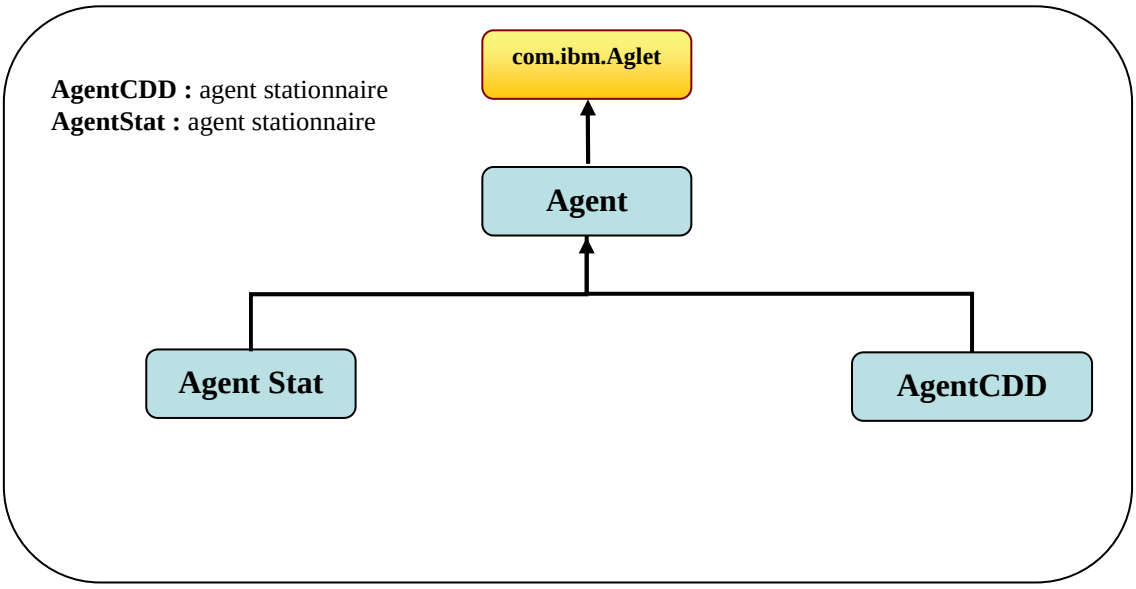

Figure 20 : Architecture du sous système AgenPrinc

IV.11.3 Algorithme de l'AgentCDD

1- Lancer AgentCDD au niveau de chaque noeud
2- Récupérer les informations de l'agent stationnaire (AgentStat)
3- Analyser les données collectées
4- Comparer les données collectées au profil normal de référence

5- Si le comportement du nœud est différent au profil normal de référence alors :

- Informer l'administrateur de l'attaque, de sa classe et de sa provenance
- Classification de l'attaque en : attaque connue ou attaque inconnue
- enregistrer l'attaque

IV.12 Réalisation

Pour implémenter l'AgenCDD, nous avons opté pour le langage JAVA. Nous gardons le même environnement de développement matériel et logiciel utilisé pour le développement de ***CARIDA***.

IV.12.1 Mise en œuvre

Pour exécuter un agent on a besoin d'une plate-forme et d'un serveur. Nous avons opté pour la plate-forme Aglets qui implémente le serveur TAHITI.

Pour lancer un agent nous devons lancer le serveur TAHITI par les deux commande ***ant*** et ***agletsd ,*** saisir le nom et le mot de passe de la boite de dialogue qui s'affiche sur l'écran et enfin le serveur TAHITI qui est lancé. À l'aide du bouton ***Creat***, notre agent est lancé après la saisie de son nom.

Dans le cas d'une intrusion l'AgentCDD détecte l'attaque et informe l'administrateur par un message d'alerte de l'attaque, son type et sa source ; du genre :

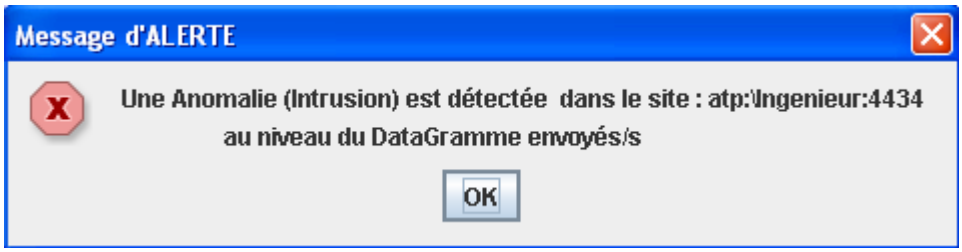

Figure 21 : Message d'alerte d'une attaque

IV.13 Conclusion

Dans cette partie, nous avons présenté notre système **CARIDA,** nous avons décrit d'une manière exhaustive la conception et la mise en œuvre de notre modèle. Premièrement nous avons décrit le domaine d'application, exposé le principe de notre modèle, l'environnement de développement, ensuite nous avons donné un aperçu du langage de programmation utilisée, et enfin, nous avons terminé par la réalisation et la description du logiciel.

Conclusion Générale

L'informatique change de manière assez profonde. Tout d'abord, l'informatique devient ubiquitaire. Au départ confinée dans les ordinateurs, elle investie les objets de la vie courante : téléphones portables, assistants personnels, maison, etc. Elle devient ainsi de plus en plus diffusée et distribuée dans de multiples objets et fonctionnalités qui sont amenées à coopérer. La décentralisation et une organisation coopérative entre modules et logiciels sont la solution. De plus, la taille, la complexité et l'évolutivité croissantes de ces nouvelles applications informatiques font qu'une vision centralisée, rigide et passive (contrôlée explicitement par le programmeur) atteint ses limites. On est ainsi naturellement conduit à chercher de donner plus d'autonomie et d'initiative aux différents modules logiciels. Le concept de système multi-agent propose un cadre de réponse à ces deux enjeux complémentaires (et à première vue contradictoires) :
Autonomie et Organisation.

Les attaques contre les réseaux informatiques et leurs ressources sont en augmentation constante et deviennent de plus en plus sophistiquées. Cette affirmation est confirmée par les rapports annuels du Computer Emergency Response Team [CERT] qui mentionnent aussi l'insuffisance des mesures destinées à centrer ces attaques et mettent en évidence la nécessité d'une amélioration continue de la protection des systèmes d'information.

Nous rappelons que l'objectif de notre travail est l'étude et la réalisation d'un système de détection d'intrusions en utilisant le Data Mining et en se basant sur la technologie d'agent mobile.

Pour cela, nous avons d'abord présenté des généralités sur les systèmes de détection d'intrusions, leur principe est d'extraire et de classifier des connaissances pertinentes d'un large volume de données diverses. Ces données sont collectées et enregistrées d'un grand nombre d'événements de sécurité, grâce à un audit de sécurité, ainsi que ses limites. Nous avons constaté que le large volume de données collectées par l'audit de sécurité qui est à extraire, à analyser et à classifier, nécessite une technologie puissante et la plus estimées à l'heur actuelle est celle du Data Mining. Cette dernière permet d'extraire d'une base de données des connaissances sous la forme de modèles de description afin de retracer le comportement actuel et / ou de prédire le comportement future des données. Ainsi que nous avons exposé des approches d'extraction de données pour la détection d'intrusions. Et pour

finir nous avons proposé un système de détection d'intrusions basé sur l'approche Data Mining que nous avons baptisé ***Classification and Association Rule in Intrusion Detection with Agent (CARIDA)*** qui tire profil des avantage du paradigme d'agent mobile est qui est implémenté avec la plateforme Aglet qui est a son tour implémentée en Java.

Notre approche identifie des anomalies et détecte des attaques provenant soit de l'intérieur ou de l'extérieur du réseau. Elle répond à toutes les attaques car elle se base sur le comportement normal du réseau donc, toutes déviation à ce comportement est identifiée comme attaque. Pour les fausses attaques (faux positif), notre système peut les identifier à l'aide de l'administrateur.

CARIDA est un bon outil pour la détection d'intrusions connues et inconnues qui est l'objectif des IDSs.
Notre système est basé sur deux points important pour faire face aux problèmes d'attaques contre les réseaux et à l'intrusion ; à savoir la technologie d'agent mobile, qui porte un grand appui à notre approche d'où sa véracité, sa robustesse, son autonomie …etc, et le data mning qui est d'une importance capitale pour le grand volume de données à analyser grâce aux règles d'associations et à identifier à l'aide de la classification.

