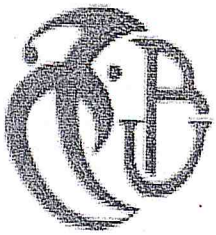
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEURE
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLAB DE BLIDA

FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE



Projet de fin d'étude pour l'obtention du diplôme Master

Filière Informatique

Option

Génie des systèmes informatique

Thème

Conception et implémentation d'un algorithme de réduction de données d'apprentissage pour la classification d'alertes à base de KNN : Cas des colonies de fourmis

---

Réaliser Par :

MEZIANE Mahdi Ayoub
MEZIANE Idir

Promoteur :
BABA-ALI Ahmed Riadh

Encadreur :
MILOUD AOUIDATE Amal

Mr. Bennoucer
Mr. Sidoumou
Mlle. Moncer

Remerciement

Tout d'abord nous tenons à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

Nous tenons à remercier chaleureusement et respectivement tous ceux qui ont contribué de prés ou de loin à la réalisation de ce modeste projet de fin d'étude, à savoir notre promoteur Mr Baba Ali qui a dirigé notre travail, ses conseils et ses commentaires précieux nous ont permis de surmonter nos difficultés et de progresser au cours de notre stage.

Notre profonde gratitude et sincère reconnaissance à notre encadreur Mme MILOUD AOUIDATE Amal, doctorante a l'USTHB, de nous avoir encadré et pour nous avoir aidé chaleureusement tout au long de notre séjour au laboratoire.

Nous adressons nos respectueux remerciements à l'ensemble de nos professeurs à l'université de « SAAD DAHLEB» de Blida.

Dédicaces

Je dédie ce modeste travail à tous ceux qui me sont chers, qui m'ont aidée par leur soutien
moral et leurs conseils, en particulier :
À Mes très chers parents pour leur soutien, leur encouragement et leur présence à mes
côtés à chaque instant de ma vie.
À mon très cher frère Tarek et à ma très chère sœur Assia, pour leur
affection ,compréhension et patience.
À mes très chères amis qui étaient présents à chaque moment par leur aide et leur
sympathie : Ayoub et Hassan .
À tous mes amis(e) et camarades, et à tous ceux que j'aime.
À mon très chère binôme Mehdi avec qui j'ai passé d'agréables moments, que j'estime
énormément ainsi que toute sa famille.
À tous ceux que je connais de prés ou de loin.

Idir

Dédicaces

Je dédie cet humble travail,

Á mes très chers parents *et à mes frères qui ont toujours été là pour moi et qui m'ont supporté avec beaucoup de patience et gentillesse durant cette période.*

Á toute ma grande famille pour leur encouragement,

À mon très cher binôme Idir avec qui j'ai passé d'agréables moments.

À tous mes amis et camarades pour leurs soutient,

Á mes enseignants,

Á toutes les personnes qui me connaissent.

Mehdi

Résumé

La détection d'intrusion est étudiée depuis environ vingt-cinq ans. Les systèmes de détection d'intrusions (IDS) sont habituellement considérés comme une deuxième ligne de défense pour se protéger contre les activités malicieuses.

Cependant, il existe un problème difficile lié à l'utilisation des IDS. Ces derniers déclenchent généralement un grand nombre de messages d'alertes qui peuvent atteindre mille alertes ou plus par jour dont la plupart sont de fausses alertes.

Nous présenterons dans ce mémoire l'étude d'une conception et implémentation d'un algorithme de réduction de données d'apprentissage pour la classification d'alertes à base de KNN.

Il s'agit du réduire la taille de la base d'apprentissage pour un système de détection d'intrusions en utilisant l'algorithme d'optimisation par colonies de fourmis et l'algorithme d'apprentissage actif.

Table des matières

Chapitre 1: Introduction Générale

Chapitre 2: Etat de l'Art

Chapitre 3: Réduction à l'aide de l'optimisation par colonies de fourmis

Chapitre 4: Implémentation et Réalisation

Chapitre 5: Test et Validation

Table des figures

CHAPITRE 1

Introduction Général

Chapitre 1 : Introduction Générale

I. Introduction

Au cours de ce chapitre nous présenterons en premier lieu la situation actuelle, nous verrons ensuite quels sont les problèmes qui nous ont amenés à la réalisation de notre travail. Ensuite nous préciserons les principaux points à atteindre à la fin de ce travail. Finalement nous présenterons un aperçu sur l'organisation du mémoire en expliquant brièvement le contenu des chapitres qui suivront.

II. Situation actuelle

Le réseau informatique est considéré comme l'une des ressources les plus critiques dans une organisation.

Si les premiers réseaux de données se limitaient à échanger des informations reposant sur des caractères entre des systèmes informatiques connectés, les réseaux modernes ont évolué pour prendre en charge: le partage de ressources (fichiers, applications ou matériels, connexion à internet, etc.), la communication entre personnes (courrier électronique, discussion en direct, etc.) et la communication entre processus (entre des ordinateurs industriels par exemple), etc.

Les attaques informatiques constituent aujourd'hui l'un des fléaux de notre civilisation moderne. Il ne se passe plus d'une semaine sans que l'on apprenne que telle entreprise ou tel institut a essuyé de lourdes pertes financières en raison d'une déficience de la sécurité de son système d'information. Par conséquent les entreprises ne peuvent plus ignorer ces risques et se croire à l'abri de telles épreuves.
Le problème aujourd'hui n'est plus de savoir ***si*** une entreprise va se faire attaquer, mais plutôt ***quand*** cela va arriver.
En effet de plus en plus d'entreprises subissent des attaques qui peuvent entraîner des pertes conséquentes. Le besoin en sécurité informatique est devenu, par conséquent, de plus en plus important. Un élément essentiel d'une bonne politique de sécurité serait l'utilisation d'un IDS
L'objectif des systèmes de détection d'intrusion consiste à identifier les attaques ou les violations de sécurité issues de la surveillance du réseau et des activités hébergées. Afin de bien comprendre le fonctionnement de tels systèmes, il est nécessaire de présenter la technologie qui régit les systèmes de détection d'intrusions.

L'IDS est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (réseau, hôte).
Il peut être comparé à une alarme domestique contre les cambrioleurs, où si une tentative d'intrusion malveillante est découverte, une réponse sera alors déclenchée, dans ce cas plus les capteurs sont paramétrés, meilleur est le système, puisque chaque capteur est chargé de

détecter un type particulier d'activité (telle que l'ouverture des portes ou des fenêtres, une détection volumétrique[1] etc.).

La détection d'intrusions suit deux approchas principales :

- l'approche par scénario
- l'approche comportementale

La première qui est l'approche par scénario, s'appuie sur la connaissance des techniques utilisées par les attaquants pour déduire des scénarios typiques.
Elle ne tient pas compte des actions passées de l'utilisateur et utilise des signatures d'attaques (ensemble de caractéristiques permettant d'identifier une activité intrusive : une chaîne alphanumérique, une taille de paquet inhabituelle, une trame formatée de manière suspecte, etc.).

La deuxième qui est l'approche comportementale, consiste à détecter une intrusion en fonction du comportement passé de l'utilisateur. Pour cela, il faut préalablement dresser un profil utilisateur à partir de ses habitudes et déclencher une alerte lorsque des événements hors profil se produisent.

III. Enoncé du problème

La détection d'intrusions est un problème difficile que les chercheurs ont appréhendé, il y a maintenant plus de trente ans.
Le problème était assez simple à exprimer lorsque ces systèmes étaient des systèmes fermés, à l'intérieur d'un périmètre défini, avec des utilisateurs connus. Le problème s'est compliqué notablement depuis que les systèmes sont devenu sans fil et ouverts (le périmètre est presque impossible à cerner), que les utilisateurs sont devenu nomades (avec des entrants, des sortants et parfois des anonymes), que les services sont devenu mobiles et que ces systèmes s'étendent par interconnexion sur de grands espaces géographiques (infrastructure de télécommunications).

Pour contrer les tentatives d'intrusion, il faut définir un schéma d'identification, d'authentification et d'autorisation destiné à des utilisateurs légitimes et aux matériels et logiciels qui se situent à l'intérieur du système. Nous pouvons aider le système de détection d'intrusions (IDS) en filtrant au préalable le trafic régulier et légitime. On définit alors une sonde intelligente qui observe, enregistre et analyse le comportement des sujets, des objets et des événements. À partir d'un modèle de normalité et conformément à certains seuils, on décrète que le comportement du système et des personnes est normal ou suspect. On en déduit des actions à mettre en œuvre, immédiatement ou non.

Ces dispositifs ont un défaut majeur (fausse alerte). Une fausse alerte positive ou négative est une erreur de jugement du système de détection, qui va réagir et renvoyer un signal alors qu'il n'y avait pas lieu de le faire.
Notons aussi qu'en plus d'augmenter en nombre, les attaques que subit le réseau présentent de plus en plus un profil inconnu, rendant encore plus lente si ce n'est impossible leur détection et ainsi la possibilité de les contrer. Face à ces nouvelles attaques, on va adopter une approche de détection se basant sur la classification des attaques et non pas leurs signatures. Dans cette approche, le système de détection possède une base de connexions modélisant

[1] Alarme volumétrique : alarme qui se déclenche quand on change le volume de protection : en entrant dans le véhicule ou en bougeant dans le véhicule quand le système de détection est armé. L'alarme ne signale que les mouvements dans le véhicule.

différentes attaques subies. L'analyse consiste à rechercher la similarité à un motif caractéristique d'une attaque dans le flux d'événements.
Cette base d'attaques est immense vue le nombre important d'intrusions qui ne cessent d'augmenter et plus la taille est grande plus l'analyse et la vérification des intrusions sera lente. D'autre part, l'utilisation d'une base réduite peut engendrer des faux-négatifs (une intrusion réelle non détectée). Alors, il faut optimiser la taille de cette base en respectant les contraintes de précisions (taux de classification).

Soit :

I = Ensemble des instances de la base DARPA
I' = Ensemble des instances de la base DARPA réduite
P = Fonction de classification (La précision)
Objectif :
Obtenir I' tel que : $|I'| \leq |I|$ et $P(I') >= P(I)$

IV. Objectif général

Le but de ce travail est d'optimiser la taille de la base d'entraînement d'un classificateur IBR (Instance Based Reasoning), afin de permettre l'utilisation de ce dernier dans la détection d'intrusions. Cela en respectant les contraintes suivantes:

- *La Précision :* il est nécessaire de garder un taux de classification maximal et de minimiser le nombre d'alertes fausses-positives
- *La Rapidité de détection:* il est nécessaire d'avoir une détection assez rapide
- *Le Temps de réponse:* il est nécessaire de minimiser au maximum le temps de réponse.

V. Organisation du mémoire

L'organisation de ce document reflète la démarche que nous avons adoptée lors de la réalisation de ce travail. Il est composé de trois chapitres après l'introduction générale :

– le deuxième chapitre décrit les systèmes de détection d'intrusions (IDS), avec une classification générale des systèmes de détection d'intrusions selon plusieurs critères. Ainsi un exemple d'IDS sera présenté. Les dernières sections de ce chapitre seront consacrées à décrire quelques méthodes de classification existantes en mettant l'accent sur l'algorithme de classification KNN.

– Le troisième chapitre décrit l'architecture proposée et sa conception.

– Le dernier chapitre est consacré à l'implémentation et à la description du Système développé.

Nous présentons en dernier lieu une conclusion dans laquelle nous proposons un ensemble de perspectives à ce travail.

CHAPITRE 2

Etat de l'Art

Chapitre 2 : Etat de l'Art

Introduction

Dans ce chapitre nous allons présenter en première partie les Système de Détection d'Intrusion (IDS), ensuite dans la deuxième partie nous allons décrire l'algorithme de classification KNN ainsi que les approches de détection d'intrusions basées sur l'algorithme KNN. Enfin, dans la troisième partie nous allons présenter des algorithmes existants qu'on va l'utiliser dans la réduction de la base d'apprentissage des IDS.

I. Partie 1 : Les systèmes de détection d'intrusion

Afin de détecter les attaques que peut subir un système, il est nécessaire d'avoir un logiciel spécialisé dont le rôle serait de surveiller les données qui transitent, et qui serait capable de réagir si des données semblent suspectes. Plus communément appelé IDS (Intrusion Détection Systems), les systèmes de détection d'intrusions conviennent parfaitement pour réaliser cette tâche.

1. Définitions

Dans cette section nous allons définir quelques notions importantes dans ce travail.

1.1) Intrusion

Une intrusion est une action ayant pour but de compromettre, sur un système d'information la confidentialité, l'intégrité, la disponibilité et l'imputabilité des ressources.
Cette action peut avoir réussi, échoué, être en cours ou être une simple phase préparatoire.
Le système d'information visé peut être un réseau, une machine, un système d'exploitation une application [1].

1.2) Système de détection d'intrusion

Un système de détection d'intrusions (IDS, de l'anglais Intrusion Detection System) est un périphérique ou processus actif qui analyse l'activité du système et du réseau pour détecter toute entrée non autorisée et ou toute activité malveillante. La manière dont un IDS détecte des anomalies peut varier ; cependant, l'objectif principal de tout IDS est de prendre sur le fait les auteurs avant qu'ils ne puissent vraiment endommager nos ressources. Les IDS protègent un système contre les attaques, les mauvaises utilisations et les compromis. Ils peuvent également surveiller l'activité du réseau, analyser les configurations du système et du réseau contre toute vulnérabilité, analyser l'intégrité de données et bien plus. Selon les méthodes de détection utilisée, il existe plusieurs avantages directs et secondaires du fait d'utiliser un IDS.

1.3) Notions de base

faux-positif : détection en absence d'attaque
Alarme générée par un IDS pour un événement légal
faux-négatif : absence de détection en présence d'attaque
Non génération d'alarme par un IDS pour un événement illégal
Log : ligne d'un fichier d'un logiciel qui enregistre les données transitant sur un système pour le surveiller ou faire des statistiques.
Fichier log : contient les événements s'étant produits sur un système.
Evasion : Technique utilisée pour dissimuler une attaque et faire en sorte qu'elle ne soit pas décelée par l'IDS
Sonde : Composant de l'architecture IDS qui collecte les informations brutes
[2]

2. Architecture classique d'un IDS

L'IDS se compose de trois composants. Un capteur est chargé de collecter des informations sur l'évolution de l'état du système et de fournir une séquence d'événements qui traduit l'évolution de l'état du système. Un analyseur détermine si un sous-ensemble des événements produits par le capteur est caractéristique d'une activité malveillante.

Un manager collecte les alertes produites par le capteur, les met en forme et les présente à l'opérateur. Éventuellement, le manager est chargé de la réaction à adopter. Nous détaillerons par la suite chacun de ces trois composants.

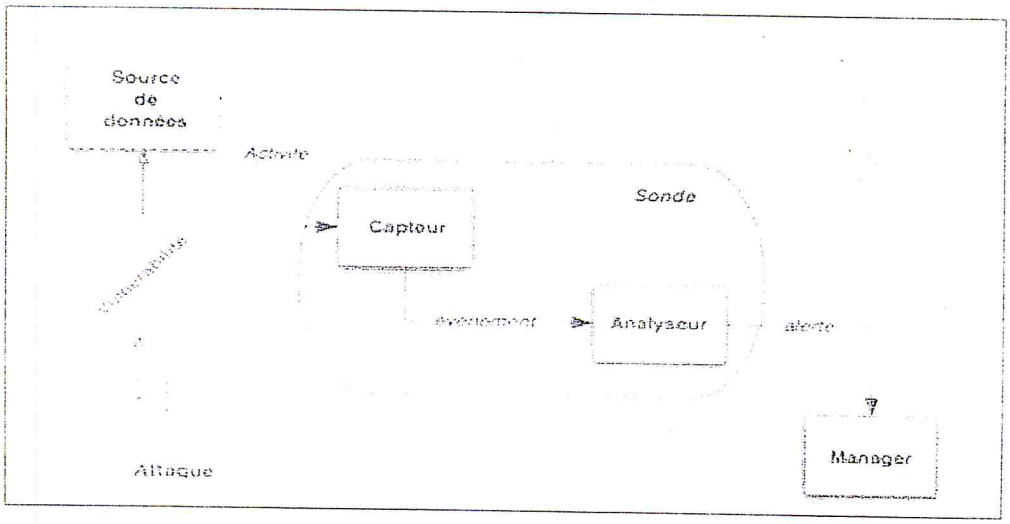


Figure 1: Architecture classique d'un IDS

2.1) Le capteur

Le capteur observe l'activité du système par le biais d'une source de données et fournit à l'analyseur une séquence d'événements qui informe de l'évolution de l'état du système. Le capteur peut se contenter de transmettre directement ces données brutes, mais en général un prétraitement est effectué.

-Les capteurs système qui collectent des données produites par les systèmes d'exploitation des machines, notamment par le biais des journaux d'audit système ou par celui des appels système invoqués par les applications. On désigne les IDS utilisant des capteurs système par l'acronyme HIDS (*Host-based IDS*).
- Les capteurs réseau qui collectent les données en écoutant le trafic réseau entre les machines par le biais d'une interface spécifique. On parle alors de NIDS (*Network-based IDS*).
- Les capteurs applicatifs qui collectent les données produites par une application particulière, avec laquelle des utilisateurs sont susceptibles d'interagir, comme un serveur web ou un serveur de base de données.

2.2) L'analyseur

L'objectif de l'analyseur est de déterminer si le flux d'événements fourni par le capteur contient des éléments caractéristiques d'une activité malveillante. Deux grandes approches ont été proposées : l'approche comportementale (*anomaly detection*) et l'approche par scénarios d'attaques (*misuse detection*) :

2.2.1- l'approche comportementale

Cette technique consiste à détecter une intrusion en fonction du comportement passé de l'utilisateur. Pour cela, il faut préalablement dresser un profil utilisateur à partir de ses habitudes et déclencher une alerte lorsque des événements hors profil se produisent. Cette technique peut être appliquée non seulement à des utilisateurs mais aussi à des applications et services.

Plusieurs métriques sont possibles : la charge CPU, le volume de données échangées, le temps de connexion sur des ressources, la répartition statistique des protocoles et applications utilisés, les heures de connexion, etc. [4]

Avantages et Inconvénients :

Dans cette partie nous allons présenter les avantages et les inconvénients de l'approche comportementale [1] :

Avantages

- Permet la détection d'attaques non connues a priori
- Facilite la création de règles adaptées à ces attaques
- Difficile à tromper

Inconvénients

- les faux-positifs sont relativement nombreux
- générer un profil est complexe :
  - durée de la phase d'apprentissage
  - activité saine du système durant cette phase ?
- en cas d'alerte, un diagnostic précis est souvent nécessaire pour identifier clairement l'attaque

2.2.2- l'approche par scénario

Cette technique s'appuie sur la connaissance des techniques utilisées par les attaquants pour déduire des scénarios typiques. Elle ne tient pas compte des actions passées de l'utilisateur et utilise des signatures d'attaques (ensemble de caractéristiques permettant d'identifier une activité intrusive : une chaîne alphanumérique, une taille de paquet inhabituelle, une trame formatée de manière suspecte, ...).

Les signatures d'attaques connues sont stockées dans une base, et chaque événement est comparé au contenu de cette base si un événement (paquet réseau, log système) correspond à une signature de la base, l'alerte correspondante est donnée par l'IDS [1].

Avantages et Inconvénients :
Dans cette partie nous allons présenter les avantages et les inconvénients de l'approche comportementale [1] :

Avantages

- simplicité de mise en œuvre
- rapidité du diagnostic
- précision (en fonction des règles)
- identification du procédé d'attaque
  - procédé
  - cible(s)
  - source(s)
  - outil
- facilite donc la réponse à incident

Inconvénients

- ne détecte que les attaques connues de la base de signatures
- maintenance de la base techniques d'évasion possibles dès lors que les signatures sont connues

2.3) Le manager

Le manager est responsable de la présentation des alertes à l'opérateur (fonction de console de management). Il peut également réaliser les fonctions de corrélation d'alertes, dans la mesure de leur disponibilité. Enfin, il peut assurer le traitement de l'incident, par exemple au travers des fonctions suivantes :

– confinement de l'attaque, qui a pour but de limiter les effets de l'attaque ;

– éradication de l'attaque, qui tente d'arrêter l'attaque ;

– recouvrement, qui est l'étape de restauration du système dans un état sain ;

– diagnostic, qui est la phase d'identification du problème, de ses causes et qui peut éventuellement être suivi d'actions contre l'attaquant (fonction de réaction). [3]

3. Les familles d'IDS

Un IDS est un ensemble de composants logiciels et matériels dont la fonction principale est de détecter et d'analyser toutes tentatives d'effractions (volontaire ou non). Il existe trois grandes familles distinctes d'IDS :

- Les NIDS (*Network Based Intrusion Detection System*), qui surveillent l'état de la sécurité au niveau du **réseau.**
- Les HIDS (*HostBased Intrusion Detection System*), qui surveillent l'état de la sécurité au niveau des **hôtes.**
- Les IDS hybrides, qui utilisent les NIDS et HIDS pour avoir des alertes plus pertinentes.

Les NIDS permettent de surveiller l'ensemble d'un réseau contrairement à un HIDS qui est restreint à un hôte.

