République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et Recherche Scientifique
Université Saad Dahlab de Blida 1

Faculté des sciences
Département d'Informatique

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Informatique

Option : Ingénieure de logiciel et Sécurité des Systèmes d'Informations

Système de détection d'intrusion avec une approche D'apprentissage automatique

Présenté par :

- BENHASSINE Oualid
- BOUTEBALI Imadeddine

Encadreur :

- Dr. Bougharra Abedelrahim

Promoteur:

- Dr. CHIKHI Nacim Fateh

Soutenu le : 16/07/2019

Devant le jury :

-Mme. FAREH	Président
-Mme. ZAHRA	Examinateur
-M. BOUGHARRA Abed El Rahim	Encadreur
-M. CHIKHI Nacim Fateh	Promoteur

Année universitaire : 2018/2019

Remerciements

Tout d’abord, nous remercieront Allah de nous avoir aidé et donné la force et la volonté de réaliser ce travail

Ensuite, nous tenons à exprimer nos plus vifs remerciements et gratitude à notre promoteur Mr Chikhi Nacim Fateh pour son encadrement continu, pour les remarques constructives Qu’il nous a fournies ainsi que pour ses précieux conseils durant toute la période de notre travail. On le remercie également pour la confiance qu’il nous a accordée et pour la grande liberté d’idées et de travail qu’il nous a donnée. Nous n’oublierons pas aussi de les remercier pour ses qualités humaines, son hospitalité et son soutien qui ont permis de bien mener bien cet ouvrage.

Nous tenons à remercier les membres du jury d'avoir bien voulu participer à l'évaluation de ce travail.

Quelques personnes ont contribué à la réalisation de ce travail et méritent des remerciements.

Je suis également redevable notre encadreur Bougharrara Abedrahim, qui croyait en notre capacité à réaliser quelque chose de grand. En nous donnant l'opportunité de poursuivre ce rêve et de le réaliser. Avec un cœur plein de gratitude.

Enfin, nous tenant à remercier nos familles pour leur encouragement, leur aide et leur grande patience avec nous.

ملخص

يمكن تعريف نظام كشف التسلل بأنه نظام آلي يتمثل دوره في اكتشاف الاختراقات في نظام الكمبيوتر، وهناك نوعان، الأول يعتمد على التوقيع والآخر يعتمد على الحالات الشاذة، الهدف من هذا العمل الماجستير هو إيجاد حلول للحد من الإنذارات الخاطئة عن الحالات الشاذة القائمة على ن.ك.ت باستخدام طريقة الفرقة التقليدية وطريقة جديدة مقترحة.

مزيج من ن.ك.ت والتعلم الآلي يعطي ن.ك.ت أفضل من ن.ك.ت المستندة إلى التوقيع، مجموعة البيانات المستخدمة هي*KDDcup-99* حقق الأسلوب الجديد المقترح نتائج جيدة مقارنةً بالمصنف القياسي

الكلمات المفتاحية: نظام كشف التسلل، تقنية الكشف، طرق التجميع، التعلم الآلي، أمن الشبكات، التراص، التعزيز، التعبئة، مصنف التصويت، ك أقرب جار، غابة عشوائية، شجرة القرار، الشبكة العصبية الاصطناعية، KDD99.

Résumé

Un système de détection d’intrusions peut se définir comme un système automatisé dont le rôle est la détection des intrusions dans un système informatique, il existe deux types, le premier à base des signatures et l’autre à base anomalies, l’objectif de ce travail de Master est de trouver des solutions pour réduire les fausses alarmes pour l’IDS base a anomalies en utilisant les méthodes d’ensemble traditionnelle et une nouvelle méthode proposée. La combinaison entre IDS et apprentissage automatique donne un IDS plus performant que l’IDS a base de signature. Le jeu de données utilisé c’est KDDcup-99. La nouvelle méthode proposée a eu de bons résultats par rapport au classificateur Standard.

Mots-Clés : Système de détection d’intrusion, Technique de détection, méthodes d’ensemble, apprentissage automatique, Sécurité des réseaux, KDD99, Stacking, Boosting, Bagging, Voting Classifier, KNN, Random Forest, Decision Tree, ANN.

Abstract

An intrusion detection system can be defined as an automated system whose role is the detection of intrusions in a computer system, there are two types, the first is signature based and the other anomalies based, the objective of This Master's work is to find solutions to reduce false alarms for anomalies based IDS using the traditional ensemble methods and a new proposed method. The combination of IDS and machine learning give a better IDS than the signature based IDS. The data set used is KDDcup-99 The new proposed method has had good results compared to the standard classifier.

Keywords: Intrusion Detection System, Detection Technique, Ensemble Methods, Machine Learning, Network Security, KDD99, Stacking, Boosting, Bagging, Voting Classifier, KNN, Random Forest, Decision Tree, ANN, KDD99.

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Vu le développement de la cybercriminalité, les failles de sécurité d'un système informatique peuvent avoir des conséquences désastreuses sur une organisation, qui peuvent aller de la perte financière aux atteintes à la réputation de cette dernière jusqu''a la faillite.

Les attaques informatiques ne concernent pas uniquement les entreprises et les sociétés mais touchent de plus en plus les gouvernements et les structures sensibles des pays et même les individus. A tel point que les services de sécurité ont clairement émis des menaces, en affirmant qu'ils envisageraient toutes les options possibles en cas de cyber-attaques.

Par conséquent, les gouvernements et entreprises mettent en avant une ligne de défense surarmée, à première vue, pour protéger les systèmes des éventuelles attaques. Cette ligne comprend des systèmes d'authentification des utilisateurs, des scanners d'emails et de vulnérabilité, des anti-virus, des pare-feu, etc. Il est à noter que le dispositif qui représente la dernière barrière de protection reste bien le système de détection d'intrusions.

Avec la multiplication des réseaux informatiques notamment Internet, la sécurité informatique devient un sujet crucial et sensible, étant donné l'accroissement des menaces d'attaques et leurs diversités ce qui a engendré un développement continu de la sécurité entrainant, par-là, l'essor des systèmes de détection d'intrusions, ce qui a mené ce dernier à inclure l'apprentissage automatique dans son processus de détection dans le but d'exploiter les données disponibles sur les attaques.

Chapitre I

Introduction à la détection d'intrusion

I.1. Introduction

Dans ces derniers temps, la nécessité de protéger les ressources informatiques se fait plus pressante de jour en jour. Afin d'arriver à assurer une protection maximale des ressources, de nombreuses technologies plus performantes les unes que les autres ont été développées. De tous les systèmes réalisés jusqu'à maintenant, nous nous sommes intéressés aux systèmes de détection d'intrusions (IDS).

Dans le chapitre courant, nous allons expliquer plusieurs termes et définitions relatifs à notre thème que nous avons jugé nécessaires à connaitre pour une bonne compréhension du sujet.

Nous résumons aussi les travaux qui ont été faits sur l'évaluation des Systèmes de Détection d'Intrusions suivis d'une petite comparaison entre ces travaux de recherche.

I.2. Définitions

I.2.1. Intrusion

Une intrusion peut être considérée comme l'ensemble d'actions qui ont pour but de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource. Ces actions de franchissement d'un accès non-autorisé ou de manipulation interdite d'une ressource, peuvent être menées par un individu externe n'ayant aucun privilège sur les ressources d'un système, ou par un individu interne qui outrepasse ses privilèges.

I.2.2. Détection d'intrusion

La détection d'intrusion est un ensemble de techniques et de méthodes employées dans l'analyse des informations collectées par les mécanismes d'audit de sécurité pour détecter toute activité suspecte au niveau du réseau et ses hôtes.

I.2.3. Système de détection d'intrusion

C'est une combinaison de logiciel et de matériel qui essaie de réaliser la détection d'intrusions. Un système de détection d'intrusions peut se définir comme un

système automatisé dont le rôle est la détection des intrusions dans un système informatique tout en examinant les audits de sécurité fournis par le système d'exploitation ou bien les outils de contrôle du réseau. Son but principal est la détection des utilisations non autorisées, les mauvaises utilisations et les abus dans un système informatique par les utilisateurs internes et externes [1].

Dans [2], un système de détection d'intrusions est défini comme étant comparé à une alarme de cambriolage. Par exemple, le système de serrure dans une voiture protège la voiture contre le vol. Mais si quelqu'un casse le système de serrure et essaie de voler la voiture, c'est l'alarme anti cambriolage qui détecte que la serrure a été cassée et alerte le propriétaire en donnant une alarme. Le système de détection d'intrusions d'une manière similaire complète la sécurité du pare-feu. Le pare-feu protège un système contre des attaques malveillantes et le système de détection d'intrusions détecterait si quelqu'un tente de passer le pare-feu et d'accéder au côté sûr du système, et alerte le gestionnaire au cas où il y aurait infraction dans la sécurité [2].

I.3. Types des systèmes de détection d'intrusion

Il existe deux types majeurs d'IDS :

I.3.1. IDSs à base de signature

Généralement, les IDS réseaux se basent sur un ensemble de signatures qui représentent chacune le profil d'une attaque. Une approche à base de signature consiste à rechercher dans un flux réseau les empreintes d'attaques connues, à l'instar des antivirus.

Une signature est définie comme une séquence d'événements et de conditions relatant une tentative d'intrusion. La reconnaissance est alors basée sur le concept de" Pattern Matching". Si une attaque est détectée, une alarme peut être remontée si l'IDS est en mode actif, sinon, l'IDS se contente d'archiver cette attaque [3].

I.3.2. IDSs à base d'anomalie

Les IDSs à base d'anomalie dont le déploiement nécessite une phase d'apprentissage pendant laquelle l'outil va apprendre le comportement "normal" des flux applicatifs présents sur son réseau. Ainsi, chaque flux et son comportement

Standard doivent être déclarés. L'IDS se chargera d'émettre une alarme si un flux anormal est détecté, mais ne pourra pas spécifier la criticité de l'éventuelle attaque. Les IDS comportementaux sont apparus bien plus tard que les IDS à signature et ne bénéficient pas encore de leur maturité. Ainsi, l'utilisation de tels IDS peut s'avérer délicate dans le sens où les alarmes remontées pourraient contenir une quantité importante de fausses alertes [3].

I.4. Familles de système de détection d'intrusion

Les IDSs peuvent se classer selon trois catégories majeures selon qu'ils s'attachent à surveiller :

I.4.1. Les NIDSs (Network Based Intrusion Detection Systems)

Un NIDS se découpe en trois grandes parties : la capture, les signatures et les alertes. Il écoute donc tout le trafic réseau, puis analyse les flux en transit sur le réseau et génère des alertes si des paquets semblent dangereux (voir figure 1).

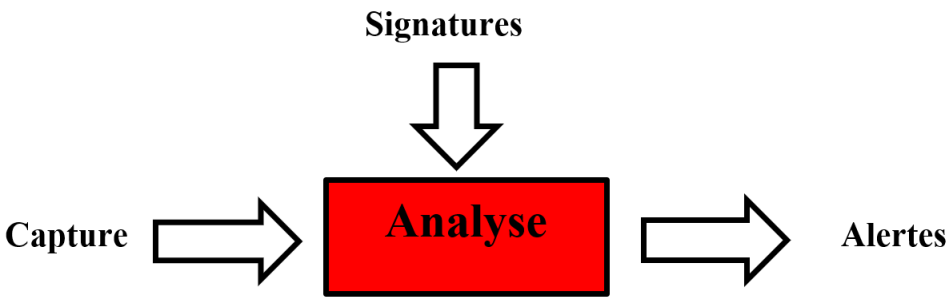


Figure 1: les composants d'un NIDS

Le rôle essentiel d'un NIDS est l'analyse et l'interprétation des paquets circulant sur ce réseau. L'implantation d'un NIDS sur un réseau se fait de la façon suivante : des capteurs sont placés aux endroits stratégiques du réseau et génèrent des alertes s'ils détectent une attaque. Ces alertes sont envoyées à une console sécurisée, pour les analyser et les traiter éventuellement [4].

Les capteurs du réseau sont placés en mode furtif (ou stealth mode), de façon à être invisibles aux autres machines. Pour cela, leur carte réseau est configurée en mode « promiscuous », c'est à dire le mode dans lequel la carte réseau lit l'ensemble du trafic, de plus aucune adresse IP n'est configurée [4].

I.4.2. Les HIDSs (Host Based Intrusion Detection Systems)

Le HIDS réside sur un hôte particulier, il analyse exclusivement l'information concernant cet hôte. Le HIDS se comporte comme un démon ou un service standard sur un hôte serveur/système. De plus, l'impact sur la machine concernée est sensible immédiatement, par exemple dans le cas d'une attaque réussie par un utilisateur. Ces systèmes de détection d'intrusions utilisent deux types de sources pour fournir une information sur l'activité de la machine : les logs et les traces d'audit du système d'exploitation : les traces d'audit sont plus précises et détaillées et fournissent une meilleure information alors que les logs qui ne fournissent que l'information essentielle sont plus petits (voir figure 2) [4].

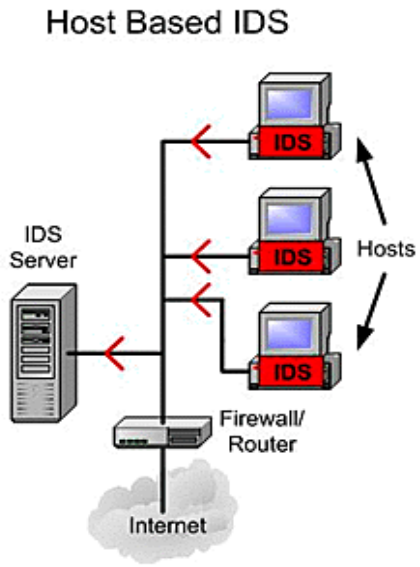


Figure 2: IDS basé sur l'hôte [3]

I.4.3. Les IDSs hybrides

Les IDS hybrides rassemblent les caractéristiques de plusieurs IDS différents. En pratique, on ne retrouve que la combinaison de NIDS et HIDS. Ils permettent, en un seul outil, de surveiller les réseaux et les terminaux. Les sondes sont placées en des points stratégiques, et agissent comme NIDS et/ou HIDS suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser le tout (Figure 3), et agréger/lier les informations d'origines multiples [5].

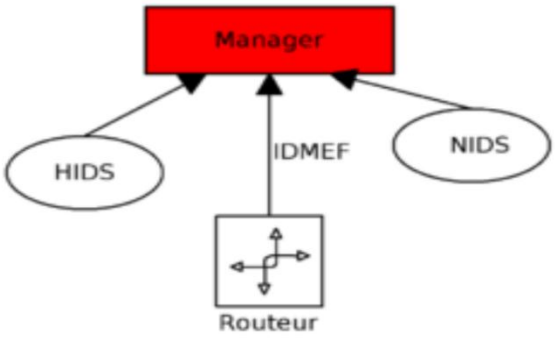


Figure 3: IDS hybride

Les IDSs hybrides sont donc basés sur une architecture distribuée, où chaque composant unifie son format d'envoi d'alerte (Exemple typique : IDMEF : Intrusion Detection Message Exchange Format) permettant à des composants divers de communiquer et d'extraire des alertes plus pertinentes.