CARIDA tire profil de deux approches à savoir, l'approche comportemental et l'approche par scénario en exploitant les avantages de chacune et remédier aux inconvénients de ces dernières. L'approche comportementale est utilisée pour la détection d'intrusions en comparant le comportement actuel du réseau au comportement normal déjà calculé et enregistré et toute déviation à celui-ci est une attaque. Pour la classification des attaques, notre système utilise l'approche par scénario en comparant la trace de l'attaque à la base des attaques enregistrées, pour définir l'attaque connue de celle inconnue.

CARIDA est un prototype évolutif et très clair pour illustrer comment on peut employer les agents mobiles pour développer des applications réparties. Il enrichi la base de signature, qui est utile pour la détection et l'identification de l'intrusion, ainsi que pour d'éventuelle travaux dans l'avenir. Il est considéré comme un maillon robuste et efficace intermédiaire entre la collecte des données distribuées dans un réseau et la réponse active. Nos résultats prouvent que CARIDA est très efficace dans la détection de nouvelles attaques.

En perspective, l’implémentation de tous les concepts étudier théoriquement comme l’audit du système d’exploitation et le scannage de trafic réseau ou la réponse active reste a compléter.

Annexe 1 Présentation de la technologie d' Agents Mobiles

1. Introduction

Bien que le concept d'agents intelligents ne soit pas vraiment nouveau, une nouvelle génération d'agents de type que l'on qualifie d'agents mobiles est en cours de développement.

Ces agents peuvent servir à nous assister dans nos tâches d'administration quotidiennes. Ils filtrent par exemple les Emails, partent à la recherche de virus, détectent les intrusions dans les réseaux, en bref effectuent les opérations d'administration de base sans plus d'intervention humaine. Ils peuvent communiquer entre eux, analyser l'information qu'ils recueillent et avertir le cas échéant l'administrateur du réseau. Et dans cette partie on va évoquer tous ces concepts et autres d'avantages.

2. Les origines

Pendant longtemps a été défini le terme d'*agent* dans les réseaux mais cette définition fait référence à une portion de code qui s'exécute sur l'élément réseau (Network Element) plutôt qu'à une fonctionnalité avancée pour l'administration de réseaux basée sur les agents intelligents et mobiles. Les agents de type Snmp ou Cmip ne sont que des mini-logiciels qui collectent et enregistrent les informations des équipements réseaux et les communiquent, sur requête, à la station d'administration par le biais des protocoles Snmp ou Cmip.

Au contraire, le concept d'agents intelligents pour l'aide à l'administration suppose que les agents sont suffisamment autonomes pour limiter l'intervention humaine dans les choix d'administration.

3. Définition

Tout d'abord, commençons par préciser quelques termes importants :

- ✓ **Agent informatique :** Est défini de manière informelle comme un programme qui peut exercer une autorité, travailler en autonomie, et rencontrer et interagir avec d'autres agents. Un agent informatique contient le code et l'information d'état nécessaires à un calcul donné. Il a besoin d'une plate-forme mettant à disposition un environnement où il pourra s'exécuter. Un agent peut être statique ou mobile.

- ✓ **Agent statique :** Les agents stationnaires demeurent solidaires d'une seule plate-forme,
- ✓ **Agent mobile :** Les agents mobiles peuvent suspendre leur exécution sur une plate-forme et se déplacer sur une autre où ils reprennent leur exécution. Les agents mobiles sont un modèle récent de calcul distribué.

  Un agent mobile est un agent qui peut se déplacer sur un réseau hétérogène sous son propre contrôle. Ce type d'agent est très utile pour surpasser les faiblesses de la connexion, mais des problèmes de sécurité reste à résoudre.

  La technologie des agents mobiles a bénéficié du travail effectué dans le domaine des ***agents intelligents*** et du développement de plates-formes capables de gérer du code mobile dans un réseau hétérogène.

4. Propriétés des agents mobiles

- ✓ **Surmonter la latence du réseau:** Les agents mobiles sont envoyés dans la zone d'activité, afin d'entreprendre des actions sur place. Cela leur permet de répondre en temps réel à des changements dans leur environnement. En plus de détecter et de diagnostiquer des intrusions, les agents mobiles peuvent aussi répondre aux attaques. Typiquement, ils peuvent collecter de l'information locale sur une attaque, stopper ou isoler un système victime d'une attaque, ou encore pister une attaque.
- ✓ **Réduire la charge du réseau:** Plutôt que de transférer les données à travers le réseau, les agents mobiles sont envoyés sur la machine où résident les données. Ainsi, c'est le calcul qui va aux données et non les données au calcul. Ceci permet de réduire la charge du réseau.
- ✓ **S'exécuter de manière autonome et asynchrone:** Dans des systèmes distribués étendus, la capacité du système, alors que des portions sont détruites ou isolées, est essentielle. Les agents mobiles existent et fonctionnent indépendamment de leur plate-forme d'origine.
- ✓ **S'adapter dynamiquement:** La capacité des systèmes d'agents mobiles à percevoir leur environnement et à réagir aux changements, est très utile pour la détection d'intrusions. Les agents peuvent migrer vers des régions plus appropriées, se cloner pour introduire de la redondance et du parallélisme, ou solliciter l'aide d'autres agents. Ceci contribue aussi dans la mise au point d'un système robuste et tolérant aux pannes.
- ✓ **Etre indépendant de la plate-forme:** Un système d'agents propose un environnement abstrait de calcul, indépendant de la plate-forme matérielle et logicielle sur laquelle il s'exécute. Cela permet un déplacement sans entrave des agents à l'intérieur d'un

domaine. Les mécanismes de réponse en bénéficient en particulier, car la détection d'une intrusion peut ainsi être suivie d'une réponse provenant de n'importe quel endroit du réseau.

5. Avantages (Caractéristiques) des agents mobiles

Un agent est généralement défini comme un système informatique logiciel qui répond aux propriétés suivantes :

- ✓ **La mobilité :** la capacité d'un agent à se déplacer dans un réseau informatique entre les machines.
- ✓ **La véracité :** la conjecture selon laquelle un agent ne communique pas de mauvaises informations sans le savoir.
- ✓ **Le bénévolat :** la conjecture selon laquelle les agents n'ont pas de buts incompatibles, et que chaque agent essaiera de faire ce qu'on attend de lui.
- ✓ **La rationalité :** la conjecture selon laquelle un agent agira de sorte à atteindre ses objectifs, au moins dans la limite de ses convictions.
- ✓ **Autonomie :** les agents opèrent sans intervention directe d'être humain ou autre, et ont un certain contrôle sur leurs actions et leur état interne.
- ✓ **Comportement social :** les agents interagissent avec d'autres agents (éventuellement humains) via une sorte de langage de communication agent.
- ✓ **Réactivité :** les agents perçoivent leurs environnements qui peuvent être le monde physique, un utilisateur via une interface graphique, une collection d'autres agents, l'Internet ou même tous à la fois ; et répondent aux changements qui apparaissent.
- ✓ **Comportement intentionnel :** les agents n'agissent pas simplement en réponse à leur environnement, ils sont capables d'avoir un comportement dirigé vers un but et de prendre des initiatives.
- ✓ **Adaptation Dynamique**: La capacité pour les systèmes d'agent mobile de sentir leur environnement et réagir aux changements qui est utiles dans la découverte de l'intrusion. Les agents peuvent se déplacer pour gagner mieux ou évitez le danger.
  Les agents peuvent ajuster aussi aux situations favorables aussi bien que défavorables.
- ✓ **Indépendance par rapport à la Plate-forme**: Les systèmes d'agent fournissent un environnement de l'informatique abstrait pour les agents, indépendant du matériel informatique et logiciel sur qu'il exécute.

6. Inconvénients des agents mobile

- ✓ Ils imposent une utilisation des ressources comme tous les IDSs.
- ✓ L'entraînement des agents prend du temps, et ils peuvent être corrompus.
- ✓ **Sécurité** : la sécurité du code mobile est moins grande par les techniques de sécurité classiques.
- ✓ **Performance** : il faut voir la rapidité avec laquelle l'agent détecte et remonte l'information d'intrusion.
- ✓ **Taille du code** : les IDSs sont complexes et les agents risquent de demander d'assez gros programmes.
- ✓ **Manque de connaissance de base** : beaucoup de plates-formes et de configurations différentes.
- ✓ **Exposition limitée** : il faudra adapter certaines structures à cette technologie.
- ✓ **Difficultés de codage et de déploiement** : il faudra un code sûr pour beaucoup de fonctionnalité.

