MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DELA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABDELHAMID IBN BADIS–MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

RAPPORT DE MINI-PROJET
Option : ISI

THÈME :

Détection des intrusions à base d'un système multi-agents et de l'apprentissage automatique

Étudiants : MEHDI Kheira et ZITOUNI Chihab Eddine

Membres de Jury :

SEHABA Karim	Président de jury	Prof
BENIDRIS Fatima	Examinatrice	MCA
HOCINE Nadia	Encadrant	MCA

Date de soutenance :26/06/2022

Année universitaire 2021-2022

Résumé

Les systèmes de détection d'intrusions ont pour objectif de protéger la sécurité des systèmes ainsi que la confidentialité et l'intégrité de données. Ces systèmes sont souvent utilisés en réseaux, d'où la complexité de l'analyse du trafic. La plupart des systèmes de détection d'intrusions sont basés sur une architecture centralisée, ce qui les rend vulnérables en cas d'attaques de déni de service. De plus, la performance et le temps de réponse du système de détection d'intrusions peuvent être négativement influencés par la densité du trafic réseau. Nous proposons dans ce projet un système de détection d'intrusions basé sur une architecture décentralisée à travers un système multi-agent. Des agents mobiles sont plus particulièrement utilisés pour analyser le trafic afin d'améliorer la performance de la détection d'intrusions. Pour l'analyse et la détection d'intrusions, nous utilisons des méthodes d'apprentissage automatiques : les arbres de décision, SVM et réseaux de neurones. Nous avons également développé une application mobile pour administrateur et superviseur le à travers pour notre système de détection d'intrusions. Le but est de faciliter la supervision de l'état de sécurité du réseau afin de pouvoir réagir efficacement en cas d'intrusions détectées.

Abstract

Intrusion detection systems aim to ensure security as well as data privacy and integrity of systems. They are often used for network security, hence the complexity of the traffic analysis. Most of intrusions detection systems are based on centralized architecture, what makes them vulnerable in case of denial-of-service attacks. Furthermore, the performance and response time of intrusions detection systems could be negatively influenced by network traffic density. In this project, we proposed an intrusion detection system based on decentralized architecture through a multi agent system. In particular, mobile agents are used to analyze the network traffic, which can improve the performance of the system. To analyze and detect intrusions, we used machine learning classifiers such as decision tree, support vector machine and neuronal networks. We've also developed a mobile application to help network administrator in supervising the network traffic in order to insure quick intervention in case of anomalies.

ملخص

الغاية من انظمة كشف التسلل هو حماية أمان الأنظمة و كذلك خصوصية و سلامة المعلومات. هاته الأنظمة عادة ما تستخدم في الشبكات الاتصال اين هنالك تعقيدات في تحليل الحركة المرورية داخل الشبكة.معظم انظمة كشف التسلل مبنية على هندسية مركزية، مما يجعلها غير محصنة في حالة هجوم من نوع الحرمان من الخدمات و علاوة على ذلك أداء و وقت أستجابة أنظمة كشف التسلل بمكن ان يتأثر سلبيا بسبب كثافة الحركة المرورية داخل الشبكة.في هذا المشروع اقترحنا نظام كشف التسلل مبني على هندسية لا مركزية من خلال نظام متعدد الوكلاء. الوكلاء متنقلين يستعملون بكثرة من أجل تحليل حركة المرور داخل الشبكات من اجل تحسين أداء كشف التسلل.لتحليل و كشف التسلل، قمنا بإستعمال مصنفات التعلم الآلي. و قمنا أيضا بتطوير تطبيق محمول و المشرف الخاص بنظام كشف التسلل الخاص بنا، و التطبيق يهدف الى تسهيل الاشراف على أمن و حالة شبكة الاتصال و الاستجابة بسرعة في حالة كشف تسلل.

Les mots clés : Systèmes de détection d'intrusions, système multi-agent, apprentissage automatique.

Remerciements

Nous tenons tout d'abord à remercier Dr. Hocine Nadia de nous avoir suggéré ce sujet intéressant, pour son encadrement avec patience, ses précieux conseils, sa disponibilité et son soutien tout au long de ce travail.
Je remercie également les membres du jury d'avoir accepté d'évaluer notre travail.

Nous tenons également à remercier Mattalah Wafaâ qui nous a soutenu depuis tout le début de ce projet et à tous ceux ce qui nous aidés de près et de loin dans de projet.

TABLE DES MATIERES

TABLE DE FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

Les ordinateurs, les smartphones et tous les appareils connectés à Internet sont devenus de plus en plus des cibles d'activités malveillantes. Par conséquent, afin de lutter contre ces activités, plusieurs approches ont été développées telles que l'utilisation des pare-feu (ou middlewares). Cependant, les pare-feu offrent généralement une protection statique et n'arrivent pas à détecter de nombreuses intrusions existantes. C'est pourquoi les systèmes de détection d'intrusion (ou IDS) ont été apparus qui permettent de surveiller différents types de menaces complexes [1].

Plusieurs IDS ont été proposés et adoptés par les organisations. Généralement, les IDS sont classés en deux catégories en fonction de la source des données analysées. Le premier type, et le plus largement utilisé dans les universités et les industries, est l'IDS basés sur le réseau (NIDS). NIDS analyse le trafic réseau en examinant l'activité des paquets pour trouver une activité suspecte. Le deuxième type des IDS est basé sur l'hôte (HIDS). Il s'appuie sur des sources de fichiers journaux de la machine, tels que les journaux système, afin de détecter les anomalies. Dans la plupart des cas, on trouve des IDS hybrides qui combinent à la fois l'analyse du réseau et les données de la machine.

Il existe de nombreux types de systèmes de détection d'intrusion utilisés et adoptés par l'industrie, notamment les IDS centralisés et les IDS distribués. Dans un IDS centralisé, l'analyse de l'intrusion s'appuie sur les paquets entrants de la machine. Une analyse simple qui permet d'observer les changements dans les propriétés statistiques du flux entrant. Ce type d'IDS n'est pas très souvent utilisé dans le cadre de systèmes avec une architecture distribuée et nécessite une analyse de réseau afin de détecter les attaques massives visant les différents appareils connectés tels que les attaques Dos (Deni of Service). Un IDS distribué se compose de plusieurs entités logicielles qui sont généralement connectées à un serveur central permettant la surveillance du réseau [6]. L'objectif est de recueillir des informations provenant de diverses sources pour détecter les attaques contre un système de réseau.

Afin d'améliorer les performances de l'IDS, les systèmes multi-agents (ou MAS) ont été largement utilisés pour créer des IDS performants en s'appuyant sur les caractéristiques des agents telles que l'intelligence, l'autonomie, la mobilité, la coopération et la flexibilité. Cependant, les résultats d'analyse des performances des systèmes de détection d'intrusion à base de MAS (ou MAS-IDS) actuels restent insuffisants en termes de temps de traitement et de tolérance aux pannes malgré l'existence de plusieurs suggestions telles que l'amélioration de l'algorithme de la détection d'intrusions et la conception du MAS. De plus, les MAS-IDS existants n'offrent pas un moyen facilitant la supervision de l'état de réseau à distance permettant ainsi de réagir rapidement en cas d'anomalies.

Ce projet de Master vise à proposer une architecture MAS pour les systèmes de détection d'intrusions qui a pour objectif d'améliorer la performance des IDS existant en réduisant le temps d'analyse et du traitement de données du trafic réseau. Le MAS-IDS proposé se basera aussi sur une application mobile permettant de faciliter la supervision réseau en temps réel, qui est une tâche complexe pour un administrateur réseau. Ce projet de Master vise à proposer une architecture MAS pour les systèmes de détection d'intrusions qui a pour objectif d'améliorer la performance des IDS existant en réduisant le temps d'analyse et du traitement de données du trafic réseau. Le MAS-IDS proposé se basera aussi sur une application mobile permettant de faciliter la supervision réseau en temps réel, qui est une tâche complexe pour un administrateur réseau. Le but est permettre de mieux faciliter le traitement des alertes afin de mieux superviser le réseau et réagir.

Ce rapport se compose de quatre chapitres :

- **Chapitre I** : Nous présenterons dans ce chapitre les types des systèmes de détection d'intrusions existants, les approches de détection d'intrusions, les différentes techniques d'implémentation et de traitement d'intrusions ainsi que les dataset les plus utilisées dans la littérature.
- **Chapitre II** : Nous décrivons le résultat de l'analyse de l'état de l'art sur les techniques de détection d'intrusions à base de systèmes multi-agents. Cette analyse nous permettra d'étudier les techniques existantes et de proposer un MAS-IDS à travers ce projet de Master.
- **Chapitre III** : Nous présenterons l'architecture générale du MAS-IDS proposée et le résultat de la phase de conception de notre système. Nous présenterons aussi le dataset utilisé par les classificateurs de notre MAS-IDS.
- **Chapitre IV** : Ce chapitre résume les différents éléments et fonctionnalités développés de notre système notamment le système multi-agent, les algorithmes d'apprentissage et l'application mobile de supervision du réseau. Nous résumons également les résultats d'analyse des intrusions à travers le dataset NSL-KDD.

CHAPITRE I : SYSTEMES DE DETECTION D'INTRUSIONS

1. Introduction

Dans ce premier chapitre, nous allons définir un système de détection d'intrusion (IDS) ainsi que les types d'IDS. Nous présenterons par la suite les attaques considérées par les IDS. Cette partie de la recherche vise à répondre à cette question « quels genre d'actions ou d'activités sont considérées comme intrusions par un IDS ? ». Ensuite, nous procéderons à une brève analyse des principales approches des techniques utilisées pour détecter les intrusions. Nous examinerons par la suite les techniques d'implémentation des IDS, ainsi que les dataset utilisés par les IDS.

2. Systèmes de détection d'intrusions

L'intrusion est une activité qui a pour but d'endommager un système d'information, de menacer l'intégrité et/ou la confidentialité des informations, et/ou de les rendre inaccessibles aux utilisateurs légitimes. De ce point de vue, on peut définir l'attaque comme toute tentative d'accès non autorisé à un ordinateur, un système informatique ou un réseau informatique, qui peut causer des dommages, manipuler, voler ou supprimer des données [1].

Pour aider les systèmes informatiques à se préparer et à faire face à ce genre d'illégales actions, les développeurs ont utilisé un système de détection d'intrusions. Un système de détection d'intrusion permet de surveiller le trafic réseau afin de détecter des anomalies. Dans ce contexte, on peut dire qu'un IDS est un système dont son but est de maintenir la sécurité et l'intégrité du système et d'identifier les mesures qui menacent l'intégrité du système et qui ne sont pas spécifiées par un simple pare-feu [6] [14].

Les systèmes IDS peuvent être globalement classés selon le type de données analysées (données de locales de l'hôte ou celle du réseau) et l'approche de détection d'intrusion utilisée (par signature ou par anomalie).

2.1 HIDS (Host Based Intrusion Detection)

Les systèmes de détection d'intrusion à base de l'hôte sont développés pour vérifier les événements, et gérer les données, via l'analyse du système informatique, telles que la configuration du système pour une seule hôte, l'activité de l'interface réseau, journaux système ou journal d'audit, le processus d'application et l'accès et la modification des fichiers [2]. Ceci est mis en œuvre, en installant des programmes (ou agents) sur le système pour générer des rapports indiquant des éventuelles activités malveillantes.

Comme nous l'avons mentionné précédemment, HIDS analyse les fichiers journaux. Ceci lui permet de déterminer si l'attaque a réussi ou non, donc il est plus précis et comporte moins d'erreurs que les identifiants basés sur le réseau. Mais cela ne veut pas dire qu'il n'a pas de point faible, son principal problème est l'intensité d'utilisation des ressources.

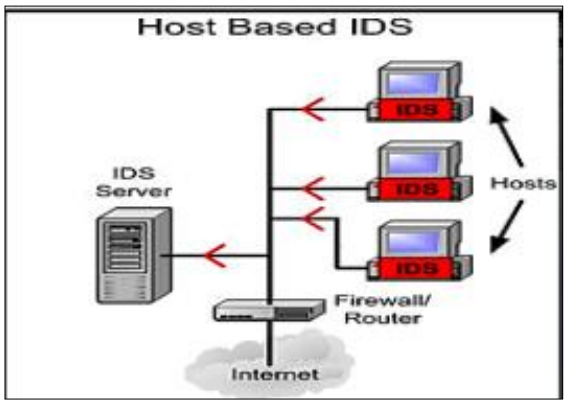


Figure 1 IDS basé sur l'hôte [3]

2.2 NIDS (Network Intrusion Detection System)

Les systèmes de détection d'intrusion à base de réseau (ou NIDS) sont utilisés pour détecter les activités suspectes en analysant le trafic réseau. NIDS utilise l'analyse au niveau des paquets d'un segment de réseau en examinant l'IP, les activités des applications et les en-têtes de paquets. Le NIDS permet ainsi de détecter différentes attaques telles que DOS, TCP SYN, etc. L'avantage de NIDS est qu'il est moins coûteux que le HIDS en termes de ressources utilisées, et réagit plus rapidement que le HIDS [4].

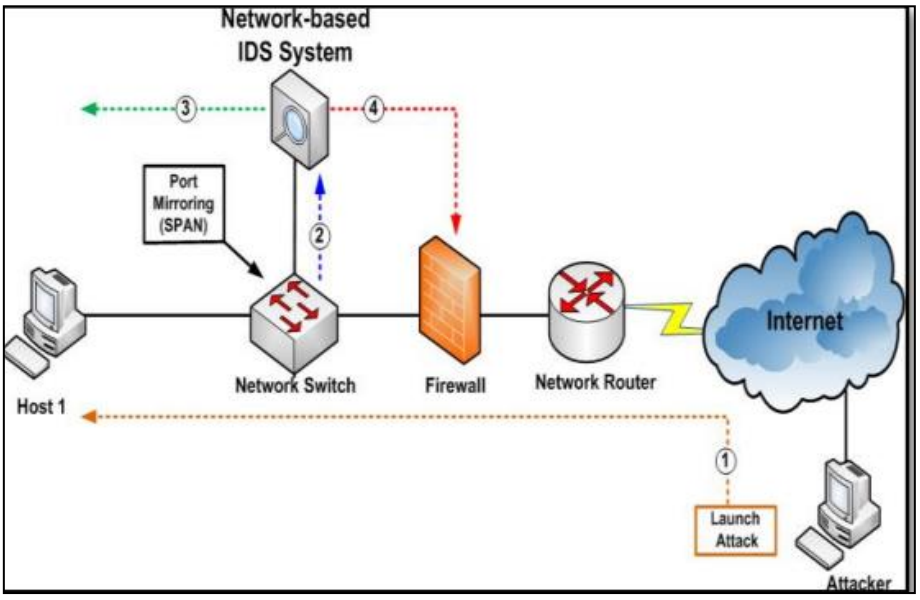


Figure 2 NIDS (IDS basé sur le réseau) [4]

2.3 IDS centralisés et IDS distribués

Les IDS sont généralement conçus selon deux types d'architectures : l'architecture centralisée et l'architecture distribuée. Dans une architecture centralisée, l'analyse des intrusions se fait par une seule entité centrale. Elle permet de surveiller l'hôte et/ou le réseau et détecter les anomalies en se basant sur la performance de cette entité centrale [16].

L'inconvénient de l'architecture centralisée pour un IDS est que si vous trouvez un point de défaillance unique et que vous le désactiver, la connexion ne peut pas se produire. De plus, l'IDS peut devenir une cible d'attaques ou non performante en cas de pannes. Une approche distribuée est donc proposée pour

améliorer la précision et la performance des IDS centralisés. Un système de détection d'intrusion distribué se compose de plusieurs entités connectées à un serveur central permettant la surveillance du réseau [6]. Un IDS distribué à la capacité de recueillir des informations provenant de diverses sources pour détecter les attaques contre un système de réseau. Vu qu'il est conçu pour fonctionner dans un environnement hétérogène, c'est ce qui rend cet IDS capable de détecter plusieurs types d'attaques.

L'approche distribuée présente plusieurs avantages par rapport à une approche centralisée, nous allons les résumer dans le tableau suivant [7] :

Tableau 1 Avantages et inconvénients des IDS distribués et centralisés [7]

IDS	**Avantages**	**Inconvénients**
IDS distribué	- Flexibilité et scalabilité. - Détecte les attaques DOS pour les réseaux à haut débit. - Réduire les coûts de calcul. - Suivi, l'analyse et le traitement des données d'attaque est plus facile et plus rapide. - Rend possible la détection d'une intrusion qui peut bloquer le trafic entrant dans l'ensemble du réseau à partir des adresses IP.	- Le flux de données entre l'hôte et l'agent IDS peut produire des surcharges de trafic réseau. - Les données dont le chemin est long de sa source à l'IDS peuvent être interceptées ou modifiées, ce qui peut donner lieu à des interprétations erronées.
IDS centralisé	- Toutes les activités de l'IDS sont contrôlées directement par une entité centrale. - Les coûts de maintenance et d'administration sont plus faibles par rapport au cas d'un système distribué.	- Impossible de détecter les événements malveillants survenant à différents endroits au même moment. - Un pirate peut neutraliser les programmes fonctionnant sur un système, rendant l'IDS inutilisable ou peu fiable.

2.4 Principales attaques considérées par les IDS

Avant de citer les attaques on doit expliquer les types d'attaques :

- **Confidentialité** : Dans ce type d'attaques, l'attaquant accède à des données confidentielles et autrement inaccessibles.
- **Intégrité** : Dans de telles attaques, l'attaquant peut altérer l'état du système et corrompre les données sans l'autorisation de l'utilisateur légitime.
- **Disponibilité** : Le meilleur exemple est une attaque par déni de service, dans laquelle l'attaquant arrête le système ou le rend indisponible pour tous les utilisateurs.
- **Contrôle**: Dans ces types d'attaques, l'attaquant peut modifier les privilèges d'accès du système car il en acquiert le contrôle total.

Nous pouvons citer les principales trois attaques ci-dessus [21, 22, 25] :

2.4.1 Les attaques DOS (déni de service, Denial of Service)

Les attaques DoS représentent un réel problème car les systèmes de sécurité réseau sont incapables de les détecter vu qu'il existe une grande similitude entre le trafic légitime et le trafic offensif. De plus, les attaquants exploitent les vulnérabilités des protocoles réseau (les protocoles TCP/IP) pour empêcher les appareils de fonctionner, restreindre les ressources du réseau et empêcher leur utilisation par les utilisateurs.

2.4.2 Les attaques DDOS (Distributed DOS)

L'attaque DDoS ne repose pas sur un protocole de réseau particulier ou une faiblesse du système. Elle exploite simplement l'énorme capacité limitée de traitement des requêtes par une machine ou un réseau [8]. L'attaque peut venir de plusieurs attaquants aussi suite à la nature ouverte du réseau Internet pour découvrir n'importe quelle machine. L'attaquant prépare ainsi progressivement un réseau d'attaque appelé botnet. Selon le code d'attaque, les machines compromises sont appelées Masters/Handler ou zombies. Les pirates envoient des instructions de contrôle aux maîtres, qui à leur tour contrôlent les zombies. Le site zombies sous le contrôle des maîtres/manipulateurs transmet des paquets d'attaque comme le montre la Figure 3, qui convergent vers la victime pour épuiser ses ressources.