I.5. Evolution des IDSs

Les systèmes de détection d'intrusions ont beaucoup évolué depuis le modèle proposé par [6]. Plusieurs techniques ont été introduites afin d'améliorer leurs performances et rendre leur détection plus précise. Dans ce qui suit nous allons présenter un ensemble de systèmes des plus connus.

I.5.1. IDES

Comme cité précédemment, le premier modèle d'IDS "IDES" (Intrusion Detection Expert System) a été réalisé par Dorothy et Peter Denning [6]. Ce modèle fait appel à des techniques statistiques afin de caractériser un comportement anormal, et se base sur un ensemble de règles pour détecter les violations. Dorothy a émis, plus tard [6], qu'il était possible de détecter des intrusions dans un système informatique indépendamment de ce dernier, des applications installées et de sa vulnérabilité, et cela à travers un modèle reformulant les comportements des utilisateurs par les systèmes de détection d'intrusion.

I.5.2. Haystack

”Haystack” [7] représente une variante de IDES. Ce prototype [7] a été développé pour la détection des intrusions dans un environnement multi-utilisateurs de l’Air Force Computer System (il s’agissait de la plateforme standard de l’Air force à l’époque) [8]. Pour détecter les intrusions, le système utilise deux méthodes de détection : la détection d’anomalies et la détection à base de signature.

I.5.3. MIDAS

MIDAS [9] a été développé par le centre national de la sécurité (NCSC), en collaboration avec le laboratoire informatique international SRI, pour fournir une détection d’intrusions pour les réseaux mainframe. Les auteurs se sont inspirés des travaux antérieurs de Denning et al. [10]. Ce modèle est construit autour de l’idée de détection d’intrusions heuristique et comporte une base de règles appelée P-Best écrite en Lisp.

Après MIDAS les recherches se sont dirigées vers l’utilisation des systèmes experts dans l’écriture des règles [11], ce qui permet une mise à jour dynamique de ces règles.

I.5.4. Discovery

L’IDS Discovery regroupe une approche hybride basée sur les statistiques et les systèmes experts [12]. Cet IDS analyse les fichiers journaux d’une application. Discovery nécessite des méthodes statistiques pour la détection de profiles et se base sur un système expert pour la détection d’intrusions.

I.5.5. Wisdom & Sense

Wisdom & Sense ou WRS [13] est un système de détection d’anomalies, qui a été développé entre 1984 et 1989. Il est intéressant de noter que WRS à sa création n’a pas été destiné à être appliqué à la sécurité informatique, mais plutôt à un problème lié au contrôle de matières nucléaires [14]. Ce système est unique dans son approche de détection d’anomalies : il étudie l’historique des données d’audit et produit un arbre de règles décrivant le comportement normal, ces règles sont ensuite introduites dans un système expert qui évalue les données de vérification récentes et déclenche une alerte lorsque les règles indiquent un comportement anormal.

I.5.6. NSM

NSM [15], [16] a été le premier système à utiliser directement le trafic réseau entant comme source de données d'audit. Cet IDS écoute passivement tout le trafic qui passe par un réseau local de diffusion et en déduit le comportement intrusif. L'idée de cette approche découle de l'observation que plusieurs autres systèmes de détection d'intrusions essayaient d'atténuer les problèmes des différentes plateformes en suivant leurs pistes d'audit.

I.5.7. Hyperview

Hyperview [17] est un système de détection d'intrusions fondé sur deux composantes principales. La première consiste en un système expert qui surveille les pistes de vérification de signes d'intrusions connues de la communauté de sécurité. La seconde est une composante de réseau de neurones qui apprend le comportement d'un utilisateur de manière adaptative et envoie une alarme quand la piste d'audit s'écarte du comportement appris auparavent.

I.5.8. DIDS

DIDS [18] est un système de détection d'intrusions distribué incorporant Haystack et NSM, vues précédemment. DIDS est constitué de trois composantes principales. Sur chaque hôte, un moniteur effectue la détection locale d'intrusion, résume les résultats et les communique au directeur DIDS. En outre, chaque segment broadcast du LAN possède ses propres moniteurs qui surveillent le trafic dans ce LAN et rapportent au directeur du DIDS leurs observations. Finalement, le directeur DIDS, qui est un système constitué du responsable de communication et d'un système expert, analyse les observations des moniteurs et en communique les résultats à l'agent de sécurité du système (SSO).

I.5.9. IDIOT

IDIOT [19] [20] [21] [22] [23] est un système qui a été développé à COAST (maintenant le Centre pour l'éducation et la recherche en sûreté de l'information et de la sécurité (CERIAS), http ://www.cerias.purdue.edu). Le principe de base derrière IDIOT est d'utiliser les réseaux de Pétri colorés pour une détection d'intrusions à base de signatures. Les auteurs suggèrent qu'une approche en couches doit être utilisée lors de l'application de techniques à base de signatures (pattern matching) pour résoudre

le problème de détection d'intrusions. Les auteurs arguent, du fait que de nombreuses techniques de filtrage disponibles, les réseaux de Petri colorés (CP-nets) seraient la meilleure technique à appliquer, car elle ne souffre pas de certaines lacunes communes dans d'autres techniques, qui ne permettent pas l'assortiment conditionnel des modèles, et ne se prêtent pas à une représentation graphique.

I.5.10. NIDES

NIDES [24] [25] est le successeur du projet IDES. NIDES suit les mêmes principes généraux que les versions ultérieures de l'IDES : il a une base forte de détection d'anomalies, complétée par un composant de système expert à base de signatures. Ce dernier est mis en œuvre en utilisant un système expert P-BEST. Le système NIDES est fortement modulaire, avec des interfaces bien définies entre les composants, construit sur une architecture client-serveur. Il est centralisé dans la mesure où l'analyse s'exécute sur un hôte spécifique et recueille des données de différents hôtes à travers un réseau informatique.

I.5.11. GrIDS

GrIDS [26] permet de construire des graphiques représentant l'activité du réseau pour faciliter la détection d'intrusions, particulièrement dans les grands réseaux. Les graphiques codifient les hôtes sur un réseau comme des nœuds, et les connexions entre les hôtes comme des raccords entre ces nœuds. Le choix du trafic fait pour représenter l'activité sous forme de bords est effectué sur la base d'ensembles de règles écrites par l'utilisateur. Les événements du réseau sont représentés à travers un mode graphique qui permet à l'observateur de déterminer la présence d'une activité suspecte. Il serait contraignant de reporter toute l'activité réseau dans un même graphique, par conséquent, le système permet pour plusieurs ensembles de règles de définir un graphe pour chaque ensemble. Dans ce cas, toutes les données recueillies sont prises en considération pour définir l'inclusion dans tous les ensembles de règles, et donc deux ensembles de règles différents pourraient rendre ainsi les mêmes données d'audit que deux graphiques différents.

I.5.12. EMERALD

L'environnement EMERALD [27], [28] est une suite d'outils distribués et modulaires permettant de traquer les activités malicieuses dans un réseau à large

échelle. EMERALD contient ainsi un module de détection d'anomalies basé sur les travaux initiés pour NIDES. Alors que NIDES est utilisé pour définir un profil utilisateur, le module intégré dans EMERALD a été étendu afin de pouvoir définir un profil réseau. EMERALD est ainsi constitué d'un ensemble de moteurs de génération de profil (engine profiler) permettant une séparation totale entre la représentation du profil et les algorithmes mathématiques utilisés pour évaluer les nouvelles observations.

I.5.13. ADAM

Le projet ADAM (Audit Data Analysis and Mining) [29] est un IDS réseau comportemental basé-anomalies. Son objectif est d'apprendre de ces attaques et de les représenter sous forme d'un jeu de règles appelé " Profils ". Il est composé de trois modules, un moteur de prétraitement, un moteur de " Mining " et un moteur de classification. Il fonctionne en temps réel et utilise un mode de Mining incrémental. L'analyse se fait en deux phases. La première phase permet de créer un modèle de profil dit " normal ". Ce modèle permet d'en générer un modèle de référence à travers des règles. Une fois cette étape exécutée, la seconde phase a pour but de capturer les flux de données entrants, de les comparer avec les règles " normales " puis lors de la détection d'un comportement suspicieux, ce dernier est automatiquement notifié soit comme attaque ou comme fausse alarme.

I.5.14. MADAM ID

Le projet MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) [30] a été créé dans le but de montrer comment les techniques de data mining pouvaient être utilisées dans la construction d'IDS dans une approche plus systémique et de façon automatisée. L'approche utilisée est de définir un système de classification intelligent qui aura la capacité de distinguer les activités normales des intrusions. Malheureusement, les moteurs de classification voient leurs performances limitées car ils ne sont qu'un chainon final de l'analyse technique. C'est dans ce but que MADAM ID propose des règles d'associations permettant de définir des attributs plus prédictifs, ce qui offre aux moteurs de classification des mises à jour plus rapides pour leurs données. L'approche utilisée par MADAM ID est une détection par

scénario. Son but principal est de recenser les données auditées et d'en définir des modèles d'intrusions ou d'activités normales.

I.6. Déficiences des IDS

La capacité d'un IDS à détecter une attaque dépend de la présence au niveau de la source de données d'un capteur et de la capacité de l'analyseur à identifier l'activité comme étant intrusive. Cependant, le nombre d'événements remarquables détectables dépend aussi du fait que l'homme fasse partie du système, car en un jour de travail, l'administrateur du système enregistre et rapporte une quantité définie de détections, aussi, certains événements détectés ne sont pas classables ou pas assez documentés, et c'est les administrateurs qui doivent prendre des décisions à propos de ces événements. Ces décisions peuvent être souvent erronées [31].

La détection d'intrusions vise, normalement, à fournir un faible taux de faux négatifs (vraies alertes négligées) et de faux positifs (fausses alarmes). Néanmoins, utilisés en milieu opérationnel, les IDS émettent un nombre important de faux positifs, et délaissent un nombre important d'attaques, créant ainsi un nombre conséquent d'alertes fausses positives et d'autres fausses négatives.

I.7. Gestion des alarmes IDS

Traiter les alarmes est l'un des axes de recherche dans le domaine de la détection d'intrusions, cependant, peu de travaux de recherche ont été réalisés dans ce domaine pour le moment. Les principaux objectifs des recherches faites peuvent se classer en deux catégories :

1. Réduction du nombre de fausses alarmes

2. Regroupement des alarmes similaires et les présenter aux utilisateurs sous une forme simplifiée, afin que les administrateurs puissent mieux comprendre les rapports. Ce qui aide aussi à mieux comprendre l'origine des fausses alarmes.

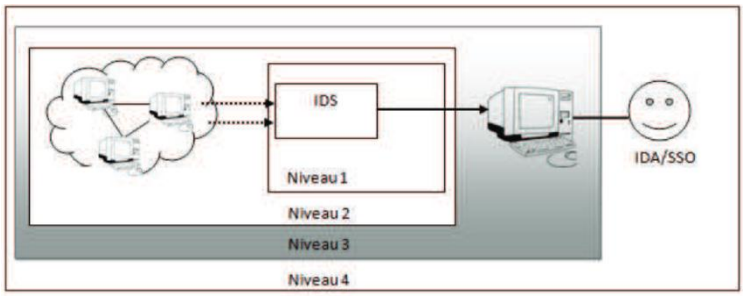


Figure 4: Evolution du périmètre pour traiter les faux positifs [32]

Après avoir mentionné les deux principaux objectifs de la recherche dans les fausses alarmes issues d'IDS, nous pensons que le data mining pourrait représenter une solution adaptée pour résoudre ce problème.

Afin de réduire le nombre de fausses alarmes, il est nécessaire de correctement classer ces dernières ainsi que les vraies alertes.

Tadeusz Pietrazek [30] a regroupé les approches relevant de la gestion d'alertes en quatre niveaux comme illustré par la (figure 4).

Nous allons dans ce qui suit présenter ces niveaux de manière ascendante.

Niveau 1 : Amélioration de l'IDS

Comme premières tentatives, la plupart des efforts se sont concentrés sur l'amélioration des capteurs et la création de capteurs permettant une meilleure détection d'intrusions ou produisant un faible taux de faux positifs. Les capteurs de détection d'intrusions ont évolué de simples moteurs de Pattern Matching aux capteurs intelligents spécialisés pouvant comprendre les protocoles de transport sous-jacents, et même des techniques sophistiquées telle que la détection du niveau de fragmentation IP ou TCP. Suivant la croissance des vulnérabilités au niveau applicatif [33], l'intelligence des IDS a elle aussi évolué afin d'accompagner cette croissance. Les IDS ont été améliorés / développés afin de traiter divers protocoles de niveau applicatif (http, RPC). Aussi, les langages de signatures sont devenus de plus en plus puissants supportant ainsi les expressions régulières (tel que Snort) et même les interactions de protocoles complexes. Ces IDS spécialisés, contrairement aux IDS généraux, se concentrent sur des types particuliers d'intrusions permettant en même temps un faible taux de faux positifs.

Enfin, nous pouvons noter que ces IDS à usage particulier doivent impérativement être complétés par des IDS supplémentaires afin d'atteindre une couverture aussi complète que possible contre toutes les attaques, ce qui soulève la nécessité de déployer et de maintenir un réseau hétérogène d'IDS complémentaires.

Niveau 2 : Tirer parti de l'environnement

Les IDS possèdent une vue limitée de l'environnement et ne peuvent donc pas distinguer, dans certains cas, avec certitude entre les attaques et les non-attaques. En utilisant des informations sur l'environnement, fournies par la vulnérabilité des scanners et des bases de données du système d'exploitation, les IDS en comparant leur environnement peuvent réduire de manière considérable le taux de faux positifs. Les principaux travaux traitant ce niveau se basent sur la corrélation d'alarmes.

Par exemple, Ptacek et Newsham [34] ont montré que, sans savoir comment les hôtes cibles gèrent certaines anomalies dans les paquets réseau, les intrus peuvent utiliser efficacement la fragmentation pour éviter d'être détectés par l'IDS réseau. Pour répondre à ces préoccupations, les approches d'active mapping [35] construisent des profils de l'environnement, qui peuvent ensuite être exploit́es par les IDS. Les approches basées sur les signatures de contexte [36], d'autre part, peuvent comprendre les interactions de protocole de niveau application et donc de déterminer l'impact d'une attaque. Un effet similaire peut être obtenu en mettant en corrélation des alertes de vulnérabilités [37] [38].