3.1) Les systèmes de détection d'intrusions "réseaux" (NIDS)

Un NIDS surveille le réseau et tente de découvrir s'il y a une tentative d'intrusion ou de déni de service. Un NIDS peut être déployé soit sur la machine cible, afin de surveiller son propre trafic, soit sur une machine indépendante pour surveiller tout le trafic réseau. Un IDS basé réseau (NIDS) surveille le trafic en des points choisis sur un réseau ou un ensemble interconnectée de réseaux. Le NIDS examine paquet par paquet le trafic en temps réel ou en temps quasi-réel, pour tenter de détecter les modèles d'intrusions.
Le NIDS peut examiner le protocole au niveau réseau, transport et/ou au niveau application.
Quelques exemples de NIDS : NetRanger, NFR, Snort, DTK, ISSRealSecure.

3.1.1) Types de capteurs réseau

Les capteurs peuvent être déployés selon l'un des deux modes : actif ou passif.

Un capteur actif est inséré dans un segment du réseau de sorte que le trafic de ce segment doit passer par le capteur. On peut obtenir un capteur actif en combinant un capteur NIDS logique avec un autre périphérique réseau, tel qu'un pare-feu ou un commutateur. La raison principale de l'utilisation des capteurs actifs et de permettre le blocage d'une attaque lorsqu'elle est détectée. Dans ce cas, le dispositif effectue la détection d'intrusions et les fonctions de prévention.

Les capteurs passifs sont par contre les plus généralement utilisés. Ils surveillent une copie du trafic réseau, car le trafic réel ne passe pas par le dispositif.
Du point de vue circulation, le capteur passif est plus efficace que le capteur actif, car il n'ajoute pas l'étape supplémentaire de traitement qui retarde le paquet.
L'interface réseau du capteur passif est une carte réseau qui ne possède pas d'adresse IP, ce qui lui permet d'être invisible sur le réseau et donc de recueillir les informations sans aucune interaction avec le protocole réseau, ce capteur possède une seconde carte réseau comportant une adresse IP qui lui permet de communiquer avec le serveur de gestion du NIDS [15].

3.1.2) Architecture d'un NIDS

La figure 2 montre l'architecture standard d'un système de détection d'intrusion réseau.
Un capteur est utilisé pour "sniffer" les paquets du réseau ou ils sont introduits dans un processus de détection qui déclenche une alarme si une intrusion est détectée.

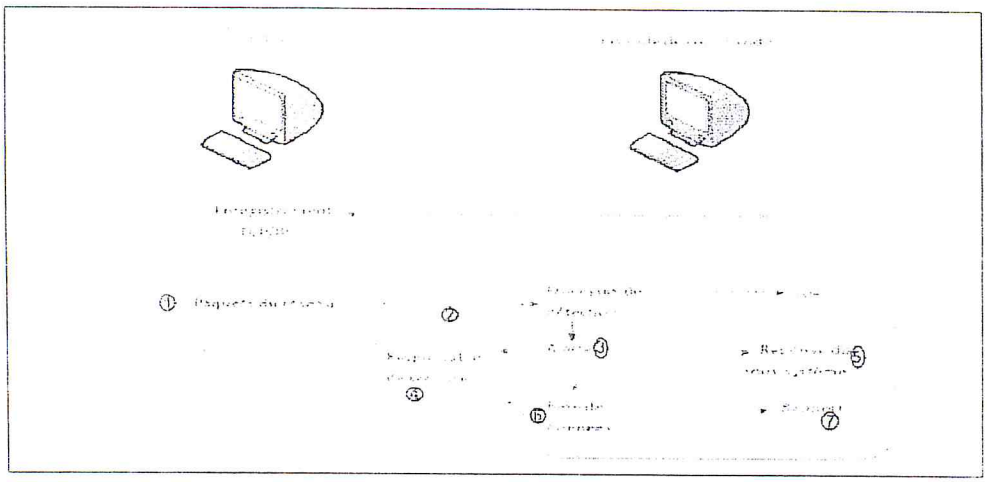


Figure 2: Architecture standard d'un NIDS

Ces capteurs sont distribués sur les différents segments critiques du réseau. Une console centrale est utilisée pour recueillir les alarmes provenant de multiples capteurs.
1. Le paquet réseau est créé quand un ordinateur communique avec les autres.
2. Le paquet est lu, en temps réel, grâce à un capteur placé sur un segment de réseau situé quelque part entre les deux ordinateurs communicants.
3. Le moteur de détection du capteur est utilisé pour identifier des modèles prédéfinis d'intrusions. Si un motif est détecté, une alerte est générée.
4. Le responsable de sécurité est informé des éventuelles intrusions. Cela peut être fait par diverses méthodes, notamment sonores, visuelles, e-mail, ou par toutes autres méthodes.
5. Une réponse à la mauvaise utilisation est générée. Le sous-système fait correspondre les réponses prédéfinies aux alertes ou peut prendre les réponses du responsable de sécurité.
6. L'alerte est stockée pour la corrélation et l'examen à une date ultérieure.
7. Les rapports résumant l'activité d'alertes sont générés.

3.1.3) Avantages et Inconvénients :

Dans cette partie nous allons présenter les avantages et les inconvénients du NIDS [5] :

- Avantages

Les IDS réseau possèdent plusieurs atouts, par exemple : les détecteurs peuvent être bien sécurisés puisqu'ils se "contentent" d'observer le trafic et permettent donc une surveillance discrète du réseau, les attaques de type *scans* sont détectés plus facilement grâce aux signatures et il est possible de filtré le trafic

- Inconvénients

Les NIDS sont très utilisés et remplissent un rôle indispensable, mais ils présentent néanmoins de nombreuses faiblesses En effet, la probabilité de faux négatifs (attaques non détectées comme telles) est élevée et il est difficile de contrôler le réseau entier. De plus, ils doivent principalement fonctionner de manière cryptée d'où une complication de l'analyse des paquets. Pour finir, à l'opposé des IDS basés sur l'hôte, ils ne voient pas les impacts d'une attaque.

- Limitations des NIDS

Outre les problèmes découlant d'attaques spécialement adaptées pour échapper ou nuire à l'IDS, la détection d'intrusion réseau est soumise à certaines contraintes. [1]

- flux chiffrés : analyse impossible
  - -IPsec
  - -Ssh
  - -https
- performances
  - -équipements adaptés au niveau du réseau
  - - perte de paquets
  - - machine de traitement suffisamment puissant
- la cible est-elle vulnérable à l'attaque
  Plusieurs politiques possibles :
  - On cherche à détecter toute attaque
  - On cherche à détecter les attaques potentiellement dangereuses, i.e. qui risquent d'aboutir

3.2) Les systèmes de détection d'intrusions de type hôte (HIDS)

Les systèmes de détection d'intrusion basés sur l'hôte (poste de travail, serveur, etc.), ou HIDS (*Host-based IDS*), analysent exclusivement l'information concernant cet hôte. Comme ils n'ont pas à contrôler le trafic du réseau mais "seulement" les activités d'un hôte, ils se montrent habituellement plus précis sur les variétés d'attaques. Ces IDS utilisent deux types de sources pour fournir une information sur l'activité : les *logs* et les traces d'audit du système d'exploitation. Chacun a ses avantages :
Les traces d'audit sont plus précises, détaillées et fournissent une meilleure information ; les *logs*, qui ne fournissent que l'information essentielle, sont plus petits et peuvent être mieux analysés en raison de leur taille. Il n'existe pas de solution unique HIDS couvrant l'ensemble des besoins, mais les solutions existantes couvrent chacune un champ d'activité spécifique, comme l'analyse de *logs* système et applicatifs, la vérification de l'intégrité des systèmes de fichiers, l'analyse du trafic réseau en direction/provenance de l'hôte, le contrôle d'accès aux appels système, l'activité sur les ports réseau, etc. [6]

3.2.1) Avantages et Inconvénients :

Dans cette partie nous allons présenter les avantages et les inconvénients duHIDS [6] :

Avantage

Les systèmes de détection d'intrusion basés sur l'hôte ont certains avantages :

- l'impact d'une attaque peut être constaté et permet une meilleure réaction;
- Les attaques dans un trafic chiffré peuvent être détectées (impossible avec un IDS réseau), - Les activités sur l'hôte peuvent être observées avec précision.

Inconvénient

Les HIDS présentent néanmoins des inconvénients, parmi lesquels :

- Les *scans* sont détectés avec moins de facilité ;
- Ils sont plus vulnérables aux attaques de type DoS
- l'analyse des traces d'audit du système est très contraignante en raison de la taille de ces dernières ;
- Ils consomment beaucoup de ressources CPU.

3.3) Comparaison entre NIDS et HIDS

Malgré la grande différence de déploiement entre le NIDS et le HIDS, leur comparaison reste possible, et, sur beaucoup de points, tels que [1] :

- la protection sur LAN : Les deux types d'IDS protègent contre les intrusions sur le LAN, mais seul le HIDS peut encore protéger le système des intrusions hors LAN, car il se trouve sur le hôte lui-même et donc filtre les paquets reçu par l'hôte qu'ils proviennent du LAN ou pas, contrairement aux NIDS se trouvant sur un segment du LAN, et qui donc ne filtrent que ce qui passe sur le réseau local.
- l'exigence en bande passante: Le NIDS est très exigent en bande passante qu'elle soit LAN ou WAN (internet), ce qui cause une grande surcharge du réseau, par contre le HIDS n'a besoin d'une bande passante que dans le WAN.
- Journalisation et taux d'échec : Les deux systèmes ont la fonctionnalité de journalisation, le taux d'échec NIDS est beaucoup plus élevé que le taux d'échec HIDS.
- Le déploiement : le déploiement de NIDS est généralement beaucoup plus simple que celui de HIDS, car pour le NIDS on utilise une sonde indépendante pour surveiller un ensemble de machines, ce qui veut dire il y'a peu d'impact sur les ressources du système surveillé. Par contre pour le HIDS une sonde pour un système à surveiller, ce qui veut dire l'utilisation d'une partie des ressources du système à surveiller.

- vulnérabilité : Un NIDS est généralement moins vulnérable qu'un HIDS, parce que le HIDS est sur la machine attaquée et si l'attaque réussie, elle peut aller jusqu'à stopper l'activité de l'IDS.
- L'activité de détection d'intrusion réseau (NIDS) est généralement transparente car on analyse une copie des paquets. Par contre, il est plus facile de repérer un HIDS (par le biais des fichiers logs par exemple)

- La détection d'intrusion réseau (NIDS) permet rarement de savoir si une attaque a abouti. Par contre, pour le HIDS on peut le savoir (par exemple si le processus visé est toujours actif ou non)

En conclusion Les rôles des NIDS et HIDS sont largement complémentaires tel qu'une architecture idéale allie les deux selon la nature des systèmes à surveiller et de l'activité qui leur est liée.

3.4) Les systèmes de détection d'intrusions "hybrides"

Généralement utilisés dans un environnement décentralisé, ils permettent de réunir les informations de diverses sondes placées sur le réseau.
Leur appellation « hybride » provient du fait qu'ils sont capables de réunir aussi bien des informations provenant d'un système HIDS qu'un NIDS.
[L'exemple le plus connu dans le monde Open-Source est Prelude.
Ce Framework permet de stocker dans une base de données des alertes provenant de différents systèmes relativement variés. Utilisant Snort comme NIDS, et d'autres logiciels tels que Samhain en tant que HIDS, il permet de combiner des outils puissants tous ensemble pour permettre une visualisation centralisée des attaques.]

[4].

4. Conclusion

De manière générale, l'efficacité d'un système de détection d'intrusion dépend de son "Adaptabilité" (Possibilité de définir et d'ajouter de nouvelles spécifications d'attaques), de sa robustesse (résistance aux défaillances) et de la faible quantité de faux positifs (fausses alertes) et de faux négatifs (attaques non détectées) qu'il génère.
Les IDS continuent d'évoluer pour répondre aux exigences technologiques du moment mais offrent déjà un éventail de fonctionnalités capable de satisfaire les besoins de tous les types d'utilisateurs.
Ils peuvent assurer une solide aide aux administrateurs de sécurité et nous donnent une bonne vision des attaques subies et ils permettent aussi d'évaluer le niveau de défense du réseau.
Ils peuvent assurer une solide aide aux administrateurs de sécurité et nous donnent une bonne vision des attaques subies et ils permettent aussi d'évaluer le niveau de défense du réseau.
Il est important de noter que **le risque nul d'être piraté n'existe pas** et qu'il faut s'avoir s'appuyer au mieux sur les outils (nouvellement) disponibles afin de tendre vers cet idéal.

"Se faire battre est excusable, mais se faire surprendre est impardonnable" N.Bonaparte

II. Partie 2 : Algorithme KNN

La faculté d'apprendre de ses expériences passées et de s'adapter est une caractéristique essentielle des formes de vies supérieures. Elle est essentielle à l'être humain dans les premières étapes de la vie pour apprendre des choses aussi fondamentales que reconnaître une voix, un visage familier, apprendre à comprendre ce qui est dit, à marcher et à parler.
L'apprentissage automatique (Machine Learning) est une tentative de comprendre et reproduire cette faculté d'apprentissage dans des systèmes artificiels. Il s'agit, très schématiquement, de concevoir des algorithmes capables, à partir d'un nombre important d'exemples (les *données* correspondant à "l'expérience passée"), d'en assimiler la nature afin de pouvoir appliquer ce qu'ils ont ainsi appris aux cas futurs.
La formulation du problème de l'apprentissage supervisé est simple : on dispose d'un nombre fini d'exemples d'une tâche à réaliser, sous forme de paires *(entrée, sortie désirée)*, et on souhaite obtenir, d'une manière automatique, un système capable de trouver de façon relativement fiable la sortie correspondant à toute nouvelle entrée qui pourrait lui être présentée.
Il y'a plusieurs types de problèmes auxquels l'apprentissage supervisé est appliqué, parmi ces taches on trouve la classification.
Dans la classification, l'entrée correspond à une instance d'une classe, et la sortie qui y est associée indique la classe. Par exemple pour le problème de détection d'intrusion, l'entrée serait le paquet provenant du réseau, et la sortie indiquerait s'il est normal ou intrusif.

1. Description de l'algorithme KNN

L'algorithme K Plus Proche Voisin (KNN de l'anglais : K NearestNeighbor) est le plus simple de tous les algorithmes d'apprentissage automatique. Il permet de faire une classification sans faire d'hypothèse sur la fonction de classe $y=f(x_1, x_2,\ldots, x_p)$ qui relie la variable dépendante' y' aux variables indépendantes $(x_1, x_2,\ldots, x_p)$. C'est une méthode de classification non-paramétrique puisqu'aucune estimation de paramètres n'est nécessaire.
On dispose de données d'apprentissage (training data) pour lesquelles chaque observation dispose d'une classe y. Si le problème est à 2 classes, y est binaire.
L'idée de l'algorithme des KNN est pour une nouvelle observation $(u_1, u_2,\ldots, u_p)$ de prédire les k observations lui étant les plus similaires dans les données d'apprentissage et utiliser ces observations pour la classer. Si on connaissait la fonction f, on aurait juste qu'à calculer $C=f(u_1, u_2,\ldots, u_p)$. Quand on parle de voisin cela implique la notion de distance. La distance la plus populaire est la distance euclidienne.
Le cas le plus simple est k=1 (cas 1-NN). On cherche l'observation la plus proche et on fixe C = y.
- L'extension de 1-NN à k-NN :
(1) Trouver les k plus proches observations
(2) Utiliser une règle d'observation

L'avantage est que de grandes valeurs de k produisent un lissage qui risque de sur-apprentissage Typiquement les valeurs de k sont quelques dizaines plutôt Si on choisit

L'avantage est que de grandes valeurs de k produisent un lissage qui risque de sur-apprentissage Typiquement les valeurs de k sont quelques dizaines plutôt Si on choisit K = n (nombre d'observations dans l'ensemble classe retenue sera celle qui a la majorité dans les données d'apprentissage, indépendamment de l'observation inconnue (u_1, u_2,..., u_p) [7].

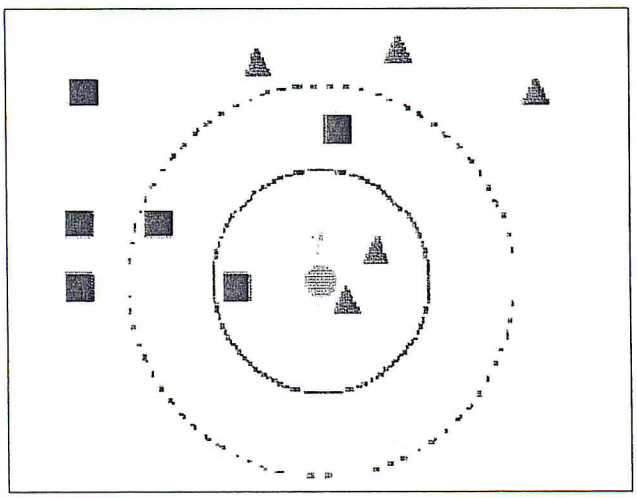


Figure 3: Classification KNN

La figure au-dessus présente un exemple du classement KNN. La prise d'essai (cercle) devrait être classée à la première classe de carrés ou à la deuxième classe de triangles.
Si k = 3, il est classé à la deuxième catégorie car il y a 2 triangles et 1 carré seulement dans le cercle intérieur.
Si k = 5, il est classé en première catégorie (3 carrés par rapport à 2 triangles dans le cercle extérieur).
On note deux aspects importants pour l'algorithme KNN :

- D'une part, à chaque nouvelle classification il est nécessaire de parcourir l'ensemble de la base d'apprentissage ce qui n'est pas nécessairement très efficace (surtout que, habituellement, on cherche à avoir la base d'apprentissage la plus grande possible afin d'avoir un meilleur classificateur).

- Et d'autre part, un point crucial de cet algorithme est la fonction de distance utilisée pour mesurer la proximité des objets. Il n'existe pas de distance/similarité universellement optimale et une bonne connaissance du problème traité guide généralement le choix de cette distance/similarité.

2. Distances et mesures de similarité

Les distances entre des caractéristiques ou des ensembles de caractéristiques sont au cœur des méthodes de classification et permettent les opérations élémentaires de transformation de caractéristiques. L'étude des différentes distances à notre disposition est par conséquent une étape incontournable. D'une manière générale nous chercherons à comparer deux vecteurs X et Y dont les composantes seront respectivement [x_0, x_1, ..., x_N] et [y_0, y_1, ..., y_N].
Avoir un outil quantitatif pour répondre à la question ***Est-ce que deux entités X et Y se ressemblent ?***
Lorsqu'on désire comparer des entités, on cherche à obtenir un scalaire indiquant la proximité de ces entités.

Lorsqu'on désire comparer des entités, on cherche à obtenir un scalaire indiquant la proximité de ces entités.
Une **distance** d sur un ensemble E est une application de E x E dans R^+ vérifiant les axiomes suivants:

- (P1) séparation: d(x, y) = 0 ⇔ x = y
- (P2) symétrie: d(x, y) = d (y, x)
- (P3) inégalité triangulaire: d(x, z) ≤ d(x, y) + d (y, z)

Les distances entre des caractéristiques ou des ensembles de caractéristiques sont au cœur des méthodes de classification et permettent les opérations élémentaires de transformation de caractéristiques. L'étude des différentes distances à notre disposition est par conséquent une étape incontournable. D'une manière générale nous chercherons à comparer deux vecteurs X et Y dont les composantes seront respectivement [x_0, x_1, ..., x_N] et [y_0, y_1, ..., y_N].
Avoir un outil quantitatif pour répondre à la question ***Est-ce que deux entités X et Y se ressemblent ?***
Lorsqu'on désire comparer des entités, on cherche à obtenir un scalaire indiquant la proximité de ces entités.
Une **distance** d sur un ensemble E est une application de E x E dans R^+ vérifiant les axiomes suivants:

- (P1) séparation: d(x, y) = 0 ⇔ x = y
- (P2) symétrie: d(x, y) = d (y, x)
- (P3) inégalité triangulaire: d(x, z) ≤ d(x, y) + d (y, z)

Distances de Minkowski (ou p-distance)
Les distances les plus répandues sont les simples distances de Minkowski dont sont extraites la distance euclidienne et la distance de Manhattan. On les calcule selon :

$$d(x,y) = \left(\sqrt{\sum_i |x_i - y_i|^p}\right)^{1/p}$$ [8].

Distance euclidienne (ou 2-distance) :

La distance euclidienne, = 2 est la plus connue ; couramment utilisée dans des espaces à 2 ou 3 dimensions, cette métrique donne des bons résultats si l'ensemble des données présente des classes compactes et isolées.

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$ [8].

Distance de Manhattan (ou 1-distance) :

$$d(x,y) = \sqrt{\sum_i |x_i - y_i|}$$

[8].