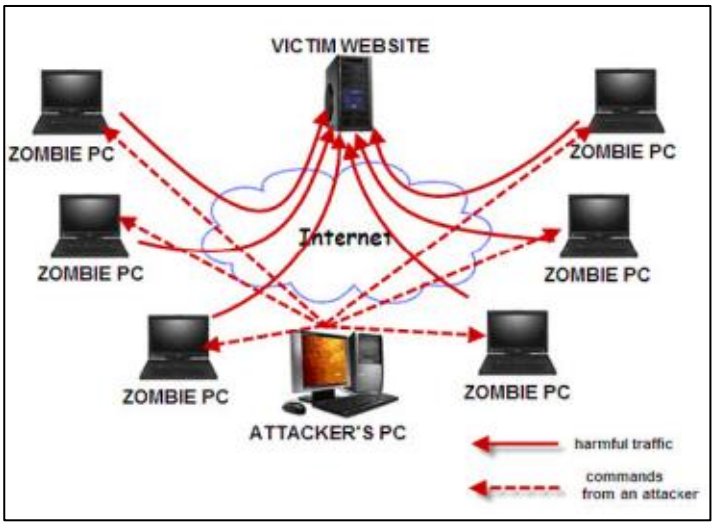


Figure 3 Architecture des attaques DDoS[10]

L'attaque DDoS cible essentiellement les ressources de calcul ou de communication de la victime, comme la bande passante, le débit et la vitesse de transmission. L'attaque DDoS peut être une attaque par inondation ou une attaque par vulnérabilité. L'attaque par inondation consomme les ressources de la victime en inondant un grand volume de paquets. Les attaques par vulnérabilité utilisent le comportement attendu des protocoles tels que TCP et HTTP à l'avantage de l'attaquant. Dans l'attaque par inondation, les attaquants encombrent le lien entre le routeur de périphérie du FAI et le routeur d'accès de la victime en inondant des paquets vers la victime. Cela a pour conséquence, priver les clients légitimes du service en raison du nombre limité d'ordinateurs.

Lorsque le taux total des demandes est supérieur au taux de service total, les demandes commencent à être mises en mémoire-tampon sur le serveur de la victime. Au fil du temps, les demandes entrantes sont abandonnées en raison du dépassement de la mémoire-tampon. Les signaux de contrôle de congestion et de contrôle de flux, forcent les clients légitimes à diminuer leur taux d'envoi de paquets, cependant, les paquets d'attaque continuent à arriver au taux de distribution spécifié par l'attaquant. Par conséquent, il

arrive un stade où la totalité de la bande passante du goulot d'étranglement est occupée par les paquets d'attaque.

Les attaques U2R (User-to-Root)

C'est un processus par lequel un utilisateur normal obtient les privilèges de root (super utilisateur) par une transition illégale défini comme une attaque de type user To root (U2 R). Pour exécuter une attaque U2R, l'intrus doit d'abord, d'une manière ou d'une autre avoir un accès local (en tant qu'utilisateur normal) à la machine de la victime par le biais de l'une des méthodes suivantes : renfilage, attaque par dictionnaire, ou ingénierie sociale, etc.

L'intrus exploite ensuite une vulnérabilité ou un bogue associé à l'environnement du système d'exploitation de la machine attaquée pour effectuer le passage du niveau utilisateur à la racine. Après avoir acquis les privilèges root, l'intrus a un contrôle total sur la machine de la victime pour installer des entrées de porte dérobée en vue d'une exploitation future, modifier des systèmes pour collecter des informations, d'autres actions potentiellement très préjudiciables. Un seul processus permet de passer légalement une transition légale du niveau utilisateur au niveau racine dans l'environnement solaristm. Il s'agit de l'utilitaire "su". Si un utilisateur obtient le Shell root sans exécuter la commande « sue », cela signifie qu'une transition U2R intrusive a eu lieu. Comme les dommages causés par les attaques U2R peuvent être potentiellement très destructeurs, il est très important de détecter à 100% les attaques de cette catégorie.

3. Approches de détection d'intrusion

Il existe de nombreuses recherches et études sur les systèmes de détection d'intrusion, mais il n'existe toujours pas de classification acceptée à l'échelle mondiale. Aujourd'hui, on trouve généralement deux types de systèmes de détection d'intrusion : Système de détection des intrusions basé sur les signatures (SIDS). Dans cette approche, si la signature de trafic suspect correspond à une signature spécifique stockée dans la base de données, une intrusion est signalée. La deuxième approche est le système de détection des intrusions basé sur les anomalies (AIDS). Il permet de déterminer le comportement normal du système, et tout écart par rapport à celui-ci est considéré comme une intrusion [13].

3.1 Approche basée sur les signatures (SIDS)

Cette approche vise à identifier les événements qui violent la sécurité du réseau. Ce type de détection utilise la méthode de comparaison pour trouver une attaque connue. Cela signifie que l'alarme est déclenchée si la signature d'une activité suspecte correspond à une signature d'intrusion précédente qui existe dans la base de données des signatures.

Une signature est une chaîne de caractères qui apparaît dans les paquets d'attaque. Une base de données de l'IDS doit stocker dans ce cas des milliers de signatures, cette densité ralentit le débit et rend l'IDS vulnérable aux attaques par déni de service (DoS). Cette vulnérabilité n'est pas la seule faiblesse de cette approche, mais le principal problème c'est l'incapacité à détecter des attaques inconnues ou celle qui ne laissent pas de signatures telles que les attaques Zéro-Day [1].

3.2 Approche basée sur la détection des anomalies (AIDS)

Cette méthode tente de détecter les intrusions en analysant les différences significatives entre un comportement normal et un comportement suspect en le considérant comme une intrusion [14].Les dispositifs de sécurité du système définissent le comportement normal d'un système en créant des profils utilisateur. Après avoir créé les fichiers, les détecteurs surveillent les données des nouveaux comportements, comparent les données avec le profil et tentent de détecter les écarts. Parmi les anomalies

détectées par les systèmes de découverte d'identité, on cite l'utilisation abusive des protocoles réseaux.Le principal avantage d'un IDS à base de détection d'anomalie est sa capacité à détecter les événements d'intrusion qui ne laissent aucune trace, comme les attaques Zéro-Day. Parce qu'il n'est pas basé sur une base de données de signatures, mais compare plutôt l'activité actuelle avec le comportement défini.

Malgré ces avantages, cela ne signifie pas qu'il s'agit d'une méthode idéale. L'une des faiblesses majeures du système de détection d'intrusion basée sur les anomalies est le taux élevé de fausses alarmes, car tout écart par rapport au comportement normal est considéré comme une intrusion. Le deuxième problème est qu'il ne peut pas classer l'activité détectée en déterminant le niveau de gravité et les conséquences de l'activité détectée. En fin, on peut trouver aussi des IDS hybrides qui combinent à la fois l'analyse par signature et par anomalie. Cette approche a été utilisée pour exploiter tous les avantages des deux approches mais l'analyse des données dans ce cas peut prendre beaucoup de temps ce qui dégrade la performance du IDS [4].

4. Techniques d'implémentation de l'analyse des intrusions

La figure 4 donne un aperçu des techniques proposées ces dernières années pour améliorer la précision de la détection des intrusions et de réduire les fausses alarmes [1].

4.1 Techniques basées sur les statistiques

Le but de ces techniques est de déterminer toute différence entre un comportement spécifique et le comportement actuel en utilisant des calculs statistiques sur : la bande passante, les protocoles, les ports et les périphériques connectés. Plusieurs techniques sont utilisées, notamment:

- **Uni varié** : "Uni" signifie "un", donc cela signifie que les données ne contiennent qu'une seule variable. La détection unie variée recherche les écarts dans chaque mesure individuelle décrivant le comportement. Cette technique est utilisée lorsqu'un profil statistique normal est créé pour une seule mesure de comportements dans les systèmes informatiques.
- **Multi varié** : dans ce type, les statistiques concernent plusieurs mesures ou variables d'un profil d'activités normales. Elles permettent d'analyser les relations entre les différentes variables. Cependant, le principal défi pour les statistiques multi variés est qu'il est difficile d'estimer les distributions pour les données de grandes dimensions.
- **Modèle de série temporelle** : une série temporelle, est une suite d'observations faites au cours d'une période de temps spécifique. La nouvelle observation est classée comme anormale si la probabilité d'occurrence à ce moment est très faible. La classification est également basée sur l'examen de la variance trouvée dans les données de la série temporelle.

4.2 Techniques basées sur la connaissance

Aussi appelée système expert. La base de connaissances doit être créée pour refléter le profil du trafic légitime. Les informations contenues dans la base de connaissances contiennent des représentations symboliques de règles de jugement d'expert dans un format qui permet au moteur d'inférence de déduire. Le principal avantage des techniques basées sur la connaissance est la capacité de réduire les fausses alarmes positives car le système possède suffisamment de connaissance sur les comportements normaux.

Cependant, un environnement informatique est en évolution dynamique, ce qui nécessite une mise à jour régulière des connaissances pour un comportement normal attendu. Cette technique est aussi coûteuse en termes de temps de traitement, car il est très difficile de collecter toutes les informations et connaissances sur l'ensemble des utilisateurs et du réseau.

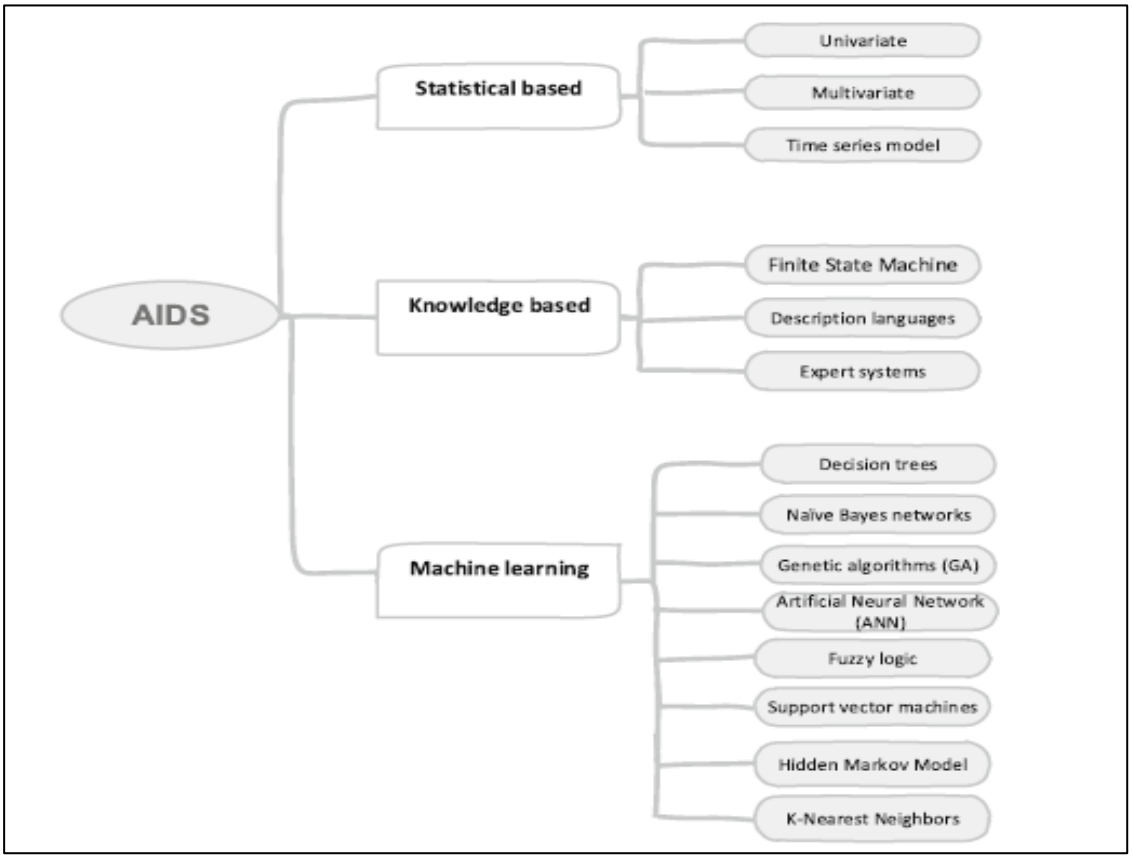


Figure 4 Classification des méthodes d'implémentation des IDS [1]

4.3 IDS basé sur l'apprentissage automatique

L'apprentissage automatique est un sous-domaine de l'informatique. Il extrait des informations à partir de grandes quantités de connaissances, et étudie des algorithmes capables de faire des prédictions basées sur des données, plutôt que de suivre les instructions du micro logiciel. L'objectif de l'utilisation de l'apprentissage automatique est de créer des identifiants plus précis, et nécessitant moins de connaissances humaines. En outre, il gère les événements d'attaque, tels que Denial of Service (DOS), Probe, U2R et R2.

Au cours des dernières années, le nombre d'AIDS qui ont utilisé des méthodes d'apprentissage automatique a augmenté [15]. L'un des principaux objectifs de l'IDS basé sur la recherche en apprentissage automatique est de découvrir des modèles, et de construire un système de détection d'intrusion basé sur l'ensemble de données.

4.3.1 Apprentissage non supervisé

L'apprentissage non supervisé est une forme d'apprentissage automatique qui se basent sur un ensemble de données non étiquetées, et assignent les éléments à certaines classes. Les données non classifiées sont étiquetés comme une intrusion car les occurrences normales devraient produire des grappes plus importantes que les anomalies.

4.3.2 Apprentissage supervisé

Les techniques IDS basées sur l'apprentissage supervisé détectent les intrusions en utilisant des données d'apprentissage étiquetées. Une approche d'apprentissage supervisée se compose généralement de deux étapes, à savoir l'entraînement et le test. Au cours de la phase d'entraînement, les caractéristiques et les classes pertinentes sont identifiées, puis l'algorithme apprend à partir de ces échantillons de données.

Dans la phase de test, le modèle formé est utilisé pour classer les données inconnues dans la classe intrusion ou normale. Le classificateur résultant devient alors un modèle qui, étant donné un ensemble de valeurs de caractéristiques, prédit la classe à laquelle les données d'entrée pourraient appartenir. Le classificateur doit non seulement traiter les données d'apprentissage, mais aussi identifier avec précision la classe d'enregistrements qu'il n'a la création de modèles de classification avec une capacité de généralisation fiable est une tâche importante de l'algorithme d'apprentissage.

5. Apprentissage supervisé pour la détection des intrusions

Parmi les méthodes d'apprentissage supervisé, nous définissions dans ce qui suit les méthodes que nous allons utiliser dans notre projet.

5.1 Arbre de décision (DecisionTree)

L'algorithme de l'arbre de décision appartient à la famille des algorithmes d'apprentissage supervisé. L'objectif d'utilisation d'un arbre de décision est de créer un modèle qui peut être utilisé pour prédire la classe ou bien la valeur de la variable cible en apprenant des règles de décision simple déduites de données antérieures. Dans les arbres de décision, pour prédire l'étiquette de classe, on commence par la racine de l'arbre. On compare les valeurs de l'attribut racine avec l'attribut de l'enregistrement. En se basant sur la comparaison, on suit la branche correspondante avec sa valeur et on passe au nœud suivant. La décision de faire des scissions stratégiques affecte fortement la précision d'un arbre. Les critères de décision sont différents pour les arbres de classification et de régression.

L'arbre de décision utilise de nombreux algorithmes afin de pouvoir décider entre diviser un nœud en deux ou plusieurs sous-nœuds. La création des sous-nœuds augmente l'homogénéité des sous-nœuds résultants .C'est-à-dire, la pureté d'un nœud peut augmenter par rapport à la cible variable. L'arbre de décision divise les nœuds sur toutes les variables disponibles, puis sélectionne la division qui donne la plupart des sous-nœuds homogènes.

Selon Kajal Rai, M. Syamala Devi et Ajay Guleria [31], l'utilisation de l'algorithme de l'arbre de décision IDS, permet à l'administrateur réseau de décider du trafic entrant (qu'il soit malveillant ou pas), tout ça en générant un modèle qui sépare les données de ce trafic. L'algorithme fournit une pondération uniforme à toutes les valeurs du domaine : il permet de prendre un nombre diminué d'attribut et fournit une précision acceptable en si peu de temps. Ils ont constaté que l'algorithme proposé pour la détection de l'intrusion basée sur les signatures est plus efficace pour trouver des attaques dans le réseau avec un petit nombre de fonctionnalités et qu'il faut moins de temps pour construire le modèle et que son efficacité dépend de la taille de l'ensemble de données et du nombre de fonctionnalités utilisées pour construire l'arbre de décision.

5.2 Support Vector Machine (SVM)

SVM est un algorithme d'apprentissage automatique supervisé qui peut être utilisé pour résoudre des problèmes de classification ou de régression. Il utilise une technique appelée astuce du kernel pour transformer les données, puis sur la base de ces transformations, il trouve une frontière optimale entre les sorties possibles. Pour simplifier les choses, il effectue des transformations de données extrêmement complexes, puis détermine comment séparer les données en fonction des étiquettes ou des sorties définies.

SVM fonctionne en mappant les données à un espace d'objets de haute dimension afin que les points de données puissent être catégorisés, même lorsque les données ne sont pas autrement séparables linéairement. Un séparateur entre les catégories est trouvé, puis les données sont transformées de manière à ce que le séparateur puisse être dessiné en tant qu'hyperplan.

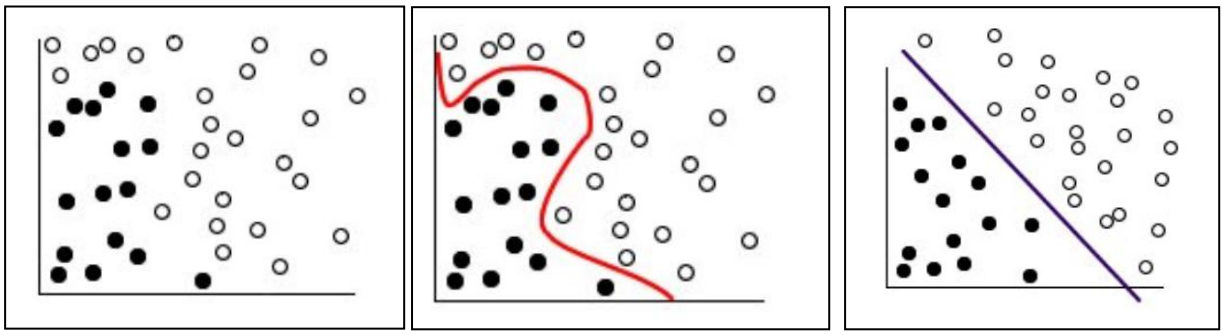


Figure 5 Les étapes du SVM

D'après Jayshree Jha & Leena Ragha [32], SVM est l'une des techniques de détection d'intrusion d'anomalies les plus populaires et les plus utilisées de nos jours. Cela est évidemment grâce à sa bonne nature de généralisation, sa capacité à surmonter la malédiction de la dimensionnalité et également pour sa vitesse qui est considérée comme l'une des principales caractéristiques de la SVM, sans oublier de citer sa capacité en détection des intrusions en temps réel. SVM est utile pour sa capacité de trouver le moins de risque en utilisant une minimisation des risques structurels, car il peut bien se généraliser avec les astuces « noyau », même dans des espaces de haute dimension dans de faibles conditions d'échantillonnage d'entraînement. Il peut également sélectionner des paramètres de configuration appropriés parce qu'il ne dépend pas du risque empirique traditionnel. Les SVM ont la capacité de mettre à jour les modèles dynamiquement à chaque classification d'un nouveau cas.

5.3 Réseaux de neurones (NN)

Les réseaux neuronaux, également connus sous le nom de réseaux neuronaux artificiels sont placés au cœur des algorithmes d'apprentissage profond. Leur nom et leur structure sont inspirés du cerveau humain, imitant la façon dont les neurones biologiques se signalent les uns aux autres.

Les réseaux de neurones sont composés d'une couche de nœud, contenant une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque nœud, ou neurone artificiel, se connecte à un autre et a un poids et un seuil associés. Si la sortie d'un nœud individuel est supérieure à la valeur seuil spécifiée, ce nœud est activé, envoyant des données à la couche suivante du réseau. Sinon, aucune donnée n'est transmise à la couche suivante du réseau.