Niveau 3 : Post-traitement d'alarmes

Le post traitement d'alarmes utilise les alertes générées par un IDS en entrée et tente d'améliorer la qualité de ces alertes à travers leur traitement. Cela est obtenu en faisant appel au datamining et aux systèmes de corrélation d'alertes. Julisch [39] en se basant sur le data mining a montré que les techniques de data mining pouvaient se montrer très efficaces dans la découverte, l'éradication et la prévention des attaques fausses positives. En outre la corrélation d'alertes permet aussi de contrer le problème de faux positifs, mais aborde aussi le problème de redondance dans le flux d'alertes issues d'un IDS.

Niveau 4 : Implication de l'analyste

Peu de systèmes traitant l'un des niveaux précédemment cités tiennent compte du fait que les alertes générées par les IDS sont transmises à l'analyste humain.

I.7.1. Post-traitement d'alarmes

Le post-traitement d'alarmes alloue l'identification de nouvelles attaques ou description d'événements suspicieux. Le moteur analyse et traite les données par rapport aux règles et politiques afin de déterminer si une donnée est malveillante ou bénigne. Le posttraitement implique la corrélation d'alarmes et la collecte d'informations provenant à partir d'autres moteurs d'analyse. Cette collecte d'informations est principalement assurée à travers des mécanismes d'extraction de connaissances. Enfin, l'unité d'intervention décide pour chaque alarme du type de réponse nécessaire : active / passive. Le post-traitement souffre néanmoins de la limitation du nombre de ressources pour l'investigation, car il ne dispose que d'un nombre limité d'alarme pour le traitement.

I.7.1.1. Post traitement par corrélation d'alarmes

La corrélation d'alarmes permet de générer de nouvelles alarmes à partir de celles existantes. Elle consiste à combiner les informations fragmentées contenues dans la séquence d'alarmer et interpréter l'ensemble du flux d'alarmes .la corrélation offre principalement le filtrage des alarmes redondantes et le filtrage des alarmes de faible priorité, elle constitue une étape à préalable une contre-mesure efficace. Afin d'expliquer l'importance de la corrélation d'alarmes, nous prendre l'exemple d'une authentification échouée, cela génère une alerte de faible intensité. Mais s'il y a une série d'authentification échouées avec des utilisateurs différents, on peut conclure à une attaque de force brute.

I.7.1.2. Post traitement par datamining

L'extraction de connaissances, communément appelée datamining, est un domaine aujourd'hui très sollicité. Nous pouvons définir le datamining comme suit :

Le Data Mining est un ensemble de méthodes et techniques qui permettent la prise de décisions, à travers la découverte, rapide et efficace, de schémas d'informations inconnus ou cachés à l'intérieur de grandes bases de données. Selon David hand, « Le datamining est l'analyse d'un ensemble d'observations qui a pour

but de trouver des relations insoupçonnées et résumer les données d'une nouvelle manière, de façon qu'elles soient plus compréhensibles et utiles pour leurs détenteurs » [13].

Diverses méthodes d'apprentissage automatique et de datamining ont été proposées pour la détection d'intrusion, et ont eu un grand succès [40] [41]. Par exemple les arbres de décision et les règles d'association floues ont été utilisées dans la détection d'intrusions, Les réseau de neurones, eux, ont été utilisés pour améliorer les performances de détection d'intrusion [42].

C. Discussion

Tableau 1: Comparaison des techniques de gestion d'alertes [10]

Technique	Points forts	Faiblesses
Corrélation d'alarmes	-Donne un aperçu sur les relations entre les alarmes, leurs causes ainsi qu'une présentation claire de ces dernières	-Ne permet pas de réduire le nombre d'alarmes fausses positives
Datamining	-Augmentation des performances des moteurs de recherche - Permet de créer des liens pertinents entre les données qui paraissent n'avoir aucune corrélation	- Le datamining est totalement dépendant de la base donnée qu'il analyse et donc de sa taille

I.8. Conclusion

Ce chapitre a fourni une brève introduction aux notions et aux différents paradigmes utilisés dans le cadre de ce travail.

Nous avons donné les définitions essentielles du système détection d'intrusion suivi de leur types, familles d'IDS et mentionné l'évolution. A la fin on a expliqué brièvement la gestion des alarmes d'IDS

D'après les études effectuées, l'algorithme de classification est très important.

Dans le chapitre suivant nous présenterons les principales techniques de classification utilisées dans la détection d'intrusion.

Chapitre II

Apprentissage automatique

II.1. Introduction

Suite au développement rapide des différentes méthodes et techniques de piratage et avec le nombre très important d'attaques effectuées dans le monde entier, la protection des données cyber structurelle a connu une très grande vulnérabilité causée par l'incapacité de suivre le rythme d'évolution de la cybercriminalité, pour cela, les chercheurs en sécurité informatique ont utilisé les différentes techniques d'apprentissage automatique, de statistique et de data mining, afin de relever les défis de la cyber sécurité.

Dans ce deuxième chapitre, nous allons présenter les principaux algorithmes d'apprentissage automatique et les différentes Techniques de détection basées sur l'apprentissage automatique.

II.2. Définition d'apprentissage automatique

L'apprentissage automatique ou statistique, aussi appelé « *machine learning* » est un domaine à la jonction des statistiques et de l'intelligence artificielle qui a pour but la résolution automatique de problèmes complexes à partir d'exemples. La démarche de conception d'un modèle par apprentissage nécessite de postuler une fonction, dont les variables sont susceptibles d'avoir une influence sur la grandeur à modéliser. Cette fonction dépend des paramètres ajustables.

L'apprentissage statistique consiste en l'ajustement de ces paramètres de telle manière que le modèle ainsi obtenu présente les qualités requises d'apprentissage et de généralisation [43].

II.3. Différents types d'apprentissage

II.3.1. Apprentissage supervisé

Dans l'apprentissage supervisé, les données fournies sont des paires : une entrée et une étiquette. On parle alors d'entrées étiquetées. Le but de l'apprentissage est d'inférer la valeur de l'étiquette étant donnée la valeur de l'entrée. On peut

distinguer deux grands types d'apprentissage supervisé : la classification et la régression [44].

II.3.1. 1. Classification

Lorsqu'on fait de la classification, l'entrée est l'instance d'une classe et l'étiquette est la classe correspondante. La classification consiste donc à apprendre une fonction f class de X= Rd dans Y= N qui associe à un vecteur sa classe. Si le nombre de classe est égal à 2, on parle alors de la classification binaire [43].

II.3.1. 2. Régression

Dans le cas de la régression, l'entrée n'est pas associée à une classe mais à une ou plusieurs quantités continues. Ainsi, l'entrée pourrait être les caractéristiques d'une personne (son âge, son sexe, son niveau d'études) et l'étiquette son revenu. La régression consiste donc à apprendre une fonction f_{regr} de $X = R_d$ dans $Y = R_k$ qui associe à un vecteur sa valeur associée [43].

II.3.1.3. Les algorithmes plus connus

II.3.1.3. 1. Régression des moindres carrés ordinaires : Les moindres carrés sont une méthode permettant d'effectuer une régression linéaire. Vous pouvez considérer la régression linéaire comme une tâche consistant à ajuster une ligne droite à travers un ensemble de points. Il existe plusieurs stratégies possibles pour ce faire, et la stratégie des « moindres carrés ordinaires » est la suivante : vous pouvez tracer une ligne, puis pour chacun des points de données, mesurer la distance verticale entre le point et la ligne et les additionner. ; la ligne aménagée serait celle où cette somme de distances est la plus petite possible [45].

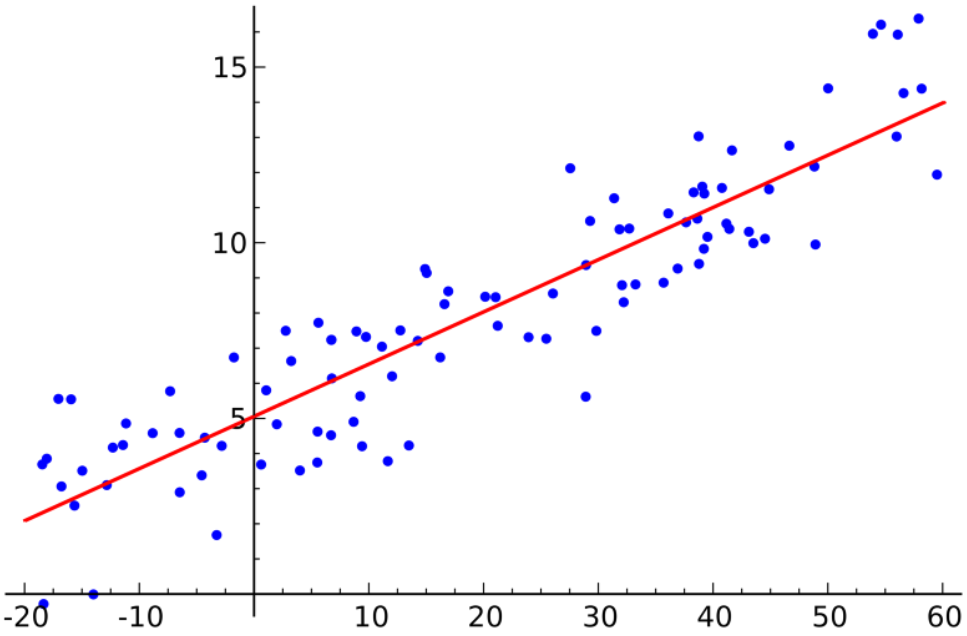


Figure 5 : Régression des moindres carrés ordinaires [45].

Linéaire fait référence au type de modèle que vous utilisez pour ajuster les données, tandis que les moindres carrés correspondent au type de mesure d'erreur que vous minimisez [45].

II.3.1.3. 2. Régression logistique : La régression logistique est un moyen statistique puissant de modéliser un résultat binomial avec une ou plusieurs variables explicatives. Il mesure la relation entre la variable dépendante catégorique et une ou plusieurs variables indépendantes en estimant les probabilités à l'aide d'une fonction logistique, qui est la distribution logistique cumulative.

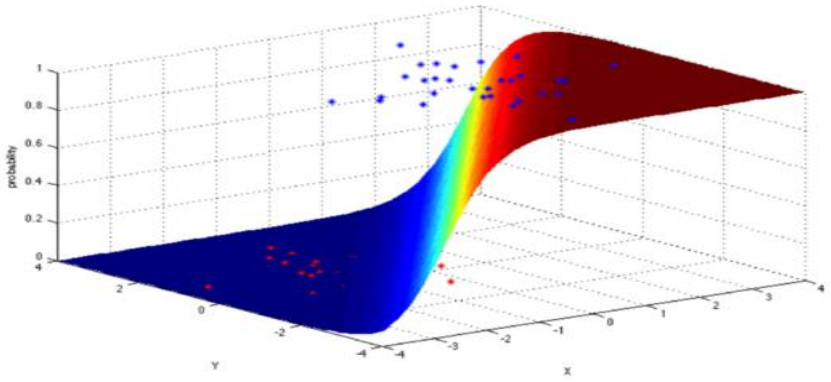


Figure 6: Régression logistique [45].

En général, les régressions peuvent être utilisées dans des applications du monde réel telles que :

- Pointage de crédit.
- Mesurer les taux de réussite des campagnes marketing.
- Prédire les revenus d'un produit donné.
- Est-ce qu'il va y avoir un tremblement de terre un jour particulier [45].

II.3.1.3.3. Machines à vecteurs de support (Support Vector Machine) : SVM est un algorithme de classification binaire. Étant donné un ensemble de points de 2 types dans N lieu dimensionnel, SVM génère un hyperplan dimensionnel (N - 1) pour séparer ces points en 2 groupes. Supposons que certains points de 2 types soient séparables linéairement. SVM trouvera une ligne droite qui sépare ces points en 2 types et située aussi loin que possible de tous ces points. En termes d'échelle, certains des problèmes les plus importants qui ont été résolus à l'aide de SVM (avec des implémentations correctement modifiées) sont la publicité écran, la reconnaissance de sites de jonction humaine, la détection de genre basée sur l'image, la classification d'images à grande échelle [45].

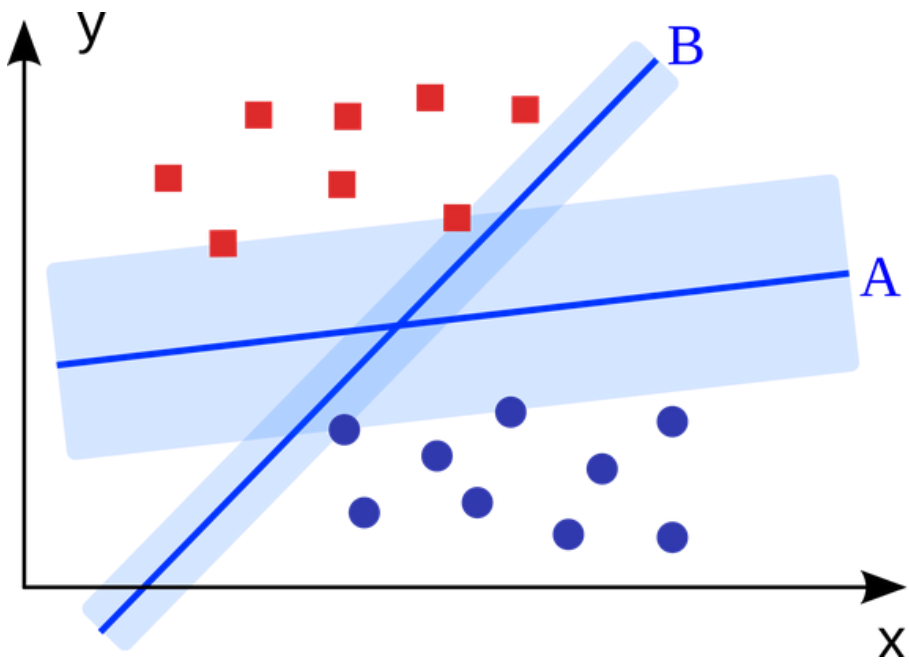


Figure 7: Support Vector Machine [45].

II.3.1.3.4. Arbres de décision : Un arbre de décision est un outil d'aide à la décision qui utilise un graphique ou un modèle arborant des décisions et leurs

conséquences possibles, y compris les conséquences d'un événement fortuit, le coût des ressources et l'utilité [45].