3. Algorithme de base

K-plus proche voisin est un algorithme d'apprentissage supervisé où le résultat est le classement d'une nouvelle instance selon la majorité des k-catégories des voisins les plus proches. Le but de cet algorithme est de classer un nouvel élément en fonction des attributs et des échantillons de formation (d'entrainement). Chaque attribut de la mémoire est sous forme de couples, composés d'entrées et de sorties, les entrées sont les valeurs des objets et les sorties leurs classes respectives.
L'idée de l'algorithme est de prédire pour une nouvelle observation les k observations lui étant les plus similaires, dans les données d'apprentissage, et d'utiliser ces observations pour affiner l'observation à une classe. La plus simple présentation de cet algorithme est exposée comme suit :

Arguments :
X : est notre objet inconnu
S : l'ensemble de tous les objets (l'espace d'étude)
Les Yi : sont les classes des objets de S
Les Yij : sont les objets de S appartenant à la classe Yi
d : est notre métrique (distance)
Entrées :X, S, d, k
Sorties :Y= classe de X
Début :
1. Pour chaque Yij dans S faire Calculer d (X, Yij)
2. Classer les distances par ordre croissant
3. Compter le nombre d'occurrences de chaque classe Yi parmi les k plus proches selon l'ordre
4. Assigner à X la classe la plus fréquente
5. Retourner Y
Fin

4. Avantages et Inconvénients

Dans cette section nous allons présenter les avantages et les inconvénients de l'algorithme KNN [9].

Avantage :

- Apprentissage rapide et incrémental
- Méthode facile à comprendre
- Se prête à la parallélisassions
- Adapter aux domaines où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières

Inconvénient

- Prédiction lente car il faut revoir tous les exemples à chaque fois.
  → Structuré la base d'apprentissage de manière à faciliter la recherche des voisins.
- Méthode gourmande en places mémoire
  → Réduire la base d'apprentissage :

  - éliminer les exemples non utiles (edited KNN)

  - condenser des groupes d'exemples sous forme de prototypes
- Sensible aux variables non pertinentes (ont autant d'influences sur la distance que les variables pertinente)
  → éliminé les variables non pertinentes

5. Problèmes de l'algorithme KNN

Il existe plusieurs problèmes pour cet algorithme dont nous allons présenter les principaux dans ce qui suit.

5.1) La taille de l'ensemble d'apprentissage

Les données provenant de problèmes d'apprentissage concrets réels apparaissent souvent en haute ou très haute dimension : c'est à dire qu'un grand nombre de variables ont été mesurées pour chaque exemple d'apprentissage [10], comme par exemple l'ensemble d'apprentissage d'un système de détection d'intrusion peut contenir des milliers ou des millions d'instances.
Avec le temps de nouvelles instances sont détectées et classées et plusieurs classes peuvent apparaître d'où la tache de l'algorithme devient de plus en plus complexe, d'une part le temps de calcul des distances entre les instances de l'ensemble d'apprentissage et la nouvelle observation sera coûteux, surtout si la valeur du k et très importante. D'autre part, avec le temps des prototypes d'apprentissage bruyants apparaissent dans l'ensemble d'apprentissage, ce sont des prototypes qui ne participent pas au classement des nouvelles observations ou qui ont un effet très léger sur le résultat final de l'algorithme ce qui nous emmène vers la réduction des données.
La réduction de données est un processus utilisée pour transformer les grands ensembles de données brutes en une forme plus condensée, sans perdre d'importantes informations.
La réduction des données se réfère à la sélection des prototypes (Prototype Selection: PS).
La sélection des prototypes est le processus de trouver des prototypes significatives qui peut aider à réduire le nombre de données en gardant une meilleur performance.

5.2) La valeur du K

La valeur du K est l'un des paramètres à déterminer car les résultats se diffèrent selon cette valeur.
Prenons cet exemple : On doit classer le nouvel élément *x*

- Si k=1, *x* sera classé dans la classe +
- Si k=5, *x* sera classé dans la classe –

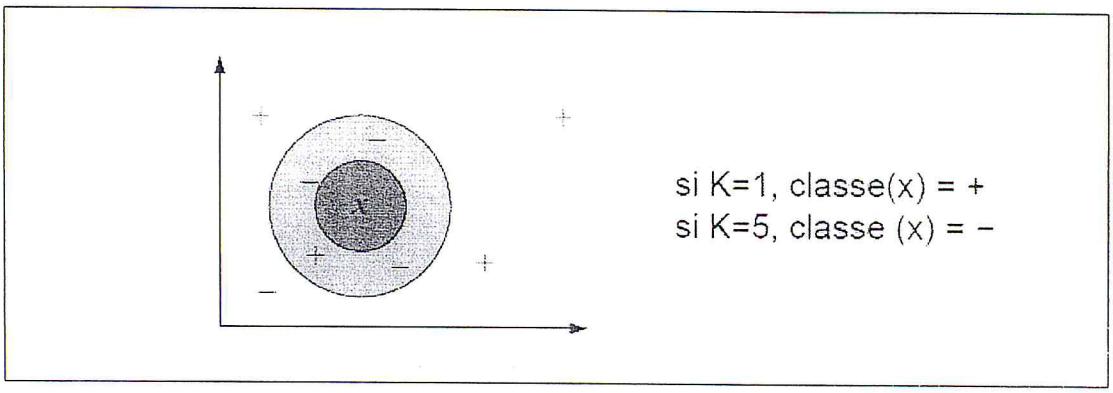


Figure 4: Choix du K

On voit que le résultat diffère selon la valeur du K, d'où l'importance de la détermination de cette valeur.

Les performances du KNN standard dépendent de la valeur du paramètre k ; si k est très petit l'algorithme devient sensible au bruit, si par contre k est trop grand l'aspect de localité devient important. Plusieurs problèmes requièrent différentes valeurs de k, en réalité nous devons déterminer k en rapport avec les données d'entraînement. L'utilisation de cette validation croisée pour optimiser k, introduit un autre problème, même pour un seul ensemble de données, la valeur optimale de k varie selon les régions de l'espace d'entrées, alors la validation croisée va sacrifier les performances sur quelques régions, pour avoir des performances générales acceptables.

6. Approches de détection d'intrusions basées sur l'algorithme KNN

La méthode proposée par Law et Kwok [11] consiste à déterminer si les séquences d'alarmes entrantes sont différentes de la situation normale, si oui, une alerte peut être signe d'attaque et une investigation plus approfondie est nécessaire, sinon le risque d'être attaqué est faible. Etant donné un grand nombre d'alarmes générées par un IDS dans un environnement sans attaques avec N types distincts d'alarmes il y a un ensemble distinct de points avec N attributs dans un espace représentant ce nombre d'alarmes de types différents dans une fenêtre temporelle. Pour détecter les modèles d'alertes anormales nouvellement arrivées, de nouveaux points de données sont créés pour ces nouvelles alarmes. La distance de ces points par rapport aux points normaux indique leur déviation de la situation normale. Le classificateur KNN a été utilisé pour classer les données et savoir si un point est normal ou non. KNN a d'abord été utilisé dans la zone de détection d'intrusions pour la détection

utilisé dans la zone de détection d'intrusions pour la détection d'anomalies d'apprentissage et la découverte du comportement du programme d'intrusion à partir des données d'audit. Les auteurs ont proposé de l'étendre à l'utilisation pour la détection de fausses alarmes en se basant sur la distance Euclidienne, qui représente la similitude entre deux points. Plus cette distance est petite plus les points sont similaires. Le processus de détection d'alarmes a utilisé les alarmes normales (alarmes relevées par l'IDS en cas d'aucune attaque) pour construire le modèle de fausses alarmes. Les alarmes entrantes sont filtrées en permanence par une procédure de filtrage utilisant ces modèles comme patrons. La modélisation proposée par les auteurs ainsi que les procédures de filtrage ont été indépendantes du processus de détection d'intrusions. Par conséquent, ce modèle peut être appliqué à la plupart des IDS commerciaux sans rien changer dans leur configuration.

Yang Li et Li Guo [12] ont proposé une nouvelle méthode de détection d'intrusions basée sur l'algorithme d'apprentissage automatique TCM-KNN, ainsi qu'une méthode de sélection de données d'apprentissage basée sur l'apprentissage actif.

Cette approche a été la première qui inclut le TCM-KNN dans la détection d'intrusions. Pour cette utilisation les auteurs ont optimisé le TCM-KNN sur deux aspects.

La première amélioration est l'introduction d'une méthode d'apprentissage actif afin de sélectionner un petit nombre de données mais de bonne qualité pour l'apprentissage, ce qui permet d'alléger la quantité de données étiquetées et de réduire la taille de l'ensemble d'apprentissage, et donc de réduire aussi le coût de calcul du TCM-KNN. Le second aspect a été la proposition d'une méthode de sélection d'attributs les plus importants et nécessaires pour le TCM-KNN. Les auteurs ont attribué à chaque point une mesure appelée mesure d'étrangeté individuelle. Cette mesure définie l'étrangeté du point par rapport au reste des points. La mesure d'étrangeté utilisée a été le rapport de la somme des k distances les plus proches de la même classe à la somme des k distances les plus proches de toutes les autres classes. Cette mesure croît lorsque la distance entre les points de la même classe augmente ou lorsque la distance entre les autres classes devient plus petite. L'algorithme comprend deux phases importantes : la phase de formation et la phase de détection. Dans la première phase trois actions principales doivent être considérées : la collecte des données pour la modélisation, la sélection d'attributs et enfin la modélisation par l'algorithme TCM-KNN et donc la construction du classificateur de détection d'intrusions. Pour la phase de détection toutes les données recueillies en temps réel du réseau doivent être prétraitées en étant vectorisées suivant les attributs sélectionnés.

Sur la base de leur précédente proposition (TCM-KNN), les mêmes auteurs ont présenté un mécanisme de sélection d'instance pour TCM-KNN basée sur EFCM (Extended CMeans Fuzzy) pour la détection d'anomalies. Ils ont introduit un algorithme de clustering, visant à limiter la taille des données de formation, réduisant ainsi le cout de calcul du TCM-KNN et améliorant le rendement de sa détection. Tout d'abord, les auteurs ont proposé de classer l'ensemble de données d'entraînement normal en introduisant trois classes : les données notables, des données obscures et les données redondantes. Les données notables ont été définies comme celles appartenant à une série de clusters. Ces données représentent ces clusters et y sont les plus centrées. Les données obscures désignent celles qui appartiennent à un cluster avec de très petits grades d'appartenance calculés par EFCM. Les données redondantes comprennent, quant à elles, les données restantes qui n'appartiennent à aucune des deux classes précédentes, et qui sont essentiellement les données qui se trouvent le long des limites des clusters. Ils ont ensuite proposé de former l'algorithme TCM-KNN avec les données notables et obscures, afin que ce dernier puisse offrir le meilleur usage possible en permettant de distinguer les anomalies des cas normaux avec un taux de détection élevé et un faible taux de faux positifs, ainsi qu'un cout de calcul réduit.

Liwei Kuang and Mohammad Zulkernine [13] ont proposé une méthode de détection d'anomalies : CSI-KNN en combinant l'étrangeté et la mesure d'isolation. L'algorithme de détection d'intrusions analyse différentes caractéristiques des données du réseau en employant deux mesures : l'étrangeté et l'isolation. La métrique d'étrangeté mesure si le comportement inconnu est plus semblable aux comportements normaux ou aux comportements anormaux. Elle effectue une détection d'anomalies en analysant la distribution d'étrangeté des données normales mais qui traite également les attributs d'anomalies en employant des données d'attaque. La métrique d'isolation identifie la similarité par rapport aux classes normales et peut détecter les attaques anormales. C'est une méthode pure de détection d'anomalies qui n'utilise que les données normales

Basée sur ces mesures, une unité de corrélation soulève les alertes d'intrusions et les estimations de confiance associées. Multiples classificateurs CSI-KNN travaillent en parallèle afin de traiter différents types de services réseau. Le composant de corrélation agrège les résultats de l'étrangeté et les modèles d'isolement et fournit une décision finale. En outre, l'algorithme fournit une confiance graduée pour chaque alerte d'intrusion.

III. Partie 3 : Algorithmes Existant

1. Clustering « DBSCAN »

Aujourd'hui, il y a plus en plus des applications dont la base de données esttrès grosse et les données apparaissent sous la forme d'un flux de données infini comme la surveillance de réseaux par exemple.

Pour exploiter efficacement des données massives, il faut trouver un traitement spécial qui réduit les données afin d'obtenir des informations nécessaires à partir de ces données. Il y a certaines méthodes pour ce faire, parmi eux on trouve la classification ou le clustering.

1.1) Notion de Clustering et de Cluster

Le clustering peut être décrit comme le regroupement d'objets en différents groupes, appelés *clusters*, de façon à ce que les données de chacun d'entre eux partagent des caractéristiques communes comme la proximité dans l'espace.

Un cluster est un ensemble d'éléments. Cet ensemble est distinct des autres. Donc chaque élément d'un cluster a de fortes ressemblances avec les autres éléments de ce même cluster, et doit être différent des éléments des autres clusters. C'est ce que l'on appelle : la forte similarité intra-classe, et la faible similarité inter-classe [15].

1.2) Les méthodes de clustering

On distingue trois grandes familles de clustering [15]:

3. Le subspaceclustering, dont le but est de cibler les clusters existant dans des sous-espaces de l'espace original.

1.3) DBSCAN

De nombreuses applications ont besoin d'une gestion de données spatiales tel que les SDBS (Spatial DatabaseSystems).
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) permet l'identification de classes, c'est à dire le regroupement des objets d'une base de données en sous-classes significatives comme il est montré dans la figure suivante :

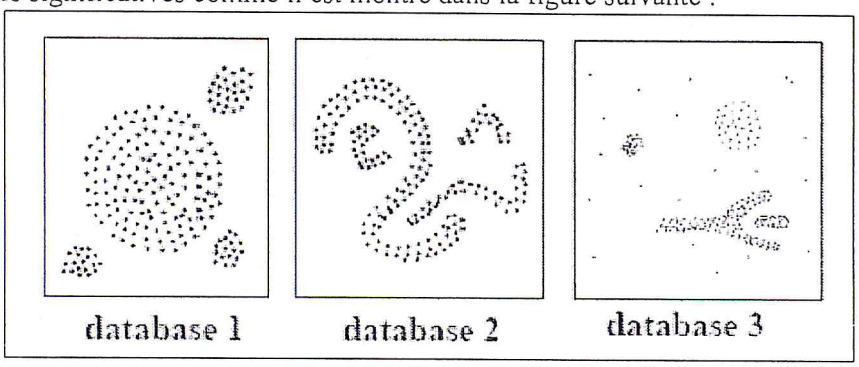


Figure 5: Clustering

On regardant ces regroupements on peut détecter les points qui appartiennent à un cluster et ceux qui n'appartiennent à aucun (bruit).

1.4) Algorithme DBSCAN

L'algorithme DBSCAN a pour but de de découvrir les clusters ainsi que le bruit (un ensemble d'objets qui n'appartiennent à aucun cluster) dans une base de donnée. Il cherche pour chaque objet le voisinage qui contient un nombre minimum d'objets, pour cela DBSCAN a besoin de 2 paramètres : Eps qui est le rayon de voisinage et MinPts qui est le nombre minimum de points qui doivent être contenus dans le voisinage.

L'idée de cet algorithme est que pour chaque point d'un cluster, ses environs pour un rayon Eps doit contenir un nombre minimum de point MinPts. C'est-à-dire pour une instance, on cherche tous ses voisins qui ont une distance inférieure à Epsilon. Si le nombre de ses voisins est au minimum égal à MinPts (le nombre des voisins supérieur ou égal à MinPts), cette instance est alors considérée comme faisant partie d'un cluster. On parcourt ensuite l'Epsilon-voisinage de proche en proche afin de trouver l'ensemble des points du cluster.

Cette forme de voisinage est déterminée par une fonction de distance entre deux points p et q comme la distance Euclidienne par exemple.

Cette forme de voisinage est déterminée par une fonction de distance entre deux points p et q comme la distance Euclidienne par exemple.

Voici l'algorithme DBSCAN [15]:

Pseudocode DBSCAN
La fonction DBSCAN commence avec un nœud non-visité, crée un nouveau groupe avecce nœud et appelle la fonction trouverGroupe pour regarder ses voisins

```
DBSCAN(D, eps, MinPts)
C = 0
        FORP dans l'ensemble D DO
           Marquer P comme visité
        N = prendreVoisin(P; Eps)
                ifsizeof(N) <MinPtsthen
                Marquer P comme BRUIT
                else
                C = prochain groupe
                Marquer P comme visité
                trouverGroupe(P, N, C, Eps, MinPts)
                end if
          END FOR
```

La fonction trouverGroupe qui est appelée par DBSCAN pour trouver les nœuds à joindrele groupe de P :

```
trouverGroupe(P, N, C, eps, MinPts)
Ajouter P au groupe C
Pour chaque point P' dans l'ensemble des voisins N
        ifP' n'est pas visité then
        Marque P' comme visité
        N' = prendreVoisin(P'; eps)
        ifsizeof(N') >= MinPtsthen
                N = N υ N'
        end if
        ifP' n'est pas un membre d'aucun groupe then
                Ajouter P' au groupe C
        end if
        end if
```

Pour trouver un cluster, DBSCAN commence par un point arbitraire p et cherche tous les voisins de p par le rayon Eps. Si le nombre de voisins de p par Eps est inférieur à MinPts, alors on marque p comme bruit. Sinon p est ajouté au cluster.

1.5) Critères de choix

On va citer les principaux critères pour qu'on ait choisi DBSCAN par rapport aux autres algorithmes de clustering :

- Rapide : DBSCAN est l'un des algorithmes de clustering les plus rapide et pour notre cas on utilise le clustering comme un prétraitement, alors c'est le critère le plus important. Dans la figure suivante une comparaison entre DBSCAN et CLARANS en fonction du temps :

number of points	1252	2503	3910	5213	6256
DBSCAN	3.1	6.7	11.3	16.0	17.8
CLAR-ANS	758	3026	6845	11745	18029
number of points	7820	8937	10426	12512	
DBSCAN	24.5	28.2	32.7	41.7	
CLAR-ANS	29826	39265	60540	80638	

Figure 6: Temps d'exécution comparé en secondes

- DBSCAN n'a pas besoin de connaitre le nombre de cluster à priori, ce qui est très important.
- DBSCAN prend en considération la notion de bruit.
- DBSCAN permet la découverte de cluster de forme arbitraire (sphérique, étiré, linéaire, allongé, etc...).
- Efficace sur les grandes bases de données.

2. Colonies de Fourmis

Dans ce chapitre nous allons présenter la méta-heuristique ACO « AntColonyOptimization » Cette méta-heuristique est relativement récente. Elle a été introduite en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994 [16].
Les métaheuristiques constituent une famille d'algorithmes inspirés de la nature. Ces algorithmes sont particulièrement utiles pour résoudre des problèmes où les algorithmes d'optimisation classiques sont incapables de produire des résultats satisfaisants. Parmi ces méthodes, les algorithmes de colonies de fourmis sont proposés pour l'optimisation des problèmes difficile. Ces algorithmes s'inspirent des comportements des fourmis observés dans la nature ; lorsque celles-ci sont à la recherche de nourriture.Si une fourmi découvre une source de nourriture, elle rentre au nid, en laissant sur son chemin une piste de phéromones. Ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre cette piste. En revenant au nid, ces mêmes fourmis vont renforcer la piste.
Dans ce qui suit nous allons détailler l'algorithme des colonies de fourmis.

2.1) Colonie de Fourmis

Dans cette section nous allons présenter le principe général de l'algorithme des colonies de fourmis.

2.1.1) Les pistes de phéromones

En marchant du nid à la source de nourriture et vice-versa (en premier temps se fait de façon aléatoire). Si une fourmi découvre une source de nourriture, elle rentre plus ou moins directement au nid, en laissant sur son chemin une piste de phéromones.. En effet, d'autres fourmis peuvent détecter les phéromones.
Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour et en revenant, ces mêmes fourmis vont renforcer la piste. D'autre part, les odeurs peuvent être utilisées par les autres fourmis pour retrouver les sources de nourritures trouvées par leurs congénères [1].
Les fourmis utilisent donc l'environnement comme support de communication : elles échangent indirectement de l'information en déposant des phéromones. Apres un certain temps, les pistes de phéromones vont s'évaporer.

7. La longue piste va finir par disparaître« l'évaporation ».
8. L'ensemble des fourmis a donc déterminé et « choisi » la piste la plus courte.

2.1.3) Communication

Les fourmis utilisent l'environnement comme support de communication : elles échangent indirectement de l'information en déposant des phéromones, le tout décrivant l'état de leur « travail ». L'information échangée a une portée locale, seule une fourmi située à l'endroit où les phéromones ont été déposées y a accès. Ce système porte le nom de « stigmergie ».