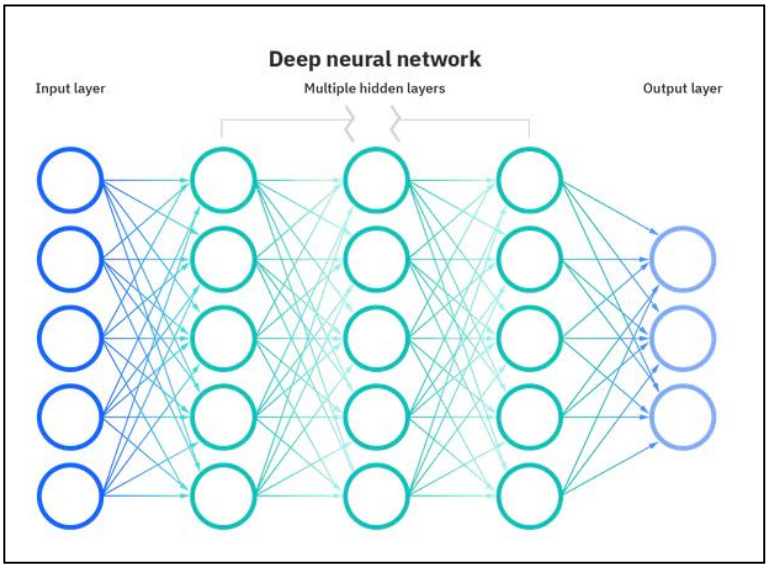


Figure 6 Schéma des réseaux neuronaux

Considérez chaque nœud individuel comme son propre modèle de régression linéaire, composé de données d'entrée, de pondérations, d'un biais (ou d'un seuil) et d'une sortie, une fois que la couche d'entrée est déterminée, des poids sont attribués. Ces pondérations aident à déterminer l'importance de chaque variable donnée, les plus grandes contribuent plus d'efficacité à la production par rapport aux autres intrants. Toutes les entrées sont ensuite multipliées par leurs poids respectifs, puis additionnées. Ensuite, la sortie passe par une fonction d'activation, qui détermine sa sortie. Si cette sortie dépasse un seuil donné, elle "allume" (ou active) le nœud, en passant des données à la couche suivante du réseau. Donc, la sortie d'un nœud devient l'entrée du nœud suivant. Ce processus de transmission de données d'une couche à la couche suivante définit ce réseau neuronal comme un réseau d'avance.

Les réseaux de neurones ont été largement utilisés pour la détection d'intrusion. Un IDS à base de NN améliore les performances des méthodes de détection basées sur les signatures [33]. Il permet aussi de réduire le taux de faux positifs dans l'application des systèmes d'intrusion réseau dans le monde réel, car des niveaux élevés de faux positifs entraînent un rapport signal/bruit extrêmement faible et rendent souvent le système bon à rien.

6. Dataset utilisés pour valider les techniques de détection d'intrusions

Le plus grand défi pour valider les techniques de détection d'intrusion est de disposer d'un ensemble de données approprié, Par exemple, un bon nombre de base de données ne peuvent pas être partagés en raison de problèmes de confidentialité [14]. Il existe quelques ensembles de données accessibles au public tel que DARPA, KDD, NSL-KDD et ADFA-LD, et ils sont largement utilisés dans les travaux de recherche. Nous décrivons dans ce qui suit les principales bases de données utilisées [16] [1]:

6.1 CAIDA (Center of Applied Internet Data Analysis 2002-2016)

Cette organisation dispose de trois ensembles de données différents, le CAIDA OC48, CAIDA DDOS, CAIDA 2016, La plupart des ensembles de données CAIDA sont très spécifiques à des événements ou des attaques particuliers et sont antonymies avec leur charge utile, leurs informations de protocole et leur destination. Un inconvénient de l'ensemble de données CAIDA est qu'il ne contient pas une diversité d'attaques. De plus, les données recueillies ne contiennent pas de caractéristiques de l'ensemble du réseau, ce qui rend difficile la distinction entre les flux de trafic anormaux et normaux.

6.2 ISCX 2012

Cet ensemble de données à deux profils, le profil Alpha qui mis en œuvre divers scénarios d'attaque en plusieurs étapes, et le profil bêta, qui est le générateur de trafic bénin et génère un trafic réseau réaliste avec un bruit de fond. Il inclut le trafic réseau pour HTTP, Protocoles SMTP, SSH, IMAP, POP3 et FTP avec charge utile complète du paquet. Cependant, il ne représente pas nouveaux protocoles réseau depuis près de 70% d'aujourd'hui les trafics réseau sont HTTPS et il n'y a pas de HTTPS traces dans cet ensemble de données. De plus, la répartition des les attaques simulées ne sont pas basées sur des statistiques du monde réel.

6.3 NSL-KDD

Cet ensemble de données est une version mise à jour du DARPA98, en traitant la partie tcpdump. Il contient différentes attaques telles que Neptune-DoS, pod-DoS, SmurfDoS et buffer-overflow.. Le trafic bénin et d'attaque sont fusionnés dans un environnement simulé. Cet ensemble de données a un grand nombre d'enregistrements redondants et est clouté par des corruptions de données qui ont conduit à des résultats de test faussés. NSL-KDD a été créé en utilisant KDD pour répondre à certains dès Les défauts de KDD.

6.4 ADFA-LD

Elle contient des milliers de traces d'appels système, qui sont collectées dans divers scénarios pour imiter les circonstances de vrais scénarios systèmes [15]. Il se compose d'une collection complète de traces d'appels système représentant les vulnérabilités et les attaques récentes au niveau du système. ADFA-LD est généré à partir d'un serveur local Linux avec le noyau Linux 2.6.38, Les attaques utilisées pour générer ADFA-LD incluent Hydra-FTP, Hydra-SSH, Adduser, Java-Meterpreter, Meter-preter et Webshell, chacune d'entre elles générant 8 à 20 traces d'attaque.

7. Conclusion

Dans ce chapitre nous avons introduit le contexte de notre projet qui est les systèmes de détection d'intrusions. Nous avons décrit les types de systèmes de détection d'intrusiond, les méthodes de détection d'intrusiond, les techniques de mise en œuvre et l'ensemble de dataset utilisés. Nous avons également présenté les principales différences entre les systèmes IDS centralisés et distribués. Dans le deuxième chapitre, nous nous intéresserons à l'IDS distribué à travers une architecture basée sur un système multi-agent.

CHAPITRE II : ETAT DE L'ART SUR LA DETECTION DES INTRUSIONS A BASE DE SYSTEMES MULTI-AGENTS

1. Introduction

Ce chapitre permettra au lecteur de faire une connaissance sur les systèmes multi-agents (MAS) ainsi que leur utilisation pour implémenter les systèmes de détection d'intrusions. On va également discuter le résultat de l'analyse des travaux et des limites des MAS actuels pour la détection des intrusions. Le résultat de l'étude de cet état de l'art nous permettra d'introduire la contribution de ce projet de Master.

2. Systèmes multi-agents (MAS)

Un système multi-agent est un environnement informatique qui rassemble plusieurs agents intelligents qui sont en interaction continue. Les MAS sont fréquemment utilisés pour résoudre les problèmes (difficiles presque impossibles) auxquels un agent individuel peut faire face. Comme pour les agents, il n'y a pas de définition catégorique pour le MAS [18]. MAS est défini comme un réseau faiblement couplé d'entités (agents) qui fonctionnent ensemble afin de trouver des solutions aux problèmes qui dépassent la capacité des entités (agents).

2.1 Agent

Les agents, plus précisément les agents intelligents, sont un paradigme pour le développement d'applications logicielles, ils prennent en charge la simulation d'interactions individuelles complexes. De nos jours, les agents font grande partie des domaines d'intérêt en informatique, intelligence artificielle (AI) et en théories des systèmes complexes. Un agent est caractérisé plus particulièrement par son processus : Perception, délibération et action (voir Figure 7). Les agents intelligents (rationnels) sont utilisés dans la majorité des applications, que ce soit en systèmes relativement petits tels que les filtres de courrier électronique, ou bien en grands systèmes complexes comme le contrôle du transport aérien ou les comportements sociaux humains.

2.2 Caractéristiques des agents

A partir de définitions précédentes, nous pouvons identifier des caractéristiques essentielles pour les agents. Ainsi, deux importantes caractéristiques font naître le concept agent [20] : la situation dans un environnement et l'autonomie.

- **La situation** : Toutes les définitions notent qu'un agent doit être situé dans un environnement. Un agent peut donc percevoir son environnement et agir sur les changements de cet environnement. Il est important de noter que cette caractéristique ne soit pas intrinsèque pour tous les systèmes informatiques. A titre d'exemple, un système expert n'a pas d'interaction avec son environnement.

- **L'autonomie** : Cette caractéristique est, peut-être, la caractéristique la plus importante des agents. On désigne par cette caractéristique la capacité de l'agent d'agir sans l'intervention d'un tiers. En d'autres termes, un agent peut atteindre ses objectifs sans demander l'aide d'un autre agent, système ou utilisateur. En plus, un agent possède le contrôle sur son état et son comportement. En effet, on ne peut pas ni modifier l'état interne d'un agent ni imposer sur cet agent l'exécution de certains comportements. Malgré que le concept d'encapsulation (c'est-à-dire la capacité de masquer les attributs) a été introduit par la programmation orientée objet, l'autonomie des agents posent des nouvelles restrictions.

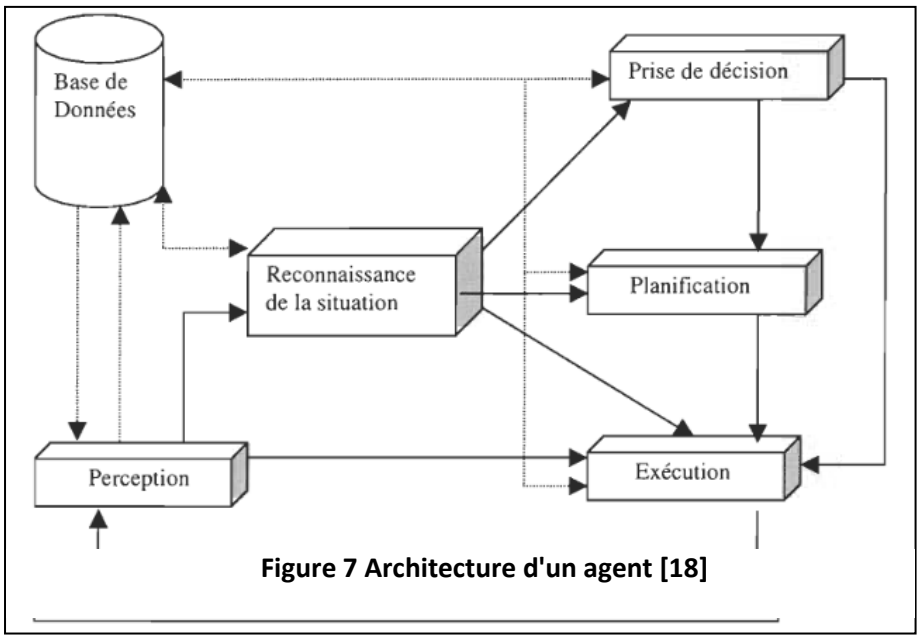


Figure 7 Architecture d'un agent [18]

De plus, l'intelligence des agents est une réflexion de la flexibilité qui caractérise les comportements de ce dernier. En effet, un agent intelligent est un agent qui assure la flexibilité de ses comportements. On entend par la flexibilité, la capacité de l'agent de changer son comportement ou sa structure afin d'atteindre ses objectifs. Afin d'assurer cette flexibilité, l'agent doit être doté au moins par trois caractéristiques [56] :

- **Réactivité :** la capacité de l'agent de percevoir son environnement et répondre aux différents changements apparus dans le temps adéquat.
- **Pro-Activité** : la capacité de l'agent de prendre l'initiative en démontrant des comportements orientés objectifs.
- **Compétences sociales** : la capacité de l'agent d'interagir avec les autres agents en utilisant un langage de communication.
- **La mobilité** : capacité de l'agent de se déplacer dans son environnement (l'environnement peut-être un réseau informatique).
- **L'adaptabilité** : capacité de l'agent de changer son comportement, sa structure ou ses buts selon la situation actuelle.
- **La rationalité** : capacité de l'agent d'atteindre ses objectifs avec le moindre coût. Un agent rationnel n'exécute pas des comportements contradictoires avec ses buts.
- **L'intentionnalité** : l'agent possède une représentation explicite de ses états mentaux.
- **La véracité** : l'agent ne communique pas des fausses informations.

2.3 Avantages des MAS

Les systèmes multi-agents offrent beaucoup d'avantages dans le développement des systèmes complexes, on peut en citer [56] :

- **La modularité** : un système multi-agents est composé d'un ensemble d'entités (les a-agents). Chaque agent est conçu comme une entité autonome indépendante des autres.
- **La réutilisabilité** : la réutilisabilité est une conséquence directe de la modularité des MAS. La modularité permet de réutiliser certains composants d'un programme dans un autre programme. Dans le cas des systèmes multi-agents, la réutilisabilité peut être appliquée à plusieurs niveaux. Naturellement, l'agent est conçu pour être une brique qu'on peut réutiliser autant que possible.
- **La facilité de maintenance** : le développement modulaire des MAS simplifie la maintenance de ces systèmes. On distingue dans la maintenance des logiciels plusieurs types comme : la maintenance perfective, la maintenance corrective et la maintenance évolutive.
- **La Scalabilité** : Les systèmes multi-agents peuvent contribuer à la scalabilité des logiciels grâce à l'autonomie des agents. En fait, un agent est une entité autonome dans le sens où elle peut atteindre ses objectifs sans l'intervention des autres entités. Un agent peut atteindre ses objectifs même si un autre agent est tombé en panne.
- **L'efficacité** : le paradigme agent offre la possibilité de développer des solutions efficaces. L'efficacité des logiciels est mesurée en fonction de ressources consommées afin de résoudre un problème. Les systèmes multi-agents représentent un modèle d'exécution distribué où les ressources sont distribuées. Ce partage permet une utilisation rationnelle des ressources.
- **L'adaptation** : le paradigme agent permet une modélisation fidèle de phénomènes réels. Les caractéristiques des agents, comme l'autonomie, la flexibilité et la sociabilité, représentent des caractéristiques intrinsèques de plusieurs systèmes réels (comme les sociétés humaines, les systèmes biologiques, les colonies d'insectes, etc.). En conséquence, l'adoption de ce paradigme permet de représenter les caractéristiques essentielles des systèmes cités auparavant.
- **Les modes d'interaction sophistiqués** : les systèmes multi-agents supportent des modes d'interaction sophistiqués comme la négociation, la coopération et la collaboration. Par contre, le paradigme objet est basé sur des interactions simples permettant seulement l'invocation des méthodes. La complexité des interactions permet la modélisation et le développement des systèmes complexes où l'interaction est une caractéristique fondamentale.

2.4 Quelques plateformes MAS existantes

En général, la mise en œuvre d'un système multi-agent exige deux langages, un langage de programmation d'agent qui sert à implémenter les agents individuels, et un autre langage assurant la communication et la coordination entre les agents. Nous présentons dans ce qui suit quelques plateformes multi-agents connues dans la littérature.

2.4.1 3APL

Il s'agit d'un langage de programmation multi-agents et de sa plateforme de développement correspondante. Il est proposé par l'Université d'Utrecht [30]. 3APL est motivé par les architectures d'agents cognitifs et fournit des constructions de programmation pour mettre en œuvre des agents individuels directement en termes d'objectifs et de plans.

2.4.1.1 Architecture de 3APL

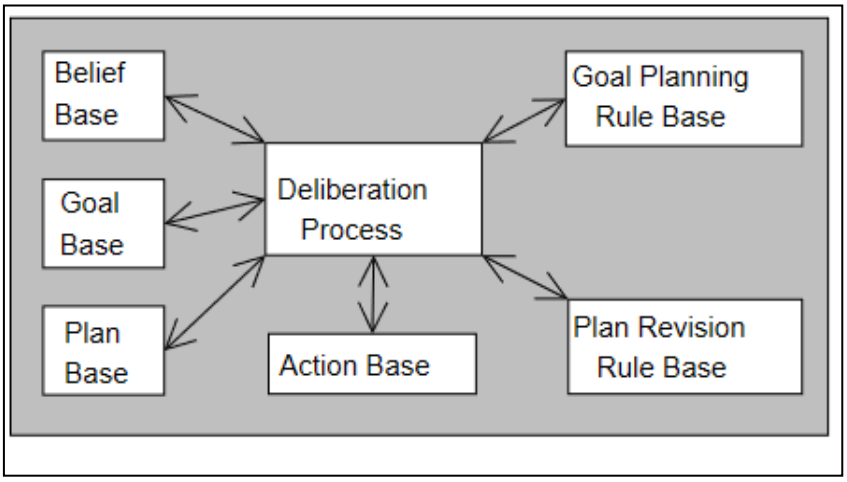


Figure 8 Agent 3APL [21]

Bien que l'architecture 3APL présente pas mal de similitudes avec d'autres architectures cognitives telles que PRS (Procedural reasoning system) proposée pour les agents BDI (voir Figure 8), ils diffèrent les unes des autres à bien des égards. Par exemple, l'architecture PRS est conçue pour planifier les objectifs des agents tandis que l'architecture 3APL est conçue pour contrôler et réviser le programme contrôle des robots.

2.4.1.2 Caractéristiques

Le langage de programmation 3APL ainsi que sa plate-forme permettent l'intégration de Prolog & Java. Les programmes Prolog peuvent être intégrés car ils peuvent être chargés dans 3APL et utilisés comme une connaissance de base. Compte tenu d'un programme Prolog chargé, l'agent peut poser des requêtes dans trois contextes différents : comme condition préalable aux actions mentales, actions d'essai dans les plans et ainsi que la garde des règles de raisonnement [21].

2.4.2 JADE

JADE (Java Agent Development framework) : est un framework conçu pour aider au développement d'applications d'agents conformes aux spécifications FIPA [24] pour les systèmes multi-agents intelligents interopérables. L'objectif de JADE est de simplifier le développement tout en assurant la conformité standard grâce à un ensemble complet de services et d'agents système. JADE est un projet open source actif, le cadre ainsi que la documentation et les exemples peuvent être téléchargés à partir de la page d'accueil de JADE. JADE est entièrement développée en Java et se réfère sur les principes de conduite suivants :

- **Interopérabilité** : JADE est conforme aux spécifications FIPA. En conséquence, les agents JADE peuvent interagir avec d'autres agents, à une seule condition : respecter la même norme.
- **Uniformité et portabilité** : JADE fournit un ensemble homogène d'API qui sont indépendantes du réseau sous-jacent de la version Java (édition, configuration et profil). Plus en détail, l'environnement d'exécution JADE fournit les mêmes API à la fois pour les environnements J2EE, J2SE et J2ME. En théorie, les développeurs d'application peuvent décider de l'environnement d'exécution Java au moment du déploiement.

- **Facilité d'utilisation** : la complexité de l'intergiciel est planquée derrière un ensemble simple et intuitif d'API.