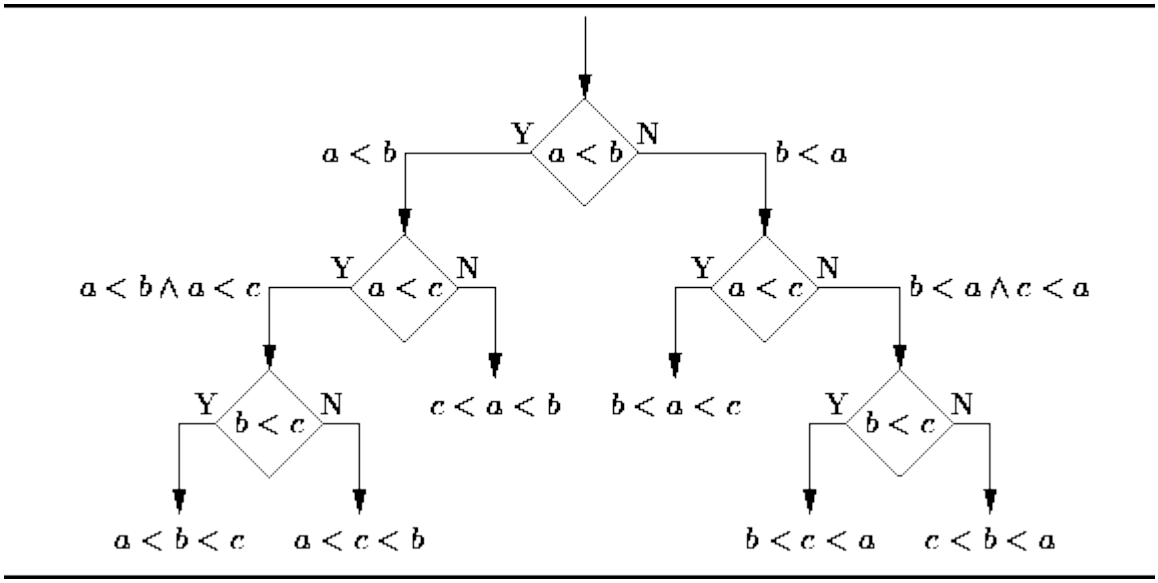


Figure 8: Graphe de l'Arbre de décision [45].

Du point de vue de la décision de gestion, un arbre de décision représente le nombre minimum de questions oui / non à poser pour évaluer la probabilité de prendre une décision correcte, la plupart du temps. En tant que méthode, elle vous permet d'aborder le problème de manière structurée et systématique pour arriver à une conclusion logique [45].

II.3.1.3.5. K-plus proches voisins (kppv) : La méthode des plus proches voisins (noté parfois k-PPV ou k-NN pour k-Nearest-Neighbor) consiste à déterminer pour chaque nouvel individu que l'on veut classer, la liste des plus proches voisins parmi les individus déjà classés. L'individu est affecté à la classe qui contient le plus d'individus parmi ces plus proches voisins. Cette méthode nécessite de choisir une distance, la plus classique est la distance euclidienne, et le nombre de voisins à prendre en compte.

Cette méthode supervisée et non-paramétrique est souvent performante. De plus, son apprentissage est assez simple, car il est de type apprentissage par coeur. Cependant, le temps de prédiction est très long, car il nécessite le calcul de la distance avec tous les exemples, mais il existe des heuristiques pour réduire le nombre d'exemples à prendre en compte [35]

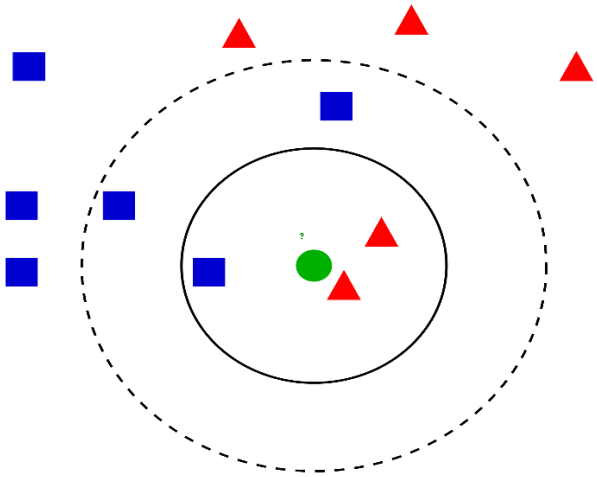


Figure 9: Exemple de classification kNN (k=3 et k=5)

II.3.1.3.6. Naïve Bayésien

Naïve Bayésien [46] est un algorithme populaire en Apprentissage automatique. Cet algorithme est utilisé pour la classification, il est particulièrement utile pour les problématiques de classification de texte.

Ce dernier est très rapide pour la classification en effet les calculs de probabilités ne sont pas très coûteux et la classification est possible avec un petit jeu de données.

Le terme P(A|B) se lit : la probabilité que l'événement A se réalise sachant que l'événement B s'est déjà réalisé.

On appelle le terme A : l'évidence. Le terme B s'appelle Outcome.

II.3.1.3.7. Artificial Neural Network (ANN)

Un réseau de neurones artificiels (RNA) [47] est un modèle informatique basé sur la structure et les fonctions des réseaux de neurones biologiques. Les informations qui circulent à travers le réseau affectent la structure de l'ANN, car un réseau neuronal change (ou apprend, dans un sens) en fonction de ces entrées et sorties.

Les ANN sont considérés comme des outils de modélisation de données statistiques non linéaires dans lesquels les relations complexes entre les entrées et les sorties sont modélisées ou des modèles sont trouvés.

II.3.2. Apprentissage non supervisé

Dans l'apprentissage non supervisé, les données sont uniquement constituées d'entrées. Dans ce cas, les tâches à réaliser diffèrent de l'apprentissage supervisé. Bien que de manière plus implicite, ces tâches sont également effectuées par les humains [31].

II.3.2.1. Regroupement ou « clustering »

Le regroupement est l'équivalent non supervisé de la classification. Comme son nom l'indique, son but est de regrouper les données en classes en utilisant leurs similarités. La difficulté du regroupement réside dans l'absence de mesure générale de similarité. Celle-là doit donc être définie en fonction du problème à traiter. L'un des algorithmes de regroupement les plus couramment utilisés est l'algorithme des k-moyennes. Le regroupement consiste donc à apprendre une fonction fregrp de Rd dans N qui associe à un vecteur son groupe. Contrairement à la classification, le nombre de groupes n'est pas connu a priori [31].

II.3.2.2. Estimation de densité

Le but de l'estimation de densité est d'inférer la répartition des données dans l'espace des entrées (ou, plus formellement, leur distribution). L'estimation de densité consiste donc à apprendre une fonction fest-dens telle que : ∫x fest-dens=1 qui associe à un vecteur sa probabilité [31].

II.3.2.3. Les algorithmes plus connus

II.3.2.3.1. Algorithmes de clustering : Le clustering consiste à grouper un ensemble d'objets de sorte que les objets du même groupe (cluster) se ressemblent davantage que ceux d'autres groupes.

Chaque algorithme de clustering est différent, et en voici quelques-uns :

- Centroid-based algorithms.
- Algorithmes basés sur la connectivité.
- Algorithmes basés sur la densité.
- Probabiliste.
- Réduction de la dimensionnalité.
- Réseaux de neurones / Deep Learning.

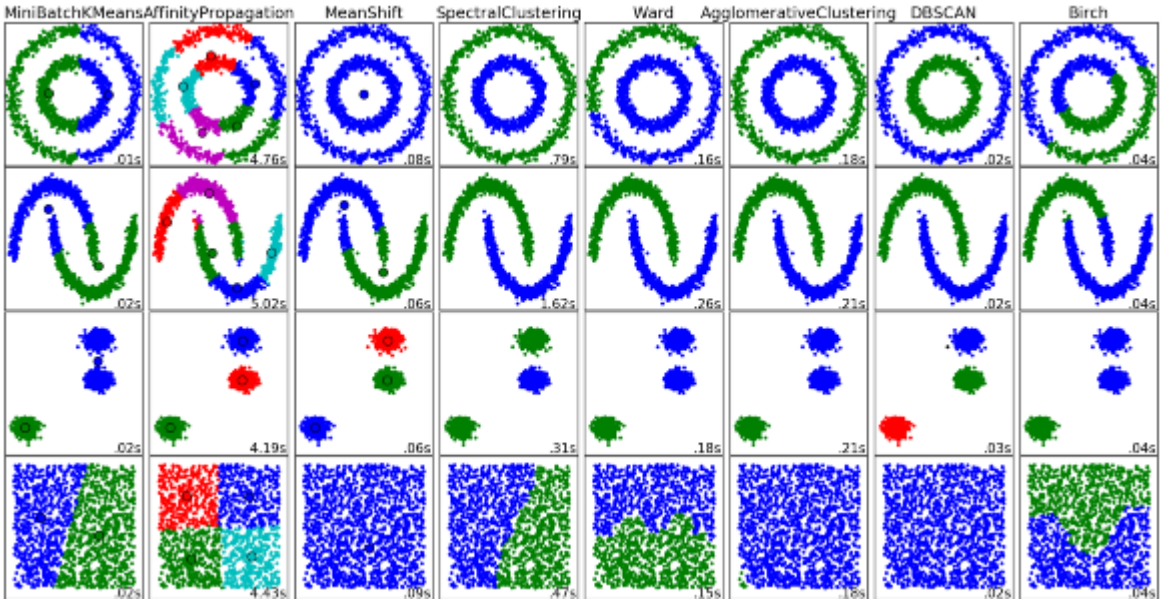


Figure 10: Clustering Algorithmes [45].

II.3.2.3.2. Analyse en composantes principales : La PCA est une procédure statistique qui utilise une transformation orthogonale pour convertir un ensemble d'observations de variables éventuellement corrélées en un ensemble de valeurs de variables linéairement non corrélées appelées composantes principales.

Certaines des applications de la PCA incluent la compression, la simplification des données pour un apprentissage plus facile, la visualisation. Notez que la connaissance du domaine est très importante lorsque vous choisissez de procéder ou non à la PCA. Il ne convient pas dans les cas où les données sont bruitées (toutes les composantes de la PCA ont une variance assez élevée) [45].

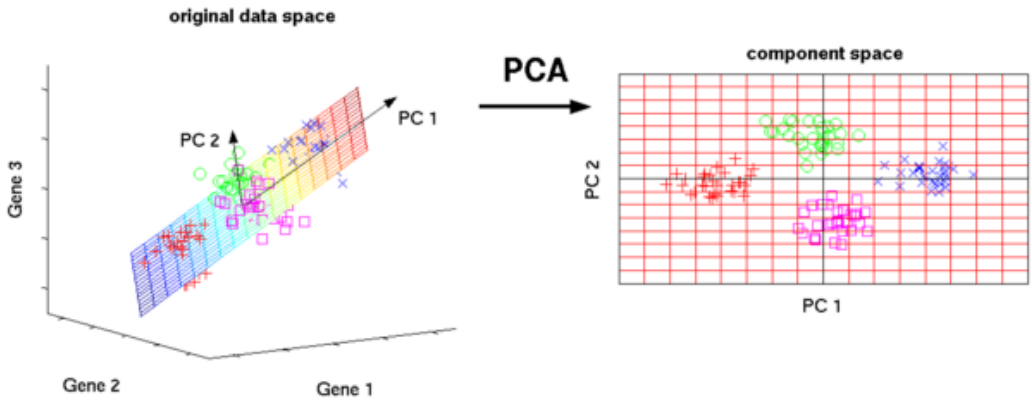


Figure 11: Principal Component Analysis [45].

II.3.2.3.3. Analyse en composantes indépendantes : La méthode ICA est une technique statistique permettant de révéler les facteurs cachés qui sous-tendent des ensembles de variables aléatoires, de mesures ou de signaux. ICA définit un modèle génératif pour les données multivariées observées, qui est généralement donné sous la forme d'une grande base de données d'échantillons. Dans le modèle, les variables de données sont supposées être des mélanges linéaires de certaines variables latentes inconnues, et le système de mélange est également inconnu. Les variables latentes sont supposées non gaussiennes et mutuellement indépendantes et sont appelées des composants indépendants des données observées.

L'ACI est liée à la PCA, mais c'est une technique beaucoup plus puissante qui est capable de trouver les facteurs sous-jacents des sources lorsque ces méthodes classiques échouent complètement. Ses applications comprennent les images numériques, les bases de documents, les indicateurs économiques et les mesures psychométriques [45].

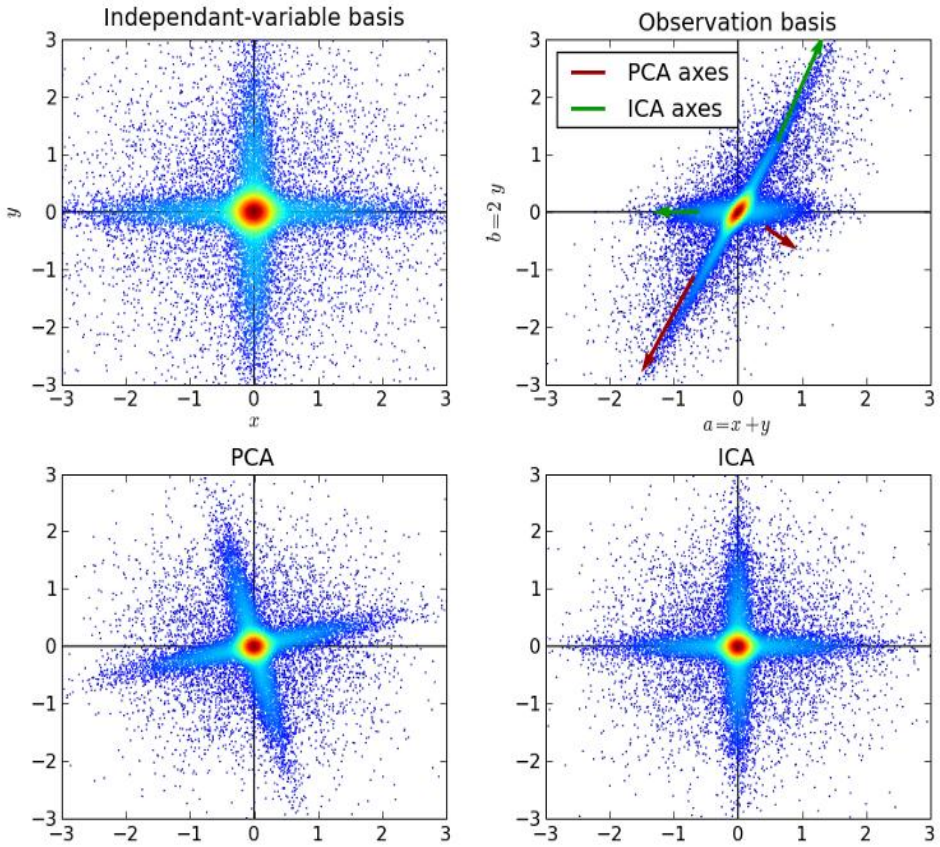


Figure 12: Independent Component Analysis [45].