Il y a d'autres ***inconvénients*** : quand les agents se déplacent, un nœud dépourvu d'agent est vulnérable pendant un moment. De plus, si les agents ont besoin d'un apprentissage, ce temps peut être long. Enfin, certains attaquants réussiront toujours à obtenir des droits pendant quelques temps avant d'être détectés.

7. La vulnérabilité de la sécurité

Les menaces de la sécurité pour l'agent mobile qui calcule le paradigme peuvent être classées dans **quatre catégories générales :** l'agent à - agent, agent à - plate-forme, plate-forme à - agent, et autre à - plate-forme.

- ✓ **La catégorie agent - à - agent** représente l'ensemble de menaces dont les agents exploitent le weaknesses de la sécurité d'où ils lancent des attaques contre autre agent qui résident sur la même plate-forme de l'agent.
- ✓ **La catégorie agent - à - plate-forme** représente l'ensemble de menaces dont les agents exploitent le weaknesses de la sécurité d'où ils lancent des attaques contre une plate-forme de l'agent où ils résident.
- ✓ **La catégorie plate-forme - à - agent** représente l'ensemble de menaces dont les plates-formes de l'agent compromettent la sécurité d'agents qui résident là.
- ✓ **Catégorie autre -à - plate-forme** représente l'ensemble de menaces dont les entités externes, y compris agents et plates-formes de l'agent situées sur le réseau ailleurs,

menacent la sécurité d'une plate-forme de l'agent, inclure son étant sous le service du système d'exploitation et communications du réseau.

8. Les familles d'agents

Il est possible d'identifier au moins quatre familles d'agents :

8.1. Les agents de gestion de profils

C'est des agents capables d'optimiser la recherche d'information et la veille sur l'Internet. Leurs rôles consistent à rechercher, manipuler et surveiller l'information issue de différentes sources. En générale, ils réagissent en fonction de renseignement fournis par l'utilisation (l'apprentissage) et peuvent lui suggérer des actions à entreprendre (la collaboration)

8.2. Les interfaces intelligents

Possèdent une capacité d'autonomie et disposent également d'une certaine capacité d'apprentissage, ces interfaces intelligente coopèrent davantage avec l'utilisateur plus tôt qu'avec d'autre agents

Ils sont définies comme des assistants personnels capables de collaborer avec l'utilisateur dans le même espace de travaille.

8.3. Le vrai agent intelligent

L'ensemble de ses attributs est doté d'une autonomie, il peut décider de partir sur un réseau (Internet ou autres) et éventuellement d'interagir et de collaborer avec d'autre agents dons le bute de ramener des informations pertinentes et fiable. Malheureusement, un tel agent n'a pas encore vu le jours.

8.4. Les agents mobiles

Leurs principale caractéristique demeure dans leurs capacité de déplacement sur les réseaux informatiques, afin d'accomplir de taches précises indépendamment de l'utilisateur.

9. Le paradigme d'Agent mobile

La mobilité du code n'est pas un nouveau concept, plusieurs mécanismes ont été conçus et mis en application pour déplacer le code entre les nœuds d'un réseau (requêtes SQL, Applets,…).

9.1. Définition

Un agent mobile est un processus, incluant du code et des données et éventuellement un état d'exécution, pouvant se déplacer entre des machines pour réaliser une tâche.

9.2. Le code de l'agent

Donc un agent est composé de son ***code*** correspondant à des algorithmes.

9.3. L'état d'un agent

Un ***état*** incluant des données. Cet ***état*** peut évoluer en cours d'exécution.

Le code et l'état de l'agent sont déplacés avec l'agent lorsque celui-ci visite les différents serveurs.

9.4. Plate-forme d'agent mobile

Un système d'agent mobile est l'infrastructure qui implémente cette technologie.

Chaque site qui veut jouet le rôle d'un support pour l'exécution, la communication, la migration et la gestion de la sécurité doit fournir un environnement convenable, c'est le ***serveur d'agent***.

Mais comme l'agent peut s'exécuter sur des machine hétérogène ayant des système d'exploitation différents, le code de l'agent doit être portable, de ce fait il est préférable que le système d'agent soit implémenté dans un langage de programmation interprété, qui fournit une machine virtuelle pour l'exécution du code de l'agent.

Donc un système d'agent doit :

- ✓ Offrir un environnement où les agents peuvent s'exécuter.
- ✓ Offrir des primitives de migration volontaire des agents.
- ✓ Supporter la communication entre les agents locaux et distants et avec l'environnement.

- ✓ Permettre l'accès aux ressources locales pour des agents spéciaux selon une politique de sécurité.

1. **L'environnement d'exécution**

Pour pouvoir s'exécuter sur un hôte, un agent doit être admis et des ressources doivent lui être allouées. Donc tout hôte qui accueille des agents doit fournir un environnement d'exécution qui permet :

- ✓ **Récupération d'un agent**

Lorsqu'un agent arrive à une machine, un événement indiquant son arrivée est généré pour prendre en charge la procédure d'initialisation et d'activation de cet agent.

- ✓ **L'activation du code de l'agent arrivée**

Une fois l'agent et arrivé, toute une série d'opération doit être effectuée commençant par le chargement de l'agent jusqu'au lancement effectif de son exécution.

- ✓ **La gestion des erreurs**

Une bonne gestion des erreurs permet d'avoir un système robuste, stable et tolérant aux fautes. Il y a plusieurs sortes d'erreurs :

a. Les erreurs (exceptions) liées à l'environnement d'exécution :

- Code inacceptable par le hôte.
- Arrêt inattendu du hôte (serveur).
- Encombrement ou bien la surcharge de l'hôte.

b. Les erreurs (exceptions) liée à l'agent

- Disparition d'un agent.
- Modification du code ou des données d'un agent.
- Les erreurs liées au réseau de communication.
- La rupture de connexion.
- La surcharge du trafic sur les réseaux.

- ✓ **L'interface utilisateur**

A travers cette interface l'utilisateur peut configurer la politique de sécurité de son serveur ; crier, activer, désactiver et détruire des agents, en plus il peut avoir des informations sur tous les agent accueillis par son serveur et les informations concernant le serveur lui même et éventuellement, des informations sur des serveurs voisins.

- ✓ **Définir une manière pour gérer les événements**

La gestion des événements est un concept capital dans le système informatique moderne. Dans un système d'agent, elle consiste à mette en place un mécanisme qui

permet d'écouter des notifications générées par les agents ou par le serveur et de répondre à ces notifications en fonction de leurs types.

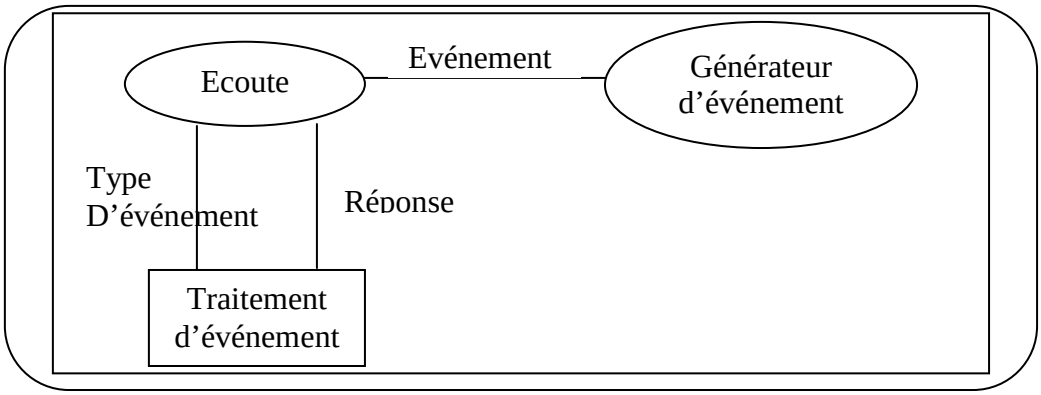

Figure 22 : Modèle de gestion d'événement

2. **La mobilité de l'agent**

L'aspect le plus intéressent dans les systèmes d'agents mobiles est la possibilité des agents de se promener dans le réseau, autrement dit c'est l'agent qui détermine quand, et où se déplace.