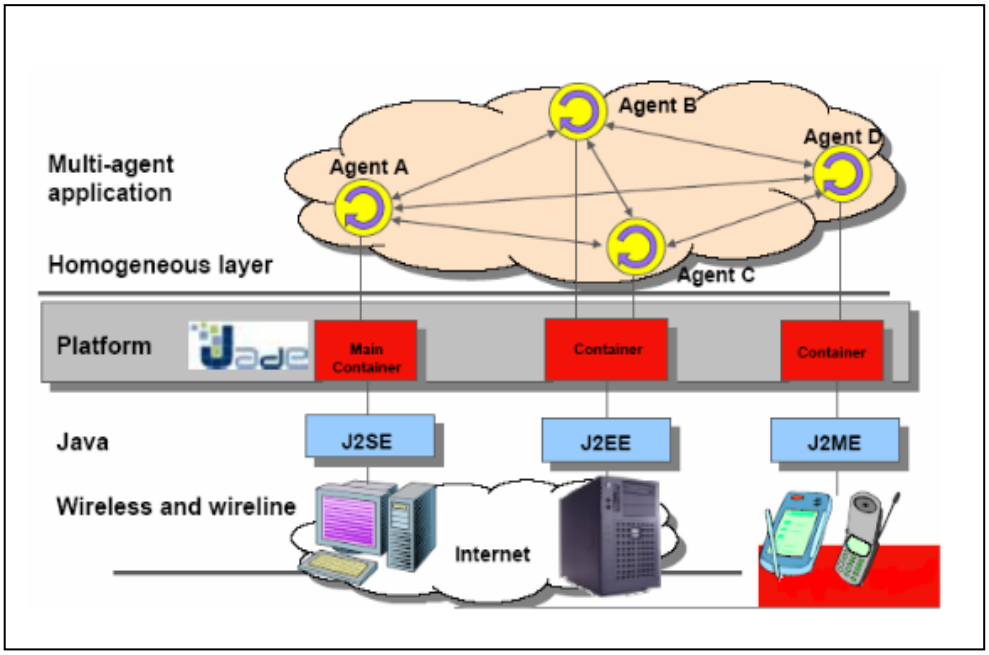


Figure 9 Architecture de JADE [23]

2.4.2.1 Architecture

JADE inclus (voir Figure 9 et 10):

- Des bibliothèques pour développer les agents spécifiques à l'application.
- La mise en œuvre des deux agents de gestion qu'une plate-forme d'agent conforme à l'APIE doit fournir, c'est-à-dire AMS (Agent Management System) et l'agent DF (Directory Facilitator).
- L'environnement d'exécution qui fournit les services de base. Ce dernier doit être activé sur l'appareil avant que les agents puissent être exécutés. Chaque instance de l'environnement d'exécution JADE est nommée conteneur (puisqu'elle "contient" des agents).
- L'ensemble de tous les conteneurs s'appelle plate-forme et fournit une couche homogène qui cache aux agents (et aux développeurs d'application) la complexité et la diversité des technologies sous-jacentes (matériel, systèmes d'exploitation, type de réseau, J VM etcL'architecture logicielle (voir Figure 10) est basée sur la coexistence de plusieurs machines virtuelles Java (VM) et la communication repose sur Java RMI (Remote Method Invocation) entre différentes machines virtuelles et la signalisation d'événements au sein d'une seule machine virtuelle. Chaque machine virtuelle est un conteneur de base d'agents [24].

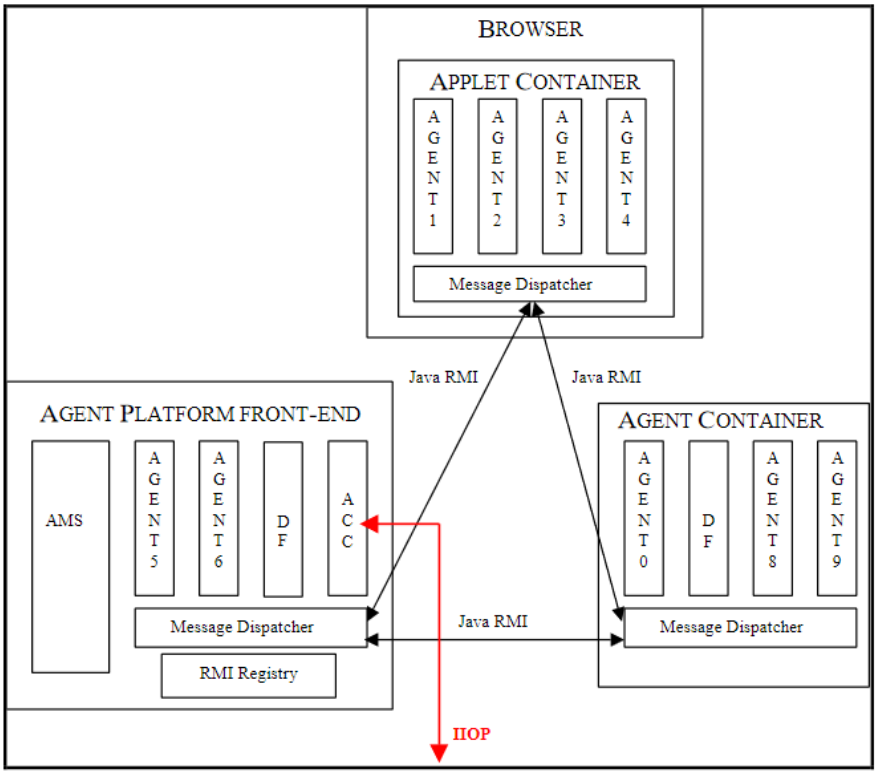


Figure 10 Architecture d'un agent JADE [24]

2.4.2.1 Caractéristiques

JADE est écrite en langage Java offrant aux développeurs d'application des fonctionnalités prête à l'emploi ainsi que des interfaces abstraites pour des tâches personnalisées et dépendantes de l'application. Elle permet de construire des systèmes d'agents conformes aux spécifications FIPA [56]. Elle permet aussi l'interopérabilité avec les agents agissant sur différentes plateformes d'agents. JADE essaie de fournir à ses utilisateurs des technologies d'agent standard tout en réduisant les frais généraux d'exécution. JADE essaie également d'être efficace en ce qui concerne la consommation de ressources : encore une fois, les comportements coopératifs aident à réduire le nombre global de threads. En outre, de nombreux objets JADE peuvent être recyclés au lieu d'être détruits et créés, ce qui réduit l'allocation dynamique de la mémoire. Le développement de JADE s'enchaîne toujours. D'autres améliorations et mises en œuvre ont déjà été prévues, y compris le soutien à la mobilité des agents tel que spécifié par la FIPA98.

2.4.3 Jack

Jack Intelligent Agents est une plate-forme d'agent qui comprend des composants [26]. JACK comprend un langage de programmation orienté agent ; une plate-forme pour exécuter des agents avec une infrastructure telle que le marshaling de messages, un serveur de noms et des outils de développement, y compris un outil de conception, un éditeur de plan graphique et un certain nombre de vues de débogage. En outre, JACK comprend un certain nombre de fonctionnalités supplémentaires telles que la possibilité de construire des équipes hiérarchiques d'agents [25].

2.4.3.1 Architecture

Le langage JACK prend en charge le raisonnement pratique de style BDI ainsi que le raisonnement d'inférence dirigé vers l'avant, et permet divers concepts d'agents tels que les attitudes mentales, la délibération, l'adaptation, le comportement réactif et proactif [21].

2.4.3.2 Caractéristiques

L'un des points forts de JACK est sa prise en charge des pratiques modernes de génie logiciel. En plus des fonctionnalités fournies par Java (objets, paquets), JACK ajoute un certain nombre de fonctionnalités qui peuvent être utilisées pour structurer un système d'agents.

Le corps d'un plan peut être divisé en un certain nombre de méthodes de raisonnement distinctes, plutôt que d'être un seul bloc monolithique de code JACK. Cela permet de structurer un seul plan à l'interne. Les capacités peuvent également contenir des sous-capacités qui permettent de spécifier des structures de modules hiérarchiques, le cas échéant. Comme JACK représente un ensemble de Java, l'intégration avec le code de ce dernier est simplifiée. JACK intégré au planificateur JSHOP (écrit en Java). JACK peut également être intégré au code C++ existant en utilisant JACOB.Le langage de l'agent JACK comprend un certain nombre d'autres fonctionnalités importantes. Le plus important est peut-être son support à la programmation « team-oriented » Le compilateur JACK est modulaire, et le langage JACK peut être étendu à l'aide de plugins, mais cet aspect n'est pas actuellement bien documenté, et l'extension du langage JACK de cette manière est difficile sans un soutien étendu de Agent Oriented Software.

2.4.4 Jadex

Jadex est un framework pour la création d'agents axés sur les objectifs suivant le modèle croyance-désir-intention (BDI). Le projet Jadex vise à rendre le développement de systèmes basés sur les agents aussi facile que possible sans sacrifier la puissance expressive du paradigme des agents. L'objectif est de construire une couche d'agent rationnelle qui se trouve au-dessus d'une infrastructure d'agent middleware et permet la construction d'agents intelligents à l'aide de bases solides d'ingénierie logicielle..

2.4.4.1 Architecture

Vu de l'extérieur, un agent est une boîte noire qui reçoit et envoie des messages. Les messages entrants, ainsi que les événements internes et les nouveaux objectifs servent de contribution au mécanisme de réaction et de délibération interne de l'agent. En fonction des résultats du processus de délibération, ces événements sont envoyés aux plans déjà en cours d'exécution ou aux nouveaux plans instanciés à partir de la bibliothèque de plans. Les plans d'exécution peuvent accéder à la base de croyances et la modifier, envoyer des messages à d'autres agents, créer de nouveaux cadres supérieurs ou sous-objectifs et provoquer des événements internes. Le mécanisme de réaction et de délibération est généralement le même pour tous les agents. Le comportement d'un agent spécifique est donc déterminé uniquement par ses croyances, ses objectifs et ses plans concrets.

- **Beliefs** : Jadex n'impose pas une représentation logique des croyances. Il permet de réutiliser les classes générées par les outils de modélisation ontologique ou les couches de mappage de base de données.
- **Goals** : Pour tout objectif qu'il a, un agent s'engage plus ou moins directement dans des actions appropriées, jusqu'à ce qu'il considère que l'objectif est atteint, inaccessible ou plus désiré.
- **Plans** : Le moteur de raisonnement gère tous les événements tels que la réception d'un message ou l'activation d'un objectif en sélectionnant et en exécutant des plans appropriés..

2.4.4.2 Caractéristiques

L'objectif général du Jadex est de fournir un moteur de raisonnement sophistiqué permettant le développement d'agents intelligents complexes arbitraires tout en essayant d'être aussi facilement utilisable que possible, le système ne sacrifie pas l'expressivité pour la simplicité. Néanmoins, les questions de génie logiciel jouent un rôle important dans la conception du système. Comme indiqué précédemment, l'un des principaux objectifs du projet est de faciliter une transition en douceur du développement logiciel vers une approche axée sur l'agent. Ceci est réalisé en recourant à des techniques établies.

3. Taxonomie des systèmes multi-agents pour les IDS

Différents travaux de recherche ont proposé l'utilisation de systèmes multi-agents afin de mettre en œuvre des IDS distribués (IDS-MAS). Les méthodes de distribution sont divisées en trois catégories principales, elles se présentent en fonction de : l'architecture, la technologie de l'agent utilisé et enfin, a technique de prise de décision pour la détection de l'intrusion. Figure 11 illustre la taxonomie des MAS-IDS existants [27].

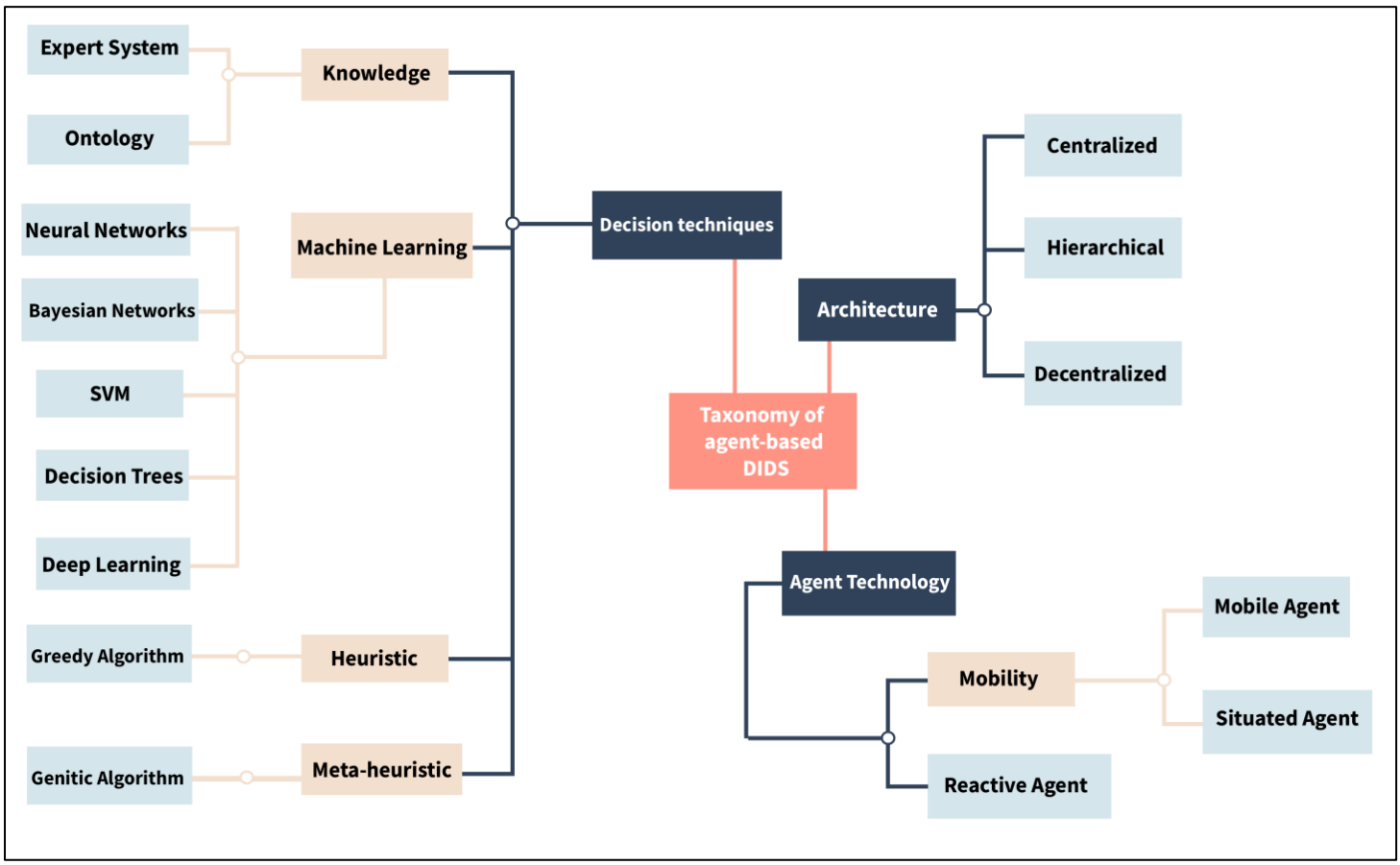


Figure 11 Taxonomie des IDS à base de MAS [27]

3.1 Architecture

Les systèmes collaboratifs de détection d'intrusion sont généralement conçus selon trois types d'architecture : centralisée, hiérarchique et distribuée.

- **Centralisé** : Les données de sécurité sont d'abord collectées dans l'entité centrale (un seul agent), puis analysées. La capacité de la bande passante et l'ajout de nouveaux appareils reste insuffisante, d'où la nécessité d'intervention humaine.
- **Hiérarchique** : Les IDS deviennent plus évolutifs lorsque les agents sont organisés d'une manière hiérarchique. Selon cette architecture, les agents collaborent et partagent leurs données d'après une structure hiérarchique. Ce modèle offre des avantages en termes d'évolutivité, scalabilité, d'extensibilité et de la tolérance aux pannes. Néanmoins, cette architecture engendre deux inconvénients principaux :
    - Comme l'analyse des données est centralisée aux plus hauts niveaux de la hiérarchie, cela rend le système non entièrement distribué.
    - Les nœuds proches du nœud racine souffrent d'une faible tolérance aux pannes et d'une vulnérabilité de surcharge.
- **Distribué** : Selon de récentes recherches, l'architecture collaborative des agents pour un IDS est l'une des architectures les plus convenables et adaptées aux systèmes complexes où il y a un grand nombre d'entités en interaction [28]. La structure de cette architecture est évolutive et ne dispose d'aucun point de défaillance. Cependant, le fait de comprendre de nombreuses voies d'accès à l'information (pas nécessaires) est le principal inconvénient d'un tel modèle.

3.2 Technologie agent dans IDS-MAS

Les IDS à base de MAS utilisent différents types d'agent, notamment [27, 6]:

- **Agents mobiles** : Les approches utilisant des agents mobiles permettent de concevoir des IDS collaboratifs qualifiés. Un agent mobile est utilisé pour :
    - **Surmonter la latence du réseau :** Les agents mobiles sont utiles pour les applications qui doivent répondre en temps réel aux changements de leur environnement, car ils peuvent être renvoyés à partir d'un contrôleur central afin d'effectuer des opérations directement au point d'intérêt distant.
    - **Réduire la charge du réseau :** Les agents mobiles offrent la possibilité de réduire la charge réseau en éliminant le besoin des données transférées dans les outils de surveillance du trafic réseau et les journaux d'audit basés sur l'hôte.
    - **Exécution asynchrone et anatomie :** L**es** agents mobiles permettent aux IDS de poursuivre leurs opérations en cas de défaillance d'un contrôleur central ou d'une liaison de communication.
    - **Structure et composition :** Les agents mobiles permettent de structurer et de concevoir un IDS.
    - **L'adaptation dynamique :** Les agents mobiles fournissent un paradigme informatique polyvalent et adaptatif car ils peuvent êtreenvoyés, clonés ou endormis à mesure que les conditions du réseau et de l'hôte changent.
    - **Scalabilité :** en créant plusieurs agents selon la taille du réseau.
- **Agents situés :** Dans ce cas un agent, qui joue plusieurs rôles, est situé sur chaque nœud du réseau. L'agent peut effectuer la tâche d'un système local de détection d'intrusion. En outre, l'agent assure l'échange de données de sécurité avec ses voisins en recueillant des informations sur les attaques en utilisant la fréquence des événements locaux des agents voisins. Le système proposé a permis d'améliorer le taux de détection. De plus, le système adopte la négociation ainsi que la propagation de la décision prise pour résoudre le problème des fausses alarmes [27].
- **Agents réactifs** : Un IDS utilisant des agents réactifs réagit rapidement aux problèmes qui ne nécessitent aucun raisonnement complexe. Il existe également des IDS - MAS qui combinent des agents réactifs et mobiles qui coopèrent afin de rendre l'IDS plus efficace et plus sécurisé.

Les résultats des travaux utilisant cette approche indiquent que le système suggéré augmente le taux de détection, réduit les faux positifs et détecte également les attaques connues et inconnues dans l'environnement Cloud computing.

3.3 Techniques de détection d'intrusions dans les IDS à base de MAS

Nous avons donné dans le chapitre I un aperçu sur les techniques d'implémentation des IDS en général. Les techniques basées sur les connaissances d'apprentissage automatique et les métaheuristiques sont les techniques les plus utilisées dans les IDS à base de MAS. Voici les principales techniques qu'on trouve dans la littérature [27] :

- **Connaissance (Knowledge-based)** : Elle est utilisée dans une conceptualisation, explicite ou bien implicite du IDS. La représentation des connaissances est basée sur une conceptualisation qui est une composition d'objets, de concepts, d'autres entités et des relations qui existent entre eux. La détection basée sur « knowledge » adapte une sorte de « knowledge gathered » sur les attaques, ce système peut être utilisé pour détecter toute attaque ou vulnérabilité du système. Si le système n'a pas la connaissance d'une attaque particulière, il n'est pas en mesure d'identifier une attaque, ce qui nécessite une connaissance significative de plusieurs attaques.

- **Systèmes experts** : Cette technique contient un ensemble de faits, de règles et de méthodes d'inférence. Chaque événement survenu dans le système est traduit en faits et règles correspondants. Les méthodes d'inférence peuvent générer des conclusions à partir des règles et des faits existants. La structure sémantique attachée à chaque événement augmentera le niveau d'abstraction des données sur les événements. Les systèmes de détection d'intrusion basés sur les signatures et les systèmes de détection d'intrusion basés sur des anomalies peuvent utiliser l'approche du système expert. Les systèmes experts offrent des moyens puissants et flexibles d'obtenir des solutions à une diversité de problèmes qui ne peuvent souvent pas être résolus par des méthodes traditionnelles.

- **Ontologies** : Certains travaux ont proposé l'utilisation des ontologies afin de représenter les connaissances sur les intrusions dans les IDS-MAS. Cette technique permet le partage et la réutilisation des connaissances entre des entités du même domaine.