II.3.3. Apprentissage semi-supervisé

Comme son nom l'indique, l'apprentissage semi-supervisé se situe entre l'apprentissage supervisé et l'apprentissage non supervisé. Certaines données sont étiquetées (c'est-à-dire sont composées d'une entrée et d'une étiquette) et d'autres ne le sont pas (seule l'entrée est fournie). Les tâches réalisées en apprentissage semi-supervisé sont les mêmes que celles réalisées en apprentissage supervisé (régression et classification), à la différence qu'il est fait usage des données non étiquetées. Un exemple où l'apprentissage semi-supervisé est adapté est la reconnaissance d'images. Il est très facile de récolter un grand nombre d'images (sur Internet, par exemple), mais il est plus difficile (que ce soit en termes de coûts ou de temps) de les étiqueter. Dès lors, avoir un algorithme capable de tirer avantages des images non étiquetées pour améliorer ses performances est souhaitable [44].

II.5. Conclusion

Dans ce chapitre nous avons passé en revue les principaux algorithmes de l'apprentissage automatique qui tend à résoudre des problèmes complexes à partir d'exemples en conjuguant les statistiques et l'intelligence artificielle.

Nous avons aussi présenté des techniques de détection d'anomalies qui sont basées sur l'apprentissage supervisé et non-supervisé, ce domaine est en perpétuelle évolution en effet de nombreux travaux et recherches sont actuellement entrepris afin d'accroitre les performances des systèmes de détection d'anomalies.

Chapitre III

Une approche d'Apprentissage Multi Niveau de détection d'intrusion

III.1. Introduction

Le problème de détection d'intrusion a suscité de nombreux travaux durant ces dernières années causées par le sur-apprentissage des types d'instances par rapport à elle-même, conduisant à un nombre très important d'approches.

Dans ce chapitre on va décrire une nouvelle technique de classification basée sur la structure hiérarchique, Cette technique permettra d'améliorer la classification dans l'IDS.

III.2. Méthodes d'ensemble

Chaque algorithme possède ses forces et ses faiblesses pour classifier de nouveaux exemples. En outre, certaines de ces forces sont complémentaires et peuvent améliorer les résultats lorsqu' elles sont combinées. Cette combinaison est surtout utile lorsque les Classificateurs se trompent sur des exemples différents. Si les algorithmes font les mêmes erreurs, cette technique perd tout son lustre et ne change presque rien aux résultats. On peut citer les intérêts de la combinaison de Classificateur comme suivant :

- Distribuer les caractéristiques sur des Classificateurs adaptés
- Exploiter la complémentarité entre Classificateur
- Prendre en compte les performances de chacun des Classificateurs
- Réduire l'importance des choix initiaux …
- Diviser pour mieux régner ... [48]

La combinaison se fait en entraînant séparément n modèles de manière à ce qu'ils soient les plus indépendants possibles l'un de l'autre et en combinant leurs réponses. Les méthodes de combinaison les plus communes sont le Bagging, le Boosting et le Stacking

III.2.1. Bagging

Bagging est une méthode proposée par [49] pour construire des ensembles de Classificateur, chaque classificateur étant entraîné sur une réplique différente de la base d'apprentissage. Les différentes répliques de la base d'apprentissage sont obtenues par bootstrap :

- Échantillon bootstrap : $L_1^*= (x_1^*, \dots, x_n^*)$ est un échantillon aléatoire de taille n obtenu par tirage aléatoire avec remise dans l'échantillon original L= $(x_1, \dots, x_n)$
- Chaque échantillon dans L peut apparaitre dans L* zéro, une, deux, trois...fois

Les différentes réplique Li* de la base d'apprentissage L sont très peu différentes de L mais suffisamment diverses pour obtenir des Classificateurs différents qu'on va pouvoir combiner.

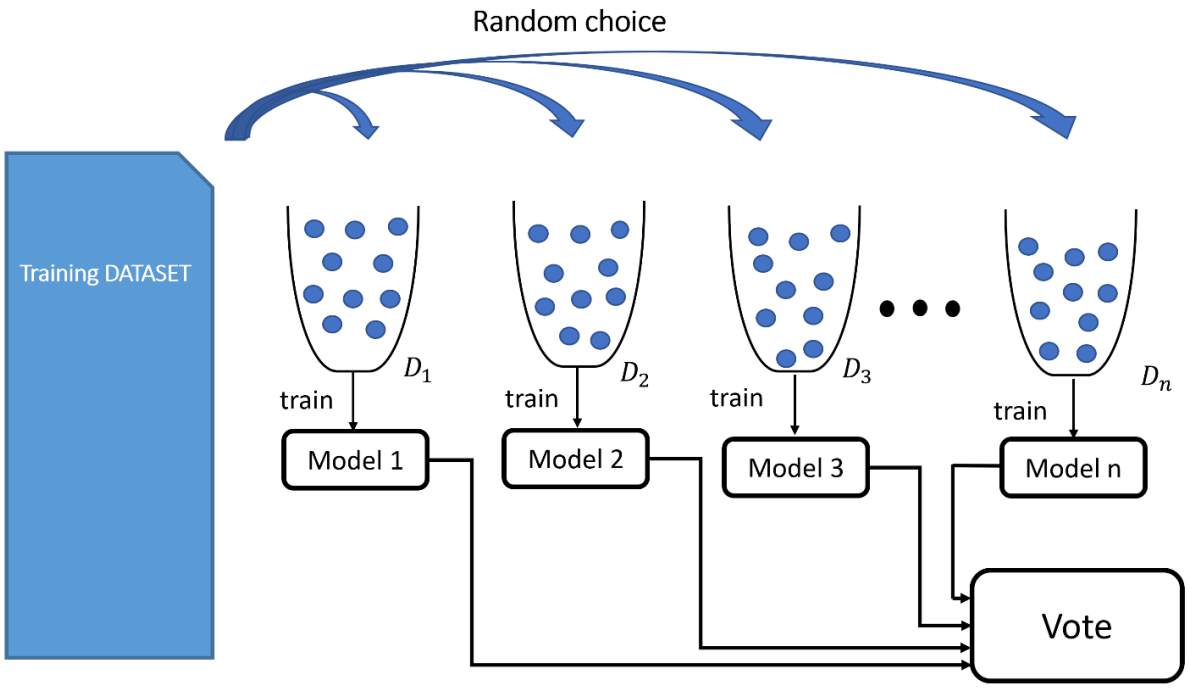


Figure 13: Schéma de la méthode Bagging

La règle de combinaison peut être n'importe quelle règle. Par exemples on peut mettre la moyenne des prédictions des différents modèles.

III.2.2. Boosting

La méthode du boosting est similaire à celle du bagging pour la création d'une famille de modèles qui sont ensuite agrégés par une moyenne pondérée des estimations. Elle diffère au niveau de la construction des modèles car chaque modèle est une version adaptative du précédent en donnant cette technique permet de produire des Classificateur "forts" (très précis) en combinant des instances "faibles" d'un Classificateur donné

L'algorithme AdaBoost [50], est le plus utilisé pour implémenter le boosting. Il construit de de façon itérative un ensemble de N Classificateur complémentaires.

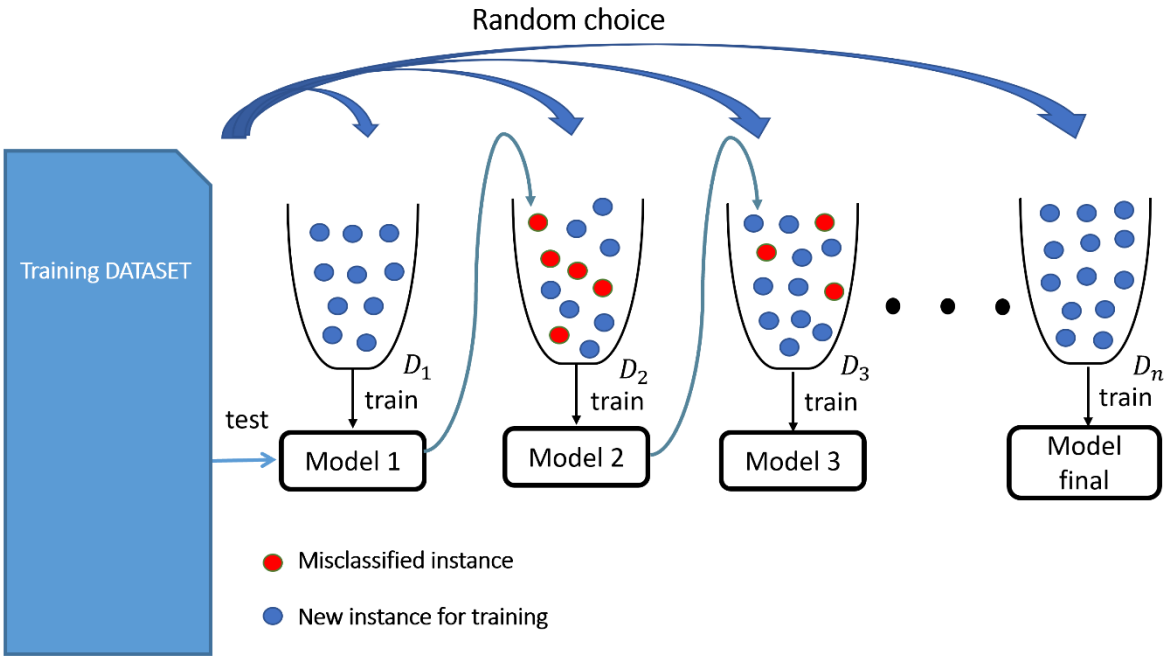


Figure 14: Schéma de la méthode Boosting

Des Classificateurs faibles sont ajoutés si nécessaire, et entrainés sur les échantillons que les Classificateurs précédents n'ont pas correctement classés. Les Classificateurs résultants dont combinés par un vote pondéré.

III.2.3. Stacking

Le principe du Stacking a été introduit par [51]. Dans cette dernière, on entraine de manière séquentielle plusieurs modèles, où le nième modèle est entrainé en tenant compte de la performance des modèles précédents. Par exemple, on a huit

Classificateurs dont sept sont entraînés indépendamment sur un même échantillon. L'apprentissage du huitième consiste à associer le meilleur des sept Classificateurs à utiliser selon les caractéristiques de l'exemple à classer [51].

Puisqu'il n'existe pas une quantité infinie de Classificateur, de plus en plus de travaux de concentrent sur la combinaison de plusieurs techniques différentes. Une telle approche nécessite généralement plus de temps pour l'apprentissage et la classification de nouveaux exemples, mais les pourcentages d'efficacité sont aussi plus élevés [53] [54] [55].

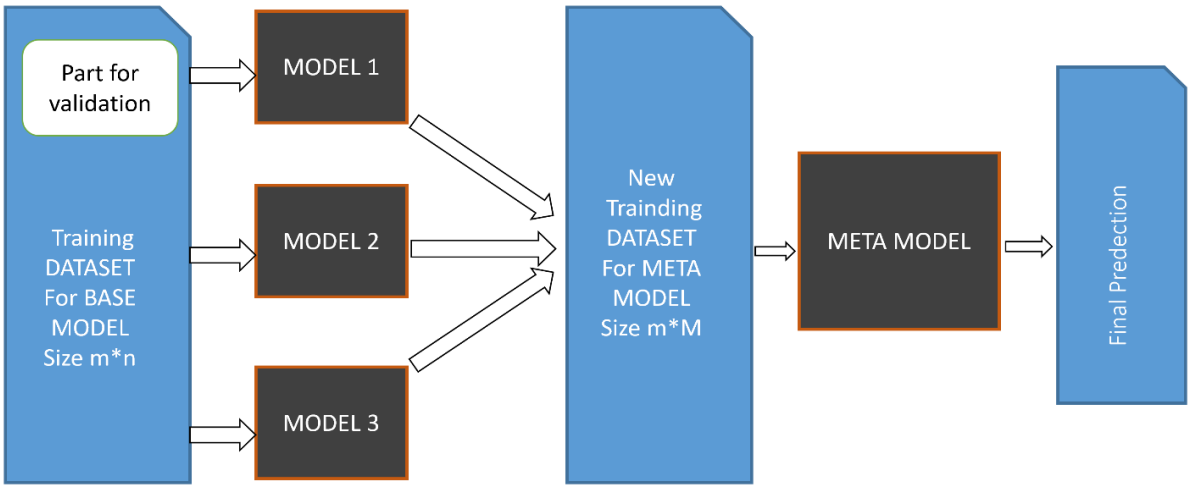


Figure 15: Schéma de la méthode Stacking

III.2.4. Forêt aléatoire d'arbre de décision

Les forêts aléatoires sont composées (comme le terme "forêt" l'indique) d'un ensemble d'arbres décisionnels binaires dans lequel a été introduit de l'aléatoire. Ces arbres se distinguent les uns des autres par le sous-échantillon de données sur lequel ils sont entraînés. Ces sous-échantillons sont tirés au hasard (d'où le terme "aléatoire") dans un jeu de données. La technique des forêts aléatoires modifie la méthode du Bagging appliquée ici aux arbres en ajoutant un critère de décorrélation entre ces arbres. L'idée de cette méthode est de réduire la corrélation sans augmenter trop la variance. Le principe consiste à choisir de façon aléatoire un sous-ensemble de variables qui sera considéré à chaque niveau de choix du meilleur nœud de l'arbre.

III.2.4.1 principe de forêt aléatoire d'arbre de décision

Considérons un ensemble d'entrainement L= $\{(X_1, Y_2), .., (X_m, Y_m)\}$,a le nombre d'attributs des exemples de X. Considérons également L_t un bootstrap contenant m instances obtenus par échantillonnage avec remplacement de L. Soit $\{h_1, \dots, h_2\}$ un ensemble de T arbres de décision. Chaque arbre h_t est construit à partir de L_t. Pour chaque nœud de l'arbre, l'attribut de partitionnement est choisi en considérant un nombre f(f<a) d'attributs choisis aléatoirement (parmi les a attributs). Pour classifier une nouvelle instance, le classificateur des forêts aléatoires effectue un vote de majorité uniformément pondéré des classificateurs de cet ensemble pour l'instance. L'algorithme illustre ce principe.

III.2.5 Classificateur de vote (Voting Classifier)

L'idée derrière le classificateur de vote est de combiner conceptuellement différents classificateurs d'apprentissage automatique et d'utiliser un vote à la majorité (Majority/Hard voting) ou les probabilités prédites moyennes (soft voting) pour prédire les étiquettes de classe. Un tel classificateur peut être utile pour un ensemble de modèles tout aussi performants afin de compenser leurs faiblesses individuelles.

Dans le vote à la majorité, l'étiquette de classe prévue pour un échantillon particulier est l'étiquette de classe qui représente la majorité (mode) des étiquettes de classe prédites par chaque classificateur individuel.

Par exemple, si la prédiction pour un échantillon donné est

classificateur 1 -> classe 1

classificateur 2 -> classe 1

classificateur 3 -> classe 2

Voting Classifier (avec vote = 'Hard') classerait l'échantillon en tant que «classe 1» sur la base du libellé de la classe majoritaire.