- ✓ **Type de mobilité**

On distingue deux types de mobilités :

a. La faible mobilité

Ce type de mobilité consiste à transférer les données et le code de l'agent seulement. Le transfère des donnée est simple, mais en ce qui concerne le transfère du code, le système d'agent doit fournir un support pour réaliser cette opération.

b. La forte mobilité

Le problème qui se pose dans la mobilité faible est la difficulté de la capture de l'état de l'agent, c'est à dire comment transférer en plus des données et du code l'état de l'exécution qui est : *le compteur ordinal, l'ensemble des fichier ouvert, la pile des appels,…etc.* on parle dans ce cas de la "***transparence de mobilité***"

- ✓ **Nature de mobilité**

Selon le degré de mobilité des ressources, on distingue trois sortes des ressources :

1. Ressource non mobile : c'est une ressource qui ne peut pas se déplacer d'un site à un autre (attacher à son site).
2. Ressource copiable : on peut faire une copie.
3. Ressource déplaçable : c'est une ressource qui peut se déplacer d'un site à un autre.

✓ **L'opération de migration de l'agent**

Durant son exécution, l'agent décider qu'il et nécessaire de se déplacer sur un autre serveur dans le réseau, donc, il fait appel à une primitive de migration, le serveur d'agent doit suspendre l'exécution d'agent, capturer ses données et éventuellement son état d'exécution, assembler le tous dans un format standard et l'envoyer à sa destination.

3. **La communication dans les plates-formes d'Agents Mobiles**

Dans tous les systèmes distribués, la communication entre ses composantes est une question très importante. Un système d'agent mobile ne fait aucune exception dans cette règle. De plus, les systèmes à agent mobile ne peuvent pas être imaginé sans communication.

Un agent a besoin de communiquer avec son propriétaire, avec d'autre agents locaux ou à distance, ou avec des services disponibles au niveau du serveur.

La communication avec le propriétaire identifié dans les attributs de l'agent qui doit être asynchrone.

Dans le cas de la communication avec un agent distant, le choix entre la communication distante et la migration sur le site distant est un problème complexe (compromis entre le coût de communication et le coût de migration) dont la résolution est laissée au programmeur dans les systèmes actuels.

✓ **Mécanismes de communications entre agent**

Pour communiquer, les agents doivent se mettre d'accord sur une approche de convocation commune.

Il existe plusieurs mécanismes de communications inter-agents. On peut citer :

- Communication par envoi de message.
- Invocation de méthodes : un agent peut invoquer d'une manière sécurisée les méthodes d'un autre agent.
- Partage de données pour échanger des informations.
- Traitement des événements : les événements sont utilisés comme des messages asynchrones.

Une approche plus sophistiquée a été élaborée dans les systèmes d'agents mobiles pour faciliter la communication entre plusieurs agents est appelée un ***Meeting***, qui consiste à regrouper un ensemble d'agents dans une même place définie par un identificateur et par une URL, et d'informer l'ensemble de chaque nouvel agent arrivé, et de déterminer est ce

qu'ils sont tous (les agents) présents, et en fin de leurs permettre (les agents) de communiquer d'une manière contrôlée.

- ✓ **Mécanismes de communications agent/serveur**

Une autre sorte de communication celle entre l'agent et le serveur.

Le serveur d'agent doit contrôler l'état de tous ses agents qui s'exécutent localement ou, sur un serveur distant, lorsque une erreur d'exécution d'agent se produit, le serveur doit terminer l'exécution de ce dernier, cette opération nécessite de connaître l'emplacement de l'agent et de lui demander de retourner à son emplacement d'origine soit pour lui permettre de continuer ou pour le tuer.

Le serveur peut utiliser un mécanisme d'événement pour signaler l'agent, ou soulever une exception à distance. Le modèle de traitement des exceptions de l'agent doit lui répondre en s'émigrant lui même.

4. La sécurité dans les plates-formes d'Agents Mobiles

Les problèmes de sécurité dans les systèmes d'agents mobiles sont divers. Ils sont une des questions les plus cruciales pour l'acceptation des systèmes d'agents mobiles. Dans un système d'agents mobile on utilise deux concepts principaux à savoir l'agent et le serveur pour identifier la source, peut causer l'attaque ainsi qu'être la victime.

- ✓ **Types d'attaque possibles**

On a quatre types t'attaques possibles dans les systèmes à agent mobile, à savoir :

1. L'attaque d'un agent contre un autre agent.
2. L'attaque d'un agent contre un serveur.
3. L'attaque d'un serveur contre un agent.
4. L'attaque externe d'un système d'agents mobiles.

- ✓ **Techniques utilisées dans les attaques**

Chaque composante peut exploiter la vulnérabilité d'une autre composante pour effectuer des attaques qui peuvent s'effectuer de plusieurs façons :

1. ***Masquage de l'identité (Masquerading)*** : Quand une composante change son identité pour gagner plus de privilège, ce masquage peut endommager les agents comme il peut endommager les serveurs.
2. ***Le déni de service*** : Les agents peuvent bloquer les services d'un serveur par la consommation d'une grande quantité des ressources locales (CPU, Mémoire, …), les attaques de déni de service peuvent être lancer intentionnellement ou à travers des erreurs de programmation.

3. **L'accès non autorisé :** Les mécanismes de contrôle de l'accès sont utilisés pour prévenir l'accès des utilisateurs (agent ou autre) non autorisés aux ressources locales pour lesquelles ils n'ont pas des permissions attribuées par la politique de sécurité.
4. **La répudiation :** La répudiation apparue quand un agent ou un serveur qui a participé dans une transaction ou une communication nie cette opération.
5. **L'écoute clandestine:** Les menaces classiques d'espionnage contiennent l'interception et le suivie de communication secrète.
6. **La modification du comportement :** Quand un agent arrive sur un serveur, il lui expose ses données ainsi que son code, le serveur peut donc changer le code de ce dernier ce qui résulte le changement radical de son comportement.

La sécurité entre serveur et agent a une double direction : d'une part, les serveurs doivent être protégés contre les agents malveillants, et d'une autre part, les agents doivent être protégés contre les serveurs malveillants.

La *première* direction, la protection contre des agents peut être résolue en utilisant la technique existante des programmes de Java et SafeTcl, qui et utilisé pour l'exécution de programmes inconnus, ces deux système emploient la même approche, connu sous le nom du modèle de sécurité Sandbox, où toutes les procédures potentiellement dangereuses sont limité par des commandes spéciales de sécurité qui décident quels sont les programmes qui peuvent employer ces procédures.

L'*autre* direction, la protection des agents contre les serveurs malveillants, est spécifique aux agents mobiles, et l'effort de recherche continus en essayant de fournir des approches dans ce domaine.

10. Etude comparative de quelques plates-formes d'agents mobiles

Pour avoir une idée sur l'état de l'art de la technologie d'agent mobile, on va évaluer quelques plates-formes en les analysants autour de quatre points : exécution, migration, communication et sécurité.

10.1 Voyager

Développé par la société ObjectSpace de Dalles en USA. Voyager est un environnement de développement d'applications réparties à base de Java supportant les objets mobiles et les agents.

✓ **Exécution**

Voyager comporte un ensemble de classes Java pour supporter la mobilité et la communication, un agent s'exécute dans une machine virtuelle Java. Un serveur doit être lancé sur chaque site, le choix d'un numéro de port différent permet de lancer plusieurs serveurs sur le même site. Les méthodes d'une classe distante sont re-routés par le serveur local, via une référence virtuelle, vers l'objet distant. Ce mécanisme permet la migration, la communication et le suivi des objets distant.

✓ **Migration**

Voyageur supporte seulement la mobilité faible, la migration et proactive et réactive.

✓ **Communication**

La communication par message est synchrone, asynchrone, et "*oneway*" (aucune réponse attendue). Le multicast est possible, y compris entre agents distants.

✓ **Sécurité**

Les agents sont divisés en deux catégories ayants des droits différents. Les agents dont le code réside dans le système de fichier local sont considérés comme sûre, et pouvant effectuer toutes les opérations, les autres ont des droits restreints. Ces droits ne sont pas directement paramétrables par l'utilisateur.

10.2 Odyssey

Odyssey est développée par Général Magic en USA, elle est implémentée sous forme de bibliothèque qui permet la construction des agents mobiles, cette plate-forme est implémentée en Java et nécessité le JDK 1.1.x.

✓ **Exécution**

Puisque Odyssey est implémenté en Java, tous les agents développés par cette plate-forme sont exécutés dans la JVM (Java Virtual Machine).