Exemple 1:

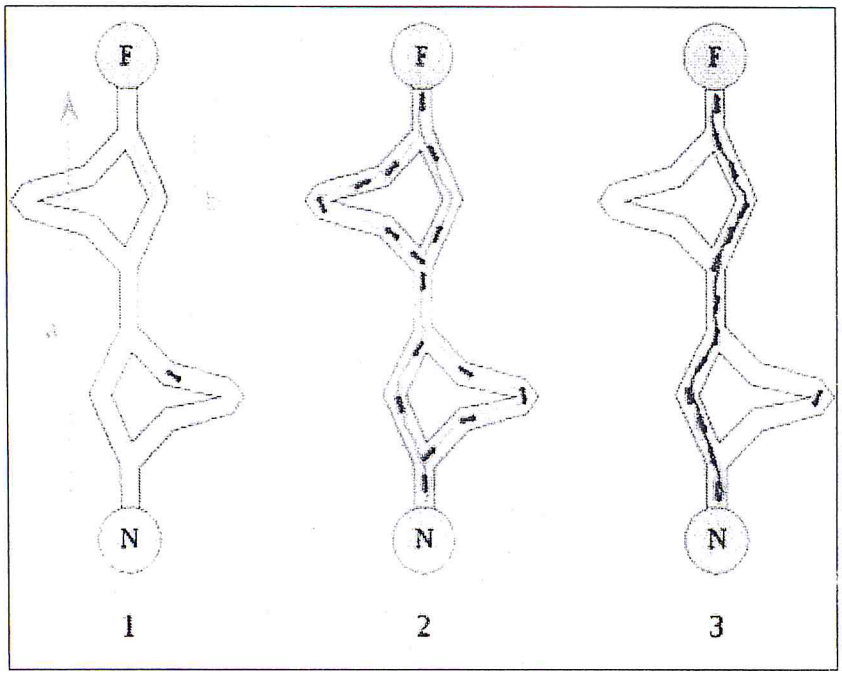


Figure 7: Le choix du plus court chemin grâce aux phéromones

1) la première fourmi trouve la source de nourriture (F), via un chemin quelconque (a), puis revient au nid (N) en laissant derrière elle une piste de phéromone (b).

2) les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.

3) les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones

Exemple 2 :

Les fourmis se déplacent entre le nid(E) et la source de la nourriture (A) sur le chemin A-E (a).

Un obstacle qui coupe le chemin (b).La fourmi qui se déplace de A vers E et se trouve en B, a 2 choix :

1. Le chemin B-C-D
2. Le chemin B-H-D

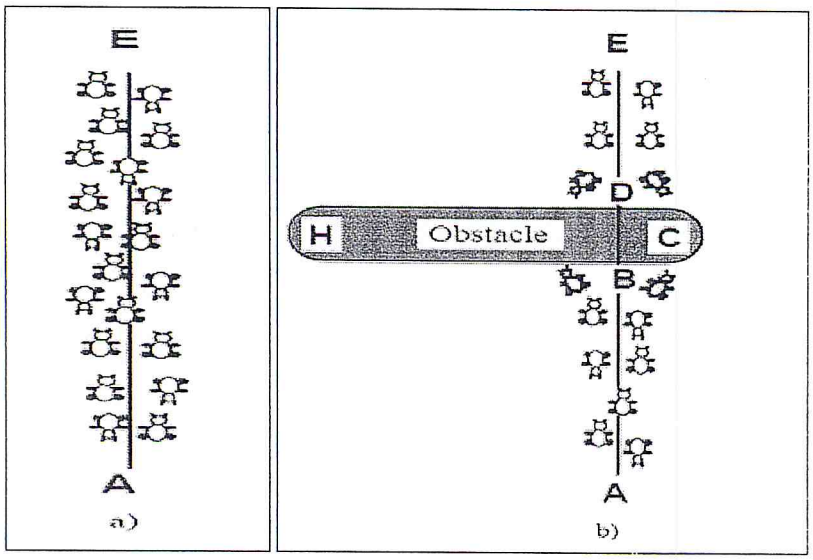


Figure 8: Un Obstacle coupe le chemin

- Le choix = f (intensités de phéromone)
- La première fourmi a des probabilités égales de suivre les deux chemins.
- Celle qui suit le chemin B-C-Darrive en premier en D.

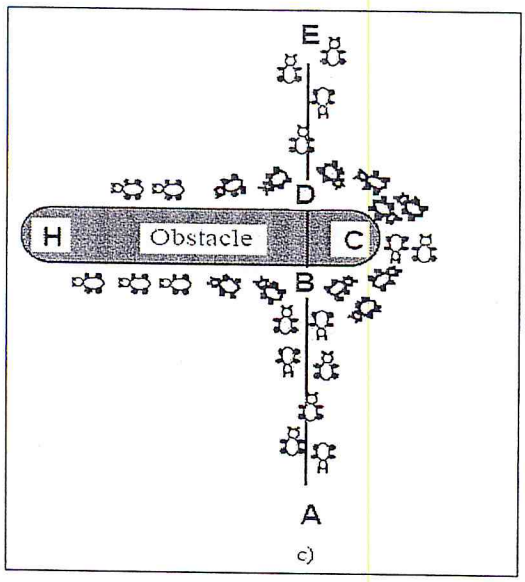


Figure 9: Le choix du nouveau chemin le plus court

- Les fourmis qui se retournent de E ont en D deux choix :

1. Le chemin D-C-B
2. Le chemin D-H-B

- Le chemin D-C-B aura une plus forte intensité de phéromone causé par :

1. La moitié des fourmis qui prennent ce chemin de retour.
2. Le nombre supérieur des fourmis qui ont suivi le chemin B-C-D et qui se retournent.

Conclusion

Dans ce chapitre nous avons présenté le fonctionnement de l'algorithme des colonies de fourmis qui est la partie la plus importantes dans notre approche.

3. Apprentissage Actif

L'apprentissage automatique est aujourd'hui devenu un outil très puissant pour la recherche d'information. Quelle que soit l'approche, elle comporte un inconvénient majeur : il faut beaucoup d'exemples. Or plus le modèle est riche, plus il faut d'éléments pour l'apprendre correctement et évaluer avec fiabilité ses performances et collecter ces éléments coûte cher en temps.
Il est donc primordial de minimiser aux maximum le nombre d'exemple d'apprentissage mais avec certaines garanties de performance.
L'une des approches pour résoudre ce problème est l'apprentissage actif.

3.1) L'apprentissage

L'apprentissage est une discipline à la frontière entre les mathématiques, les statistiques, l'informatique et la théorie de l'information. Son objectif est d'extraire et d'exploiter automatiquement l'information présente dans un jeu quelconque de données. Ces données sont de plus en plus abondantes et souvent de plus en plus complexes avec le développement des moyens informatiques, de télécommunication, des capteurs électroniques et autres compteurs. L'apprentissage consiste donc à développer, analyser et implémenter des méthodes qui permettent à une machine (au sens large) de traiter ces données et d'évoluer grâce à un processus d'inférence de décision issu de l'observation de celles-ci.
Le jeu de données initial sur lequel se base l'apprentissage possède un rôle important dans la capacité à bien apprendre [17].

On a deux types d'apprentissages :

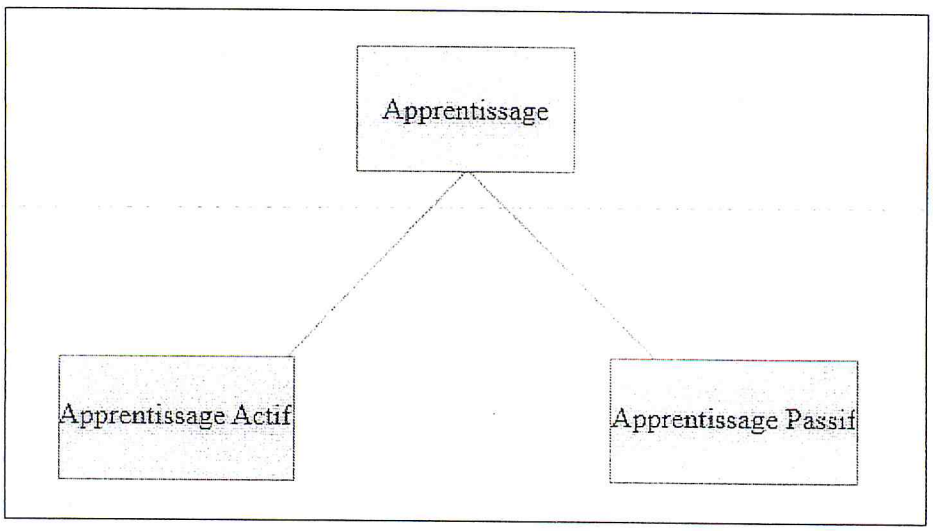


Figure 10: Type d'apprentissage

3.1.1) L'apprentissage Passif :

Il est constitué de 3 modes [17] :

- Apprentissage Non Supervisée : utilise des données non étiquetées. Dans ce cas, le modèle ne reçoit aucune information lui indiquant qu'elles devraient être ses sorties ou même si celles-ci sont correctes. Il doit donc découvrir par lui-même les relations existantes entre les exemples d'apprentissages.

- Apprentissage Semi-Supervisée :

Le mode d'apprentissage semi-supervisé manipule conjointement des données étiquetées et non étiquetées. Parmi les utilisations possibles de ce mode d'apprentissage, on peut distinguer le "clustering" semi-supervisé et la classification semi-supervisée :

- le "clustering" semi-supervisé cherche à regrouper entre elles les instances les plus similaires en utilisant l'information apportée par les données étiquetées. A l'issue de l'apprentissage, deux instances qui ont des étiquettes différentes n'appartiennent pas au même groupe.
- la classification semi-supervisée se base dans un premier temps sur les données étiquetées pour séparer les instances en fonction de leurs étiquettes. Ensuite, les données non étiquetées sont utilisées pour affiner le modèle prédictif.

- Apprentissage Semi-Supervisée : Lors d'un apprentissage supervisé, le modèle s'entraîne sur des données étiquetées. Pour ces exemples d'apprentissage, le modèle connaît la réponse attendue.

Un algorithme d'apprentissage est utilisé pour régler les paramètres du modèle en se basant sur l'ensemble d'apprentissage.

3.1.2) L'apprentissage Actif:

L'apprentissage actif permet au modèle de construire son ensemble d'apprentissage au cours de son entraînement, en interaction avec un expert (humain). L'apprentissage débute avec peu de données étiquetées. Ensuite, le modèle sélectionne les exemples (non étiquetés) qu'il juge les plus "instructifs" et interroge l'expert à propos de leurs étiquettes [17].

Supposant que les éléments de l'ensemble sur lequel on apprend, n'offrent pas tous le même degré d'information. Certains sont plus intéressants que d'autres. On va essayer donc d'identifier ces éléments important, c'est à dire qu'ils contiennent plus d'information que les autres.

3.2) Algorithme générale de l'apprentissage actif

Dans cette section nous allons présenter le schéma général de l'Active Learning :

Soit T L'ensemble global
Soit L inclut dans T l'ensemble des données avec les labelles
Soit U inclut dans T l'ensemble des données sans les labelles
Soit S le superviseur

- Pour chaque instance x de U :
-Le superviseur va décider s'il va ajouter x à L ou non
-Arrêter quand on a testé tous les éléments de U

On a l'ensemble L qui contient les instances avec les labelles et l'ensemble U contient les instances sans les labelles.
On essaye de choisir parmi tous les éléments de U ceux qui améliorent notre ensemble c'est-à-dire il réduit l'erreur et augmente la précision parce qu'on ne garde que les instances significatives et on les ajoute à l'ensemble L.
Enfin on va obtenir un ensemble réduit L et aussi performant que l'ensemble initial T.

La particularité de l'apprentissage actif réside dans leur capacité à interagir avec leur environnement. Son objectif est de sélectionner les exemples qui améliorent notre modèle tout en respectant des contraintes de performance.

Conclusion

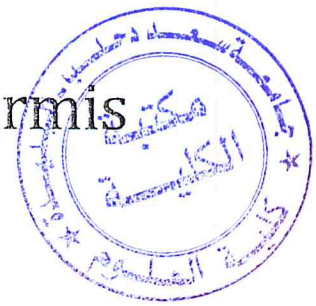
Dans ce chapitre, nous avons présenté les systèmes de détection d'intrusion (IDS), ensuite on décrit l'algorithme de classification KNN ainsi que les approches de détection d'intrusion à base de KNN, et enfin nous avons présenté trois algorithmes que nous allons utiliser dans notre approche.

Dans le chapitre suivant, nous allons présenter notre approche pour la réduction de la base d'apprentissage utilisée par les IDS.

CHAPITRE 3

Réduction à l'aide de l'optimisation par colonie de fourmis

Chapitre 3 : Réduction à l'aide de l'optimisation par colonies de fourmis



I. Introduction

Aujourd'hui et avec le développement de la technologie, les attaques ont augmenté d'une façon exponentielle. Afin de détecter les attaques que peut subir un système, il est nécessaire d'avoir une bonne politique de sécurité. Un élément essentiel d'une bonne politique de sécurité serait l'utilisation d'un IDS qui a pour rôle de détecter les attaques.

Pour réaliser sa tâche, l'IDS a besoin d'une base de données qui contient les modèles des paquets qui transitent sur le réseau. Parmi les bases de données on trouve la base KDD'99.

Ces bases de données sont gigantesques et contiennent des redondances et des instances bruités qui influent sur la classification, ces instances sont la cause des faux positifs (paquet normal classé comme attaque).

Notre approche consiste alors à réduire cette base de données en reprenant les instances les plus significatives, ce qui réduit les faux positifs et garde ou bien augmente le taux de classification.

II. Ensemble Train et Test

La base de d'apprentissage KDD'99 est un ensemble de modèles (on l'appelle aussi instance) signé comme normal ou attaque, chacune de ces instances contient 41 attributs, ils représentent les caractéristiques d'un état normal ou anormal.

La base dans son format original contient 4000000 instances et sa taille est de 700Mo.

Effectuer des tests sur cette base est très couteux en termes de temps d'exécution et nécessite des machines très performantes.

Une version réduite de la base de données originale est aussi disponible, elle contient 494020 instances (10% de la taille original), elle contient la même distribution des instances que la base original. Pour notre projet nous avons utilisé cette version réduite.

Notre travail consiste à réduire cette base en prenant les instances les plus significatives, cela va nous permettre de gagner du temps et de minimiser les erreurs causé par les instances bruitées.

Pour réaliser cela la base initial sera divisée en deux sous-ensembles, le premier sera nommer « Train » l'ensemble d'entrainement. Il contiendra des instances choisis aléatoirement et sans remise de la base de données initial, le second se nommera « Test », il contiendra le reste des instances (voir la figure suivante).

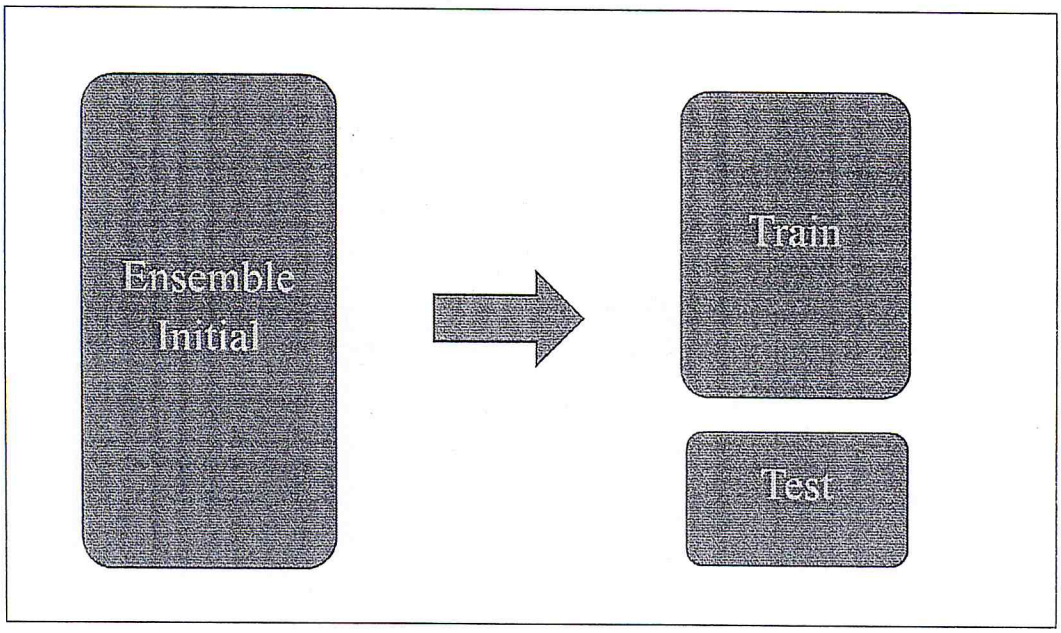


Figure 1: Ensemble Train et Test

III. Algorithmes

Notre travail va être divisé en 3 étapes :

- Le Clustering DBSCAN (prétraitement)
- Les Colonies de fourmis
- L'Active Learning

1. DBSCAN

Le clustering est un prétraitement dans notre travail, c'est juste pour déterminer les instances qui serviront comme un point de départ pour l'algorithme des colonies de fourmis. DBSCAN est l'un des algorithmes de clustering les plus rapides et précis c'est pour cela nous l'avons choisi.

Nous avons utilisé DBSCAN pour générer les clusters (regrouper chaque ensemble d'instance qui ont les mêmes caractéristiques dans un seul groupe) puis on prend aléatoirement une instance de chaque cluster et aussi une instance des NOISE (des instances n'appartenant à aucun des clusters, généralement ce sont des attaques rare), sur lesquelles on posera une fourmi pour débuter notre l'algorithme.

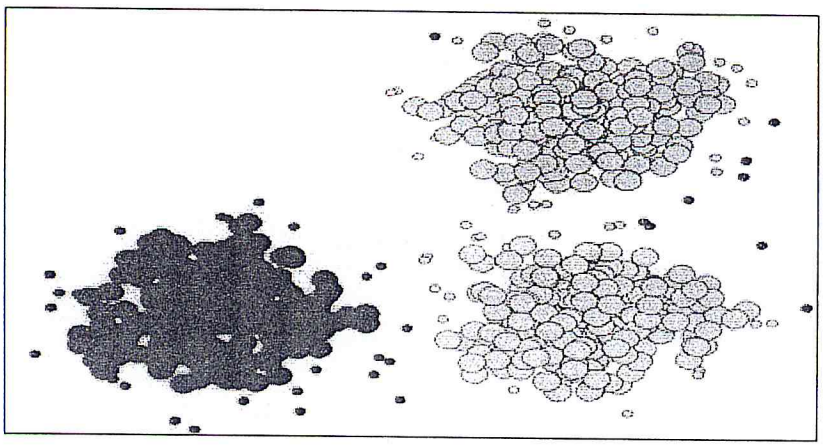


Figure 2: Clusters DBSCAN

2. Colonies de Fourmis

Dans cette section nous allons expliquer l'algorithme d'optimisation avec la colonie de fourmis que nous avons utilisé.

Notre approche consiste à travailler sur la base d'apprentissage utilisé par l'IDS pour classer les différents paquets interceptés. On cherche à réduire cette base tout en gardant ou améliorant les performances existantes (le taux de classification et le faux positif).

Le principe général de l'algorithme est qu'on doit démarrer des instances fournis par DBSCAN et que chaque fourmis va nous générer une solution.

Dans ce qui suit nous allons détailler notre algorithme :

V : la matrice Vue [Nombre d'instances, Nombre d'instances]

Ph : la matrice phéromone [Nombre d'instances, Nombre d'instances]

Mat : Matrice principal [Nombre de cluster, Nombre d'instances]

Au début nous : -Calculons la matrice vue : (1/distance)

-Initialisons la matrice de phéromone

-Initialisons la matrice principale (avec des -1)

Pour chaque fourmi i faire :

(1) Choisir le $V_i * Ph_i$ le plus grand

Choisir le facteur alpha aléatoire (0 ou 1)

Si alpha = 0 Alors

- mettre un 0 dans la case correspondante dans la matrice principale et aller vers (1);

Si alpha =1 Alors

- mettre un 1 dans la case correspondante dans la matrice principale (Mat_i) ;

-augmenter la valeur de phéromone dans la case correspondante (Ph_i) ;

- déplacer vers l'instance choisi et aller à 1;

On va continuer de boucler jusqu'aucun (-1) n'existe dans la matrice principale.

Description

Cet algorithme sera appliqué sur l'ensemble « Train ».

Au début nous initialisons la matrice phéromone (avec un taux de 0.3 dans toute la matrice et la diagonale avec des 0) qui sera partagé entre toutes les fourmis, nous calculons la matrice vue (1/Distance euclidienne) et nous initialiserons la matrice principale (avec des -1).

Matrice Phéromone :

Nombre d'instances

0	0.3	0.3	0.3	0.3
0.3	0	0.3	0.3	0.3
0.3	0.3	0	0.3	0.3
0.3	0.3	0.3	0	0.3
0.3	0.3	0.3	0.3	0

Nombre d'instances

Matrice Vue :

Nombre d'instances

1/D	1/D	1/D	1/D	1/D
1/D	1/D	1/D	1/D	1/D
1/D	1/D	1/D	1/D	1/D
1/D	1/D	1/D	1/D	1/D
1/D	1/D	1/D	1/D	1/D

Nombre d'instances

Matrice Principale :

Nombre de Clusters

(Nombre de fourmis)

-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1

Nombre d'instances

Chaque fourmi choisira l'instance sur laquelle elle va se déplacer selon la formule de sélection suivante :

La plus grande valeur de phéromone * Vue (V_i * Ph_i le plus grand)

C'est à dire on démarre du principe que le plus proche est le meilleur.

Ensuite on génère le facteur aléatoire alpha qui prendra 0 ou 1:

Si alpha=0 :

L'instance ne sera pas prise et ne sera pas visité prochainement, alors on met un 0 dans la matrice principale et on fera une nouvelle sélection sans prendre en considération les instances déjà traité.

Si alpha=1 :

On met un 1 dans la matrice principale ce qui signifie que l'instance sera prise, on augmente le taux de phéromone avec cette instance là pour laisser une trace (c'est-à-dire que cette instance aura une priorité d'être choisi par rapport aux autres) et on avance à la prochaine instance.