En cas d'égalité, Voting Classifier sélectionnera la classe en fonction de l'ordre de tri croissant. Par exemple, dans le scénario suivant

classificateur 1 -> classe 2

classificateur 2 -> classe 1

l'étiquette de classe 1 sera attribuée à l'échantillon.

Contrairement au vote à la majorité (Hard Voting), le Soft Voting renvoie l'étiquette de classe en tant qu'argmax de la somme des probabilités prédites.

Des poids spécifiques peuvent être attribués à chaque classificateur via le paramètre poids. Lorsque des poids sont fournis, les probabilités de classe prédites pour chaque classificateur sont collectées, multipliées par le poids du classificateur et moyennées. L'étiquette de classe finale est ensuite dérivée de l'étiquette de classe avec la probabilité moyenne la plus élevée.

III.3. Structure hiérarchique des attaques

L'architecture des attaques est définie sous forme d'arbre qui contient des nœuds de catégorie d'attaque, et pour chaque nœud on retrouve dans le niveau de profondeur suivant plusieurs types d'attaques qui figurent dans le schéma ci-dessous (figure 16)

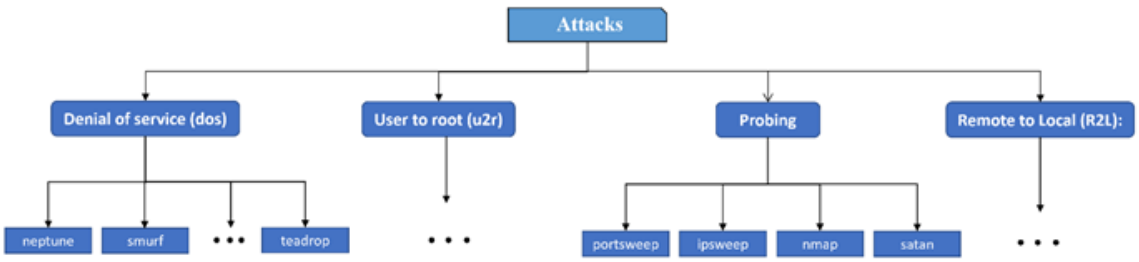


Figure 16: Exemple de la structure hiérarchique des attaques

III.4. Technique proposée (Apprentissage Multi Niveau)

Après avoir mené une étude approfondie sur les approches de l'apprentissage automatique, c'est à cet effet que nous proposons une autre approche similaire à ceux étudiées. D'après la structure hiérarchique qui exprime les catégories et les types d'attaques, de l'exploitation de ce dernier on a créé une nouvelle technique de classification d'instance d'intrusion appelé Apprentissage Multi Niveau (AMN) qui se base sur le regroupement des types d'attaques en catégorie.

III.4.1. Apprentissage (entrainement)

La première étape consiste à faire appel au jeu de données utilisé qui a N types d'étiquettes, après avoir généré d'elle un deuxième jeu de données on modifie le N

types vers M catégories d'étiquettes. La modification apportée permet au Modèle de Sélection de faire plus de précision et moins d'erreur.

En parallèle, on crée M jeux de données que chacune représente une partition du jeu de données de N types tout en contenant les instances de la même catégorie (regrouper les types d'attaques de la même catégorie dans un seul jeu de données D_i).

Avec le M jeux de données on génère un Modèle de Sélection qui s'occupe de identifier la catégorie, et pour chaque D_i jeu de données on génère un Modèle Classificateur de Type (voir la figure 17).

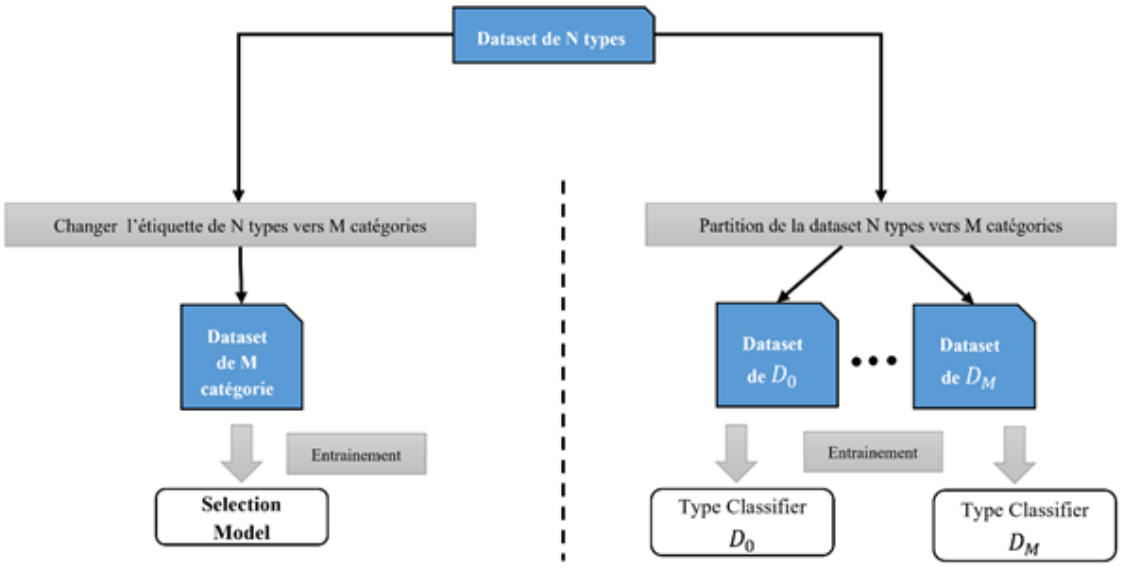


Figure 17: Schéma d'entraînement de l'Apprentissage Multi Niveau

III.4.2. Classer (Production)

La classification consiste à introduire la nouvelle instance X_i dans le Modèle de Sélection qui prédite sa catégorie Y_i, l'Apprentissage Multi Niveau utilise la prédiction obtenue pour choisir le Modèle Classificateur de Type D_i qu'elle lui réintroduit l'instance X_i pour aboutir sur le type d'attaque finale y_i (voir figure 18).

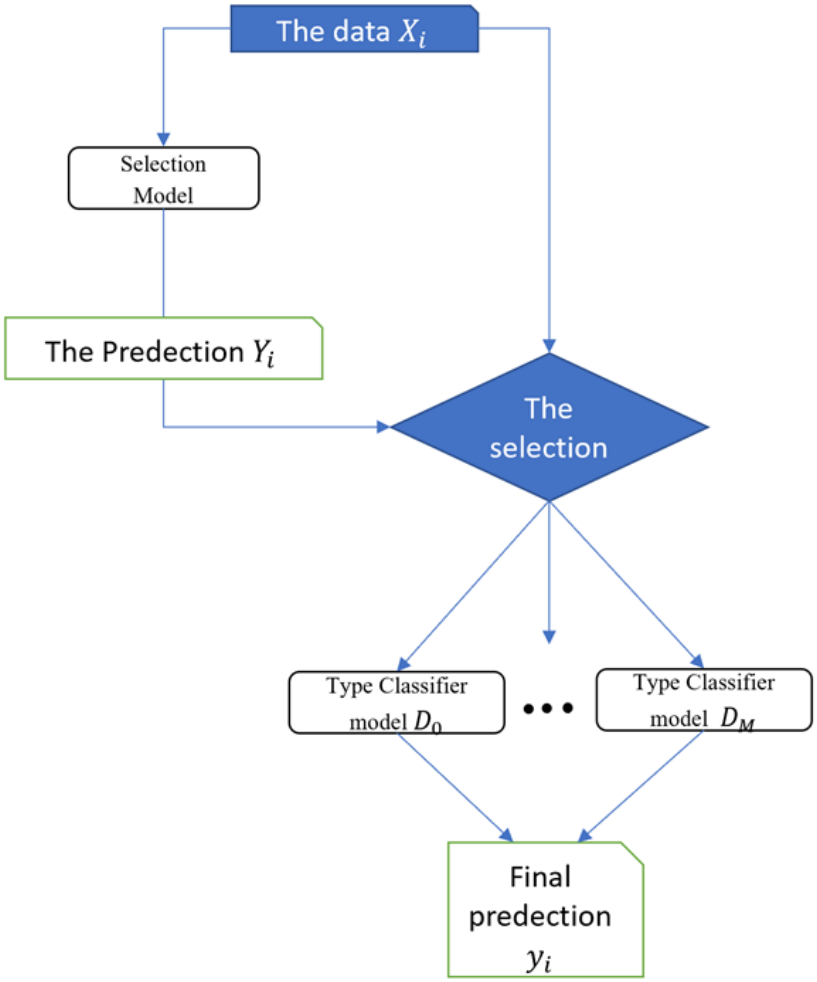


Figure 18:Schéma de classification de l'Apprentissage Multi Niveau

III.5. Conclusion

A ce niveau nous avons mis en relief et expliqué brièvement les différentes méthodes d'ensemble telles que Bagging, Boosting, Stacking, Random Forest et Voting Classifier.

Ensuite nous avons introduit la notion de structure hiérarchique des attaques (anomalies) afin de l'assimiler et proposer une nouvelle technique de détection, surnommée Apprentissage Multi Niveau (AMN) qui a été explicitée au cours du chapitre et dans lequel nous avons montré comment procéder à l'entrainement et ensuite comment effectuer la classification.

Chapitre IV

Tests et résultats

IV.1. Présentation de la base de données traitée

IV.1.1. Présentation des données KDD-Cup 1999

Les données utilisées pour nos expérimentations sont des données réelles issues de la base KDD-Cup 1999. Elles ont été préparées et contrôlées par les laboratoires MIT Lincoln pour le programme d'évaluation de détection d'intrusion DARPA 1998. Ces données ont aussi été utilisées pour le concours de détection d'intrusions de KDD 1999. Chaque connexion est étiquetée en tant que connexion normale ou attaque, avec le type spécifique d'attaque. Les attaques trouvées sont classées selon quatre catégories principales comme la montre la figure 19 :

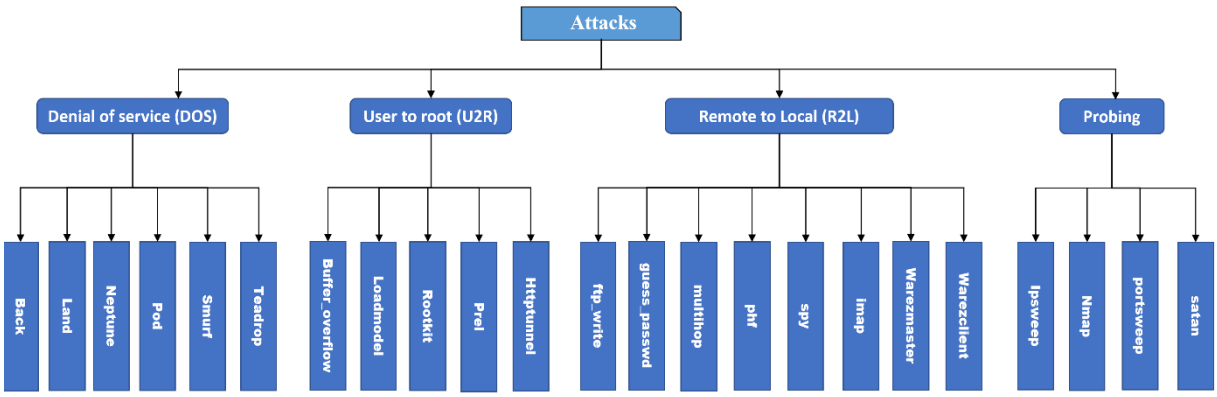


Figure 19: les catégories et les types d'attaques

- La catégorie DOS (Denial Of Service) : provoque un déni de service via des requêtes d'écho ICMP, manipulées à une adresse de diffusion d'un réseau.
- U2R (User to Root attacks*) :* l'attaquant essaie d'avoir les droits d'accès au système par le biais d'un poste.
- R2L (Remote to Local access) : Ce type d'attaque essaie d'exploiter la vulnérabilité du système afin de contrôler la machine distante.
- Probe (sondage et surveillance*) :* Ces actions ne sont pas vraiment des attaques puisqu'elles ne sont pas "destructrices" elles n'empêchent pas une entité de fonctionner correctement, mais permettent d'acquérir des informations parfois cruciales pour mener une attaque de plus grande envergure plus tard.

Ces données sont décrites au moyen de différentes variables explicatives. Pour une meilleure compréhension, celles-ci ont été classifiées en cinq catégories. Les variables d'une même machine décrivent seulement les connexions faites durant les deux dernières secondes et ayant le même destinataire que la connexion courante. Les variables de même service décrivent seulement les connexions faites durant les deux dernières secondes et ayant le même service que la connexion courante. On trouve aussi des variables de connections TCP individuelles. Enfin, il existe des variables qui indiquent un comportement anormal dans les données, ainsi le nombre de tentatives d'ouverture échouées. Il s'agit des variables de contenu. Les variables explicatives sont décrites dans le tableau en annexe (Les *variables de la base KDD-Cup 1999)*

Les données KDD-Cup 1999 sont construites à partir des données collectées par le programme d'évaluation de détection d'intrusions de DARPA en 1998. Ces données qui correspondent à environ quatre giga-octets de données binaires TCPdump compressées, contiennent sept semaines du trafic de réseau. Ceci a été transformé en environ cinq millions de connexions. Les données d'apprentissage KDD-Cup 99 possèdent 4,900,000 connexions étiquetées normale ou attaque. Chaque connexion contient 41 variables descriptives (Tableau 2). Le tableau suivant, donne la répartition exacte d'un échantillon de 10% des données utilisées lors de la compétition, nommé LS-10%, selon les différentes étiquettes de connexions.

Tableau 2: Nombre d'instances et distribution dans KDDCup'99 Train 10%

	Données d'entrainement	
Catégorie	Nombre d'instance	Pourcentage
Normal	97277	19.69
Dos	391458	79.23
U2R	52	0.01
Probe	4107	0.83
R2L	1126	0.22
Total	494020	100

A partir du tableau ci-dessus, nous remarquons que la classe "Normal" ne représente qu'environ 20% de l'ensemble de la base.

IV.1.2. Attributs utilisés

Chaque 'connexion' est caractérisée par 41 attributs tels que sa durée, le type du protocole, etc. Ces attributs ont été fixés suite à un travail de fouille de données dans KDD99 est considérée comme étant une 'connexion' normale ou bien une attaque.

Les attributs caractérisant chaque 'connexion', sont détaillés dans le tableau 3. Certains attributs sont de type discret (admettant un nombre fini de valeurs), d'autres sont de type continu.

Tableau 3: Les attributs du jeu de données KDD99 [58].