✓ **Migration**

Java fournit une manière de capturer l'état d'exécution d'un thread. Ainsi, quand un agent d'Odyssey et transporté d'un système à un autre, le thread de l'agent et remis en marche à la destination à un point prédéfini. Par conséquent Odyssey fournit une classe worker qui est une sous-classe de la classe agent, par ce fait l'agent est structuré comme un ensemble de paires destination-tâches, à chaque destination, l'agent exécute la tache associée. Odyssey emploie le service de sérialisation d'objet fourni par Java pour transférer les agents.

✓ **Communication**

La communication dans Odyssey est possible entre les agents réunis dans le même endroit alors que la communication à distance est impossible.

✓ **Sécurité**

La sécurité n'est pas documentée.

10.3 Aglets

Aglets (contraction d'agent et applet) et le nom donné par le Tokyo Research Laboratory d'IBM à un environnement de programmation et d'exécution d'agents mobiles écrits en langage Java. Il est disponible sous forme d'une classe Java.

✓ **Exécution**

L'environnement Aglets comporte un ensemble de classes Java pour le support de la mobilité et de la communication. Une Aglets est un objet Java. Il s'exécute dans une machine virtuelle Java. Sur chaque machine susceptible d'accueillir des agents, un serveur (graphique ou en mode ligne de commandes) doit être lancer pour fournir l'environnement d'exécution et le suivi des agents. Plusieurs serveurs peuvent coexister sur la même machine avec des numéros de port différents.

✓ **Migration**

L'Aglets supporte la migration proactive avec la mobilité faible (et la gestion des exceptions). Ceci provient de l'architecture actuelle de la machine virtuelle Java, qui ne permet pas à un programme d'accéder directement à son état d'exécution. C'est pourquoi le programmeur doit sauver dans des variables les informations nécessaires à la reprise de l'exécution dans l'état approprié. Ainsi la migration exécute successivement la méthode *onDispatching()* sur le site de départ, puis *onActial()* et *run()* sur le site d'arrivée. Les communications entre sites s'effectuent par le protocole TCP sur un port configurable via un mécanisme de désignation de type URL.

✓ **Communication**

La communication est par envoie de message entre les Aglets d'un même site ou des sites différents. Les messages peuvent être synchrones, asynchrones, et "*oneway*" (aucune réponse attendue). Enfin le multicast est possible pour un ensemble d'aglets situés sur le même site, l'environnement d'exécution envoyant le message d'un type donné aux aglets du site ayant souscrit à la réception des messages de ce type.

✓ **Sécurité**

La sécurité des aglets bénéficie des mécanismes de sécurités propres à Java. Da plus, les serveurs d'aglets ont la possibilité de définir les droits d'accès au système de fichier, les droits d'utilisateurs du réseau, les droits d'ouverture de fenêtres, …etc., il n'est pas possible par contre de limiter l'accès aux ressources CPU et mémoire.

10.4 Agent TCL de Datmouth college(UK)

La plate-forme Agent TCL est une variante de l'interpréteur TCL pour supporter le mécanisme de mobilité forte, le support de la mobilité est donc obtenu par la modification du noyau TCL pour offrir une commande qui permet de suspendre l'exécution d'un script TCL, capture son état, le déplacer vers une autre machine et enfin exécuter l'instruction qui vient juste après la primitive de migration de ce script.

✓ **Exécution**

L'environnement d'exécution consiste en deux serveurs :

1. Le démon pour gérer la migration et la communication à distance.
2. Le gestionnaire des ressources pour contrôler l'accès aux ressources locales.

✓ **Migration**

L'agent peut migrer volontairement en utilisant sa primitive de migration. La modification du TCL permet la capture de l'état d'exécution donc, la forte mobilité.

✓ **Communication**

Deux types de communications sont possibles :

1. L'envoi de message : permet la communication asynchrone.
2. L'établissement des canaux de communication : les agents peuvent communiquer directement après l'établissement d'un canal de communication entre eux.

✓ **La sécurité**

La sécurité dans la plate-forme agent TCL est assurée par la spécification de la sécurité propre au langage, en plus les modules de sécurités indépendants pour le contrôle des ressources locales.

10.5 Comparaison entre les plates-formes

Pour exécuter un agent on a besoin d'une plate-forme qui est un environnement d'exécution, de migration et de communication pour les agents (mobile et stationnaire). Il

existe plusieurs plates-formes, chacune d'entre elles représente des avantages et des inconvénients :

Plate-forme	Avantages	Inconvénient
Voyageur	-Support de persistance -MAJ automatique de la référence des objets pour les agents migrés. -Concept de diffusion (broadcast) des messages. -Evénements distribués	-Support de sécurité insuffisant.
Odysey		-Support de sécurité insuffisant -Pas de communication distante. -Support de persistance insuffisant. -L'agent perd les référence des objets dés qu'il se déplace.
Aglet	-Un GUI pour la gestion et la configuration. -Mécanismes haut niveau pour le contrôle d'accès.	-Pas d'authentification des agents (pas de sécurité externe).
Agent TCL	-Capture de l'état de l'exécution. -Sécurité interne et externe.	-Support de persistance insuffisant. - Documentation incomplète. -Portabilité limitée. - L'agent perd les références des objets dés qu'il se déplace. -Toutes les clés publiques doivent être connues.

Tab 5 : ***Comparaison des plate-formes***

Voyager est un environnement de développement d'applications réparties à base de Java supportant les objets mobiles et les agents. Voyager a un support de persistance, une mise à jours automatique de la référence des objets pour les agents migrés et un concept de diffusion des messages, mais le problème d'insuffisance du support de sécurité se pose.

Le problème d'insuffisance du support de sécurité et de persistance s'impose pour Odyssey, ainsi que l'absence de la communication distante et la perte des références des objets lors du déplacement de l'agent.

Aglets a une interface graphique conviviale pour la gestion et la configuration ainsi que le haut niveau pour le contrôle d'accès. Etant donné que les agents communiquent par envoi de messages, donc pas d'authentification des agents ce qui inflige l'absence de la sécurité externe.

Agent TCL a une sécurité interne et externe ainsi que la possibilité de capturer l'état de l'exécution. Mais les problèmes d'insuffisance de persistance, de limitation de la portabilité et la perte des références des objets lors du déplacement de l'agent s'impose, ainsi que l'obligation de la divulgation des clés publique ce qui rend le système (réseau) vulnérable et exposé aux intrusions.

Annexe2 La plate-forme Aglets

1. Introduction

La plate-forme Aglets est la prochaine génération dans l'évolution du code mobile sur Internet, c'est l'introduction d'un code qui peut migrer dans l'Internet avec son état (donnée), elle a été développée par IBM en 1997, elle permet la mise en place d'applications distribuées en Java grâce à sa bibliothèque de développement ASDK.

Les Aglets (Agents Applets) sont des objets Java mobile qui peuvent se déplacer d'une machine à une autre.

Une Aglets s'exécute sur un hôte, peut arrêter son d'exécution, se déplacer vers un autre hôte distant et recommencer l'exécution. Quand une Aglet se déplace, elle emporte avec elle son code (byte code) ainsi que son état (données).

Pour exploiter la plate-forme Aglets d'une manière efficace, il faut comprendre son architecture. On présente dans ce chapitre un aperçu sur les principaux composants de celle-ci, ainsi que des différents aspects concernant la communication, la migration et la sécurité de la plate-forme Aglets.

2. Choix de la plate-forme

Le choix de la plate-forme Aglets repose sur plusieurs raisons : [6]

- ✓ **La maturité**

La plate-forme Aglets fournie tous les concepts des agents mobiles, c'est une plate-forme stable et cohérente, offre en plus des fonctionnalités de base, des utilitaires (les itinéraires, Messagerie…). Elle fournie aussi un serveur d'agent avec une interface graphique agréable appelé TAHITI.

- ✓ **Une plate-forme implémentée en Java**

Comme elle est écrite 100% en Java, elle hérite donc toute la force de ce langage purement orienté objet en plus de la portabilité et de la sécurité. Java offre des mécanismes qui supportent la mobilité du code.

- ✓ **La disponibilité**

Aglets n'est pas un produit commercial, elle est disponible avec son code source. La plate-forme est constituée d'un ensemble de packages appelés ASDK et d'exemple de la spécification d'Aglets écrit par ses auteurs OSHIMA et GUENTER KARJOTH deux

célèbres auteurs dans le développement de cette plate-forme.