On refait le même travaille jusqu'à ce que il ne reste aucun (-1) dans la matrice principale (c'est-à-dire que toutes les instances seront traités).

Après n itérations on décrément le taux de phéromone pour toute la matrice (c'est l'action d'évaporation).

En résultat nous aurons la matrice principale qui contiendra les solutions générés par chaque fourmi (la ligne est une suite de 0 et de 1 donc si la i^{eme} colonne est égale à 1 alors la i^{eme} instance est prise dans l'ensemble sinon elle ne sera pas prise).Apres on va passer à l'étape d'évaluation.

Evaluation des résultats

Pour l'évaluation nous avons utilisé le *k*-plus proche voisin (KNN) que nous l'avons présenté dans le chapitre 2 (partie Algorithme KNN). Son rôle donc est de classer l'ensemble « Test » avec l'ensemble « Train ».

Pour cela il y'a deux importants paramètres à déterminer :

- La valeur du *K*
- La fonction de distance à utiliser

La distance Euclidienne est la fonction de distance la plus utilisé. Elle considère l'instance comme un vecteur de taille fixe et elle calcule la distance entre deux instances.

La valeur du K est un paramètre très crucial et il a une influence sur le résultat de la classification. Pour classer une instance X, on va déterminer les K plus proches voisins ensuite on choisira la classe la plus fréquentes parmi ses K voisins. Cette dernière représente la classe de X.

Nous allons classifier l'ensemble « Test » avec chacun des ensembles résultant de l'algorithme fourmis (chaque ligne de la matrice principale qui représente l'ensemble « Train » réduit) en utilisant le 1-NN c'est-à-dire, pour chaque instance du « Test » on calcule la distance Euclidienne avec toutes les instances de l'ensemble « Train » et on choisira la plus proche instance (celle qui a la plus petite valeur). L'instance du Test va être classée avec la classe de sa plus proche voisine. Si la classe de l'instance Test et sa plus proche voisine est la même (une bonne classification) un compteur est augmenter (ClasseJuste++).

Le taux de classification TC est calculé comme suit :

TC = (ClasseJuste÷ Taille de l'ensemble Test) × 100

Apres l'évaluation de tous les ensembles solution, nous choisirons l'ensemble qui a le meilleur taux de classification (meilleur ensemble).

3. Apprentissage Actif (Active Learning)

Apres que nous avons expliqué l'algorithme de fourmis, dans cette section on va présenter la dernière partie de notre approche qui est l'Active Learning.

Nous allons utiliser l'Active Learning avec l'algorithme de colonie de fourmis pour réduire la taille de la base d'apprentissage utilisé par un IDS pour améliorer ses performances.

Le principe de l'algorithme est de démarrer d'un petit ensemble et on augmentant au fur et à mesure cet ensemble on ajoutant les instances les plus significatives (qui améliorent ce dernier).

Voilà l'algorithme :

TF : Taux voulu

L: Ensemble d'instances (résultant de DBSCAN)

T: Ensemble Train (avec les labelles)

S: Ensemble Test (sans les labelles)

TC : Taux de classification

Début :

Tant que Evaluer (L, S) < TF fait

Pour chaque fourmi démarrant de Lfaire

-Sélectionner une instance i de l'ensemble T (en suivant la même méthode de sélection de l'algorithme de fourmi)

-Ajouter cette instance i à L

- TCi =Evaluer (L, S)

- Enregistré L'instance i et TCi.

- Retirer l'instance i de L

Fin pour

Ajouter l'instance qui donne le meilleur taux de classification TC définitivement è L.

Fin Tant que

On continuera d'exécuter ce processus jusqu'à la fin des instances de l'ensemble train

Description

Au début nous avons 3 ensembles :

-L'ensemble « Train » qui contient des instances avec leurs labelles

-L'ensemble « Test » qui contient des instances sans leurs labelles

-L'ensemble « L » qui contient les instances généré par DBSCAN

Nous allons fixer un taux qui va être une condition d'arrêt c'est-à-dire si notre ensemble L arrive à ce taux on va arrêter et retourner L.

A partir de chaque instance de L une fourmi va démarrer et elle va choisir où est ce qu'elle ira (avancer d'un pas).Apres le choix nous allons évaluer l'ensemble L avec chacune des

instances choisis (c'est-à-dire classer l'ensemble « Test » avec le nouvel ensemble L).On ajoute ensuite l'instance qui améliorent le plus l'ensemble L (qui nous donne le meilleur taux de classification).

On répète ce processus jusqu'à ce que le taux fixé soit atteint ou bien lorsqu'on arrive à la fin de l'ensemble Train.

IV. Conclusion

Dans ce chapitre, nous avons présenté notre approche pour la réduction de la base d'apprentissage qui est basée sur l'algorithme des colonies de fourmis ainsi que l'apprentissage actif

Dans le chapitre suivant, nous allons présenter les étapes d'implémentation et de réalisation de notre approche.

CHAPITRE 4

Implémentation et réalisation

Chapitre 4 : Implémentation et Réalisation

I. Introduction

Après avoir décrit notre solution, nous aborderons dans ce chapitre la partie implémentation de notre système. En première partie, nous présenterons l'environnement de travail et les outils de développement utilisés. Ensuite, nous détaillerons l'implémentation des différents modules constituant notre système et les différents échanges entre ces modules. Enfin, nous présenterons notre application ainsi que son fonctionnement.

II. Environnement de développement

Dans cette section nous présenterons les différents outils utilisés pour réaliser notre solution.

1. Langage de développement

Le choix du langage représente une étape très importante dans la réalisation de n'importe quelle application. C'est dans cette étape qu'on fait la correspondance entre les solutions que le langage nous offre et les résultats souhaités.

Les modules conçus ont été réalisés sous Java dont les principales propriétés, notamment celles qui concernent notre projet, sont résumées dans les points suivants :

Pour implémenter notre solution nous avons utilisé JAVA dont les principales propriétés, notamment celles qui concernent notre projet, sont résumées dans les points suivants :

- Java est un langage orienté objet: il est centré complètement sur les objets et fournit un ensemble prédéfini de classes facilitant la manipulation des entrées-sorties, de la programmation réseaux et de l'interface utilisateur.
- Le langage Java est distribué, il est conçu pour développer des applications en réseau. Les manipulations des objets distants ou locaux se font de la même manière.
- Sécurité: La sécurité dans Java est un aspect primordial, il ne supprime pas tous les problèmes de sécurité mais les réduits fortement.

- Le langage Java est portable: Le code développé en Java est indépendant des plateformes, pourvu qu'il existe une machine virtuelle pour chaque plateforme de travail (JVM : machine virtuelle Java).

Nous avons utilisé un éditeur de Java appelé Netbeans. C'est un environnement de développement intégré (Integrated Development Environnement).On peut y écrire, compiler et exécuter les programmes dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques, et aussi un outil libre permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

2. Outils de développement

- La librairie JPCAP :

JPCAP est une bibliothèque open source composée d'un ensemble de classes Java pour capturer et envoyer les paquets du réseau, elle est basée sur Libpcap/Winpcap. Elle facilite :

– La capture des paquets bruts directement du réseau.

– Le sauvegarde des paquets capturés dans un fichier, et la lecture des paquets capturés à partir d'un fichier.

– L'identification automatique des types de paquets et produit les objets correspondants à Java (pour l'Ethernet, les paquets IPv4, IPv6, ARP/RARP, TCP, UDP, et ICMPv4).

– Filtration des paquets selon des règles personnalisées par l'utilisateur avant de les expédier à l'application.

– Envoie des paquets bruts dans le réseau.

La librairie JPCAP doit être installée sur les deux machines, afin de capturer l'ensemble des paquets réseau (sonde et manager).

- Weka (Waikato Environment for Knowledge Analysis):

C'est un ensemble d'outils permettant de manipuler et d'analyser des fichiers de données, implémentant la plupart des algorithmes d'intelligence artificielle. Il est écrit en java, disponible sur le web, et s'appuie sur le livre

Data Mining, practical machine learning tools and techniques with Java implementations
Witten & Frank
Editeur : Morgan Kauffman

Il se compose principalement :

- De classes Java permettant de charger et de manipuler les données.
- De classes pour les principaux algorithmes de classification supervisée ou non supervisée.
- D'outils de sélection d'attributs, de statistiques sur ces attributs.
- De classes permettant de visualiser les résultats.

On peut l'utiliser à trois niveaux :

- Via l'interface graphique, pour charger un fichier de données, lui appliquer un algorithme, vérifier son efficacité.
- Invoquer un algorithme sur la ligne de commande.
- Utiliser les classes définies dans ses propres programmes pour créer d'autres

méthodes.

- La Librairie JFreeChart :

C'est une librairie Java gratuite permettant de créer des graphiques et des diagrammes de très bonne qualité. Cette API est open source et sous licence LGPL. Conçu pour être inclus dans des applications, applets, servlets et JSP. JFreeChart est distribué avec les codes sources complets sous les termes d'une License GNU Lesser General Public Licence, ce qui permet à JFreeChart d'être utilisé dans des applications libres ou propriétaires.

Exemples de graphiques disponibles sous Jfreechart :

- Graphiques
- Diagramme camembert
- Diagramme de Gantt
- Histogrammes

Thermomètres, compas, compteur de vitesse, etc…

III. Concept Utilisé

Comme il est expliqué dans le chapitre précédent, notre approche a pour but de réduire la taille de la base d'apprentissage utilisée par l'IDS pour classer les paquets interceptés et pour le faire, nous allons utiliser l'algorithme des colonies de fourmis avec l'apprentissage actif ainsi que l'algorithme KNN (pour la classification).

1. Notion Utilisé

Pour débuter, nous définirons quelques concepts qui seront utilisés durant ce chapitre :

a. Les ensembles Train et Test

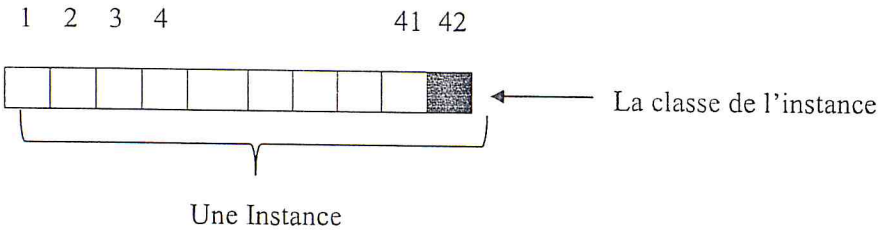
Pour réaliser notre solution, la base de données initiale sera divisée en deux sous-ensembles, le premier qui se nommera « Train » ou l'ensemble d'apprentissage, c'est celui qu'on va réduire. Le second se nommera « Test » qui sera utilisé pour évaluer notre ensemble résultant.

b. Instance

Les bases de données utilisées par les IDS appelées aussi « Benchmarck » contiennent des instances qui sont des modèles de paquets qui transitent dans le réseau.

Chaque instance est composée de deux parties :

- Les attributs qui définissent l'instance : ce sont les 41 premiers attributs qui sont présente dans une instance de l'ensemble « Train » ou « Test »
- La classe de l'instance : c'est l'attribut numéro 42, il indique si cette instance est considérée comme un comportement normal ou anormal du système.

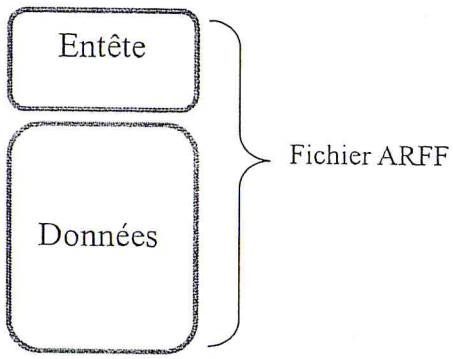


Plusieurs bases d'apprentissage existent, parmi le plus utilisés on trouve KDD CUP 99.

KDD CUP 99 est un ensemble qui regroupe une version des données collectées dans DARPA'98, il contient 4.900.000 instances signé comme normal ou attaque.

En général la taille de ces base d'apprentissage est très grande par exemple la KDD CUP 99 contient 4.900.000 instances et elle dépasse les 700Mo. Pour pouvoir simplifier l'utilisation et la visualisation des données présentes dans la base, nous avons utilisé le logiciel WEKA dans sa version 3.6 qui comporte son propre format de fichier appelé ARFF (.arff).

c. Fichier ARFF



Le format ARFF se caractérise par un entête qui définit les différents attributs de l'instance ainsi que leur type et les valeurs que peuvent prendre.

Dans tout ce qui suit nous avons utilisé ce type de format (.ARFF) pour représenter nos fichiers. Etant donné que le langage JAVA ne supporte pas le format ARFF, il s'avère nécessaire d'utilisé les classes fourni par WEKA dans 3 parties :

- la lecture des fichiers
- le calcul de la distance euclidienne
- le clustering (DBSCAN)

2. La lecture d'un fichier ARFF

Vu qu'on travaille sur des fichiers de grand taille, et comme la lecture des fichiers Texte (.txt) en java est trop lourde en matière de temps d'exécutions, d'espace mémoire et de manipulation de données ; nous avons utilisé la classe weka.core.converters.ArffLoader pour lire les fichiers en format .ARFF.
Au début il faudra importer les deux packages suivant :

```
import weka.core.Instances;
import weka.core.converters.ArffLoader;
```

Figure 1: Packages Weka pour lire un fichier ARFF

La classe « weka.core.converters.ArffLoader » permet de converti le contenu texte du fichier ligne par ligne (la ligne représente une instance), en un objet de types weka.core.Instances facile à manipuler et plus léger comme il est montré dans la figure suivante :

```
ArffLoader loder = new ArffLoader();
Instances DN= null;

        loder.setFile(new File(chemin));

        DN = loder.getDataSet();
```

Figure 2: Lire un fichier ARFF

Chemin : c'est le chemin du fichier à lire. Cette méthode nous permettra de gagner du temps par rapport à l'utilisation d'un fichier texte (.txt) dont la manipulation est très lourde (les opérations de lectures et d'écritures).

3. Le calcul de la distance Euclidienne

La distance la plus connue est la distance Euclidienne, qui définit l'espace cartésien.

$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (Pour des vecteurs de dimension n).

Dans notre implémentation, nous aurons besoin de calculer la distance Euclidienne entre deux instances. WEKA nous donne la possibilité de calculer la distance euclidienne entre deux objets de type instance avec leur 42 attributs (comme il est expliqué ci-dessus).

Pour cela il nous faudra importer les packages suivants :

```
import weka.core.EuclideanDistance;
import weka.core.Instances;
```

Figure 3: Packages distance Euclidienne

Tout d'abord il normalise les valeurs des attributs de type numérique puis il procède au calcul de la distance.
Nous allons faire appel à la classe weka.core.EuclideanDistance à chaque fois que nous aurons besoin de calculer la distance euclidienne de la façon suivante :

```
EuclideanDistance euclideanDistance = new EuclideanDistance(instances);
euclideanDistance.distance(instances.instance(i), instances.instance(j) ;
```

Figure 4: Calcul de la distance Euclidienne

Cette dernière instruction calcul la distance euclidienne entre l'instance « i » et « j ».

4. DBSCAN

DBSCAN est un algorithme de partitionnement de données, Il s'agit d'un algorithme basé densité dans la mesure où il s'appuie sur la densité estimée des clusters pour effectuer le partitionnement.
L'algorithme DBSCAN utilise 2 paramètres : la distance *Epsilon* et le nombre minimum de points $MinPts$ devant se trouver dans un rayon *Epsilon* pour que ces points soient considérés comme un cluster.
L'idée de base de l'algorithme est : pour un point donné, de récupérer son Epsilon-voisinage et de vérifier qu'il contient bien MinPts points ou plus. Ce point est alors considéré comme faisant partie d'un cluster.
C'est-à-dire pour une instance, on cherche tous ses voisins qui ont une distance inférieure à *Epsilon*. Si le nombre de ses voisins est au minimum égal à *MinPts* (le nombre des voisins supérieur ou égal à *MinPts*), cette instance est alors considérée comme faisant partie d'un cluster. On parcourt ensuite l'Epsilon-voisinage de proche en proche afin de trouver l'ensemble des points du cluster.
Pour utiliser cet algorithme, il faut tout d'abord importer les packages suivants :

```
import weka.clusterers.AbstractClusterer;
import weka.clusterers.forOPTICSAndDBScan.DataObjects.DataObject;
import weka.clusterers.forOPTICSAndDBScan.Databases.Database;
```

Figure 5: Packages pour utiliser DBSCAN

On a utilisé les paramètres par défauts (Epsilon =0.9 et Minpoints=6).
La classe weka.clusterers.DBScan nous donne plusieurs informations comme : le nombre de clusters, chaque instance à quelle cluster appartient ...etc.
Pour cela nous avons adapté cette classe à nos besoins, c'est-à-dire elle nous rend chaque cluster avec ses instances ainsi que les NOISE (instances qui n'appartiennent à aucun cluster) pour ensuite choisir une instance de chaque cluster qui va être utilisé comme points de départ (les instances qui contiennent les fourmis) de l'algorithme de colonie de fourmis.
On lance le DBSCAN de cette façon:

```
Lecture lecture = new Lecture();
Instances instances = new Instances(lecture.lecture(chemin));
ConfigDbscan configDbscan= new ConfigDbscan();
DBScan dbs= new DBScan();
dbs.setMinPoints((int) configDbscan.getMinPoints());
dbs.setEpsilon(configDbscan.getEpsilon());
dbs.buildClusterer(instances);
```

Figure 6: Utiliser le DBSCAN

En premier temps, il faut lire le fichier avec la fonction lecture que nous avons défini précédemment qui va converti le contenu de notre fichier en un objet de type instance. Deuxièmement, il faut faire la configuration, c'est-à-dire déterminer la valeur de *Epsilon* et *MinPts*. Enfin, la dernière instruction nous permet de lancer le BDSCAN et de créer les clusters.

IV. Implémentation des algorithmes

Après avoir présenté tous les outils et fonctions nécessaire pour nos algorithmes, nous allons maintenant expliquer l'implémentation de notre solution

Notre application comprend 3 fonctionnalités principales :

- Classificateur KNN
- Optimisation en utilisant l'algorithme de fourmis
- Optimisation en utilisant l'apprentissage actif

1. Le classificateur KNN

L'algorithme K-NN est une méthode de classification, il prend comme paramètres un ensemble d'apprentissage « Train » et un ensemble « Test ».

Pour classer une instance de « Test », il faut passer par les étapes suivantes :

- Premièrement lire le fichier « Train » et « Test » avec la fonction de lecture définit dans la partie précédente.
- Deuxièmement calculer la distance Euclidienne (comme il est expliqué précédemment) entre cette instance et toutes les instances de l'ensemble « Train ».
- Selon le nombre des voisins « k » :

Si k=1 : nous choisirons l'instance la plus proche, c'est-à-dire celle qui a la plus petite valeur de distance. L'instance du Test va être classée avec la classe de sa plus proche voisine.

Si K >1 : On va choisir les k plus proches instances. L'instance du Test va être classée avec la classe la plus fréquente de ses k plus proche voisines.

2. Optimisation par colonies de fourmis

Le principe général de l'algorithme de fourmis est que chaque fourmi va nous générer une solution

Dans ce qui suit nous allons expliquer comment on a implémenté cet algorithme : nous travaillerons avec deux fichiers : le fichier « Train » qui sera réduit et le fichier « Test ».

Tout d'abord, il faut que nous déterminions où on mettre les fourmis parce qu'on ne peut pas mettre sur chaque instance une fourmi (vu que le nombre d'instance est très élevé). Mais ce choix doit être judicieux afin d'assurer qu'on démarrera avec des instances significatives (des endroits stratégiques).

Pour cela, on a utilisé l'algorithme de clustering DBSCAN. On applique cet algorithme sur l'ensemble « Train » pour le découper en clusters (groupes), ensuite on va choisir une instance de chaque cluster ainsi des NOISE (instances qui n'appartiennent à aucun cluster) pour être les points de départ de l'algorithme (les instances fourmis).

Après avoir choisis les endroits des fourmis, ces dernieres vont démarrer pour générer les solutions.

Etant donné que les fourmis réelles font leur travail en parallèle tout en communiquant via l'environnement (les pistes de phéromones), il était nécessaire de trouver une méthode pour modéliser ce comportement dans notre implémentation. Afin de le réaliser, nous avons opté pour l'utilisation des Threads (tous les fourmis démarrent en même temps).

Threads:

Les « threads » ou « processus légers » sont des unités d'exécution autonomes qui peuvent effectuer des tâches, en parallèle avec d'autres threads.

Un thread peut être créé de deux manières :

- Héritage de la classe Thread : on définit une classe qui hérite de Thread et on redéfinit la méthode run.
- Implantation de l'interface Runnable : on définit une classe qui implante l'interface Runnable (ce qui consiste à définir une procédure public void run()).

Nous avons utilisé la première méthode Héritage de la classe Thread).

Chaque fourmis est un thread, et son travail est définit dans la méthode run () :

```
class X extends Thread {

   Public void run () {
       // Travail de la fourmi
   }
}
```

Le travail de la fourmi est bien détailler dans le chapitre précédent. Ce qui est important à dire c'est que la matrice des phéromones doit être partagée entre tous les threads (La matrice phéromone représente dans la nature l'environnement dans lequelle les fourmis communiquent).