Attribut	Désignation	Description
A1	La durée	Longueur (nombre de secondes) de la connexion
A2	Type de protocole	Type du protocole, par exemple tcp, udp, etc.
A3	Service	Service réseau sur la destination, par exemple, http, telnet, etc.
A4	Src_bytes	Nombre d'octets de données de la source à la destination
A5	Dst_bytes	Nombre d'octets de données de la destination à la source
A6	Drapeau	État normal ou d'erreur de la connexion
A7	La terre	1 si la connexion est de / vers le même hôte / port ; 0 sinon
A8	Incorrect_Fragment	Nombre de "mauvais" fragments
A9	Urgent	Nombre de colis urgents
A10	Chaud	Nombre d'indicateurs "chauds"
A11	num_failed_logins	Nombre de tentatives de connexion infructueuses
A12	log_in	1 si connecté avec succès ; 0 sinon
A13	num_compromised	Nombre de conditions "compromises"
A14	root_shell	1 si la racine est obtenue ; 0 sinon
A15	su_attentée	1 si la commande `` su root " a été tentée ; 0 sinon
A16	num_root	Nombre d'accès `` root "
A17	num_file_creations	Nombre d'opérations de création de fichier
A18	num_shells	Nombre d'invites de Shell
A19	num_access_files	Nombre d'opérations sur les fichiers de contrôle d'accès
A20	num_outbound_cmds	Nombre de commandes sortantes dans une session ftp
A21	is_hot_login	1 si le login appartient à la liste `` chaude "; 0 sinon
A22	is_guest_login	1 si le login est un "invité" ; 0 sinon
A23	compter	Nombre de connexions au même hôte que la connexion actuelle au cours des deux dernières secondes
A24	serror_rate	% de connexions comportant des erreurs `` SYN "
A25	reerror_rate	% de connexions comportant des erreurs `` REJ "
A26	same_srv_rate	% de connexions au même service
A27	diff_srv_rate	% de connexions à différents services

A28	srv_count	Nombre de connexions au même service que la connexion actuelle au cours des deux dernières secondes
A29	srv_serror_rate	% de connexions comportant des erreurs `` SYN ''
A30	srv_rerror_rate	% de connexions à comportant des erreurs "REJ"
A31	srv_diff_host_rate	% de connexions à différents hôtes
A32	dst_host_count	Nombre de connexions pour le même hôte
A33	dst_host_svr_count	Nombre de connexions pour le même hôte utilisant le même service
A34	dst_host_same_srv_rate	% de connexions pour le même hôte utilisant le même service
A35	dst_host_diff_srv_rate	% de connexions pour le même hôte utilisant le différent service
A36	dst_host_same_src_port_rate	% de connexions pour le même hôte ayant port src
A37	dst_host_srv_diff_host_rate	% de connexions pour le même hôte et le même service utilisant différents hôtes
A38	dst_host_serror_rate	% de connexions pour le même hôte ayant l'erreur "SYN"
A39	dst_host_srv_serror_rate	% de connexions pour le même hôte et le même service ayant l'erreur "SYN"
A40	dst_host_rerror_rate	% de connexions pour le même hôte ayant l'erreur "REJ"
A41	dst_host_srv_rerror_rate	% de connexion pour le même hôte et le même service ayant l'erreur "REJ"

IV.1.3. Pourquoi le jeu de données du KDD99 ?

Nous avons choisi le jeu de données du KDD99 puisqu'il est le plus utilisé pour l'évaluation des IDSs avec apprentissage automatique, le tableau 4 montre combien d'articles existent pour chaque jeu données d'IDS entre 2010 et 2015 :

Tableau 4: Les jeu de données plus utilisés pour IDS (entre 2010 et 2015) [53].

Dataset Name	**Article Count**
KDD99	133
NSL-KDD	23
DARPA	9
ISCX	3
Kyoto	3

IV.2. Algorithmes et Méthodes d'ensembles utilisés

Une fois l'approche finalisée, nous l'avons appliquée sur un jeu de données (KDD99) ainsi que les algorithmes et méthodes d'apprentissage les plus communément utilisés et les plus efficaces, nous citons les listes exhaustives de ces derniers : KNN, Decision Tree et Random Forest pour les algorithmes. Boosting, Bagging, Voting Classifier et Stacking, pour les méthodes. Nous effectuons une étude comparative des résultats obtenus par les méthodes et algorithmes cités ci-dessus et l'approche que nous proposons.

IV.3. Mesure de performance

Afin d'évaluer notre proposition, Nous avons utilisé un principe d'indicateurs de performances la précision.

La précision (Accuracy) est la capacité d'un système à classer une instance inconnue. Elle représente la capacité de ce système à distinguer les paquets normaux de ceux intrusifs. La précision d'un système peut être calculée comme le taux de connexions correctement classées parmi toutes les connexions.

$$\text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$$

TP (vrais positifs) : Nombre de connexions qui ont été correctement classées comme attaque.

TN (vrais négatifs) : Nombre de connexions qui ont été correctement classées comme normales.

FP (faux positifs) : nombre de connexions normales qui ont été classées comme attaques.

FN (faux négatifs) : nombre de connexions d'attaques qui ont été classées comme normales.

IV.3.1. Validation croisée

Pour la validation, nous avons utilisé un modèle de validation croisée ou cross validation appelé K-fold. Afin de mener à bien les expériences, chaque ensemble de données a été divisé en dix parties égales. Pour le premier passage on teste une partie par rapport aux neuf restantes celles-ci constituent les données d'entrainement, une fois ce dernier terminé, cette partie est recyclée dans l'entrainement suivant qui prend en charge, bien entendu, la deuxième partie constituant les dix parties précédemment divisées.

IV.4. Résultats

Avant de tester nous avons effectué un filtrage du jeu de données KDD99 qui nous a conduit à éliminer les instances dupliquées pour éviter le sur-apprentissage et avoir un classificateur plus performant. Le tableau 5 nous montre les données traitées

Tableau 5: Jeu de données après filtrage

	Jeu de données après filtrage	
Catégorie	Nombre d'instance	Pourcentage
Normal	87832	60.33
Dos	54571	37.48
U2R	52	0.03
Probe	2131	1.47
R2L	999	0.69
Total	145585	100

Dans le but de comparer entre l'Apprentissage Multi Niveau et celles existantes nous présentons les résultats sous forme graphique suivie d'un tableau représentant les valeurs des approches.

La figure 20 représente l'apprentissage du AMN.

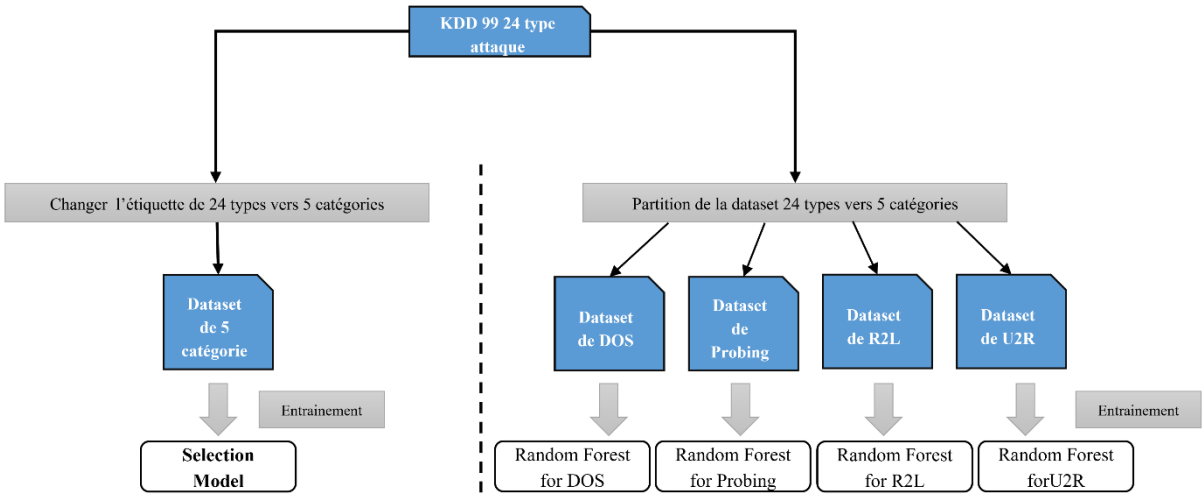


Figure 20: Apprentissage Multi Niveau avec KDD99

Pour le Modèle de sélection on a testé la majorité des classificateurs ce qui nous a permis d'avoir les résultats. Concernant les modèles classificateurs de types, les tests ont montré que KNN et Random Forest ont eu à la fois les meilleurs et très similaires résultats pour toutes les catégories d'attaques, mais prenant en compte le temps de prédiction, Random Forest s'avère plus rapide que KNN (voir figure 21), ce qui nous a conduit a opté pour les Random Forests pour la partie modèles de classifications de types.

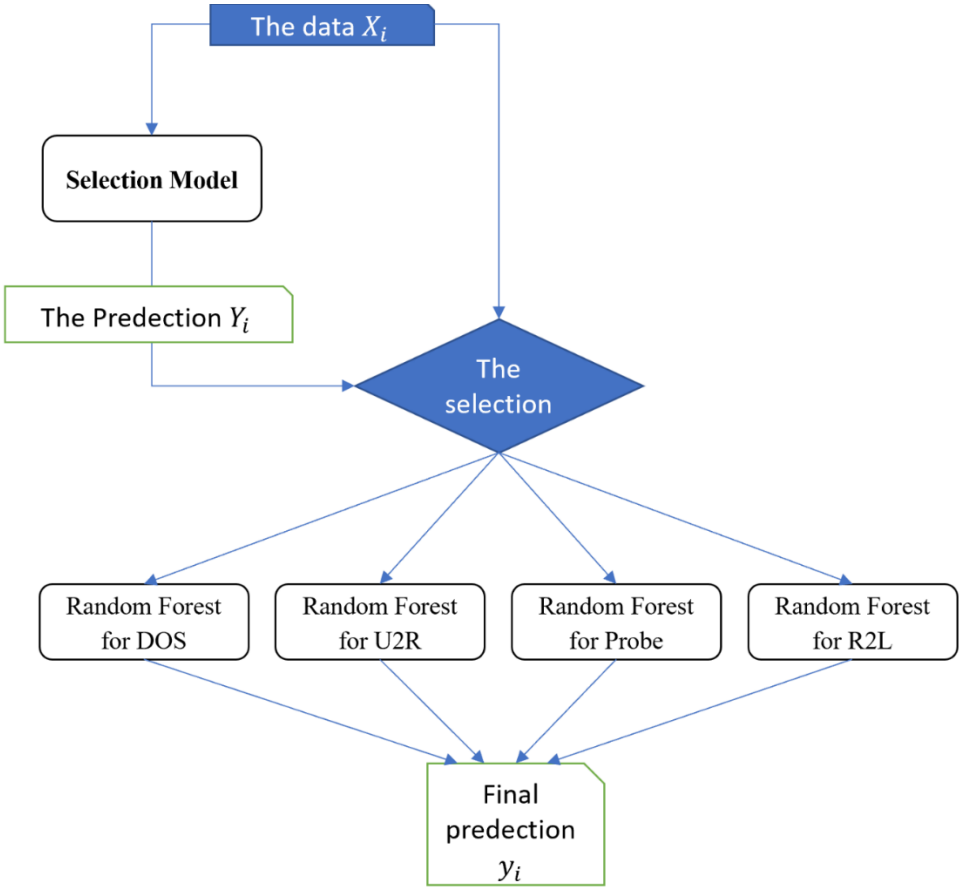


Figure 21: La classification avec AMN pour KDD99.

IV.4.1. Comparaison des résultats des classificateurs et l'AMN

Afin de collecter les résultats des tests on a fait une comparaison entre les classificateurs KNN, Decision Tree et Random Forest avec les mêmes classificateurs mais on a utilisé l'AMN pour prouver son impact.

IV.4.1.1 Résultats

Les figures 22 et 23 illustrent les différents résultats de test avec KNN Standard et KNN appliqué comme modèle de sélection dans l'AMN.

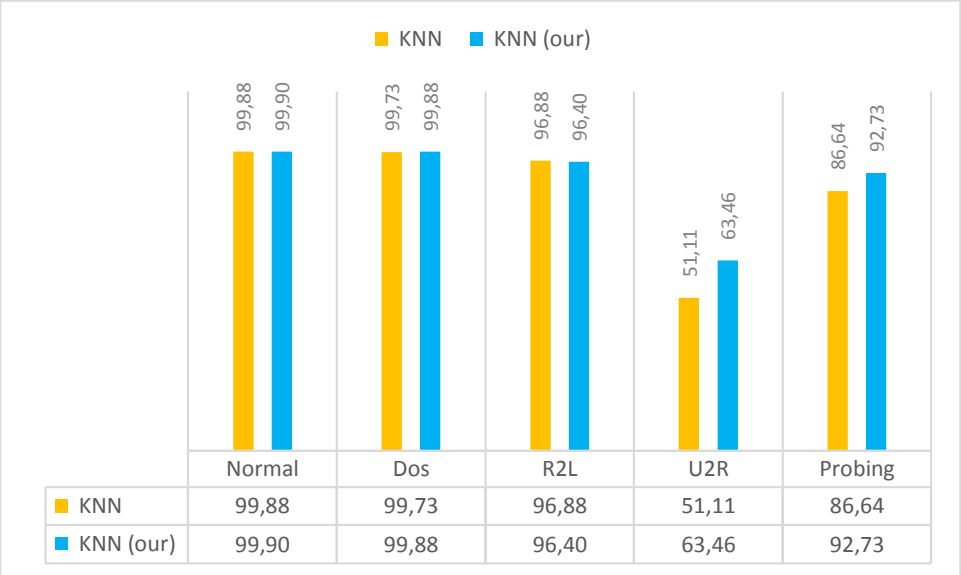


Figure 22 : La précision pour chaque catégorie d'attaque de kNN (k=1).

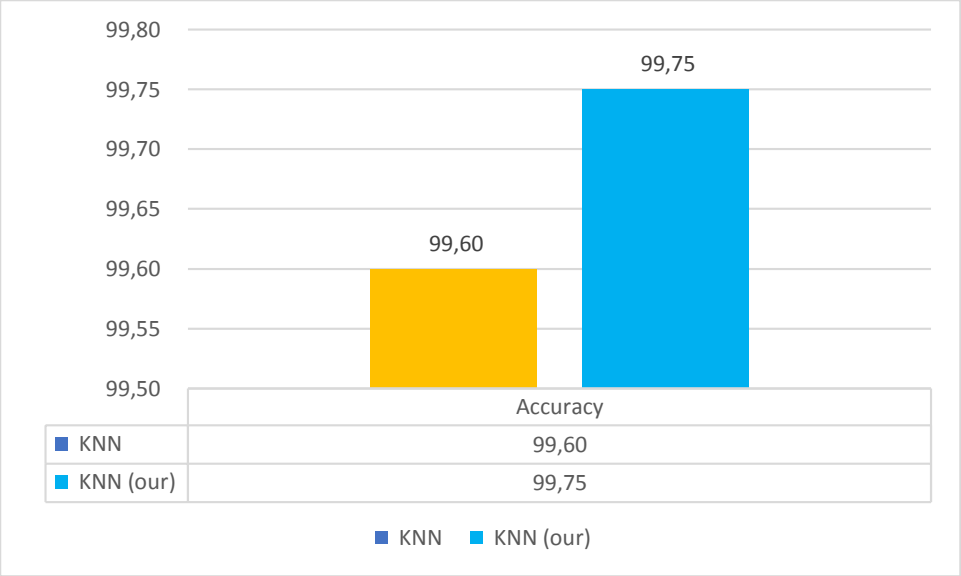


Figure 23: La précision (Accuracy Score) de kNN (k=1).