3. Les objectifs de la plate-forme Aglets

- ✓ Fournir un modèle facile et compréhensible pour la programmation des agents mobiles sans avoir besoin de modifier la machine virtuelle Java.
- ✓ Conception d'une architecture extensible et réutilisable.
- ✓ Conception d'une architecture conforme avec la technologie Java/Web.
- ✓ Support dynamique de communication des agents.

4. Architecture de la plate-forme Aglets

Aglets a une architecture spécifique, consiste à avoir un ensemble de serveurs connectés, chaque serveur est conçu autour d'un runtime java, qui fourni des services aux agents accueillis localement. Un serveur est contrôlé par un utilisateur via une interface graphique (GUI) ou une console ; il peut créer, envoyer…etc. des agents. [6]

4.1 Les éléments de la plateforme Aglets

- ✓ **Aglets**

C'est un objet mobile Java qui visite les serveurs où les agents sont autorisés dans un réseau informatique. Une aglets est ***autonome*** puisqu'elle peut répondre à son exécution dés son arrivée à destination et ***réactif*** car elle répond (réagi) à des événements de son environnement.

- ✓ **Proxy**

Un proxy est un représentant d'une aglets. Il sert à protéger l'aglets contre les accès directs à ses méthodes publiques. Le proxy fournit également la transparence de l'emplacement de l'aglets, c'est à dire qu'il peut cacher le vrai emplacement de l'aglets.

- ✓ **Contexte**

Le contexte est le lieu d'exécution de l'aglets, c'est un objet stationnaire qui fournit des moyens pour mettre à jour et contrôler les aglets dans un environnement d'exécution uniforme, cet environnement est protégé contre les aglets malveillantes.

Un serveur peut accueillir plusieurs contextes, chaque contexte a un nom et peut être localisé par la combinaison de son nom et de l'adresse du serveur qu'il l'accueille.

- ✓ **Hôte**

Un hôte est une machine capable d'héberger plusieurs serveurs. L'hôte est

généralement un nœud dans un réseau.

- ✓ **Identificateur**

Une identification est reliée à une aglets. Cette identification est globale et unique et non changeable tout au long de la vie de l'aglets.

4.2 Cycle de vie d'une Aglets

Les ingénieurs d'IBM ont mis en place diverses situations dans lesquelles une aglets peut se trouver. Elle peut être en cours de *création*, d'*exécution*, de *déplacement*, de *libération* ou bien *stocké* sur le disque.

Découper le cycle de vie d'une aglets en étapes (ou situation) distinctes permet de bien définir ce qui peut être effectué sur une aglets en fonction de la partie du cycle de vie dans laquelle se trouve cette dernière.

Les types du comportement des aglets ont été implémentés de manière à répondre aux principaux besoins des agents mobiles.

- ✓ **Création**

La création se fait dans un contexte. Un identifiant unique lui est assigné. L'initialisation et l'exécution de l'aglets commencent immédiatement.

- ✓ **Clonage**

Le clonage est la création d'un duplicata dans le même contexte que l'origine. Un identifiant différent lui est alors attribué.

- ✓ **Déportation (Dispatching)**

C'est le transfert d'une aglets d'un contexte à un autre, on dit que l'aglets est migrée vers son nouveau contexte.

- ✓ **Récupération**

L'aglets déposée est récupérée dans son contexte d'origine.

- ✓ **Activation et Désactivation**

La désactivation d'une aglets est une interruption temporaire de son exécution et le stockage de son état dans un support secondaire de stockage. L'activation est l'opération inverse.

- ✓ **Libération ou Déstructure**

C'est la fin de vie de l'aglets et le retirer du contexte.

- ✓ **Messagerie**

La messagerie entre les aglets consiste à l'*émission*, la *réception* et le *traitement* des messages synchrones ou asynchrones

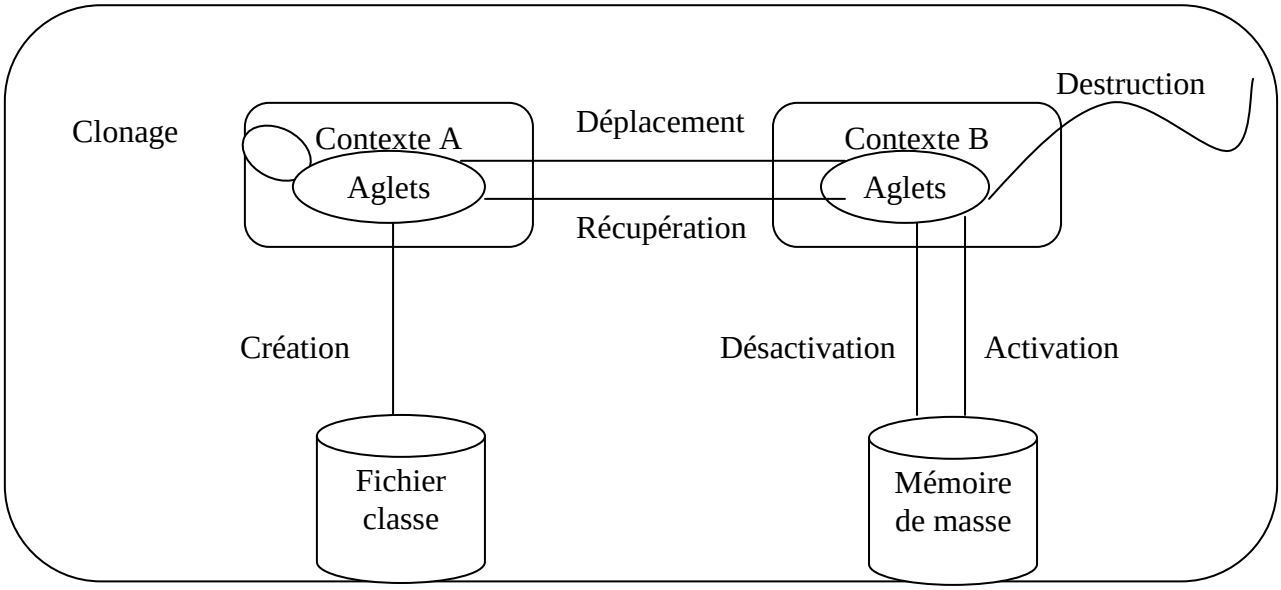

Figure 23 : Modèle du cycle de vie d'une Aglets

4.3 La migration d'une Aglets

La plate-forme Aglets supporte la migration proactive avec mobilité faible. Cela dérive de l'architecture de la machine virtuelle Java.

Une aglets peut se migrer à une autre destination. La migration cause la suspension de son exécution, la sérialisation de son état interne et son code dans un format standard et en fin le transfert vers la destination. A la réception, l'aglets est reconstruit et recommence l'exécution du début.

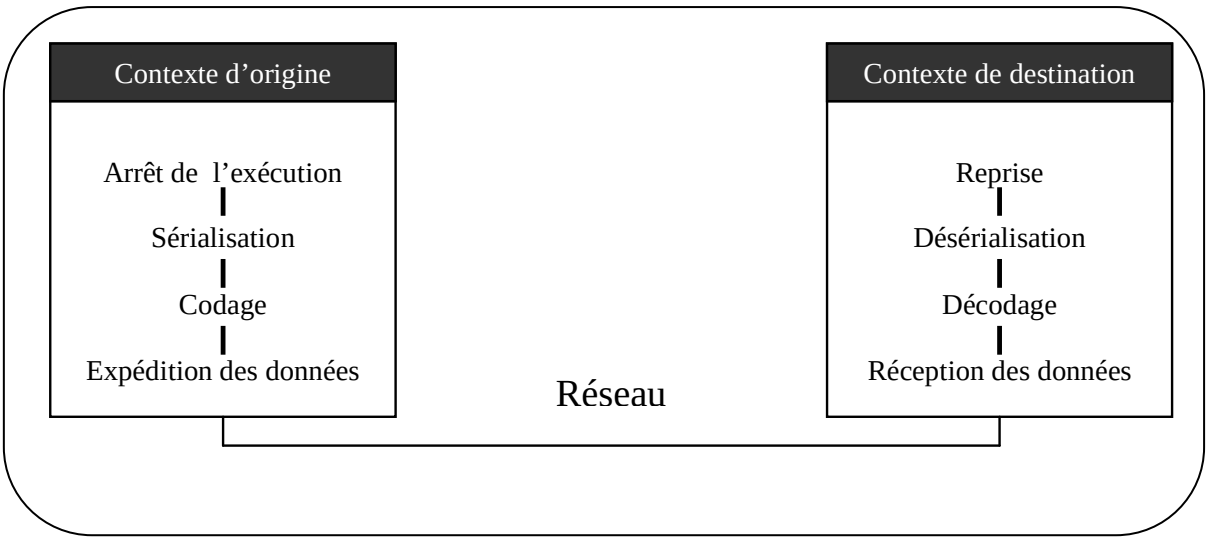

Figure 24: Le transfert d'une Aglets

- ✓ **Le protocole de transfert d'agents :**

La plate-forme Aglets est décomposée en deux couches :

1. ***La couche d'exécution*** : C'est la couche qui est responsable de l'exécution des aglets.
2. ***La couche de transport*** : C'est la couche qui est responsable de transporter une aglets à la destination sous forme d'un flux d'octets qui contient des définitions de classe (code) aussi bien que l'état de l'aglets ordonné dans un format standard. Les fonctionnalités de cette couche sont assurées par le protocole de transfert d'agent ATP, qui est un protocole simple de niveau application du modèle de référence OSI.