- **Machine Learning** : Les techniques d'apprentissage automatique ML qui ont été largement utilisées dans la construction des IDS-MAS sont les réseaux de neurones, les réseaux baysiens, support vector machines, arbres de décision et apprentissage profond.

4. Facteurs qui influencent la performance de la détection d'intrusion à base d'un MAS

Parmi les facteurs qui influencent la performance des MAS-IDS sont [27]:

Distributions d'agents et de tâches : Dans les architectures IDS multi-agents, des sous-tâches sont attribuées aux agents en fonction de la structure organisationnelle choisie, cette dernière déterminera la manière dont les agents capteurs seront placés. La distribution de plusieurs capteurs peut augmenter l'évolutivité du système en augmentant le nombre d'agents à collecter et à prétraiter simultanément de grands volumes de données [31]. Pour les agents d'analyse, les architectures IDS-MAS se présentent sous deux différentes formes : des analyseurs individuels et multiples. L'utilisation de plusieurs analyseurs encourage l'équilibrage de charge en divisantla charge de travail entre plusieurs agents d'analyseurs pour une exécution parallèle. La création de plusieurs agents avec différentes techniques d'analyse permet

d'examiner les types sophistiqués d'attaques qui ne peuvent pas être détectés par une seule technique d'analyse.

Collecte et synchronisation des données : Il existe également quelques études qui comprennent de nombreuses tâches telles que la collecte de données, l'agrégation, la synchronisation et la préparation àl'analyse par un ou plusieurs agents d'analyse [31]. L'agrégation des données peut faire face à plusieurs problèmes : le fusionnement des données de différentes sources, le fusionnement et la synchronisation des données par adresse IP (source et destinataire), timestamps et finalement, le type de protocole. Les processus de collecte et de synchronisation des données affectent les performances de détection de différentes manières : La complexité des méthodes d'agrégation peut dégrader les performances de détection en augmentant les coûts du traitement. Les méthodes d'agrégation des données affectent également la précision de la détection par le niveau de qualité des fonctionnalités générées.

Analyse des données : La plupart des architectures IDS multi-agents dépendent de l'apprentissage à agent unique. Les techniques adaptées avec un tel composant d'analyse de données peuvent affecter les performances de détection des intrusions et des attaques de deux façons : Le coût de traitement de certaines techniques telles qu'ANN est bien très élevé, ce qui pourrait engendrer un retard au niveau du temps de réponse. La composante d'analyse est également affectée par les caractéristiques sélectionnées de la technique d'analyse choisie dans la classification des données.

Gestion & coordination : L'un des inconvénients des IDS-MAS est que l'architecture IDS est complètement un-configurable et non évolutive sans un administrateur. De conséquence, le système ne peut pas changer, s'adapter ou s'étendre par lui-même pour faire face aux changements environnementaux.

Partage des connaissances : Il existe trois schémas de partage de connaissances utilisés avec les architectures IDS multi-agents : Certaines architectures adaptent une base de connaissances partagée. Le problème de ce schéma est que les architectures IDS multi-agents deviennent sensibles, le concentrateur peut faire face à des erreurs et à des pannes. Les architectures adaptent des systèmes de base de connaissances distribuées. Cela ajoute également des frais de communication supplémentaires à l'architecture du système. Toutes les architectures existantes utilisaient des schémas d'échange de messages. Dans ce schéma, il pourrait y avoir des coûts de communication extrêmes entre les agents s'ils utilisaient un protocole de coopération inefficace.

5. Limites des MAS existants pour la détection d'intrusion

Les architectures IDS-MAS possèdent des caractéristiques avantageuses, mais elles souffrent également de limitations qui peuvent éventuellement dégrader les performances globales des intrusions et des détections d'attaques [30].

Les limites qui peuvent avoir un impact sur le temps de réponse et la précision des intrusions et de la détection des attaques sont les suivantes:

- **Scalabilité** : D'après des études, l'évolutivité des architectures IDS-MAS souffrent de limitations dans les méthodes proposées. Par conséquent, il est nécessaire d'élaborer et d'évaluer des modèles, des frameworks et des approches pour faire de l'évolutivité une caractéristique essentielle des architectures
- **Equilibre de la distribution des charges d'analyse en MAS (Load-balancing)** : Les mécanismes d'équilibrage de charge utilisés avec les architectures IDS-MAS souffrent de limitations. Il est donc nécessaire de mener des expériences et de proposer des modèles, des cadres et des mesures pour créer et évaluer cette caractéristique sur les architectures IDS multi-agents.

6. Conclusion

Les recherches effectuées dans ce chapitre nous ont permis d'avoir une vue globale sur les IDS à base de systèmes multi-agents existants, les systèmes multi-agents en général, les agents, ses architectures ainsi que les types des plateformes les plus connus. Nous nous sommes basés sur une taxonomie IDS-MAS, qui nous a aidés à analyser les travaux de recherche existants afin de pouvoir situer la contribution de ce projet de Master.

CHAPITRE III : CONCEPTION ET DONNEES D'APPRENTISSAGE UTILISEES

1. Introduction

Dans ce chapitre, nous proposerons un système de détection d'intrusions basé sur un système multi agents, en particulier les agents mobiles. Nous présenterons d'abord l'architecture du système, la partie conceptuelle puis le dataset utilisé pour l'apprentissage automatique.

2. Langages et environnements de développement

2.1 Langage de programmation Java

Pour implémenter notre système, nous avons choisi le langage de programmation Java développé par Sun Microsystems. Ce langage a réussi à susciter l'intérêt de nombreux développeurs à travers le monde. En fait, Java fonctionne avec une machine virtuelle appelée JVM (Java Virtual Machine) quilui permet de s'adapter à n'importe quelle machine. Java peut fonctionner aussi bien sur un PC que surun MAC, sur un téléphone ou même sur une carte à puce. Le choix de ce langage était fait naturellement suite à notre sélection de la plate-forme multi-agent JADE.

2.2 JADE (Java Agent Development Framework)

JADE est une plateforme facilitant la création des systèmes multi-agents entièrement implémentés en langage Java. Les agents exécutant JADE communiquent à l'aide du langage ACL (Agent Communication Language). La description de cette plateforme se trouve en Section 4.2.4.

Figure 12 illustre l'interface graphique de JADE. Chaque agent étant lié à un centenaire (ou Container).

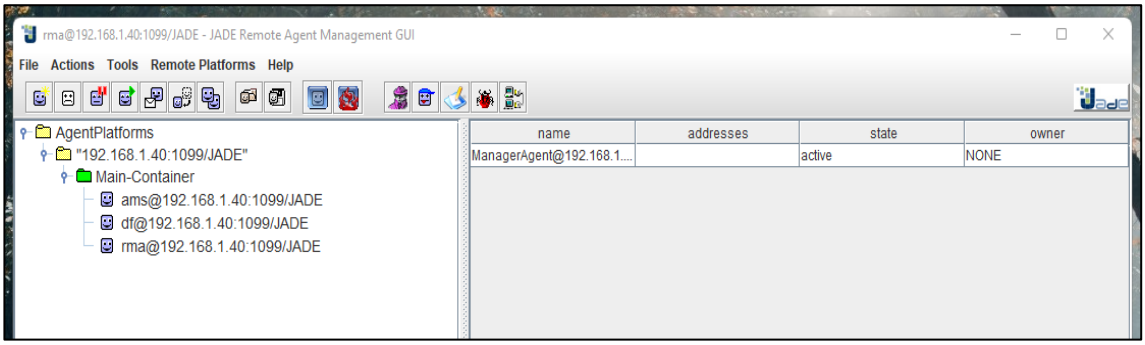


Figure 12 Interface graphique de jade

2.3 Weka

On a utilisé à travers la plateforme Jade des bibliothèques de Weka. Weka est une suite de logiciels d'apprentissage automatique écrits en Java et développés à l'Université de Waikato en Nouvelle-Zélande. La licence publique générale GNU rend Weka disponible gratuitement [34]. Weka peut être utilisé comme un outil externe et/ou en ligne via une interface graphique, comme montre la Figure 13, ou bien à travers des Jar à associer à une application. Dans notre projet, nous n'avons utilisé que les jars pour faire appel aux classificateurs depuis Jade.

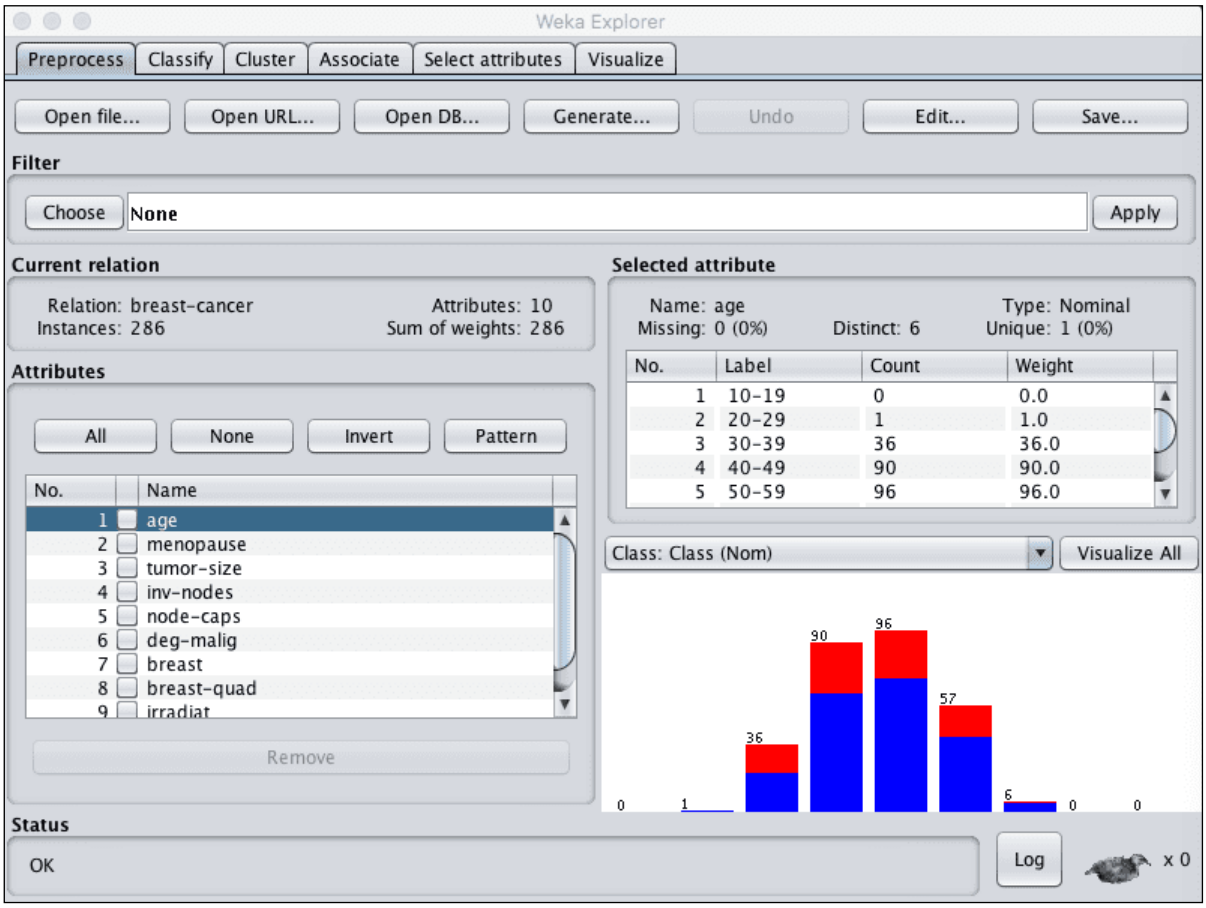


Figure 13 Interface graphique de Weka

2.4 Développement mobile sous Android

Afin de développer la partie supervision et contrôle de notre MAS-IDS par l'administrateur réseau, nous avons développé une application mobile avec Java Android en utilisant Android Studio.

Android Studio est un environnement de développement des applications mobiles. Il est disponible pour les systèmes Linux, Mac et Windows. La première version stable (1.0) de cet outil était disponible en décembre 2014 [36].

Le langage de programmation est Java ou Kotlin. Android Studio est simplement l'endroit où vous allez écrire, éditer et enregistrer vos projets et les fichiers qui les composent. Dans le même temps, Android Studio vous donnera accès au SDK Android ou au « Kit de développement logiciel » [37]. Considérez cela comme une extension du code Java qui lui permet de fonctionner correctement sur les appareils. Android Studio vous permet également d'exécuter votre code, soit via un émulateur ou via un matériel connecté à votre machine. Vous pourrez alors également « déboguer » le programme pendant son exécution et obtenir des commentaires afin de pouvoir résoudre plus rapidement le problème.

3. Principe et architecture MAS-IDS proposée

Les réseaux sont devenus plus complexes en raison de l'augmentation du nombre d'utilisateurs et du développement des services. Cette évolution doit s'accompagner de l'invention de nouveaux types d'attaques, ce qui rend l'approche centralisée inadapté. L'approche centralisée entraîne l'apparition de faiblesses dans le système informatique. De plus, dans le cas d'un flux important d'événements, cela peut entraîner un temps de traitement lent et un retard de réponse, et parfois conduire à un processus de

tolérance aux pannes et la récupération des données perdues. Et le dernier point est que la centralisation freine l'extension du système. Notre proposition dans le cadre de projet de Master se base sur l'utilisation d'un système multi-agent. Les agents mobiles nous permettent de faire l'analyse des intrusions en mode distribué.

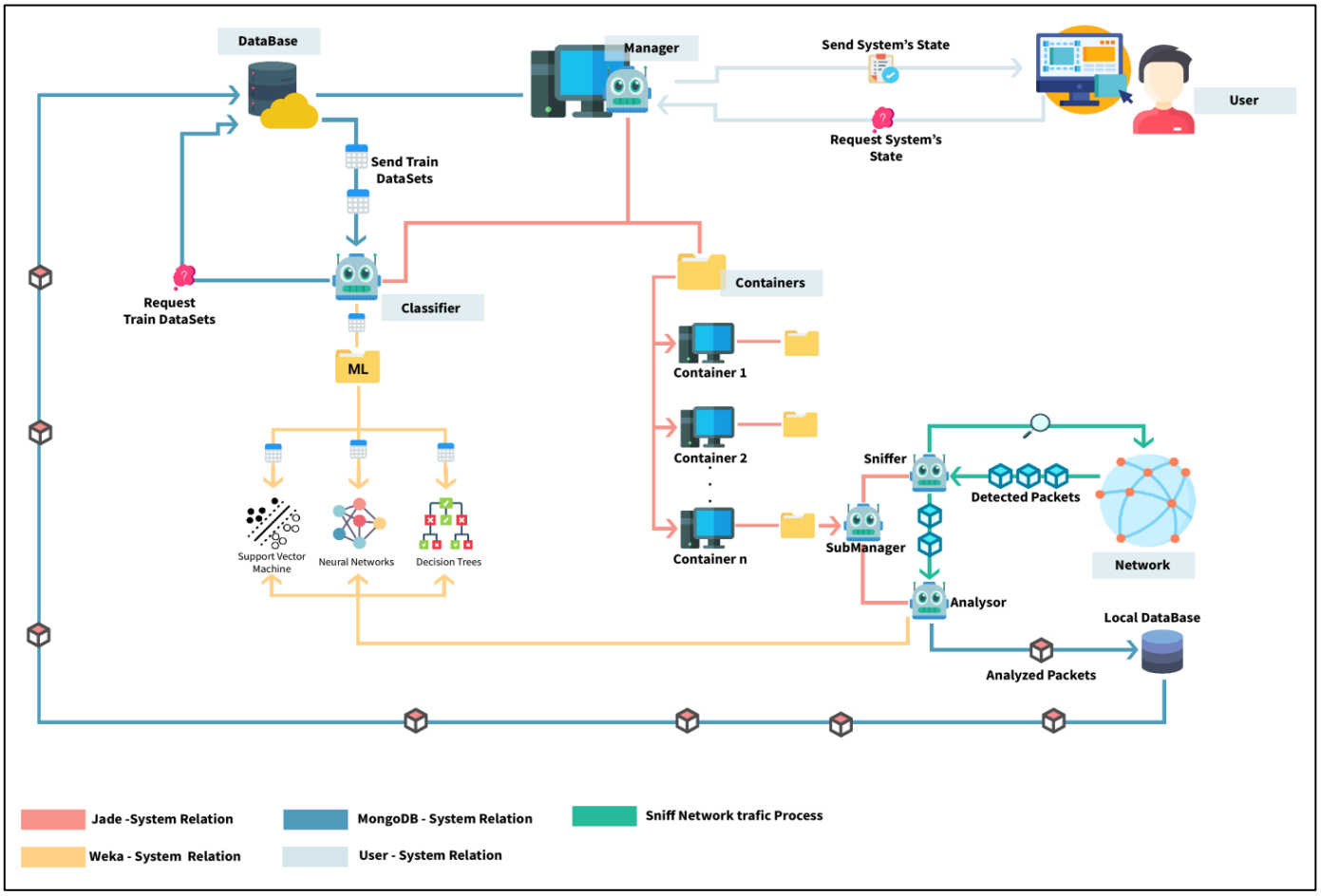


Figure 14 L'architecture générale du MAS-IDS proposé

L'approche que nous proposons vise à surveiller le système et détecter les anomalies, en tenant compte du temps de traitement. Notre solution devrait apporter les avantages suivants :

- ✓ Une approche distribuée de la détection des intrusions.
- ✓ Système utilisant la technologie des agents mobiles pour détecter les intrusions ce qui facilite la mise en œuvre du MAS à travers le code mobile.
- ✓ Pour rendre le modèle évolutif, nous utiliserons l'apprentissage automatique. En particulier, chaque agent (supervisant une machine) est responsable sur l'analyse réseau. L'analyse s'effectue à travers la classification supervisée en utilisant différents algorithmes tels que : les réseaux de neurones, support vector machine et arbres de décision.

La Figure 14 illustre le schéma global du MAS-IDS. Le système est basé sur une architecture multi-agents. Les agents suivants sont considérés :

- **Agent Manager :** Il est responsable sur tout dans le système. Son rôle est de créer les conteneurs ou « containers » selon la charge du réseau, et gérer l'état du réseau. Il a la capacité de communiquer avec tous les agents, avec les messages ACL (Agent Communication Language).

- **Agent classificateur :** Il prépare les classificateurs ML et il récupère les données d'entrainement du classificateur depuis la base de données.
- **Un conteneur :** Il permet d'associer à la fois unSub-Manager Agent, Agent Analyseur et un Agent Sgnifer.
- **Sub Manager Agent:** Il est responsable de la création à la fois du Sniffer et l'analyseur. Il gère également le conteneur auquel il appartient. On peut le considérer comme un manager local pour une machine.
- **Sniffer Agent :** Cet agent suit le trafic réseau et envoya que les paquets pertinent à l'analyseur.
- **Agent Analyseur :** Il fait l'analyse des paquets entrants (s'il est une anomalie ou non), puis il la sauvegarde dans la base de données (MongoDB).

L'avantage de l'architecture MAS-IDS par rapport aux MAS-IDS existants est qu'elle est flexible et permet de mieux gérer le cas de défaillance de l'entité Manager. En effet, la communication entre les agents permet de vérifier en temps réel les états des différents agents du système, y compris le Manager. En cas de non réponse d'un agent, dans un intervalle de temps important, un nouveau agent Manager du MAS est élu afin d'assurer le fonctionnement du MAS-IDS.

De plus, la tâche d'analyse est complètement distribuée en réseau. Les agents se coordonnent uniquement suite à un taux élevé d'anomalies (déterminé par un seuil) afin de réduire d'une part le temps dédié à la communication, et d'autre part le nombre de fausses alarmes qui doivent être traitées par l'administrateur. En fin, le MAS-IDS proposé fournit à l'administrateur une interface de supervision de l'état réseau, lui facilitant ainsi la visualisation de l'état du MAS-IDS pour pouvoir réagir rapidement.