Quand une fourmi doit choisir sa destination, elle prend en considération deux paramètres : La distance et la quantité de phéromone. C'est-à-dire plus la valeur de phéromone entre deux instances (entre l'instance fourmi et une autre instance) est élevé plus cette dernière est prioritaire dans le choix. En plus, quand une fourmi se déplace vers une autre instance elle met à jour (renforce) la quantité de phéromone entre elles.

A la fin de l'algorithme, chaque fourmi va nous donner un ensemble réduit (une solution). Ensuit on va évaluer chacun des ensembles résultant avec le classificateur KNN. C'est-à-dire on va classer l'ensemble « Test » (qui contient des instances sans les classes) avec chaque ensemble résultant. Le classificateur KNN va nous rendre le taux de classification de chaque ensemble. L'ensemble qui aura le grand taux de classification sera le meilleur.

3. Optimisation par l'Apprentissage Actif

Dans cette partie nous allons expliquer comment on a implémenté l'algorithme de l'apprentissage actif dans notre approche.

L'Apprentissage Actif a besoin de 3 ensembles :

- L'ensemble « Train » qu'on va le réduire.
- L'ensemble « Test ».
- Un ensemble qu'on va l'appelé « L ».

Le principe général de l'algorithme est qu'on démarrera avec un petit ensemble (L) et on va l'améliorer au fur et à mesure.

Comment l'ensemble « L » est construit :

L'ensemble « L » est l'ensemble solution (il va contenir notre ensemble réduit). Comme il est déjà dit, on démarre avec un petit ensemble « L » et on l'améliorera au cours de l'algorithme. Il est donc nécessaire de démarrer avec un ensemble qui contient des bonnes instances. Pour cela, on appliquera l'algorithme de clustering DBSCAN sur l'ensemble

« Train » pour le découper en clusters (groupes), ensuite on choisira une instance de chaque cluster ainsi des NOISE (instances qui n'appartiennent à aucun cluster). Ces instances vont être l'ensemble « L » de départ. Avec cette méthode on est sûr d'avoir choisi des instances significatives dès le départ.

Remarque : Les instances qui sont dans « L » sont aussi dans « Train ». C'est-à-dire on remplit « L » au fur et à mesure avec des instances du « Train » sans qu'on les enlève du dernier.

Maintenant nous allons commencer l'algorithme de l'apprentissage actif.

Cet algorithme (comme il est expliqué dans le chapitre précédent) prend en paramètre un taux qui est notre objectif on l'appel TV (le taux voulu). C'est-à-dire si l'algorithme arrive à ce taux au cours de son exécution il va s'arrêter.

L'algorithme consiste à :

Premièrement on évaluera l'ensemble « L » qui contient une instance de chaque cluster (c'est-à-dire classer l'ensemble « Test » avec « L » en utilisant KNN) ;

Si le taux de classification est supérieur ou égale au taux voulu Alors on arrête et l'ensemble « L » qui sera notre solution finale.

Sinon Chaque instance de « L » sera une fourmi dans l'ensemble « Train ». On démarre l'algorithme de fourmis (expliqué dans la partie précédente), mais cette fois on avance d'un seul pas. C'est-à-dire, chaque fourmi choisi sa distance (l'instance ou elle ira). On choisira la meilleure instance de chaque pas.

Par exemple si l'ensemble « L » contient 10 instances, ces instances vont être des fourmis dans l'ensemble « Train ». Ces 10 fourmis vont avancer d'un pas, c'est-à-dire chaque fourmi choisit une instance de destination. Maintenant on a 10 nouvelles fourmis, nous choisirons la meilleure parmi elles pour l'ajouter définitivement à « L ».

Comment nous choisirons la meilleure instance ? Nous ajouterons la première instance à « L » et nous classerons l'ensemble « Test » avec le nouveau « L » et nous garderons le taux de classification pour cette instance TC_1. L'instance est retirée de « L ».

On répète ce processus avec les 9 instances restantes. En résultat, nous aurons les taux de classification de chaque instance TC_1, TC_2, TC_3,..., TC_{10}. L'instance qui aura le meilleur taux de classification sera ajoutée définitivement à « L ».

Si les instances d'un pas dégradent le taux de classification, aucune instance ne sera prise et on passera au pas suivant.

Avec cette méthode on sera sûr que nous ajouterons que les instances qui vont améliorer notre ensemble solution.

V. Présentation de l'application

En premier lieu nous allons présenter les fonctionnalités principales de l'application ensuite le déroulement de l'exécution des algorithmes d'optimisation des colonies de fourmis et de l'apprentissage actif.

La figure suivante montre l'interface graphique qui apparait lors du lancement de l'application et qui nous donne la possibilité de choisir la tâche que nous voulons exécuter.

L'interface ci-dessous apparait lors du lancement de l'application, elle comporte les champs suivants :

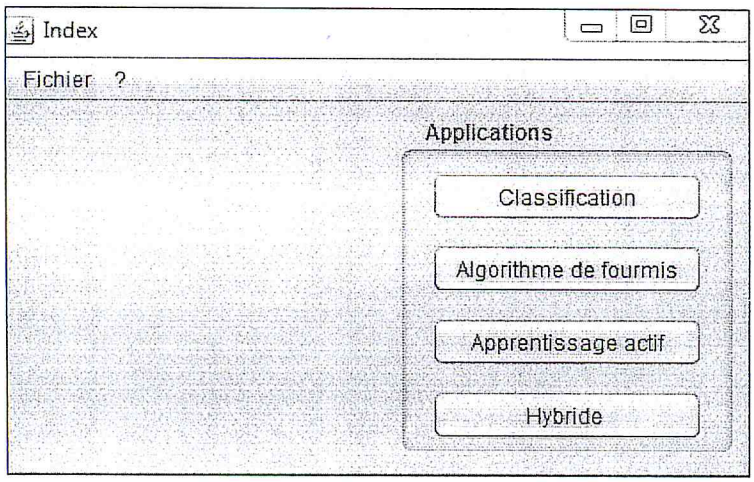


Figure 7: L'interface principale de l'application

- Bouton «Classification»: il nous mène a l'interface de notre classificateur KNN (figure 2)
- Bouton «Algorithme de fourmis»: permet d'ouvrir l'interface d'exécution de l'algorithme de fourmis (figure 5)
- Bouton «Apprentissage actif»: permet d'explorer l'interface de l'algorithme d'optimisation Apprentissage actif (figure 7)
- Bouton «Hybride» pour aller vers une interface qui nous permet d'exécuter les deux algorithmes en même temps (figure 8)

1. Interface du classificateur KNN

Cette interface nous permet de faire la classification KNN entre deux ensembles « Train » et « Test ».

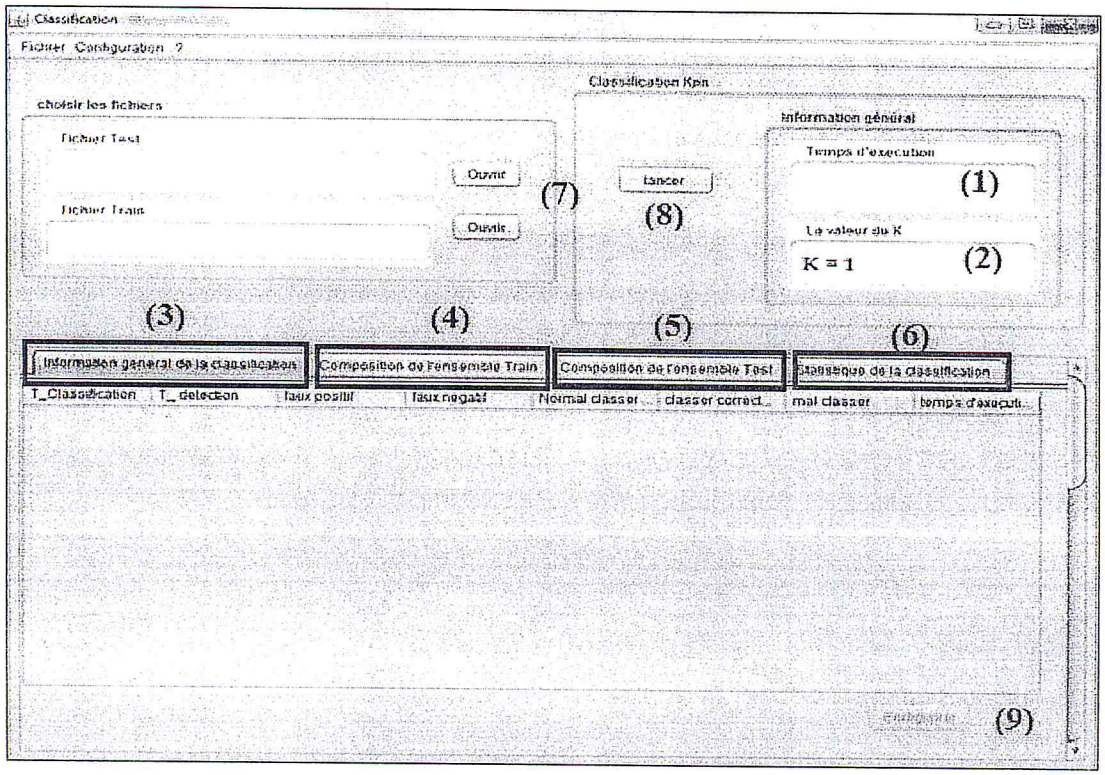


Figure 8: Interface Classificateur

(1) le temps d'exécution de la classification.
(2) la valeur du K (notre classificateur est à base de K- plus proche voisin).
(3) les différentes informations de la classification (taux de classification, taux de détection,..)
(4) la composition de l'ensemble Train.
(5) la composition de l'ensemble Test.
(6) les statistiques de la classification affichée en détail.
(7) pour charger les ensemble Train et Test en format ARFF (Figure 4)
(8) lancer la classification, une fois terminé il nous affiche la composition de chaque fichier et les statistiques de classification par famille.
(9) Enregistrer le tableau des résultats de classification dans un fichier Excel .xls

- Configuration -> KNN : permet de configurer les paramètres de KNN (Figure 3).

Cette interface permet de changer les paramètres de classificateur KNN, elle contient :
-un champ pour saisir la valeur du k (nombre de voisins).
-une liste déroulante qui permet de choisir le type de classification (par famille ou par classe).

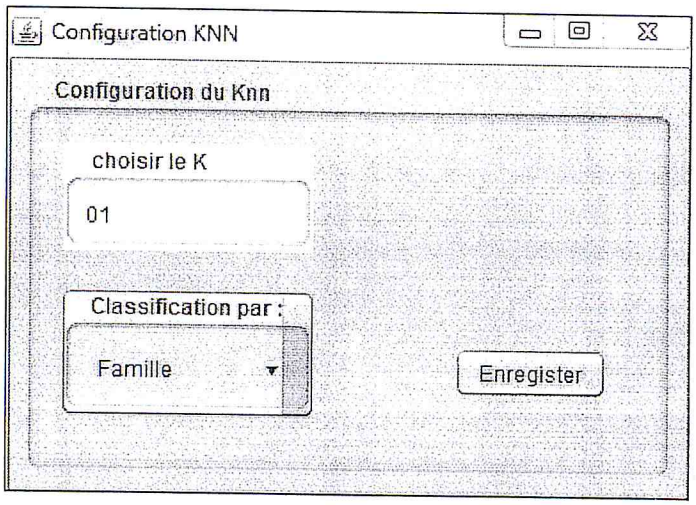


Figure 9: Paramètres de l'algorithme KNN

- Bouton «Enregistrer»: permet d'enregistrer la nouvelle configuration.

Charger les fichiers Train et Test :

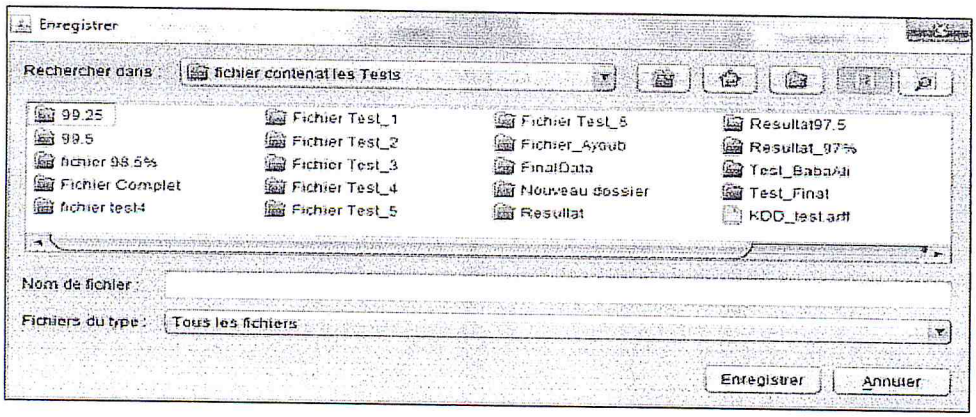


Figure 10: Charger le fichier Train et Test

2. Interface de l'algorithme de fourmis

L'interface ci-dessus permet de lancer l'algorithme de classification elle comporte :

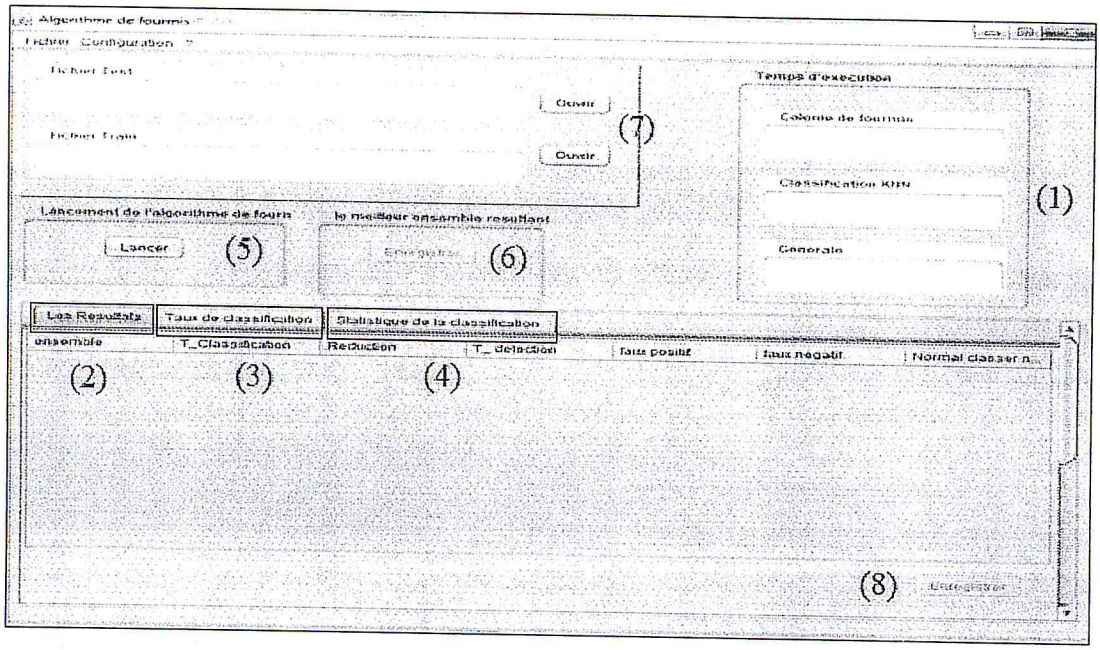


Figure 11: L'interface de l'algorithme de fourmis

(1) Les champs de temps d'exécution de l'algorithme de fourmis, le temps pris pour classifier tous les ensembles résultant et enfin le temps des deux opérations.
(2) Tableau qui contient les résultats de et les taux de chaque ensemble résultant
(3) Un onglet qui affiche les bars de la classification
(4) Un onglet qui contient les statistiques de meilleur ensemble résultant
(5) Lancer l'algorithme
(6) Bouton pour enregistrer le meilleur ensemble résultant avec le format ARFF (.arff)
(7) Pour charger les ensemble Train et Test en format ARFF (Figure 4)
(8) Enregistrer le tableau des résultats de classification dans un fichier Excel .xls

- Fichier ->Enregistrer->Ensemble : permet d'enregistrer le meilleur ensemble résultant dans un fichier .arff

- Fichier ->Enregistrer->Tableau: permet d'enregistrer le tableau résultant après l'exécution de l'algorithme de fourmis (le tableau contiendra les statistiques de tous les ensembles résultants)

- Configuration -> DBSCAN : permet de configurer les paramètres du DBSCAN (Figure6)

Cette interface permet changer les principaux paramètres du DBSCAN

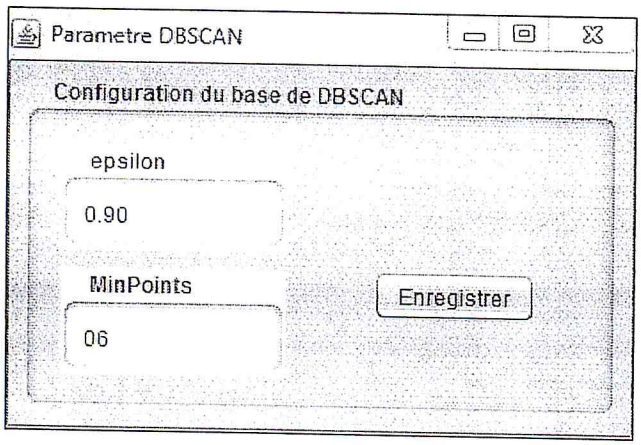


Figure 12: Paramètres DBSCAN

3. Interface de l'algorithme d'apprentissage actif

La figure suivante montre l'interface graphique de l'exécution de l'algorithme d'apprentissage actif :

L'interface est similaire à celle de l'algorithme de colonie de fourmi (figure 1) seulement que dans cette interface il y a un champ du taux que l'on fixe comme une condition d'arrêt pour l'algorithme d'apprentissage actif.

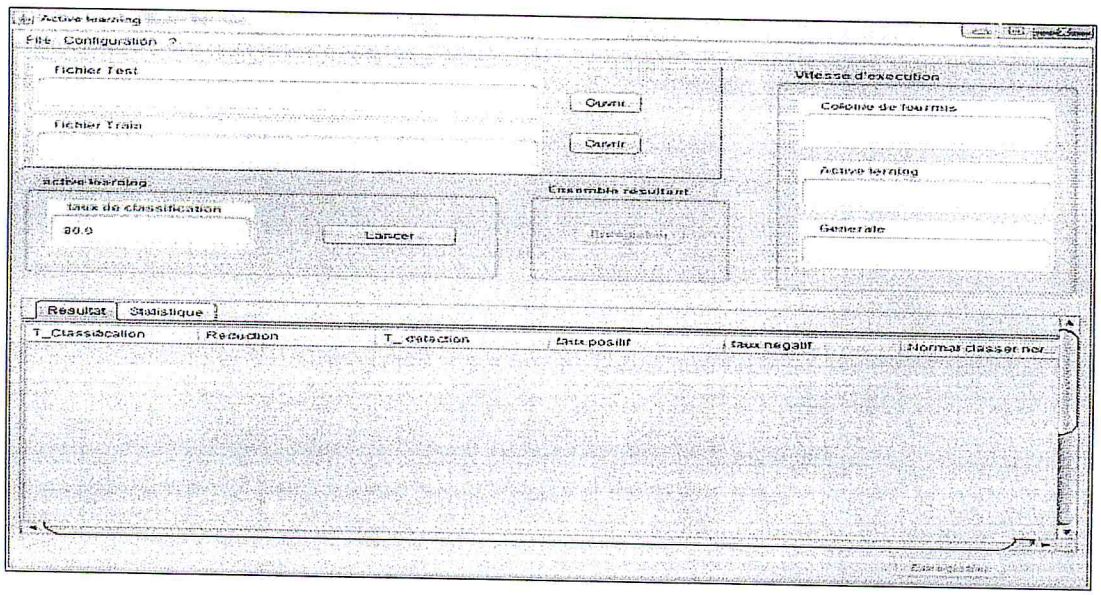


Figure 13: L'interface de l'algorithme d'apprentissage actif

4. Interface hybride

Cette interface nous permet de lancer les deux algortihmes d'optimisation en meme temps elle comporte :

-un champs qui affiche le temps d'execution de l'algorithme de colonie de fourmis, Classification KNN pour les ensembles resultant et temps de l'apprentissage actif.

- un champs pour fixer le taux de classification pour l'apprentissage actif comme condition d'arret.

-un tableau dans lequel s'affiche les statistiques de classification des ensembles resultant de l'algorithme de fourmis et celui de l'apprentissage actif.