La modélisation de ce protocole est basée sur le protocole HTTP, en plus, il a été conçu pour transmettre un agent indépendamment de sa plate-forme. Les requêtes sur lesquelles est basé ce protocole sont :

- ***Dispatch*** : C'est une requête destinée au système d'agent à distance, pour reconstruire l'agent à partir du contenu de la requête, et de le lancer à nouveau. Si la requête est satisfaite, l'émetteur arrête l'agent et libère toutes les ressources utilisées par ce dernier.
- ***Retract*** : C'est une requête où le système d'agent à distance, renvoie l'agent spécifié à l'émetteur d'origine. Le récepteur de l'agent est le responsable de la reconstruction et la mise en marche de l'agent. Si l'agent est transmet avec succès l'émetteur stoppe l'agent et libère toutes les ressources utilisées par celui-ci.
- ***Fetch*** : Similaire à GET du HTTP, c'est une requête où le récepteur doit récupérer et envoyer l'information identifié.
- ***Message*** : Est utilisé pour envoyer des messages entres les agents identifiés par leurs identificateurs ID.

D'une manière générale, ce protocole est basé sur le modèle requête/réponse, et ne dispose d'aucune règle pour la communication entres les agents.

4.4 Le modèle de communication de l'Aglets

La communication est basée sur l'échange d'objets de classe ***Message.*** Ce modèle de communication est indépendant de la location de l'aglets et peut être synchrone ou asynchrone.

Quand une aglets désire envoyer un message, doit obligatoirement passer par le proxy du destinataire.

Chaque aglets possède un gestionnaire de message ***MessageManager*** qui lui permet de les traiter un par un et dans l'ordre de leurs arrivées respectives. Cet ordre peut être changé par l'aglets en modifiant les priorités des messages dans la file d'attente, ceci est possible grâce à la méthode ***setPriority*** (les priorités ont des valeurs de 1 à 10).

- ✓ ***La classe Message***

Chaque message est caractérisé par la catégorie (***Kind***) à laquelle il appartient et au contenu (***Arg***) de n'importe quel type. La création d'une instance message nécessite l'affectation d'une valeur à son paramètre ***Kind***.

Les constructeurs les plus utilisés de la classe :

1. *public Message (String Kind)* : Crée un message d'un type donné (Kind). Un tableau (Hashtable) par défaut est assigné pour ce type de message.
2. *public Message (String Kind, Object Arg)* : Crée un message d'un type donné (Kind) avec un argument de type Object.

Figure 25 : L'objet Message

- ✓ ***Type de messages***

Il y a quatre types de messages :

1. ***Message synchrone*** : La méthode ***AgletProxy.sendMessage(Message msg)*** permet l'envoie de messages synchrones, elle est donc bloquante car elle attend la réponse du récepteur. Le destinataire du message doit implémenter un gestionnaire des messages reçus. Cette fonctionnalité est rendue possible par la surcharge de la méthode ***Aglet.handelMessage(Message msg)***. La méthode ***Message.sendReply*** permet de retourner une réponse à l'émetteur.
2. ***Message asynchrone*** : La messagerie asynchrone est implémentée grâce à la notion de ***futur objet***. L'envoi d'un message asynchrone retourne un lien vers la réponse si cette dernière n'existe pas encore. Ce lien permet de tester si la réponse

est arrivée ou pas. Cette technique permet à l'aglets d'envoyer un message sans être obligé d'interrompre son exécution en attente de la réponse.

3. ***Multicast (diffusion des messages)*** : Permet à une aglets de s'abonner à un ou plusieurs groupes au sein d'un même contexte. Chaque groupe est identifié par une catégorie de messages.

   Le message Multicast fournit un mécanisme puissant pour la communication et la collaboration des aglets, pour qu'une aglets bénéficie de ce type de message il procède comme suit :

   - ✓ Inscription dans un seul ou plusieurs messages Multicast.
   - ✓ Implémentation de porteur (handeler) de chaque type de message.

4. ***Message simple (unidirectionnel)*** : Il permet l'envoi un message sans attente de réponse.

5. Les Aglets et la sécurité

Dans un réseau en pleine expansion tel que l'Internet, le problème de la sécurité trouve tout son sens dans l'utilisation des agents. On peut identifier trois cibles possibles : [43]

5.1 L'agent

L'agent qui visite une machine pourrait être la cible d'une tentative d'exécution d'information. Le même scénario avec un agent malveillant. Etant donné que la communication entre les aglets et les agents se fait par le biais de la messagerie, cette dernière peut être une cible d'attaque.

5.2 Le hôte

Un agent malveillant pourrait visiter un hôte dans le but d'accéder ou de corrompre ses fichiers. Une entité malveillante pourrait envoyer un nombre important d'agent vers un serveur dans le but de le surcharger.

5.3 Le réseau

Multiplication sans fin d'agent dans le but d'encombrement et de surcharger le réseau. Donc, la sécurité est essentielle pour les aglets, car l'acceptation d'un agent hostile peut endommager l'ordinateur.

5.4 Modèle de sécurité d'aglets

La plate-forme aglets est renforcée par un modèle de sécurité, ce modèle est basé sur les mécanismes suivants :

1. **Les principales**

   Une principale sert comme une identité qui peut représenter l'aglets, le contexte, le domaine (ensemble des serveurs) etc.

   Le rôle de chaque type des principales est d'assurer la conformité et la porté de tous l'ensemble (aglets, contextes, domaine,…).

2. **Les permissions**

   Les permissions définissent la possibilité d'exécution d'une aglets par :

   - ✓ La limitation des ressources consommées.
   - ✓ Restriction d'accès à tel ou tel type de ressource.

   Ces permissions sont basées 100% sur java et concernent :

   - ✓ L'accès aux fichiers (read/write).
   - ✓ L'établissement d'une connexion à une base de données.
   - ✓ L'établissement d'une connexion réseau.
   - ✓ La création d'une icône sur le bureau.

3. **Protection**

   Comme les ressources sont protégées par des permissions, il faut aussi protéger les aglets contre les accès non autorisés des autres entités.

   Par exemple, le seul responsable de détruire et libérer une aglets est l'administrateur (owner).

5.5 Variables d'environnement

Avec les variables d'environnements, un utilisateur peut préciser à la plate-forme Aglets les différentes propriétés du système d'exploitation telle que :

- ✓ Le chemin des aglets locales AGLET_PATH.
- ✓ Le chemin des aglets exportables AGLET_EXPORT_PATH.
- ✓ Le chemin où se trouve l'ASDK (Aglets Software Développement Kit).
- ✓ HOME qui est le chemin spécifique à un utilisateur.
- ✓ JAVA_HOME qui est le chemin où se trouve le JDK (Java Développement Kit).

Bibliographie

[1]: R. Agrawal, T. Imielinsky, et A. Swami. ***"Mining association rules between sets of items in large databases"***. In Proc. of the ACM SIGMOD'93, 1993.

[2]: Jean-Marc Alliot, Nicolas Durand "***Algorithmes génétiques***".Mars 14, 2005

[3]: D. Anderson and T. Lunt and H. Javitz and A. Tamaru and A. Valdes. " ***Detecting unusual program behaviour using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES) "***. Technical Report, SRI-CSL-95-06, Computer Science Laboratory, SRI International, Mai 1995.