4. Modélisation UML

Nous présentons dans cette section les principaux diagrammes UML que nous avons établit lors de la partie conception de notre projet.

4.1 Diagramme de cas d'utilisation

La figure 15 montre le digramme de cas d'utilisation générale du MAS-IDS.Voici la description de chaque cas d'utilisation :

- **Agent Manager**
  - Créer les conteneurs: pour chaque machine dans le réseau Agent Manager créer un conteneur qui contient Sub Manager, Sniffer Agent et Agent analyseur.
  - Gérer l'état du réseau : consiste à vérifier la charger de réseau.
- **Agent classificateur**
  - Préparer les classificateurs ML : pour commencer l'analyse du réseau.
  - Récupérer les données d'entrainement du modèle (training data) : il prend une copie des données, cette copie elle va être envoyée à Agent Analyseur après.
- **Sub Manager Agent**
  - Créer l'agent Sniffer et l'Analyseur : Sub Manager Agent créer l'agent Sniffer et l'Analyseur dans son conteneur pour gérer l'état du réseau.
  - Gérer le conteneur: vérifié l'état du réseau dans le conteneur auquel il appartient.

- **Sniffer Agent**
  - Sniffer le trafic réseau : Il s'agit de surveiller l'état du réseau, dans le conteneur où il se trouve. par l'analyse des paquets.
  - Envoyer les paquets à l'agent analyseur : après l'analyse des paquets, le sniffer envoie les paquets l'analyseur.
- **Agent Analyseur**
  - Traitement des paquets reçus (si c'est une anomalie ou pas) : Après avoir obtenu une copie des algorithmes auprès de l'agent classificateur. Ensuite Il les traite et, en fonction du résultat du traitement, les classe comme des intrusions ou pas.
  - Sauvegarder es données en BDD : une fois le traitement terminé et la décision prise, les données sont enregistrées dans la base de données.

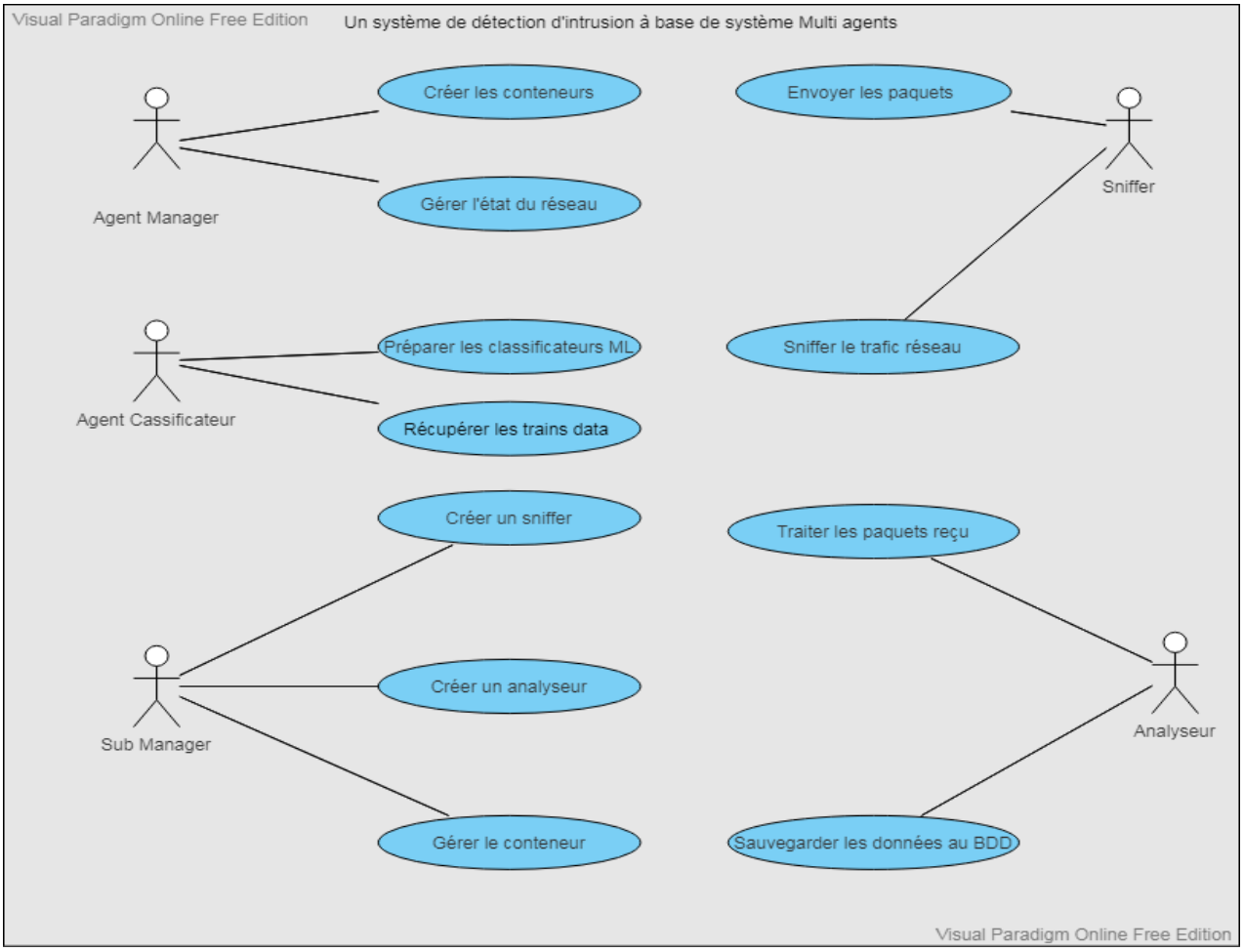


Figure 15 Le digramme de cas d'utilisation

4.2 Le Diagramme de séquence

Figure 16 illustre le digramme de séquence générale simplifié de notre MAS-IDS.

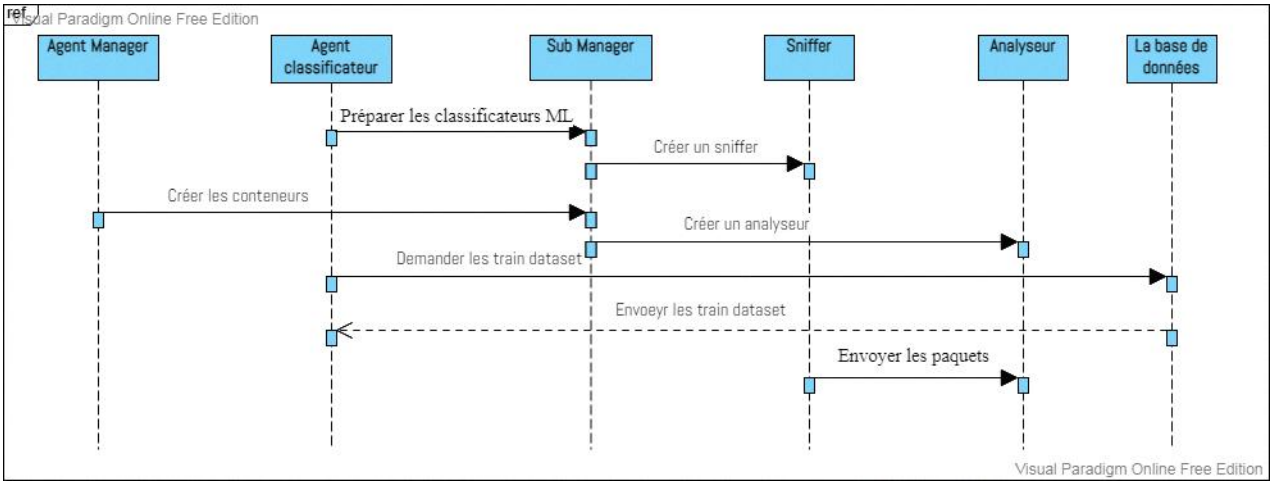


Figure 16 le diagramme de séquence

Voici le processus général de communication entre les agents :

- Le MAS-IDS fait appel à l'agent Manager. Lors de l'initialisation du système, l'agent Manager commence par la création des conteneurs pour chaque machine dans le réseau. Chaque conteneur contient un Sub-Manager, Sniffer agent et Agent analyseur.
- L'Agent classificateur demande les données d'entrainement (training data) et il les récupère depuis la base de données. Il prépare ensuite les classificateurs MLS (SVM, NN, DT) pour commencer l'analyse du système.
- Le Sub-Manager peut être considérer comme un Manager local. Il crée l'agent sniffer et l'analyseur, et vérifie l'état de son conteneur.
- Le sniffer analyse les paquets entrants de la machine et fait la partie prétraitement de données. Il envois ensuite ces données à l'analyseur pour la classification.
- Une fois que l'analyseur termine sans traitement, il envoie une copie à la BDD.

5. Les données d'apprentissage

Afin de tester les performances de notre système, nous avons utilisé l'ensemble de données NSL-KDD [35]. L'ensemble de données NSL-KDD est la version raffinée de l'ensemble donné KDD Cup 99. Les chercheurs l'ont utilisé pour créer des systèmes efficaces de détection d'intrusion, en réalisant plusieurs types d'analyses. La base de données contient plusieurs fichiers. Le Tableau 2 donne un aperçu des fichiers existants.

Nous avons effectué une analyse minutieuse de l'ensemble de données NSL-KDD a été effectuée, en utilisant différentes techniques d'apprentissage automatique disponible dans l'outil WEKA. Le but de cette analyse est de former et de tester diverses attaques existantes.

S.No.	Nom du fichier	Description
1	KDDTrain+.ARFF	La base NSL-KDD complète avec des étiquettes binaires au format ARFF
2	KDDTrain+.TXT	L'ensemble d'entrainement optimal complet NSL-KDD, y compris les étiquettes de type d'attaques au format CSV.
3	KDDTrain+_20Perce nt.ARFF	Un sous-ensemble de 20 % du fichier KDDTrain+.arff
4	KDDTrain+_20Perce nt.TXT	Un sous-ensemble de 20 % du fichier KDDTrain+.txt
5	KDDTest+.ARFF	L'ensemble de test NSL-KDD complet avec des étiquettes binaires au format ARFF
6	KDDTest+.TXT	L'ensemble de test NSL-KDD complet, y compris les étiquettes de type d'attaques au format CSV
7	KDDTest-21.ARFF	Un sous-ensemble du fichier KDDTest+.arff qui n'inclut pas les enregistrements avec un niveau de difficulté de 21 sur 21
8	KDDTest-21.TXT	Un sous-ensemble du fichier KDDTest+.txt qui n'inclut pas les enregistrements avec un niveau de difficulté de 21 sur 21

Tableau 2 Liste des fichiers d'ensemble de données NSL-KDD et leurs descriptions

Type d'ensemble de données	Nombre total de :					
	Dossiers	Classe normale	Classe DoS	Classe de sonde	Classe U2R	Classe R2L
KDD Train+ 20%	2519 2	13449	923 4	2289	11	209
		53.39 %	36. 65 %	9.09%	0.04 %	0.83 %
KDD Train+	1259 73	67343	459 27	11656	52	995
		53.46 %	36. 46%	9.25%	0.04 %	0.79 %
KDD Test+	2254 4	9711	745 8	2421	200	2754
		43.08 %	33. 08 %	10.74 %	0.89 %	12.22 %

Tableau 3 Les données normales et les attaques

L'ensemble de données NSL-KDD contient un nombre raisonnable d'enregistrements, ce qui permet d'effectuer des expériences sur l'ensemble complet. Dans chaque enregistrement, il y a 41 attributs dépliant différentes caractéristiques du flux et une étiquette attribuée à chacun soit comme un type d'attaque, soit comme normal.

Le tableau 3 montre détails des données (paquets) considérées comme non intrusions ou « normales » et les détails des classes d'attaques qui figurent dans l'ensemble de données NSL-KDD . Le 42ème attribut contient des données sur les différentes 5 classes existantes.

L'ensemble de données NSL-KDD contient de nombreux types d'attaques, qui se répartissent en quatre catégories [35] :

- **DOS :** le déni de service est une catégorie d'attaque, qui épuise les ressources de la victime, la rendant ainsi incapable de traiter les demandes légitimes.
- **Probing(Sondage) :** l'objectif de l'attaquant est d'obtenir des informations sur la victime distante, par exemple à traversle balayage des ports. Les fonctionnalités les plus pertinentes pour détecter une telle attaque sont : "la durée de connexion" et "le nombre d'octets de la source".

- **U2R :** l'accès non autorisé aux privilèges du super utilisateur local (root) est un type d'attaque, par lequel un attaquant utilise un compte normal pour se connecter à un système victime et tente d'obtenir des privilèges root/administrateur en exploitant une vulnérabilité chez la victime, par exemple à traversdes attaques par débordement de tampon.
- **R2L :** accès non autorisé depuis une machine distante, l'attaquant s'introduit dans une machine distante et accède localement à la machine victime. Par exemple, cela consiste à deviner le mot de passe.Les fonctionnalités pertinentes pour détecter un tel type d'attaques sont "la durée de la connexion" et "le service demandé". D'autres fonctionnalités au niveau de l'hôte sont "le nombre de tentatives de connexion infructueuses**".**

Chaque catégorie d'attaques de NSL-KDD contient un ensemble de micro-attaques (voir le tableau 4)

Classe d'attaque	Type d'attaque
DoS	Back, Land, Neptune, Pod, Smurf,Teardrop,Apache2, Udpstorm, Processtable, Worm (10)
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

Tableau 4 Les micro-attaques

6. Conclusion

Nous avons présenté dans ce chapitre l'architecture générale du MAS-IDS proposé ainsi que les données d'apprentissage utilisées pour la détection d'intrusions dans le MAS-IDS proposé.

CHAPITRE IV : IMPLEMENTATION ET TEST DU MAS-IDS

1. Introduction

Dans ce chapitre, nous décrivons d'abord l'environnement de programmation et les outils utilisés durant notre projet. Ensuite nous présenterons les résultats des méthodes d'apprentissage automatique : Arbre de décision, SVM et réseaux de neurones sur le dataset NLS-KDD. Puis on présentera l'interface de supervision réseaux liée à notre MAS-IDS développé. Enfin, nous expliquons le fonctionnement de notre système et son efficacité à détecter les intrusions.

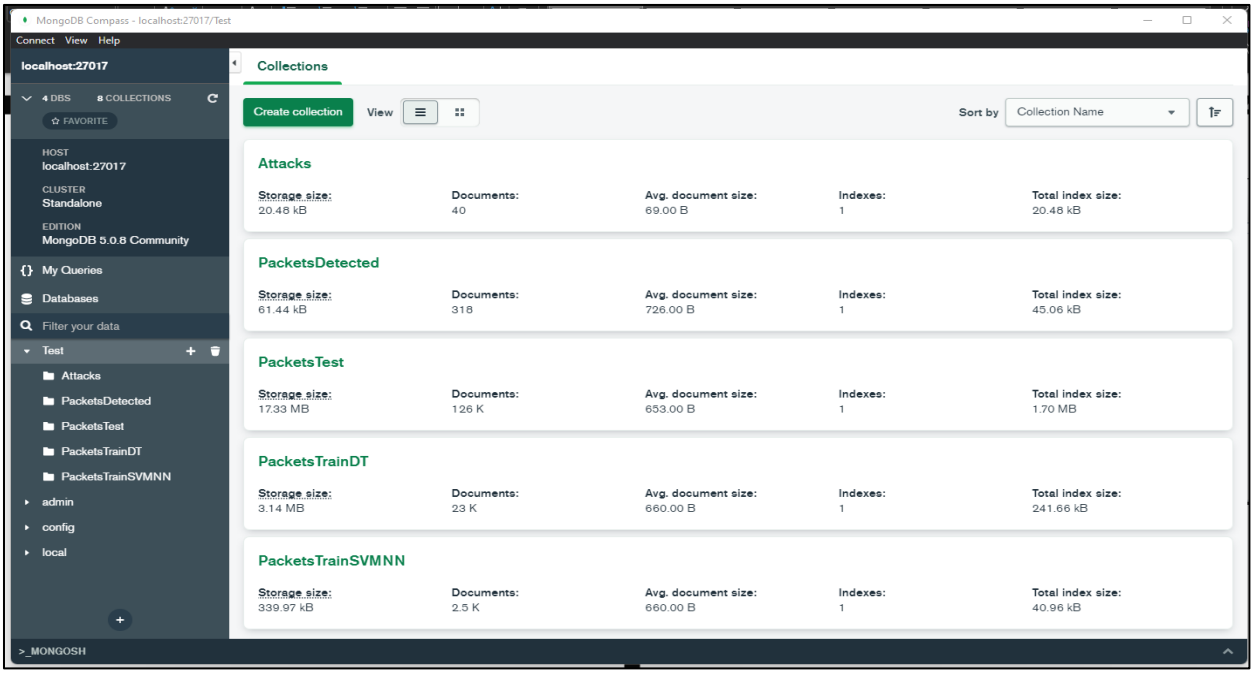
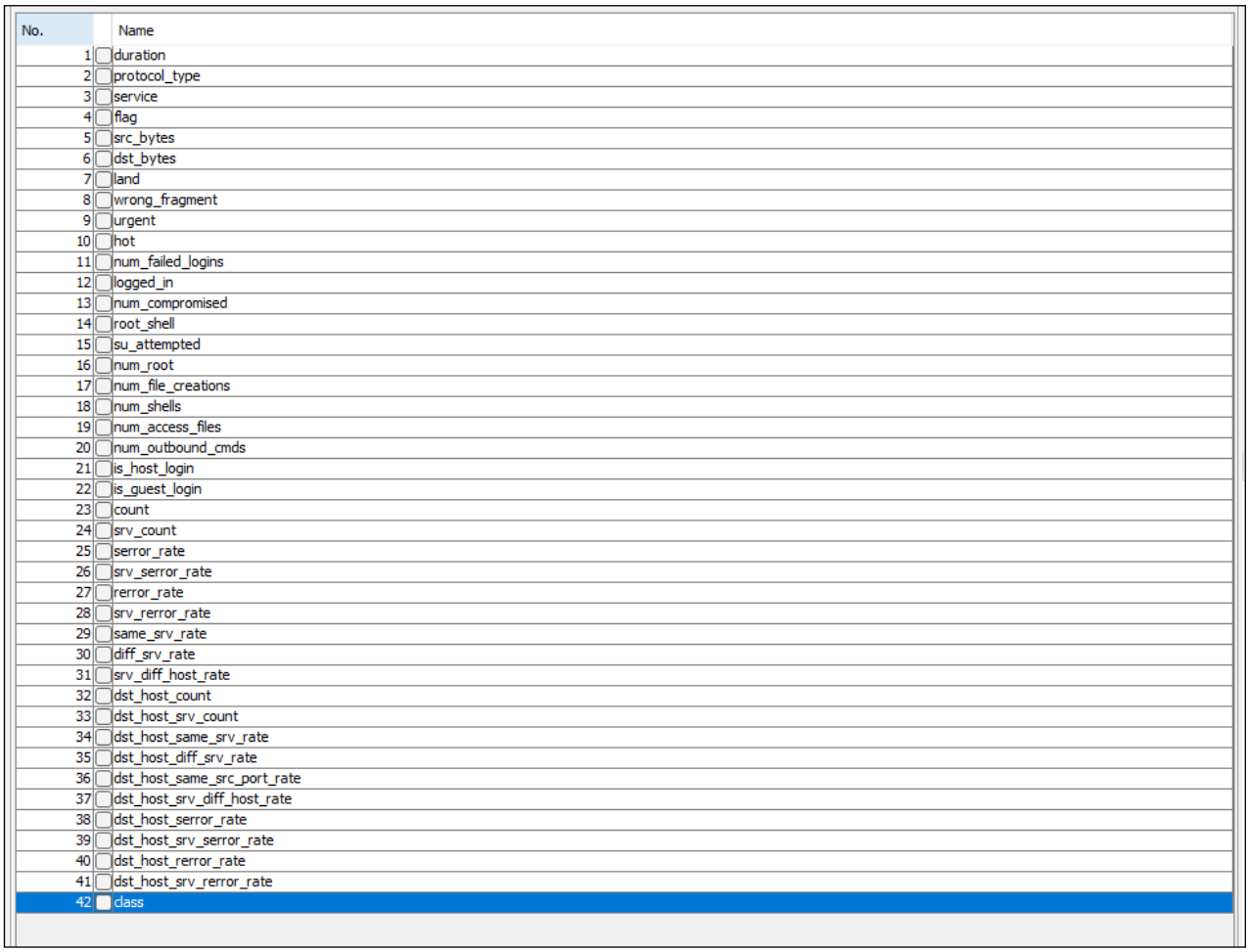