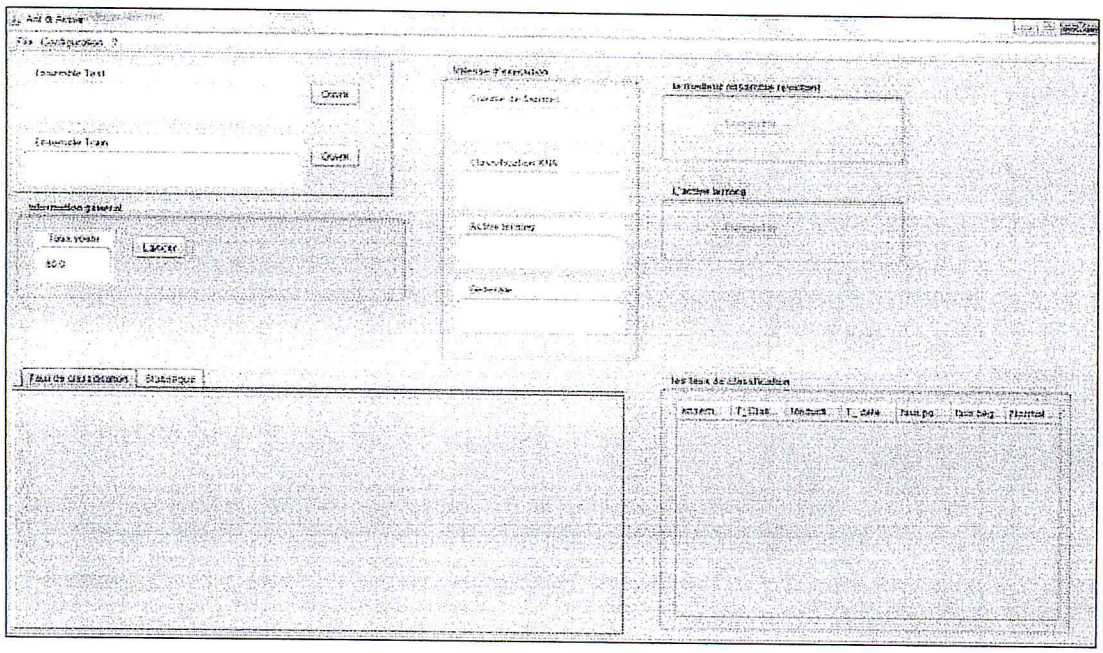


Figure 14: L'interface Hybride

VI. Conclusion

Dans ce chapitre, nous avons décrit l'implémentation de nos algorithmes d'optimisations et les interfaces de l'application réalisée.

Dans la partie implémentation nous avons présenté les outils et le langage de programmation utilisés pour mettre en œuvre notre solution. Nous avons utilisé principalement le langage Java et L'environnement NetBeans pour le développement des algorithmes d'optimisation. Nous avons également utilisé WEKA pour la lecture et l'utilisation des fichiers ARFF. Nous avons achevé ce chapitre par l'explication de l'exécution des algorithmes et la classification à base de KNN ainsi que la présentation de notre l'application.

CHAPITRE 5

Test et validation

Chapitre 5: Test et Validation

I. Introduction

Les tests de vérification et la validation des besoins représentent une phase très importante du développement d'un logiciel. Ils visent à identifier les bugs et les anomalies qui découlent d'une mauvaise expression des besoins ou d'erreurs de programmation.

Notre travail est sur la réduction de la base d'apprentissage utilisé par l'IDS. Pour cela nous avons choisis de travailler sur la base d'apprentissage KDD'99, c'est-à-dire on va appliquer notre approche sur cette base.

Mais pour valider notre approche, il faut tester nos résultats (ensembles résultant) avec des ensembles conçu spécialement pour ça. Ces ensembles de validations contiennent des nouvelles attaques qui ne figurent pas dans le KDD'99 pour voir si notre ensemble (réduit) pourra détecter ces nouvelles attaques ou non.

Pour cela, nous allons travailler avec un ensemble qui s'appelle « Corrected ».

Dans ce chapitre nous allons débuter par définir l'ensemble KDD'99, ensuite on va présenter les différents tests que nous avons faits ainsi que leurs résultats, en termine avec une discussion des résultats.

II. Présentation des données KDD

L'évaluation des systèmes de détection d'intrusion est une problématique d'actualité très active. Le nombre important d'intervenants impliqués dans cette discipline (laboratoires de recherche, institutions gouvernementales et non gouvernementales, universités, développeurs du monde commercial, etc) et la sensibilité du sujet, rendent indispensable la standardisation d'une méthodologie d'évaluation.

D'une part, pour rendre compte des performances des systèmes commercialisés, d'autres part, pour offrir des moyens aux chercheurs travaillant sur ce problème afin d'élaborer et d'évaluer de nouvelles approches. C'est dans cette optique que l'agence américaine DARPA a initié un programme d'une série d'évaluation annuelle dont la finalité est l'élaboration d'une méthodologie d'évaluation standard.

1. Historique

L'agence américaine DARPA (Defence Advanced Research Projects Agency) a lancé un programme afin d'élaborer des méthodologies d'évaluation pour les systèmes de détection d'intrusions.

La première campagne d'évaluation dans ce programme a eu lieu en 1998 et réalisée par les laboratoires MIT Lincoln (Massachusetts Institute of Technology). La deuxième campagne a eu lieu en 1999 et une troisième en 2000. L'objectif de ces campagnes était de fournir une base d'évaluation contenant du trafic normal et différentes variantes d'attaques que les chercheurs travaillant sur la détection d'intrusions peuvent utiliser.

Lors de la conférence mondiale sur la découverte de connaissances dans les bases de données KDD 1999, une compétition concernant la détection d'intrusions a eu lieu. Pour les besoins de la compétition, l'un des participants a fourni la base de données KDD'99 [KDD 1999], qui est une version formatée des données brutes DARPA'98. Depuis, la base de données KDD'99 a été largement utilisée en détection d'intrusions, notamment lorsque des techniques de fouille de données sont utilisées. Contrairement aux données DARPA'98 pour lesquelles une base de données formatée est disponible, aucune base formatée n'est disponible pour DARPA'99. C'est pourquoi, ces données n'ont pas connu une large utilisation comme DARPA'98 bien qu'elles contiennent plusieurs variantes d'attaques, notamment les nouvelles. Concernant les données DARPA'2000, elles sont très utilisées en corrélation. [18]

2. Présentation des données

La base de données KDDcup'99 a été développée par MIT Lincoln Lab en 1998, cette base contient un nombre d'enregistrements de communication égal à 4.900.000. Chaque communication dispose de 41 attributs et elle est classifiée en cinq classes: normal et quatre comportements d'attaques (*Dos, Probe, U2r, R2l*). [19] *Dos* (Déni de service), *R2L* (accès non autorisé d'une machine à distance, par exemple devinant le mot de passe), *U2R* (accès non autorisé aux privilèges d'un super-utilisateur tel que *buffer overflow*) et *Probe* (sondage et surveillance tel que port scanning).

Les différentes attaques existantes, classées en catégories, sont décrite dans le tableau suivant :

Types d'attaques	Catégories d'attaques
apache2, arppoison, back, Crashiis, land, mailbomb, Neptune (SYN Flood), pod (Ping of Death), processtable, selfping, smurf, sshprocesstable, Syslogd, tcpreset, teardrop, udpstorm	DOS
Buffer overflow, loadmodule, perl, rootkit, anypw, casesen, eject, ffbconfig, fdformat, loadmodule, ntfsdos, perl, ps, sechole, xterm, yaga	U2R
dictionary, ftpwrite, guest, httptunnel, imap, named ncftp, netbus, netcat, phf, ppmacro, sendmail, sshtrojan, xlock, Xsnoop	R2L
ipsweep, mscan, NTinfoscan, nmap, queso, portsweep, saint, satan, resetscan	Probe

Figure 1: Répartition des attaques dans DARPA'99

Les données DARPA'99 comportent cinq semaines de trafic réseau et autres données d'audit fournies par les laboratoires Lincoln. Les données des trois premières semaines sont réservées pour l'apprentissage et celles des deux dernières semaines sont fournies pour le test. Les cinq semaines sont décrites comme suit :
– Les données collectées pendant les deux semaines 1 et 3 contiennent uniquement du trafic normal. Toutes ces données sont réservées pour l'apprentissage.
– Les données de la semaine 2 contiennent du trafic normal et différents types d'attaques. Elles sont également destinées pour l'apprentissage.
– Les deux dernières semaines (4 et 5) contiennent des données de test contenant du trafic normal et plusieurs attaques présentes également dans la base d'apprentissage, mais également plusieurs nouvelles attaques (qui ne sont pas présentes dans la base d'apprentissage). [18]

3. Les types d'attaques

Dans cette section nous allons présenter les 4 familles d'attaques Dos, U2R, U2L, Probe.

- DOS (Deny Of Service)

Le déni de service, connu sous le titre anglophone de "Denial Of Service" ou encore DOS, est une attaque réalisée dans le but de rendre indisponible durant une certaine période les services ou ressources d'une organisation. Généralement, ce type d'attaque à lieu contre des machines, serveurs et accès d'une entreprise afin qu'ils deviennent inaccessibles pour leurs clients. Le but d'une telle attaque n'est pas d'altérer ou de supprimer des données, ni même de voler quelconque information. Il s'agit ici de nuire à la réputation de sociétés présentes sur Internet en empêchant le bon fonctionnement de ses activités.

La réalisation d'un déni de service n'est pas très compliquée, mais pas moins efficace. Il est possible d'attaquer tout type d'équipements réseau tel que les serveurs, routeurs et Switchs. Cela touche 99% de la planète car la plupart des dénis de service exploitent des failles liées au protocole **TCP/IP**. Nous pouvons diviser les impacts des attaques DOS en deux catégories :

- Les dénis de service par saturation qui consistent à submerger une machine de requêtes, afin qu'elle ne soit plus capable de répondre aux demandes réelles.

- Les dénis de service par exploitation des vulnérabilités qui consistent à exploiter une faille du système cible afin de le rendre inutilisable.

Le principe de ces attaques est d'envoyer des paquets ou des données de taille ou de constitution inhabituelle, afin de provoquer une saturation ou un état instable des équipements victimes et de les empêcher ainsi d'assurer les services réseau qu'elles sont censés offrir. Dans certains cas extrêmes, ce type d'attaque peut conduire au crash de l'équipement cible.

- *Exemple DOS*

Il existe de nombreux types d'attaques par déni de service. Le simple fait d'empêcher un système de rendre un service peut être qualifié de « déni de service ». En voici les principaux :

Ping flooding : cette attaque consiste à envoyer un flux maximal de « ping » vers une cible.
Attaque Ping of Death : le principe est d'envoyer un paquet ICMP avec une quantité de données supérieure à la taille maximale d'un paquet IP. Si la cible n'est pas adaptée à gérer ce type de paquet, il peut se produire un crash.

SYN flood : cette technique consiste à saturer une machine en envoyant une multitude de paquets TCP avec le flag SYN armé. Cela créera une multitude de connexions en attente, demandant un grand nombre des ressources du système.
UDP Flood : l'attaquant envoie un grand nombre de requêtes UDP sur une machine. Le trafic UDP étant prioritaire sur le trafic TCP, ce type d'attaque peut vite troubler et saturer le trafic transitant sur le réseau.
L'attaque ARP : elle consiste à s'attribuer l'adresse IP de la machine cible, c'est-à-dire à faire correspondre son adresse IP à l'adresse MAC de la machine pirate dans les tables ARP des machines du réseau. Pour cela il suffit en fait d'envoyer régulièrement des trames ARP REPLY en diffusion, contenant l'adresse IP cible et la fausse adresse MAC. Cela a pour effet de modifier les tables dynamiques de toutes les machines du réseau. Celles-ci enverront donc leurs trames Ethernet à la machine pirate tout en croyant communiquer avec la cible
L'attaque Teardrop : cette attaque utilise une faille propre à certaines piles TCP/IP. Cette vulnérabilité concerne la gestion de la fragmentation IP. Ce problème apparaît lorsque la pile reçoit le deuxième fragment d'un paquet TCP contenant comme donnée le premier fragment. La pile TCP/IP peut s'avérer incapable de gérer cette exception et le reste du trafic.
Smurfing : consiste à envoyer un ping en diffusion sur un réseau (A) avec une adresse IP source correspondant à celle de la cible (B). Le flux entre le port ping de la cible (B) et du réseau (A) sera multiplié par le nombre de machines sur le réseau (A), conduisant à une saturation de la bande passante du réseau (A) et du système de traitement des paquets de (B).
L'attaque land : consiste à envoyer des paquets TCP comportant une adresse source IP et un numéro de port identiques à ceux de la victime. Le host attaqué pense alors qu'il parle à lui-même ce qui généralement provoque un crash.
Les bombes e-mail : le mail bombing consiste à envoyer de gros ou de nombreux fichiers à un utilisateur pour saturer sa boîte de réception de courrier électronique.
L'attaque Unreachable Host : cette attaque envoie des messages ICMP "Host Unreachable" à une cible, provoquant la déconnexion des sessions et la paralysie de la victime, même si ils sont envoyés à faible cadence.

- Probe

Consiste en la collecte d'informations par le biais d'outils comme whois, Arin, DNS lookup. La collecte d'informations sur le système cible peut s'effectuer de plusieurs manières, comme par exemple un scan de ports grâce au programme Nmap pour déterminer la version des logiciels utilisés, ou encore un scan de vulnérabilités à l'aide du programme Nessus. Pour les serveurs web, il existe un outil nommé Nikto qui permet de rechercher les failles connues ou les problèmes de sécurité.

Des outils comme firewalk, hping ou SNMP Walk permettent quant à eux de découvrir la nature d'un réseau.

-Exemple Probe

L'attaque Nmap : Elle est conçue pour détecter les ports ouverts, les services hébergés et les informations sur le système d'exploitation d'un ordinateur distant.

- R2L (Remote to User)

Des attaquants n'ayant pas un accès autorisé sur la machine cible, arrivent à accéder aux ressources du système comme s'ils étaient des utilisateurs autorisés [18].

- U2R (User to Root)

Cette catégorie d'attaques concerne un abus de privilèges de la part d'un utilisateur autorisé pour accéder aux privilèges du super utilisateur [18].

III. Tests et Résultats

Les tests sont effectués en deux phases :

- La première consiste à réduire l'ensemble train et évaluer les résultants avec l'ensemble « corrected ».

- La deuxième évalue le meilleur ensemble résultant dans un réseau réel (IDS).

Nous décrivons les résultats en utilisant les termes suivants :
Taux de détection, Taux de Classification, Taux de faux positifs et le Taux de faux négatifs.
Chaque mesure est définie ci-dessous :

- Taux de détection : le taux de détection est le rapport entre les intrusions bien classées et le nombre total des intrusions de test. Il est calculé par la formule suivante:

Taux de détection= [(Le nombre des attaques détectées) ÷ (Le nombre Totale des attaques)] × 100

- Taux de Classification : le taux de classification est le rapport entre des connexions correctement classifiées et le nombre total des connexions de test. Il est calculé par la formule suivante :

Taux de classification = [(Le nombre des connexions correctement classifiées) ÷ (Le nombre Totale des connexions de test)] × 100

- Taux de faux positifs : le taux de faux positifs est le rapport entre le nombre des connexions normales classées en tant qu'attaque et le nombre total des connexions normal. Il est calculé par la formule suivante :

Taux de faux positifs = [(le nombre des connexions normales classées attaques) ÷ (le nombre total des connexions normal)] × 100

- Taux de faux négatifs : le taux de faux négatifs est le rapport entre le nombre des attaques classées normal et le nombre total des attaques. Il est calculé par la formule suivante :

Taux de faux positifs = [(le nombre des attaques classées normal) ÷ (le nombre total des attaques)] × 100

A. Partie 1 : Réduction et évaluation des ensembles

La base de d'apprentissage KDD'99 est un ensemble de model (on l'appelle aussi instance) signé comme normal ou attaque, chacun de ces instances contient 41 attributs, ils représentent les caractéristiques d'un état normal ou anormal.

La base dans son format original contient 4900000 d'instances et sa taille est de 700Mo.

Effectuer des tests sur cette base est très couteux en termes de temps d'exécution et nécessite des machines très performantes.

Une version réduite de la base de données originale est aussi disponible, elle contient 494020 instances (10% de la taille original), elle contient la même distribution des instances que la base original.

Pour les tests nous allons travailler avec une base qui contient 4942 instances qui est une version réduite de la base 10% avec le logiciel Weka qui garde la même distribution c'est-à-dire les mêmes caractéristiques.

Pour cela, on va diviser la base initiale (4942) en deux sous-ensembles, le premier on va le nommer « Train » l'ensemble d'entrainement et il contient 3943 instances. Il contiendra des instances choisis aléatoirement et sans remise de la base initial, le second on va le nommer « Test », il contiendra le reste 999 instances. Notre approche sera appliquer sur l'ensemble « Train », ensuite on va utiliser l'ensemble « Test » pour évaluer nos résultats.

Mais pour valider notre approche, il faut tester nos résultats (ensembles résultant) avec des ensembles conçu spécialement pour ça. Ces ensembles de validations contiennent des nouvelles attaques qui ne figurent pas dans le KDD'99 pour voir si notre ensemble (réduit) pourra détecter ces nouvelles attaques ou non.

Pour cela, nous allons travailler avec un ensemble qui s'appelle « Corrected ».

L'ensemble «Corrected» contient 310992 instances, mais pour nos tests nous allons utiliser une version réduite avec weka (pour garder la pondération de l'ensemble et ne perdre aucune famille d'attaque) qui contient 2996 instances (1% du corrected).

Les trois fichiers de test se compose de :

	Nbr d'instance	Normal	Dos	U2r	R2l	Prob
«Train»	3943	762	3097	15	31	38
«Test»	999	190	775	9	16	9
«Corrected»	2996	588	2272	19	62	55

Figure 2: Composition des ensembles Tests

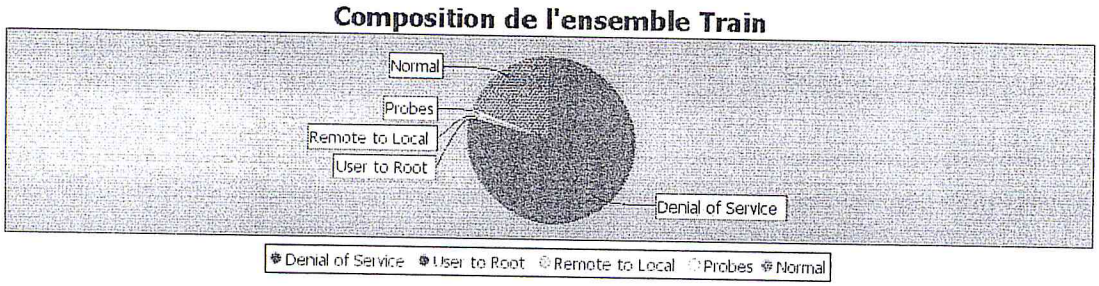


Figure 3: composition de l'ensemble «Train»

L'ensemble «Train» sert comme une base d'apprentissage pour les IDS c'est pour cela il doit contenir la plus part des familles d'attaque pour connaitre les attaques qui transitent dans le réseau.

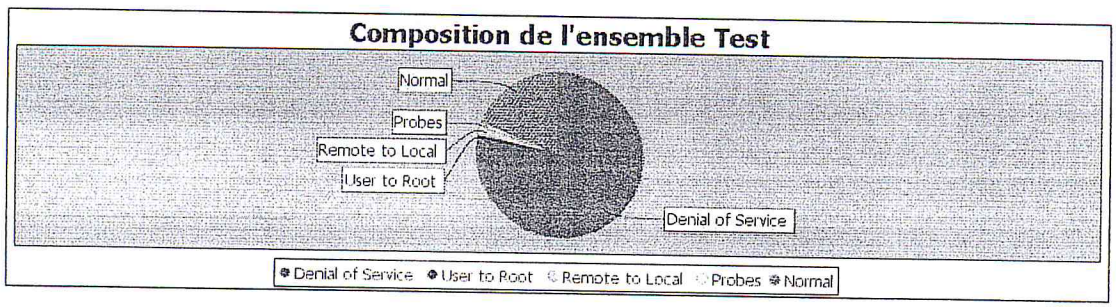


Figure 4: composition de l'ensemble «Test»

L'ensemble «Test» est utilisé pour l'évaluation de l'ensemble «Train» lors de la réduction. Il contient la plus part des attaques ainsi que des paquets normaux, il représente en quelque sorte les paquets transitent dans le réseau.

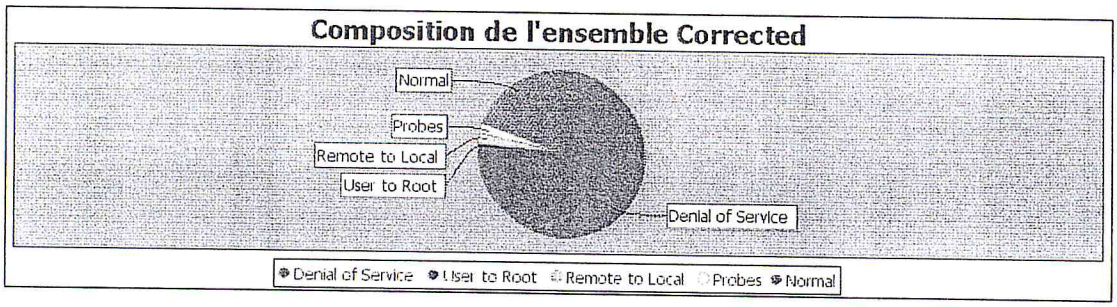


Figure 5: composition de l'ensemble «Corrected»

L'ensemble «Corrected» utilisé pour la validation des ensembles résultant, il contient des attaques qui ne figurent ni dans l'ensemble «Test» ni dans le «Train». Il est similaire à un réseau réel d'une telle manière qu'on anticipe les nouvelles attaques.