[4]: D. Anderson and T. Frivold and A. Valdes. "NIDES: A Summary".

[5]: James Anderson. ***"Computer security threat monitoring and surveillance"***. Technical report, James Anderson company, For Washington, Pennsylvania, April 1980.

[6]: Bettahar Aoued.***"Le Aglets d'IBM "***Université de Montréal Cours IFT6802-H2003.

[7]: P.Besse. ***" Apprentissage Statistique & Data mining "***. Institut de Mathématique de Toulouse. Laboratoire de qtatistique et probabilités-UMR CNRC5583.

[8]: Philippe Biondi : ***" Architecture expérimentale pour la détection d'intrusions dans un système informatique"***. Avril-Septembre 2001.

[9]: Boudaoud Karima. ***" Détection d'intrusions : une nouvelle approche par systèmes multi agents"***. Ecole polytechnique Fédérale de Lausanne, 2000.

[10]: P. Chambet. ***"Firewalls et applications web : architecture et sécurisation – pourquoi les firewalls sont impuissants face aux attaques web"***. Linux Magazine, novembre 2002.

[11]: W.W. Cohen. ***"Fast Effective Rule Induction"***. In Proceedings o/the 12th International Conference on Machine Learning, Lake Taho, CA, 1995.

[12]: Dary Alescandra & Pena Maldonado. ***"Data Minig : A new Intrusion Detection Aproach"***. GIAC, Security Essantials Certification Practical Assignement [SANS Indtitute], June 2003.

[13]: E. Davalo, P. Naim ***"Des réseaux de neurones"*** EYROLLES, 2ème édition. 1993

[14]: H. Debar, M. Dacier & A.Weps. ***"A revised taxonomiy for intrusion-detection systems"***. IBM Research report, 1999.

[15]: H.Debar. ***" Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques"***. Thèse de doctorat, Université de Pris 6, 1993.

[16]: D.E. Denning. ***"An Intrusion Detection Model".*** In IEEE Transactions on Software Engineering, February 1997

[17]: D.E Denning. ***"An intrusion-dection model"***. In : proccedings of the IEEE Transactions on software engineering, Septembre 1987.

[18]: G. Dreyfus , ***" Réseaux de neurones "***, EYROLLES. 2002

[19]: A. El Ouaazizi et al. ***"Line fitting in noisy data using genetic algorithm".*** 3ème Conférence Internationale sur le Contrôle Qualité par Vision Artificielle QCAV'97, 1997.

[20]: U.M. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy. ***"Advances in Knowledge Discovery and Data Mining"***. AIII/MIT Press, 1996

[21]: .J. Frawley, G. Piatetsky-Shapiro, et C.J. Matheus. ***"Kowledge discovery in databases: an overview"***. AI Magazine, 1992.

[22]: Jeremy Franck. ***" Artificial Intelligence and Intrusion Detection : Current and Future Directions"***. Divising of Computer Science. Unversity of California at Davis, 1994.

[23]: Mostafa HANOUNE, Fouzia BENABBOU ***"Modelisation informatique de clients douteux, en utilisant les techniques de datamining"***.

[24]: Guy G. Helmer, Johnny S. K. Wong, Vasant Honavar & Les Miller . ***"Intelligents Agents for Intrusion Detection"*** , Proceedings, IEEE Information Technology Conference, 1998.

[25]: Guy Helmer, Johnny S. K. Wong, Vasant Honavar & Les Miller. ***"Lightweight Agents for Intrusion Detection"***, the Journal of Systems and Software, 2000.

[26]: Guy Helmer, Johnny S.K. Wong, Vasant Honavar & Les Miller. ***"Automated Discovery of Concise Predictive Rules for Intrusion Detection"***, the Journal of Systems and Software 60, 2002.

[27]: J. Hérault et C. Jutten, Ed. Hermès***" Réseaux de neurones et traitement dusignal"***, 1994.

[28]: K. Ilgun. ***" USTAT: A Real-Time Intrusion Detection System for UNIX"***. Master Thesis, University of California, Santa Barbara, November 1992

[29]: H.S.Javitz, A. Valdes, T.F. Lunt, A. Tamaru, M. Tyson & J. Lowrance. ***"Next generation intrusion dection expert system (nids) "***. Rapport technique a016, Rationales, SRI, 1993

[30]: H.S. Javitz and A. Valdes, ***"The SRI IDES Statistical Anomaly Detector"***.

[31]: W. Lee and S.J. Stolfo and K. Mok. ***"Mining Audit Data to Build Intrusion Detection Models"***. In Proceedings of the International Conference on Knowledge and Data Mining, Aout 1998.

[32]: W. Lee and S. Stolfo. ***"Data Mining Approaches for Intrusion Detection"***. In Proceedings of the 7th USENIX Security Symposium, 1998.

[33]: W. Lee and S.Stolfo and K. Mok. ***"A Data Mining Framework for Building Intrusion Detection Models"***. In Proceedings of the IEEE Symposium on Security and Privacy, 1999.

[34]: Lodivic Mé & Cédric Michel. ***" La détection d'intrusion : bref aperçu et derniers développements"***. Mars 1999

[35]: Ludovic, Mé., ***"Audit de sécurité par algorithmes génétiques".*** Thèse de Doctorat, Université de Rennes 1, France, 7 Juillet 1994.

[36]: T.F. Lunt and R Jagannathan. ***"A Prototype Real-Time Intrusion-Detection Expert System"***. In Proceedings of the IEEE Symposium on Security and Privacy, 1988,

[37]: Lovidi Mé. ***"Détection des intrusions dans les systèmes d'information : la nécessaire prise en compte des caractéristique du système surveillé"***. Habilitation à diriger des recherches, université de Rennes 1 l'institut de formation Supérieure en informatique et en Communication de Rennes1, 2003.

[38]: Cédric Michel. ***"Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène"***. Thèse de doctorat, Université de Rennes1, 2003.

[39]: R. Mukkamala and J. Gagnon and S. Jajodia. *"Integrating Data Mining Techniques with Intrusion Detection"*. In Proceedings of the XIII Annual IFIP WG 11.3 Working Conference On Database Security, Seattle, WA, juillet 1999.

[40]: Nicolas Noblis . ***" Un modèle de Case-based Resoming pour la détection d'intrusions"*** Septembre 2004.

[41]: N. Pasquier. ***"Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données"***. Doctorat d'université, Université de Clermont-Ferrand II, France, 2000.

[42]: Alain B & P B ***"Data mining I Exploration Statistique"***. Version septembre 2005

[43]: Patrice.bellot. ***"Les Aglets : des agents en java"*** DESS Commerce Electronique/DESS TAII.2002/03

[44]: P.A. Porras and P.G. Neumann ***"EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances"***. In Proceedings of the National Information Systems Security Conference, 1997.

[45]: P.A. Porras. ***"STAT: A State Transition Analysis for Intrusion Detection".*** Master Thesis, Computer Science Department, University of California, Santa Barbara, 1992.

[46]: Leonid Portonoy. ***"Intrusion Detection with Unlabeled Data Using Clustering"***, ACM Workshop on Data Mining Applied to Security (DMSA 2001), 2001.

[47]: R. Rakotomalala, Revue MODULAD ***"Arbres de Décision "***, Equipe de recherche en Ingénierie des Connaissances. Laboratoire ERIC. 2005

[48]: Renders, J.M., ***"Algorithmes génétiques et Réseaux de Neurones"***. Editions HERMES, 1995.

[49]: I. Sager et al. ***" Cyber Crime"***. In Business Week, Février 21, 2000.

[50]: S. Smaha. ***"Haystack audit trail analysis system"***. Status Report HS-STAT.TXT Haystack Laboratories, Colorado, Aout.1990

[51]: G. Vigna and R. Kemmerer. ***"NetStat: A Network-Based Intrusion Detection Approach"***. In Proceedings of the 14t Annual Information Theory : 50 Years of Discovery Computer Security Application Conference, Dec. 1998.

[52]: I.H. Witten and E. Frank. ***"Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations"***. Morgan Kanfmann, 2000.

[53]: J. Zimmermann, L. Mé, and C. ***"Bidan. Introducing reference flow control for intrusion detection at the os level"***. In Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID). Springer Verlag, 2002.

[54]: ***http://www.toulouse.inra.fr/centre/esr/CV/bontemps/WP/algogene.pdf***