Figure 17 Les paquets sauvegarder dans MongoDB

2. Etapes de traitement de données et analyse des intrusions

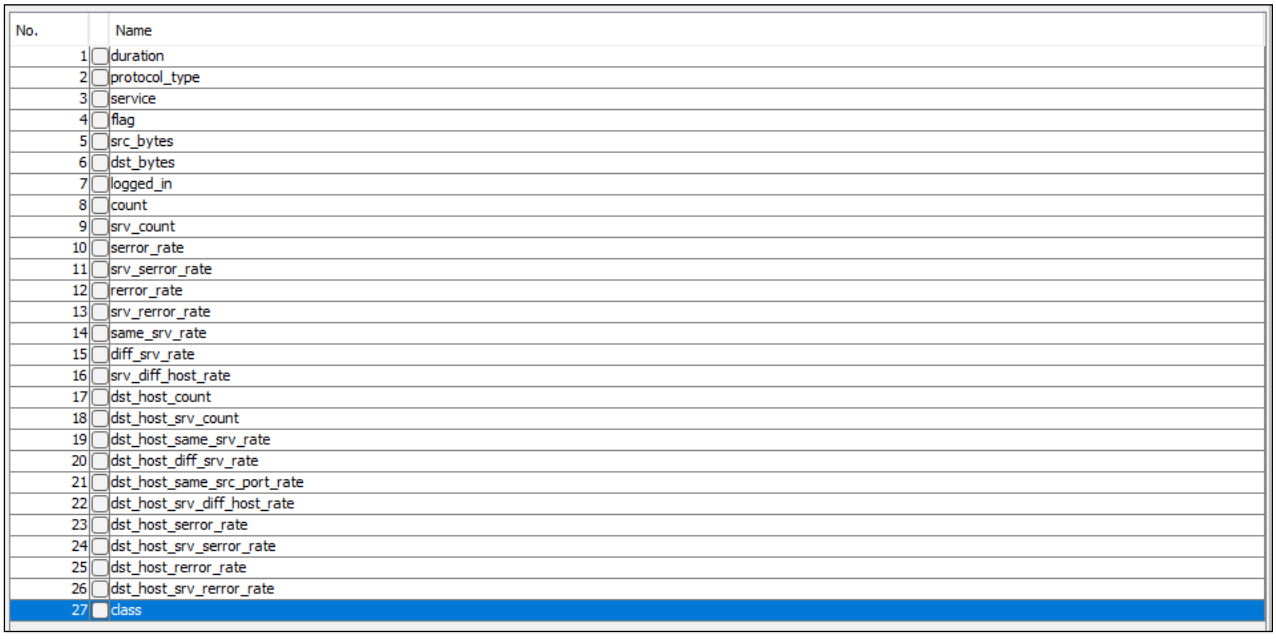
2.1 Enregistrement des données entrantes

Nous avons utilisé MongoDB pour l'enregistrement des paquets (voir Figure 17). MongoDB est un système de gestion de base de données (SGBD) à usage général puissant, flexible et évolutif. Il combine l'évolutivité avec des fonctionnalités telles que les indexes secondaires, le tri, l'agrégation et les indexes géo spatiaux. L'intérêt de MongoDB réside dans sa grande évolutivité, ainsi que dans sa contribution à un système performant. Il est compatible avec tous les systèmes d'exploitation, et ses bibliothèques sont disponibles pour différents langages.



No.	Name
1	duration
2	protocol_type
3	service
4	flag
5	src_bytes
6	dst_bytes
7	land
8	wrong_fragment
9	urgent
10	hot
11	num_failed_logins
12	logged_in
13	num_compromised
14	root_shell
15	su_attempted
16	num_root
17	num_file_creations
18	num_shells
19	num_access_files
20	num_outbound_cmds
21	is_host_login
22	is_guest_login
23	count
24	srv_count
25	serror_rate
26	srv_serror_rate
27	rerror_rate
28	srv_rerror_rate
29	same_srv_rate
30	diff_srv_rate
31	srv_diff_host_rate
32	dst_host_count
33	dst_host_srv_count
34	dst_host_same_srv_rate
35	dst_host_diff_srv_rate
36	dst_host_same_src_port_rate
37	dst_host_srv_diff_host_rate
38	dst_host_serror_rate
39	dst_host_srv_serror_rate
40	dst_host_rerror_rate
41	dst_host_srv_rerror_rate
42	class

Figure 18La liste des attributs dans NLS-KDD



No.	Name
1	duration
2	protocol_type
3	service
4	flag
5	src_bytes
6	dst_bytes
7	logged_in
8	count
9	srv_count
10	serror_rate
11	srv_serror_rate
12	rerror_rate
13	srv_rerror_rate
14	same_srv_rate
15	diff_srv_rate
16	srv_diff_host_rate
17	dst_host_count
18	dst_host_srv_count
19	dst_host_same_srv_rate
20	dst_host_diff_srv_rate
21	dst_host_same_src_port_rate
22	dst_host_srv_diff_host_rate
23	dst_host_serror_rate
24	dst_host_srv_serror_rate
25	dst_host_rerror_rate
26	dst_host_srv_rerror_rate
27	class

Figure 19 La liste des attributs sélectionnés pour l'apprentissage

2.2 Prétraitement et sélection des attributs

Le dataset NLS-KDD est déjà disponible en version prétraitée. Les données non complètes et redondantes qui peuvent influencer la performance des classificateurs sont déjà supprimées. De plus, les ensembles de tests et d'entrainement optimaux sont préparés.

Dans cette étape de prétraitement, nous avons effectué une sélection manuelle des attributs sur la base de leur simple corrélation avec la variable dépendante. Les figures18 et 19 montrent les attributs de NLS-KDD et ceux sélectionnés pour l'apprentissage.

2.3 Les algorithmes d'apprentissage

Dans cette étape, nous avons déjà entraîné notre système avec l'ensemble de test. Nous avons utiliséles trois types de classificateurs WEKA pour l'apprentissage automatique. Une description détaillée du principe de chaque méthode se trouve en Section 4 du second chapitre.

2.3.1 Arbre de décision (Decision Tree)

Nous résumons le principe de la méthode de l'arbre de décision comme suit : vous définissez un attribut au nœud racine, puis ajoutez un enfant pour chacune des valeurs possibles. Le processus est répété pour chaque branche, en utilisant uniquement les attributs qui accèdent à la branche. Pour obtenir la feuille la plus homogène il faudra prendre en compte le critère de pureté de la feuille. Le meilleur attribut est celui qui a la meilleure corrélation avec la distribution de la catégorie.

La figure 20 montre la méthode de Weka utilisée pour la création d'un modèle à l'aided'un arbre de décision.

```
J48 j48 = new J48();
j48.buildClassifier(TrainDataDT);
```

Figure 20 créer un modèle de DT

2.3.2 Réseaux de neurones

L'objectif de cette méthode est d'identifier les relations de base dans un ensemble de données, à travers un processus similaire au travail du cerveau humain connu sous le nom de mimétisme. Le concept de réseau de neurones permet à ne pas repenser les critères de sortie pour obtenir les meilleurs résultats. La figure 21 donne un aperçu sur les méthodes utilisées sur Weka pour faire appel à ce classificateur.

```
MultilayerPerceptron multilayerPerceptron = new MultilayerPerceptron();
multilayerPerceptron.setLearningRate(0.1);
multilayerPerceptron.setMomentum(0.2);
multilayerPerceptron.setTrainingTime(2000);
multilayerPerceptron.setHiddenLayers("3");
multilayerPerceptron.buildClassifier(TrainDataNN);
```

Figure 21 créer un modèle de NN

2.3.3 SVM

C'est une méthode très utilisée en classification supervisée de données. Elle se base sur la transformation des données sur de nouvelles dimensions, puis cherche à trouver une séparation optimale des données afin que les points de données puissent être catégorisés, même lorsque les données ne sont pas autrement séparables linéairement. Un séparateur entre les catégories est trouvé, puis les données sont transformées de manière à ce que le séparateur puisse être dessiné en tant qu'hyperplan [98]. La figure 22 montre l'appel de la méthode SMO dans Weka pour la classification SVM.

```
SMO smo = new SMO();
smo.buildClassifier(TrainDataSVM);
```

Figure 22 créer un modèle de SVM

2.4 Résultats de l'analyse des intrusions avec le dataset NLS-KDD

Nous avons comparé la performance des algorithmes de classification sur différentes variables indépendantes, notamment, par catégorie d'attaques, par micros attaques et si c'est une anomalie ou pas.

Les figures suivantes (Figure 22, 23 et 24) montrent les résultats des classificateurs. Nous avons eu de bons résultats dans les trois cas, donc on a choisi à intégrer dans notre MAS-IDS la classification à base de la variable indépendante « micros attaques » parce qu'il n'y a pas de différence de résultat, et aussi les micros attaques permettent de connaître le type d'intrusion qui peut endommager notre système

Nous notons que les résultats de SVM et NN sont similaires (presque les mêmes F-MEASURE et précision), mais les meilleurs résultats sont obtenus à partir de DT. Malgré la grande quantité de données traitées nous avons obtenu une précision de 100% et F-MEASURE 1 qui est le maximum. Mais cela n'enlève rien aux résultats obtenus par SVM et NN, parce qu'une précision de 95% et F-MEASURE plus que 0.9 ne peut pas être négligé. Il faut noter aussi que NN nécessite un temps important d'entrainement par rapport aux autres algorithmes.

----------Arbre de decision----------

F-Measure	0.993917180067067	
Correctly Classified Instances	22388	99.308 %
Incorrectly Classified Instances	156	0.692 %
Kappa statistic	0.9859	
Mean absolute error	0.011	
Root mean squared error	0.0741	
Relative absolute error	2.2386 %	
Root relative squared error	14.9621 %	
Total Number of Instances	22544	

----------Reseau de Neurones----------

F-Measure	0.9957591178965225	
Correctly Classified Instances	2462	99.1942 %
Incorrectly Classified Instances	20	0.8058 %
Kappa statistic	0.9151	
Mean absolute error	0.0162	
Root mean squared error	0.0863	
Relative absolute error	16.2285 %	
Root relative squared error	38.7377 %	
Total Number of Instances	2482	

----------Support Vector Machine----------

F-Measure	0.987602437486867	
Correctly Classified Instances	2423	97.6229 %
Incorrectly Classified Instances	59	2.3771 %
Kappa statistic	0.7007	
Mean absolute error	0.0238	
Root mean squared error	0.1542	
Relative absolute error	23.869 %	
Root relative squared error	69.2048 %	
Total Number of Instances	2482	

Figure 24 Résultats de classification avec la variable Normal/Anomalie

----------Arbre de decision----------

F-Measure	0.9978356489562242	
Correctly Classified Instances	22361	99.1883 %
Incorrectly Classified Instances	183	0.8117 %
Kappa statistic	0.9881	
Mean absolute error	0.005	
Root mean squared error	0.0502	
Relative absolute error	1.849 %	
Root relative squared error	13.5981 %	
Total Number of Instances	22544	

----------Reseau de Neurones----------

F-Measure	0.9634230503795721	
Correctly Classified Instances	2376	95.7293 %
Incorrectly Classified Instances	106	4.2707 %
Kappa statistic	0.9397	
Mean absolute error	0.037	
Root mean squared error	0.1183	
Relative absolute error	12.9949 %	
Root relative squared error	31.3583 %	
Total Number of Instances	2482	

----------Support Vector Machine----------

F-Measure	0.9540150995195608	
Correctly Classified Instances	2352	94.7623 %
Incorrectly Classified Instances	130	5.2377 %
Kappa statistic	0.9258	
Mean absolute error	0.2424	
Root mean squared error	0.3197	
Relative absolute error	85.0806 %	
Root relative squared error	84.7051 %	
Total Number of Instances	2482	

Figure 23 Résultats de classification avec la variable caégorie d'attaques

```
------------------Arbre de decision-------------------------

F-Measure                   1.0
Correctly Classified Instances       22333               99.0641 %
Incorrectly Classified Instances       211                0.9359 %
Kappa statistic                          0.9877
Mean absolute error                      0.0007
Root mean squared error                  0.019
Relative absolute error                  1.9037 %
Root relative squared error             13.801  %
Total Number of Instances            22544
```

```
------------------Reseau de Neurones-------------------------

F-Measure                   0.8382352941176471
Correctly Classified Instances        1992               80.2579 %
Incorrectly Classified Instances       490               19.7421 %
Kappa statistic                          0.7921
Mean absolute error                      0.0215
Root mean squared error                  0.0932
Relative absolute error                 45.2046 %
Root relative squared error             60.4177 %
Total Number of Instances             2482
```

```
------------------Support Vector Machine-------------------------

F-Measure                   0.8378378378378377
Correctly Classified Instances        2177               87.7115 %
Incorrectly Classified Instances       305               12.2885 %
Kappa statistic                          0.8708
Mean absolute error                      0.0475
Root mean squared error                  0.1528
Relative absolute error                 99.7833 %
Root relative squared error             99.0464 %
Total Number of Instances             2482
```

Figure 25 Résultats de classification avec la variable micro-attaques

3. Implémentation de la partie MAS

Afin de pouvoir utiliser les outils cités précédemment, il faut importer leur bibliothèque, à savoir : «Weka.jar » et « Jade.jar ».

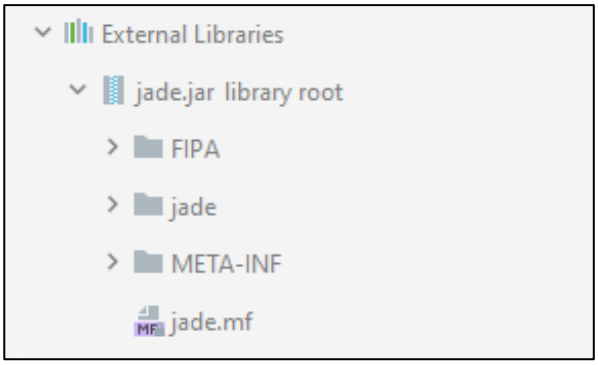


Figure 26 Les fichier Jar utilisés

3.1 Création des agents

La figure 23 représente une partie du code source qui permet la création des agents de notre MAS-IDS, l'agent Manager plus précisément.

```
Runtime runtime = Runtime.instance();
Profile profile = new ProfileImpl();
ContainerController containerController;

 profile.setParameter(Profile.MAIN_HOST,  s1: "localhost");
 profile.setParameter(Profile.GUI, s1: "true");
 containerController = runtime.createMainContainer(profile);

    AgentController agentController = null;
    try {
        agentController =   containerController.createNewAgent( nickname: "ManagerAgent", className: "ManagerAgent", args: null);
        agentController.start();

    } catch (StaleProxyException e) {
        e.printStackTrace();
    }
```

Figure 27 Exemple de création de l'agent Manager en Jade

3.2 La mobilité de l'agent

L'utilisation de la mobilité de l'agent est la seule solution pour éviter les problèmes rencontrés dans d'autres modèles traditionnels, tels que le volume de données, le rappel de procédure et l'évaluation à distance. La figure 25 montre un exemple de la mobilité de l'agent. Après avoir navigué et capturé les résultats de performance, les agents analysent le nœud du réseau.

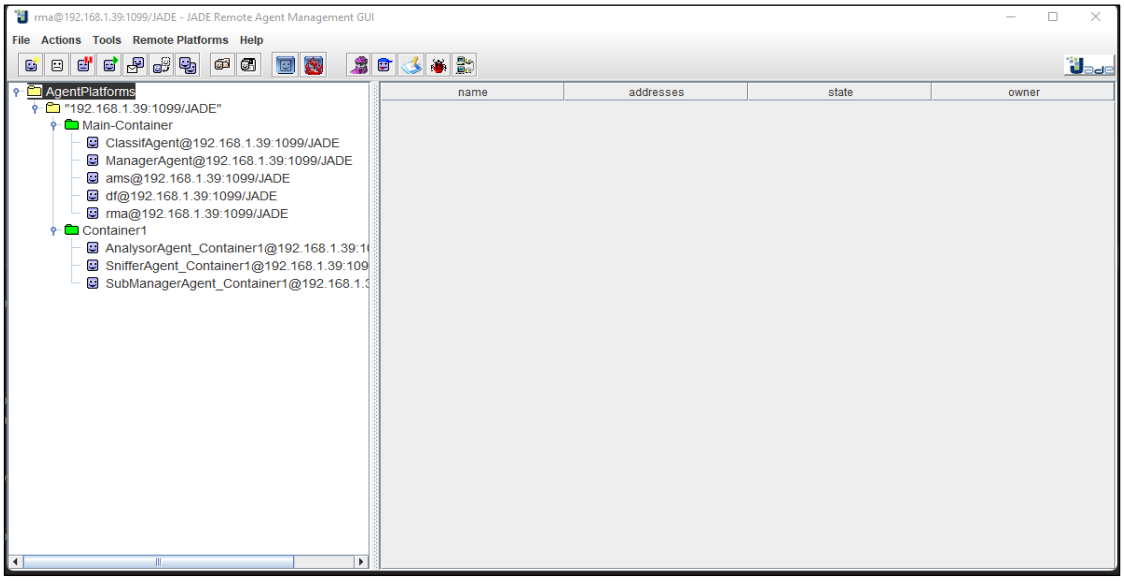


Figure 28 Les agents après la mobilité

4. Interface de supervision de l'IDS-MAS

L'interface de supervision (ou la console de supervision) permet d'accéder aux informations en temps réel du MAS-IDS. Son objectif est d'assister les administrateurs dans la gestion et la supervision du réseau via le MAS-IDS. Les principales données visualisées sont représentées dans les figures suivantes.