1. Le résultat des Tests avec les ensembles résultants

Après l'application des algorithmes d'optimisation (Colonie de fourmis et apprentissage actif) nous avons obtenu des ensembles de différentes tailles. Pour évaluer leur performance nous les avons classés avec l'ensemble «Test» et «Corrected».

La composition des Ensemble résultant :

	Réduction	Nombre d'instance	Normal	Dos	U2r	R2l	Probe
Ensemble_1	77.38%	3053	383	2621	9	21	19
Ensemble_2	0.53%	21	9	9	0	1	2
Ensemble_3	9.35%	396	9	349	3	4	4
Ensemble_4	8.44%	333	8	310	4	7	4
Ensemble_5	88.5%	3490	261	3202	9	13	5

Figure 6: Composition des ensembles résultants

Le tableau ci-dessus contient les ensembles obtenu lors de la réduction avec leur taille, le pourcentage de la réduction par rapport à l'ensemble «Train» et la composition de ses instances par famille.

Ensemble_1: est obtenu après l'application de l'algorithme des colonies de fourmis sur l'ensemble « Train ». Il représente 77% de l'ensemble «Train».

Ensemble_2: est obtenu avec l'algorithme de l'apprentissage actif, nous avons fixé le taux de classification à 97% comme condition d'arrêt. Il représente 0.53% de la taille général de l'ensemble «Train».

Ensemble_3: est obtenu avec l'algorithme de l'apprentissage actif, nous avons fixé le taux de classification à 98.5% comme condition d'arrêt. Il représente 9.35% de la taille général de l'ensemble «Train».

Ensemble_4: est obtenu avec l'algorithme de l'apprentissage actif, nous avons fixé le taux de classification à 99.25% comme condition d'arrêt. Il représente 8.44% de la taille général de l'ensemble «Train».

Ensemble_5: est obtenu avec l'algorithme de l'apprentissage actif, nous avons fixé le taux de classification à 99.5% comme condition d'arrêt. Il représente 88.5% de la taille général de l'ensemble «Train».

Les résultats de la classification

Pour K=1 :

Le résultat de classification avec l'ensemble «Test»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	100%	100%	0.0%	0.0%	7.842 seconde
Ensemble_2	97.5%	99.01%	0.0%	00.99%	0.178 seconde
Ensemble_3	98.7%	99.38%	0.53%	0.62%	1.073 seconde
Ensemble_4	99.3%	100%	0.0%	0.03%	0.989 seconde
Ensemble_5	99.7%	100%	0.0%	0.01%	6.721 seconde
«Train»	99.9%	100%	0.0%	0.0%	10.10 seconde

Figure 7: Classification 1NN avec l'ensemble «Test»

Le résultat de classification avec l'ensemble «Corrected»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	94.46%	94.64%	2.38%	5.36%	23.35 seconde
Ensemble_2	93.83%	94.64%	2.38	5.36%	0.519 seconde
Ensemble_3	94.13%	95.35%	4.93%	4.65%	2.895 seconde
Ensemble_4	94.13%	95.53%	5.44%	4.07%	2.743 seconde
Ensemble_5	94.36%	95.35%	3.23%	4.65%	19.43 seconde
«Train»	94.39%	94.56%	2.55%	5.44%	29.56 seconde

Figure 8: Classification 1NN avec l'ensemble «Corrected»

Pour K=3 :

Le résultat de la classification avec l'ensemble «Test»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	99.3%	99.51%	0.0%	0.49%	8.87 seconde
Ensemble_2	95.3%	96.66%	7.37%	3.34%	0.18 seconde
Ensemble_3	97.70%	98.27%	0.53%	1.73%	1.14 seconde
Ensemble_4	98.20%	98.64%	0.0%	1.36%	0.96 seconde
Ensemble_5	99.10%	99.51%	0.0%	0.49%	7.39 seconde
«Train»	99.6%	100%	0.53%	0.0%	10.43seconde

Figure 9: Classification 3NN avec l'ensemble «Test»

Le résultat de la classification avec l'ensemble «Corrected»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	94.43%	94.31%	1.70%	5.96%	24.13 seconde
Ensemble_2	88.95%	94.39%	24.66%	5.61%	0.511 seconde
Ensemble_3	93.59%	93.69%	2.55%	6.31%	3.014 seconde
Ensemble_4	93.62%	94.27%	2.55%	5.73%	2.862 seconde
Ensemble_5	93.39%	95.18%	3.57%	4.82%	20.24 seconde
«Train»	94.66%	94.35%	1.53%	5.56%	30.86 seconde

Figure 10: Classification 3NN avec l'ensemble «Corrected»

Pour K=5 :

Le résultat de la classification avec l'ensemble «Test»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	98.10%	98.52%	0.0%	1.48%	8.48 seconde
Ensemble_2	95.5%	94.93%	0.53%	5.07%	0.18 seconde
Ensemble_3	96.20%	96.29%	0.0%	3.71%	1.12 seconde
Ensemble_4	97.60%	97.90%	0.53%	2.10%	0.96 seconde
Ensemble_5	91.49%	90.61%	0.53%	9.93%	7.39 seconde
«Train»	98.8	99.26%	0.53%	0.74%	10.44seconde

Figure 11: Classification 5NN avec l'ensemble «Test»

Le résultat de la classification avec l'ensemble «Corrected»:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_1	94.66%	94.44%	1.36%	5.56%	24.79 seconde
Ensemble_2	78.77%	74.29%	2.04%	25.71%	0.511 seconde
Ensemble_3	87.12%	85.09%	1.87%	14.91%	3.252 seconde
Ensemble_4	93.49%	93.81%	3.40%	6.19%	2.862 seconde
Ensemble_5	86.92%	86.25%	1.87%	13.75%	20.24 seconde
«Train»	94.56%	94.27%	2.04%	5.73%	30.35 seconde

Figure 12: Classification 5NN avec l'ensemble «Corrected»

2. Analyse des résultats

Nous avons utilisé différentes valeurs de K pour voir l'influence du choix de cette valeur sur la classification. Par la suite, nous allons se focaliser sur K=1 pour l'évaluation de nos résultats.

D'après les expérimentations précédentes, le meilleur ensemble est l'ensemble_4 qui est obtenu avec l'algorithme de l'apprentissage actif avec un taux fixé égale à 99% comme condition d'arrêt. Il représente 8.44% de la taille général de l'ensemble «Train» et il contient 333 instances, alors que l'ensemble « Train » contient 3943 instances. Ainsi que ses résultats sont très bon par rapport aux autres ensembles.

Classification (K=1) avec l'ensemble « Test »:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_4	99.3%	100%	0.0%	0.03%	0.989 seconde
«Train»	99.9%	100%	0.0%	0.0%	10.10 seconde

Figure 13: Classification 1NN de l'enselmble_4 avec «Test»

D'après ces résultats, on voit que notre ensemble réduit garde les mêmes performances de l'ensemble initial, par contre notre ensemble est très rapide.

Classification (K=1) avec l'ensemble « Corrected »:

	Taux de classification	Taux de détection	Faux positif	Faux négatif	Temps d'exécution
Ensemble_4	94.13%	95.53%	5.44%	4.07%	2.743 seconde
«Train»	94.39%	94.56%	2.55%	5.44%	29.56 seconde

Figure 14: Classification 1NN de l'enselmble_4 avec «Corrected»

Comme nous l'avons dit, pour valider une solution il faut la tester avec l'ensemble « corrected » parce qu'il contient des nouvelles attaques qui n'existent pas dans l'ensemble « Train ».
D'après les résultats, on voit qu'on est arrivé à améliorer plusieurs choses :

- Le temps de classification avec l'ensemble « Train » se fait 29.56 secondes par contre avec l'ensemble réduit il se fait avec 2.743 secondes, ce qui implique qu'avec notre ensemble l'IDS sera plus rapide et dans ce cas il pourra classer les paquets en temps réel.
- Les faux négatifs : Après la réduction le taux de faux négatifs (attaque classé comme normal) est moins, ce qui implique que notre ensemble est plus précis parce qu'on ne garde que les instances significatives.
- Taux de détection : Avec notre ensemble le taux de détection (combien d'attaque classer juste) a augmenté, ce qui est très important parce que l'ensemble « corrected » contient des nouvelles attaques. Notre ensemble est plus précis.

En conclusion, on peut dire que notre ensemble réduit est plus rapide et plus précis que l'ensemble « Train » parce qu'on ne garde que les instances significatives.

B. Partie 2 : Tests dans un réseau réel

Pour tester la performance de notre meilleur ensemble résultant (ensemble_4) nous avons implémenté un IDS et utilisé l'ensemble_4 comme une base d'apprentissage.

Notre travail se compose de deux parties:

- Partie 1 : Nous lisons le fichier .ARFF a envoyé (ensemble «Test») ensuite nous encapsulons les instances dans des paquets TCP, ces derniers seront envoyés par réseau (JPCAP).

- Partie 2: L'IDS capte les paquets, il les décapsule puis les converti en des instances de type Weka.core.instance. Ces derniers seront classés en utilisant notre ensemble réduit (l'ensemble_4) comme une base d'apprentissage. Si une attaque est détectée elle sera signalée.

1. Scénario d'attaque :

Nous avons fait quatre phases d'attaque alternée par des phases normales:

-Première phase : pendant 62.221 secondes nous avons envoyé 70 attaques DOS, 6 attaques U2R, 5 attaques R2L, 9 attaques Probe, 24 attaques R2L et 2 attaques U2R.

Cette phase est alternée par 196 paquets Normal.

-Deuxième phase : pendant 53.514 secondes nous avons envoyé 273 attaques DOS, 5 attaques U2R, 11 attaques Probe, 1 attaque U2R, 1 attaque R2L, 15 attaques DOS, 1 attaque R2L, 18 attaques Probe, 4 attaques R2L, 3 attaques U2R, 1 attaque R2L, 3 attaques Probe, 5 attaques R2L et 18 attaques DOS.

La deuxième phase est alternée par 196 paquets Normal

-Troisième phase: pendant 222.058 seconde nous avons envoyé 1428 attaques DOS, 2 attaques U2R, 2 attaques Probe, 14 attaques R2L et 12 attaques Probe.

Cette phase est alternée par 196 paquets Normal

-Quatrième phase: pendant 28.556 seconde nous avons envoyé 6 attaques R2L et 3 attaques DOS.

Après la capture des paquets, l'IDS les classes en utilisant notre ensemble réduit (Ensemble_4) comme base d'apprentissage.

2. Résultat des Tests

Nous allons détailler les résultats du test dans un réseau réel.

	Emis	Détecté	Taux de classification
DOS	2273	2202	97%
U2R	20	5	25%
R2L	62	11	18%
Probe	56	32	57%

Figure 15: les familles d'attaques dans un réseau réel

Ce tableau ci-dessus comporte les différentes familles d'attaques envoyé et capté par l'IDS puis classer.

Nous avons envoyés 2273 paquets DOS, parmi eux 2202 paquets ont été détectés (Classé correctement). Le taux de classification de cette famille est de 97%.

Pour U2R : Nous avons envoyés 20 paquets, parmi eux 5 paquets ont été détectés (Classé correctement). Le taux de classification de cette famille est de 25%.

Pour R2L : Nous avons envoyés 62 paquets, parmi eux 11 paquets ont été détectés (Classé correctement). Le taux de classification de cette famille est de 18%.

Pour Probe : Nous avons envoyés 56 paquets, parmi eux 32 paquets ont été détectés (Classé correctement). Le taux de classification de cette famille est de 57%.

La figure ci-dessous représente les différent taux de classification de chaque famille.

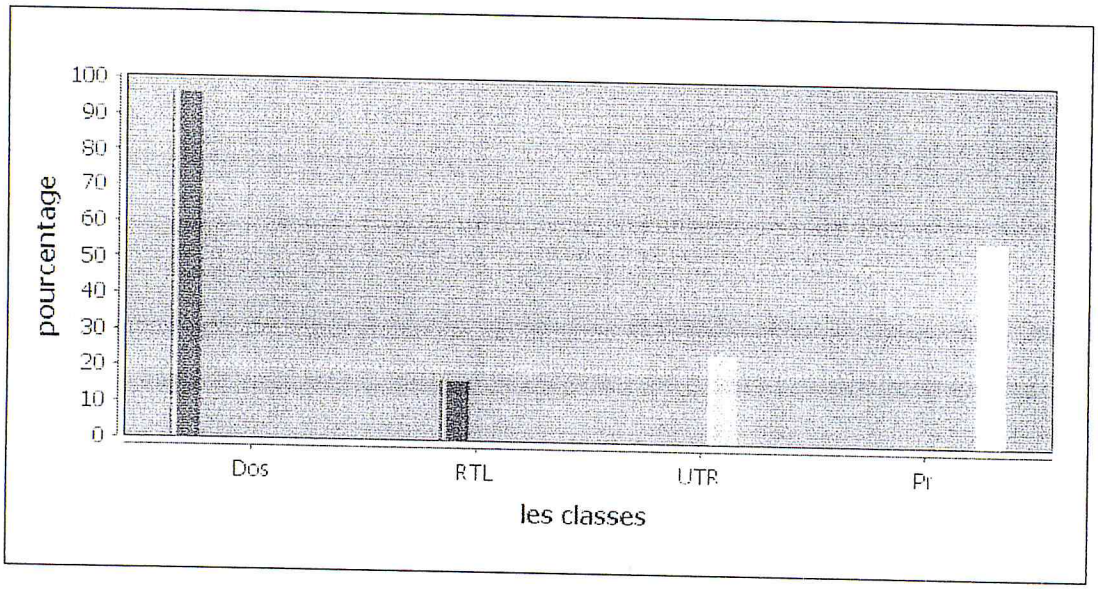


Figure 16: le taux de classification des attaques

Le taux de classification de la famille DOS est de 97%, Probe : 57%, U2R : 25% et la famille R2L : 18%.

Ces différents taux de classification reviennent en premier lieu à la composition de l'ensemble d'apprentissage que nous avons utilisé (Ensemble_4). Il se compose de 93% d'attaques DOS, 1% Probe, 2% R2L, 1% U2R. C'est pour cela les différents taux de classifications se varient selon la composition de l'ensemble.

3. Analyse des résultats

Un IDS est un outil qui a pour vocation la surveillance d'un ou plusieurs réseaux de machine, Il permet de détecter des attaques ou des évènements suspects.
Il a besoin d'une base d'apprentissage performante pour qu'il puisse effectuer ses taches en temps réel. Cette base doit être réduite et ne contient que des instances significatives.

L'utilisation de l'ensemble «Train» comme base d'apprentissage engendrera des pertes de paquets et un retard dans la classification (à cause de sa taille). Pour cela, nous l'avons remplacé avec notre ensemble réduit (Ensemble_4).

D'après les résultats du test, le taux de classification est de 93.4%.
Tous les paquets envoyés ont été capturés et classé (aucun paquet n'a été perdu), ce qui valide notre approche (l'ensemble réduit est performent).

Notre ensemble se caractérise par :

- Sa taille : Comme il est réduit, le temps d'exécution est minime ce qui nous permet une classification en temps réel.

- Sa précision : Comme il ne contient que des instances significatives, le taux des faux positifs est minimisé.

IV. Conclusion

Dans ce chapitre, nous avons décrit l'ensemble KDD'99 et les tests qui ont été effectué avec pour voir l'efficacité de nos algorithmes ainsi que leur validation.

Dans la première partie nous avons présenté l'ensemble KDD'99, sa composition et les familles d'attaques qu'il contient ensuite nous avons parlé des ensembles utilisés lors des tests et l'évaluation des ensembles résultant.

Nous avons achevé ce chapitre par un ensemble de tests effectué dans un réseau réel, ils nous ont permis en premier lieu de voir la différence entre les résultats de l'ensemble « Train » et notre ensemble ainsi que les améliorations apportées par la réduction, en second lieu nous avons vu les fruits de notre travaille notamment la classification en temps réel dans un IDS en utilisant notre meilleur ensemble résultant.

CONCLUSION GÉNÉRALE

Conclusion Générale

Les systèmes de détection d'intrusions réseau représentent une technologie très récente et en cours de développement et c'est un domaine qui intéresse de près les scientifiques ainsi que les industrielles. Ces systèmes commencent à prendre une importance capitale dans notre environnement informatique actuel, en revanchent ils produisent un nombre important des fausses alertes du à différentes raisons. L'une des principales raisons est la composition de la base d'apprentissage utilisée pour la classification d'alertes.

Parmi les bases d'apprentissages les plus utilisées, on trouve la KDD CUP 99 qui est constitué d'un ensemble de model (on l'appelle aussi instance) signé comme normal ou attaque. Cette dernière présente des défauts majeurs comme la redondance des données, l'existence des instances bruitées (ce qui fausse la classification) ainsi que sa grande taille.

Le projet qui nous a été confié consiste en la réduction des données d'apprentissage pour la classification d'alertes à base de KNN. Nous avons présenté quelques approches existantes qui traitent le problème de la réduction et la classification des données.

Dans ce mémoire nous avons proposé une approche de réduction des données d'apprentissages en se basant sur l'algorithme d'optimisation avec les colonies de fourmis et l'algorithme d'apprentissage actif. Le principe général de l'algorithme des colonies de fourmis est que chaque fourmi parcourt tous l'ensemble et nous génère une solution. L'algorithme d'apprentissage actif vient pour améliorer les résultats de l'algorithme précédent de manière qu'il ne prend que les instances significatives de l'ensemble d'apprentissage afin de le réduire au maximum et garder une bonne précision. Pour l'évaluation des résultats, nous avons utilisé un classificateur basé sur l'algorithme des K plus proche voisin(KNN).

Pour valider notre approche, nous avons mené des tests en deux phases :

La première consiste à réduire l'ensemble de départ et évaluer ses résultats avec un classificateur 1NN pour vérifier la précision. La deuxième phase consistait à évaluer notre meilleur ensemble dans un réseau réel.

D'après les résultats, nous avons obtenu un ensemble performant qui se caractérise par :

- Sa petite taille : Il représente 8.44% de la taille de l'ensemble initial, ce qui nous permet une classification en temps réel.

- Sa précision : Comme il ne contient que des instances significatives, le taux des faux positifs est minimisé.

En perspectives, il serait intéressant de tester notre approche sur des ensembles plus grands.

BIBLIOGRAPHIE

Bibliographie

[1] Vincent GLAUME, " Détection d'Intrusion Réseau et Système ", vol. 92, pp.7-68, 2008.

[2] Odile PAPINI, " DÈTECTION D'INTRUSIONS", Université de Toulon et du Var.

[3] Guillaume Hiet, " Détection d'intrusions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous Linux pour les programmes Java ", université de RENNE 1, vol.152, pp.19-21, 2008.

[4] David Burgermeister et Jonathan Krier, " Les systèmes de détection d'intrusions ", vol.61, pp.13-17, 2006.

[5] Nicolas Baudoin et Marion Karle, " IDS et IPS ", vol.30, pp.11, 2003-2004

[6] Nathalie Dagorn, " Détection et prévention d'intrusion : présentation et limites ", Rapport de recherche, Université de Nancy1.

[7] Nicolas Turenne, " Méthode des KNN (K nearest neighbours, ou *k plus proches voisins*)", 2006.

[8] Alain PUJOL, " Contributions à la Classification Sémantique d'Images ", Ecole Centrale de Lyon, 48 et 49, 12 juin 2009

[9] " Apprentissage par estimation de densité Naïve Bayes et KNN ", Geneva Lab, vol.33, pp.31-32, 29 Septembre 2009

[10] Pascal Vincent, " Modèles à noyaux à structure locale ", Université de Montréal Faculté des études supérieures, vol.163, pp.29, Octobre 2003

[11] Yang Li and Li Guo. Tcrn-knn algorithm for supervised network intrusion detection. *Computers & security*, 26:459467, 2007.

[12] Li Y. Optimizing network anomaly detection scheme using instance selection mechanism.*Global Telecommunications Conference*, GLOBECOM: 17, 2009.

[13] Kwok Ho Law and Lam For Kwok. Ids false alarm filtering using knn classifier. In *WISA'04 Proceedings of the 5th international conference on Information Security Applications, pp. 114-121*, 2005.

[15] Benjamin DEV`EZE et Matthieu FOUQUIN, DATAMINING C4.5 – DBSCAN, 14-16 rue Voltaire 94270 Kremlin Bicêtre, 11, 2005.

[16] COSTANZO Andrea LUONG Thé Van MARILL Guillaume, « Optimisation par colonies de fourmis », vol. 33, pp. 4-6, 2006.

[17] Benoît Gandar, Gaëlle Loosli et Guillaume Deffuant, Sélection de points en apprentissage actif Discrépance et dispersion : des critères optimaux, MajecSTIC, novembre 2009.

[18] Karima SEDKI, " Raisonnement sous incertitude et en présence de préférences : Application à la détection d'intrusions et à la corrélation d'alertes ", Université d'Artois, vol.175, pp.139-141, 04 Décembre2008.

[19] Mr SEDJELMACI Sid Ahmed Hichem, " MISE EN OEUVRE DE MECANISMES DE SECURITE BASES SUR LES IDS POUR LES RESEAUX DE CAPTEURS SANS FIL ", L'UNIVERSITE DE TLEMCEN, vol.127, pp.58, 2012.