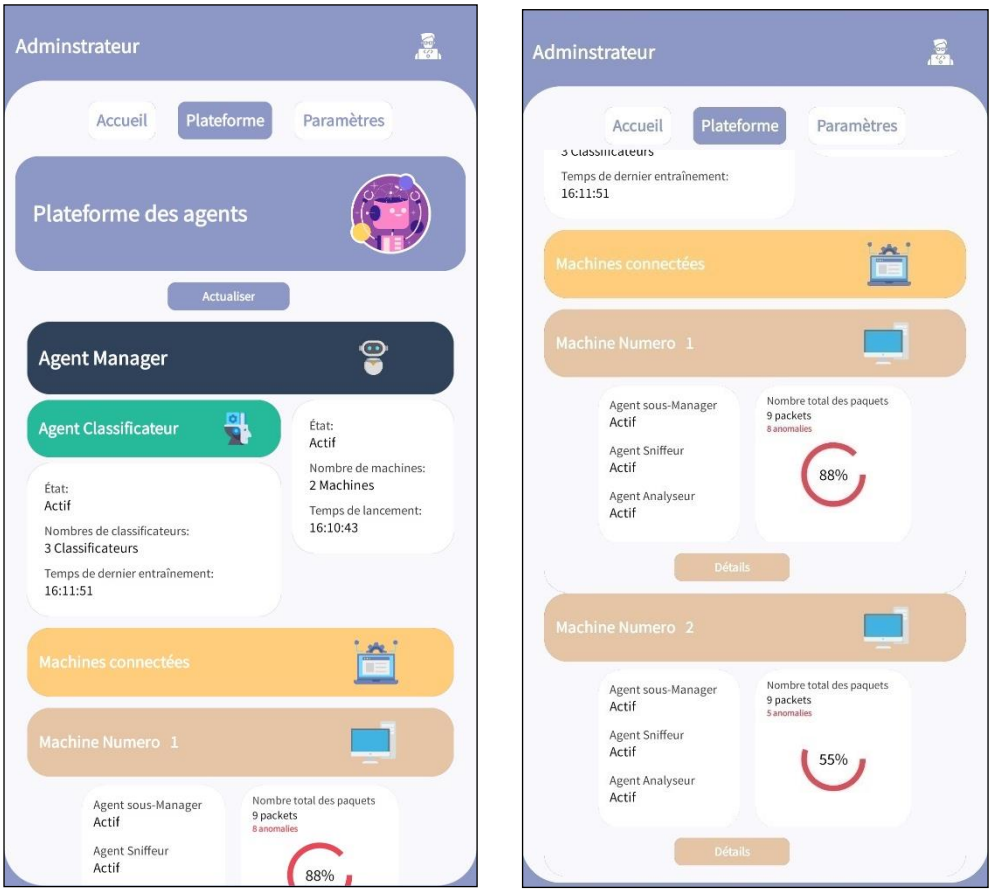


Figure 30 L'état des machines de système

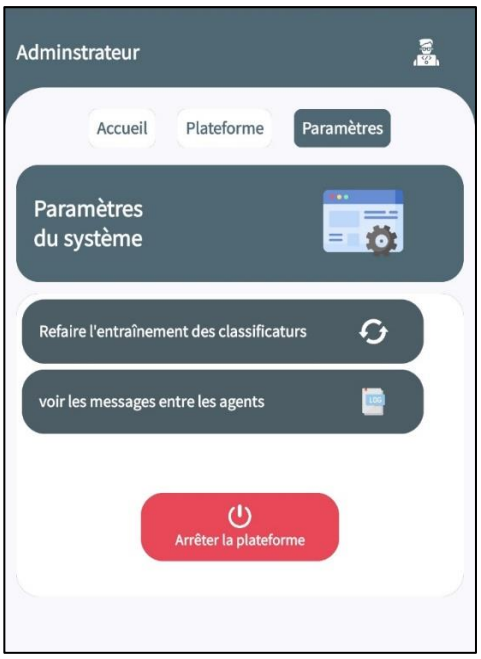


Figure 29Paramètres du système

La figure 29 donne un aperçu de notre système en termes de nombre d'appareils et de statut des agents, alors que la figure 30 montre comment l'interface donne l'accès aux différents paramètres de notre système. Notre système offre la possibilité de relancer le traitement des classificateurs. Et aussi l'administrateur peut voir les messages entre les agents pour assurer le bon fonctionnement des agents du système.

L'administrateur peut également vérifier les messages échangés entre les agents. La figure 31 illustre la visualisation des messages entre les agents.

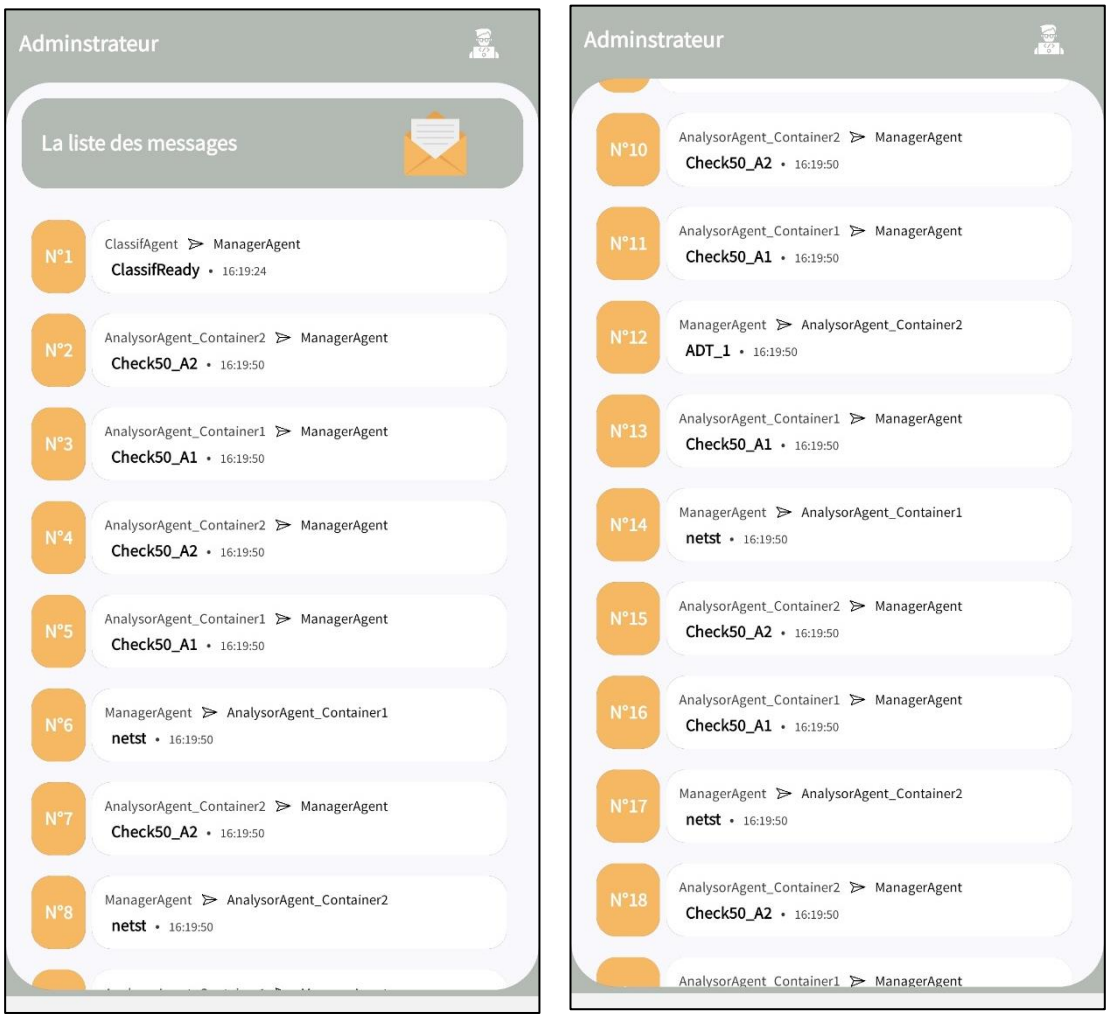


Figure 31 Les messages entre les agents

La vérification des messages entre les agents est importante pour vérifier l'état des agents du système.

Nous avons également donné à l'administrateur la possibilité d'obtenir des informations sur les algorithmes de classification utilisés ainsi que les derniers résultats obtenus. La figure 32 offre une vision des classificateurs utilisés avec leurs résultats.

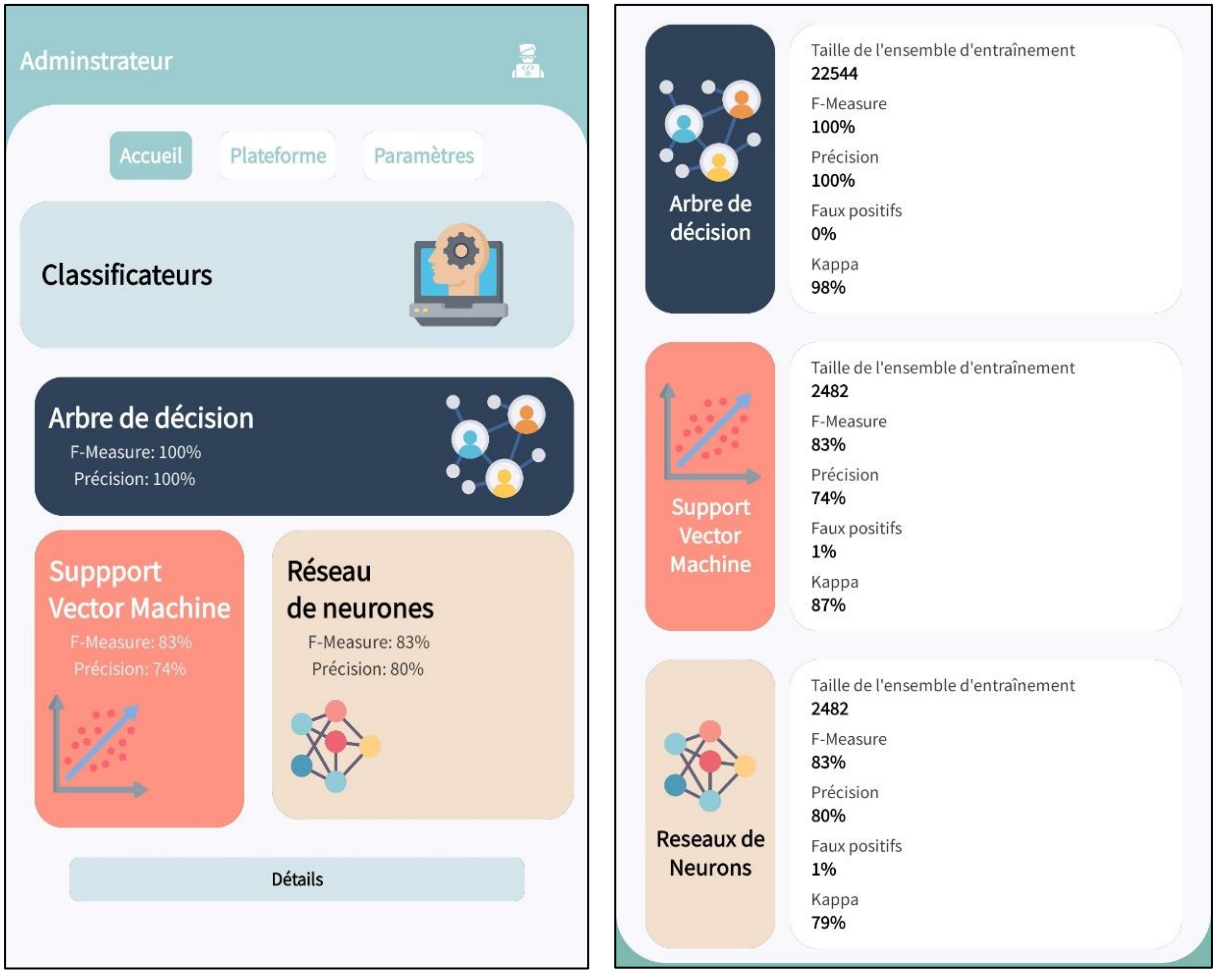


Figure 32 Les classificateurs utilisés

La figure 33 nous donne plus d'informations sur l'efficacité des classificateurs, en termes de tailles d'ensemble traites, la précision, F-MEASURE et les pourcentages des faux positifs.

Figure 33 Informations et statistiques d'une machine

Grâce à cette interface, nous pouvons connaître l'état de chaque appareil de notre système, et les statistiques nous permettent également de connaître le pourcentage d'activités suspectes, en plus de connaître le type d'attaques ciblant le système. Et enfin, le pourcentage d'utilisation de chaque classificateur utilisé est aussi affiché.

5. CONCLUSION

Le rôle d'IDS est la surveillance du trafic réseau contre les activités suspectes. Dans ce contexte on a proposé un système de détection d'intrusion basé sur les agents mobiles, en utilisant des différents types de classificateurs. Nous avons également proposé une application mobile qui permet à un administrateur de contrôler et superviser facilement l'état du réseau et les anomalies à travers le MAS-IDS.

CONCLUSION GENERALE

De nos jours, on constate que le développement des logiciels malveillants est de plus en plus fréquent, donc la nécessité de procurer une préservation de confidentialité des données échangées est une priorité. En raison de ce développement rapide, les systèmes de détection d'intrusion affrontent de plus en plus de défis à identifier les vulnérabilités. La surveillance distribuée du trafic réseau et les rapports rendent le système de détection d'intrusion plus performant. Plusieurs techniques ont été proposées afin de concevoir des IDS distribués. Dans ce projet Master, notre intérêt était orienté vers l'utilisation des systèmes multi-agents pour concevoir un IDS performant. Dans le cadre de ce projet de Master, nous avons développé un MAS-IDS basé sur des agents mobiles. Le système multi-agent proposé possède un agent « Manager » qui va diriger le fonctionnement du système ainsi que sa réaction face aux attaques. Cet agent fait appel à différents classificateurs : arbre de décision, SVM et NN.

A l'inverse des IDS existants, notre architecture MAS-IDS est flexible et permet de mieux gérer le cas de défaillance des entités responsables sur l'analyse en réseau. En effet, la communication entre les agents permet de vérifier en temps réel les états des différents agents du système. De plus, la tâche d'analyse est complètement distribuée en réseau et ne dépends pas de l'agent Manager. Les agents se coordonnent uniquement suite à un taux élevé d'anomalies (déterminé par un seuil) afin de réduire d'une part le temps dédié à la communication, et d'autre part le nombre de fausses alarmes qui doivent être traitées par l'administrateur. Enfin, le MAS-IDS proposé fournit à l'administrateur une interface de contrôle et de supervision, lui facilitant ainsi la visualisation de l'état du MAS-IDS pour pouvoir réagir rapidement.

Parmi les perspectives de ce travail est de pouvoir déployer note MAS-IDS afin d'analyser sa performance dans un vrai réseau. De plus, il sera intéressant de conduire une expérimentation permettant d'analyser les performances du MAS-IDS proposé dans ce projet de Master par rapport à d'autres MAS-IDS existants afin d'améliorer la solution proposée.

Enfin, à travers ce projet de Master, on a eu l'occasion de se familiariser avec le développement des systèmes multi-agents, les méthodes de machine Learning utilisées pour la détection d'intrusion, ses avantages ainsi que ses inconvénients. Nous avons également eu l'occasion à faire une initiation à la recherche scientifique à travers l'analyse des principaux travaux de recherche en MAS-IDS.

BIBLIOGRAPHIE

[1] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1), 1-22.

[2] Vokorokos, L., & Baláž, A. (2010, May). Host-based intrusion detection system. In *2010 IEEE 14th International Conference on Intelligent Engineering Systems* (pp. 43-47). IEEE.

[3] Mishra, P., Varadharajan, V., Pilli, E. S., & Tupakula, U. (2018). Vmguard: A vmi-based security architecture for intrusion detection in cloud environment. *IEEE Transactions on Cloud computing*, *8*(3), 957-971.

[4] Parvat, A., Dev, S., Kadam, S., & Chavan, J. (2017, August). Network Intrusion Detection System Using Ensemble of Binary Deep Learning Classifiers. In *International Conference on Smart Trends for Information Technology and Computer Communications* (pp. 3-10). Springer, Singapore.

[5] Estrin, D., Govindan, R., Heidemann, J., & Kumar, S. (1999, August). Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (pp. 263-270).

[6] Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of network and computer applications*, *36*(1), 42-57.

[7] Alves, R. C., Oliveira, D. A., Segura, G. A. N., & Margi, C. B. (2019). The cost of software-defining things: A scalability study of software-defined sensor networks. *IEEE Access*, *7*, 115093-115108.

[8] Wu, M. F. (2008, March). Protocol-based classification for intrusion detection. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering* (No. 7). World Scientific and Engineering Academy and Society.

[9] Chang, R. K. (2002). Defending against flooding-based distributed denial-of-service attacks: A tutorial. *IEEE communications magazine*, *40*(10), 42-51.

[10] Arora, K., Kumar, K., & Sachdeva, M. (2011). Impact analysis of recent DDoS attacks. *International Journal on Computer Science and Engineering*, *3*(2), 877-883.

[11] Allen, L., Heriyanto, T., & Ali, S. (2014). *Kali Linux–Assuring security by penetration testing*. Packt Publishing Ltd.

[12] Gunawan, T. S., Lim, M. K., Kartiwi, M., Malik, N. A., & Ismail, N. (2018). Penetration testing using Kali linux: SQL injection, XSS, wordpres, and WPA2 attacks. *Indonesian Journal of Electrical Engineering and Computer Science*, *12*(2), 729-737.

[13] Denis, M., Zena, C., & Hayajneh, T. (2016, April). Penetration testing: Concepts, attack methods, and defense strategies. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (pp. 1-6). IEEE.

[14] Saxena, A. K., Sinha, S., & Shukla, P. (2017, May). General study of intrusion detection system and survey of agent based intrusion detection system. In *2017 International Conference on Computing, Communication and Automation (ICCCA)* (pp. 471-421). IEEE.

[15] Luo, B., & Xia, J. (2014). A novel intrusion detection system based on feature generation with visualization strategy. *Expert Systems with Applications*, *41*(9), 4139-4147.

[16] Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[17] Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, *31*(3), 357-374.

[18] Rocha, J., Boavida-Portugal, I., & Gomes, E. (2017). Introductory chapter: multi-agent systems. In *Multi-Agent Systems*. IntechOpen.

[19] Côté, M. (1999). Une architecture multiagent et son application aux services financiers. *Mémoire présenté à la Faculté des Etudes Supérieures de l'Université de Laval, Faculté des sciences et de génie, département informatique*.

[20] Marir, T. (2016). Les Systèmes Multi-Agents. *oum lbouaghi, unversité larbi Ben Mhidi*, *2017*.

[21] Dastani, M., van Riemsdijk, M. B., & Meyer, J. J. C. (2005). Multi-Agent Programming-Languages, Platforms and Applications, chapter Programming multi-agent systems in 3APL.

[22] FIPA Specifications (2000), http://www.fipa.org, accessible le 25/05/2022.

[23] Poggi, A., & Tomaiuolo, M. (2010). Integrating peer-to-peer and multi-agent technologies for the realization of content sharing applications. In *Information Retrieval and Mining in Distributed Environments* (pp. 93-107). Springer, Berlin, Heidelberg.

[24] Bellifemine, F., Poggi, A., & Rimassa, G. (1999, April). JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM* (Vol. 99, No. 97-108, p. 33).

[25] Jarvis, J. (2003). JACK Intelligent Agents™ JACK Teams Manual.

[26] Silva, L. D., & Padgham, L. (2004, December). A comparison of BDI based real-time reasoning and HTN based planning. In *Australasian Joint Conference on Artificial Intelligence* (pp. 1167-1173). Springer, Berlin, Heidelberg

[27] Bougueroua, N., Mazouzi, S., Belaoued, M., Seddari, N., Derhab, A., & Bouras, A. (2021). A survey on multi-agent based collaborative intrusion detection systems. *Journal of Artificial Intelligence and Soft Computing Research*, *11*(2), 111-142.

[28] Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, *57*(10), 2266-2279..

[29] Saeed, I. A., Selamat, A., Rohani, M. F., Krejcar, O., & Chaudhry, J. A. (2020). A Systematic State-of-the-Art Analysis of Multi-Agent Intrusion Detection. *IEEE Access*, *8*, 180184-180209.

[30] Dastani, M., Birna Riemsdijk, M. V., & Meyer, J. J. C. (2005). Programming multi-agent systems in 3APL. In Multi-agent programming (pp. 39-67). Springer, Boston, MA.Rai, K., Devi, M. S., & Guleria, A. (2016). Decision tree based algorithm for intrusion detection. *International Journal of Advanced Networking and Applications*, *7*(4), 2828.

[31] Jha, J., & Ragha, L. (2013). Intrusion detection system using support vector machine. *International Journal of Applied Information Systems (IJAIS)*, *3*, 25-30.

[32] Shenfield, A., Day, D., & Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. *Ict Express*, *4*(2), 95-99.

[33] BRIKA, A. B., Par, S., & Pal, C. Data Mining avec Weka.

[34] Kumar, V., Chauhan, H., & Panwar, D. (2013). K-means clustering approach to analyze NSL-KDD intrusion detection dataset. *International Journal of Soft Computing and Engineering (IJSCE)*, *3*(4), 1-4

[35] Benbourahla, N. (2015). *Android 5-Les fondamentaux du développement d'applications Java*. Éditions ENI.

[36] Developers Android, disponible sur https://developer.android.com/distribute/best-practices/develop/build-with-firebase?hl=fr accessible le 25/05/2022.