Le tableau 6 représente tous les tests des classificateurs standard et avec AMN.

Tableau 6 : Pourcentage des classificateurs et l'AMN

Classifier	Accuracy	Normal	Dos	R2L	U2R	Probing
KNN	99.60	99.88	99.73	96.88	51.11	86.64
Decision Tree	99.85	99.90	99.96	95.60	57.69	97.98
Random Forest	99.91	99.98	99.97	96.00	55.77	98.22
KNN with AMN	99.75	99.90	99.88	96.40	63.46	92.73
Decision Tree with AMN	99.88	99.90	99.97	96.40	65.38	99.01
Random Forest with AMN	99.92	99.98	99.97	95.60	63.46	98.64

IV.4.2 Comparaison des résultats des classificateurs et l'AMN avec utilisation de 'Voting Classifier'

Il existe plusieurs classificateurs pour le vote, parmi ceux-ci seuls trois ont été sutilisés comme électeurs pour tester l'AMN et voir l'amélioration éventuelle des résultats par rapport au vote Standard.

IV.4.2.1 Résultats

Les figures 24 et 25 illustrent les différents résultats de test avec vote Standard (électeurs : Decision Tree, KNN et ANN) et le même vote comme modèle de sélection dans l'AMN.

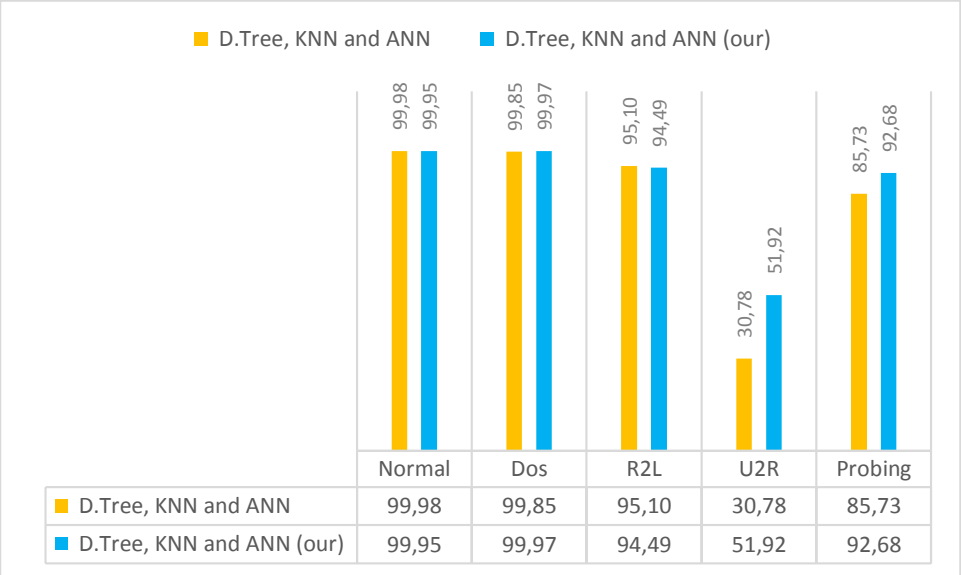


Figure 24: La précision pour chaque catégorie d'attaque de Decision Tree, KNN et ANN en utilisant Voting Classifier.

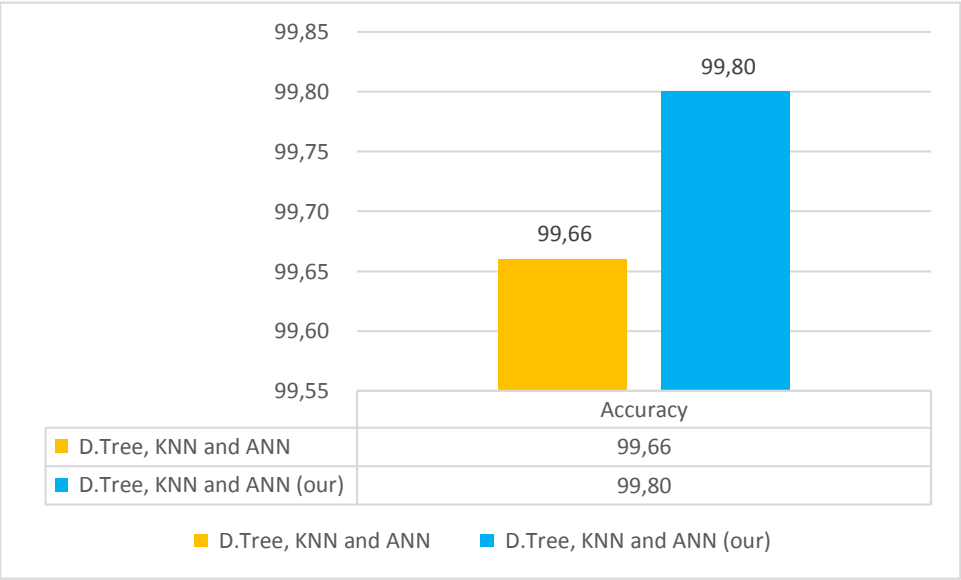


Figure 25: La précision de Decision Tree, KNN et ANN en utilisant Voting Classifier.

Le tableau 7 représente tous les tests du Voting Classifier standard et avec AMN.

Tableau 7 : Pourcentage des classificateurs et l'AMN avec utilisation de 'Voting Classifier'

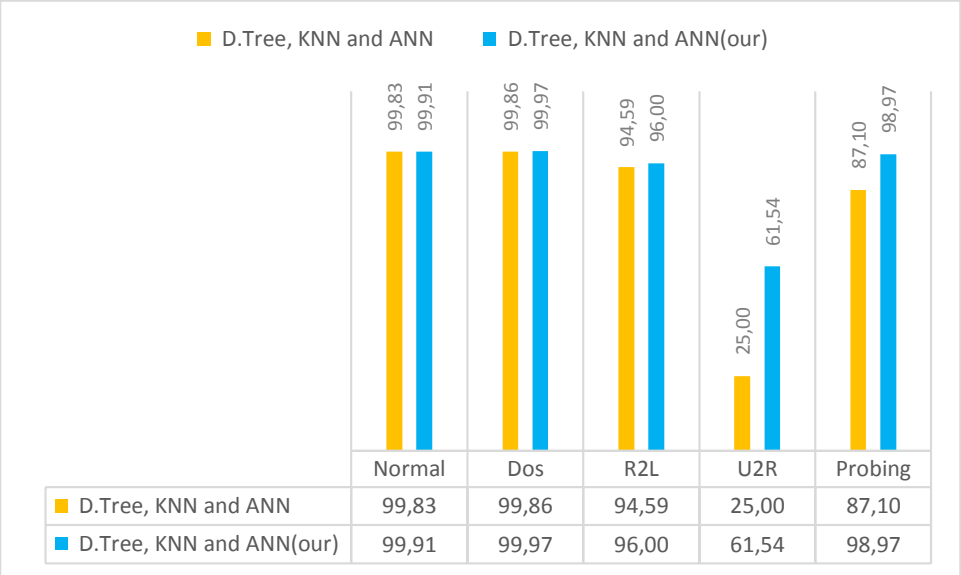
Voting Classifier	Accuracy	Normal	Dos	R2L	U2R	Probing
RDF,KNN and D.Tree	99.91	99.98	99.97	96.40	57.69	97.89
RDF,KNN and ANN	99.67	99.99	99.86	94.79	30.77	85.88
RDF,D.Tree and ANN	99.88	99.98	99.95	94.79	50.00	97.37
D.Tree, KNN and ANN	99.66	99.98	99.85	95.10	30.78	85.73
RDF,KNN and D.Tree with AMN	99.92	99.97	99.97	96.30	59.62	98.87
RDF,KNN and ANN with AMN	99.80	99.97	99.97	94.19	50.00	92.68
RDF,D.Tree and ANN with AMN	99.90	99.97	99.98	94.59	53.85	98.55
D.Tree, KNN and ANN with AMN	99.80	99.95	99.97	94.49	51.92	92.68

IV.4.3 Comparaison des résultats des classificateurs et l'AMN avec utilisation de 'Stacking'

Dans le Stack nous avons utilisé trois modèles de base et le Naïve Baise comme Meta Model pour faire la pile (a stack), parmi les modèles de base nous avons utilisé KNN, Decision Tree, Random Forest et ANN. Ici notre but consiste à tester l'AMN et voir l'amélioration éventuelle des résultats par rapport au Stack Standard.

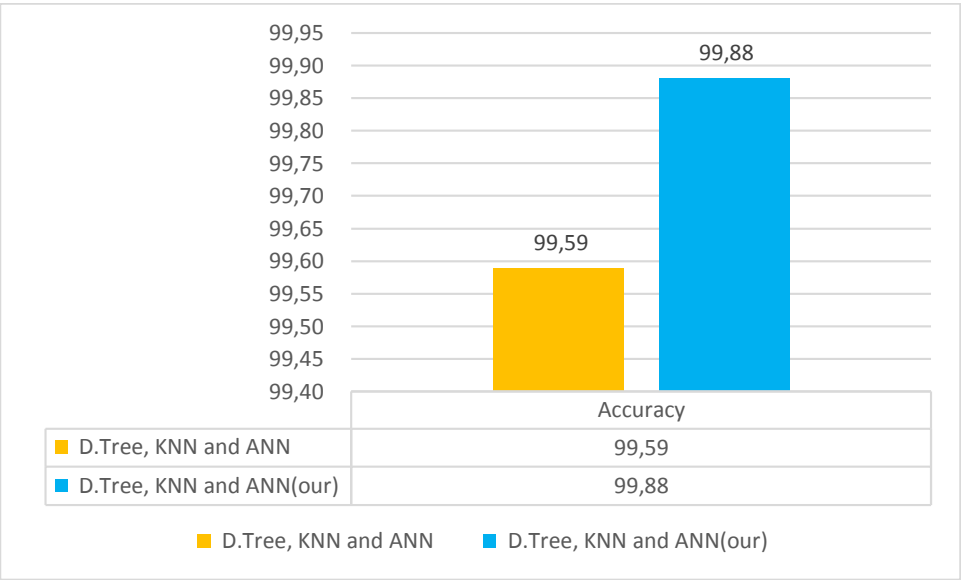
IV.4.3.1 Résultats

Les figures 26 et 27 illustrent les différents résultats de test avec Stacking Standard (Modèles de base : Decision Tree, KNN et ANN) et le même Stack comme modèle de sélection dans l'AMN.



	Normal	Dos	R2L	U2R	Probing
D.Tree, KNN and ANN	99,83	99,86	94,59	25,00	87,10
D.Tree, KNN and ANN(our)	99,91	99,97	96,00	61,54	98,97

Figure 26: La précision pour chaque catégorie d'attaque de Decision Tree, KNN et ANN en utilisant Stacking.



	Accuracy
D.Tree, KNN and ANN	99,59
D.Tree, KNN and ANN(our)	99,88

Figure 27: La précision (Accuracy Score) de Decision Tree, KNN et ANN en utilisant Stacking.

Le tableau 8 représente tous les tests du Stacking standard et avec AMN.

Tableau 8 : Pourcentage des classificateurs et l'AMN avec utilisation de 'Stacking'

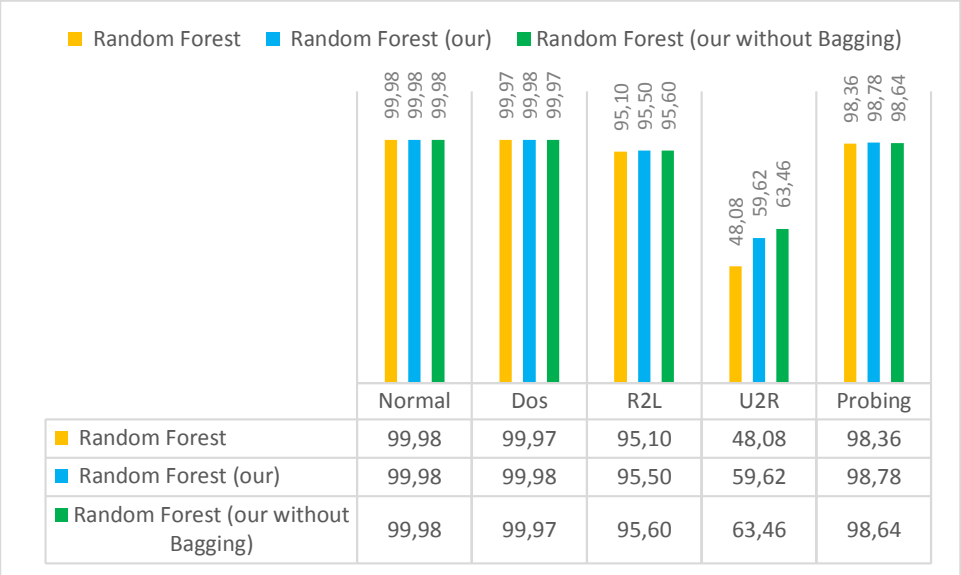
Stacking	Accuracy	Normal	Dos	R2L	U2R	Probing
RDF,KNN and D.Tree	99.61	99.83	99.85	93.73	30.78	88.87
RDF,KNN and ANN	99.62	99.90	99.89	95.30	38.46	85.73
RDF,D.Tree and ANN	99.84	99.9	99.95	94.87	51.92	98.12
Tree D., KNN and ANN	99.59	99.83	99.86	94.59	25.00	87.10
RDF,KNN and D.Tree with AMN	99.92	99.98	99.98	95.50	69.23	98.55
RDF,KNN and ANN with AMN	99.76	99.93	99.87	95.60	59.61	92.73
RDF,Tree D. and ANN with AMN	99.87	99.91	99.97	95.60	63.46	98.83
Tree D., KNN and ANN with AMN	99.88	99.91	99.97	96.00	61.54	98.97

IV.4.4 Comparaison des résultats des classificateurs et l'AMN avec utilisation de 'Bagging'

Pour le Bagging nous envisageons l'utilisation des classificateurs suivants : Random Forest, Decision Tree, KNN et ANN. Ici notre but consiste à tester l'AMN, le Bagging comme Modèle de Sélection dans la technique proposée et le Bagging Standard avec comparaison des trois résultats obtenus.

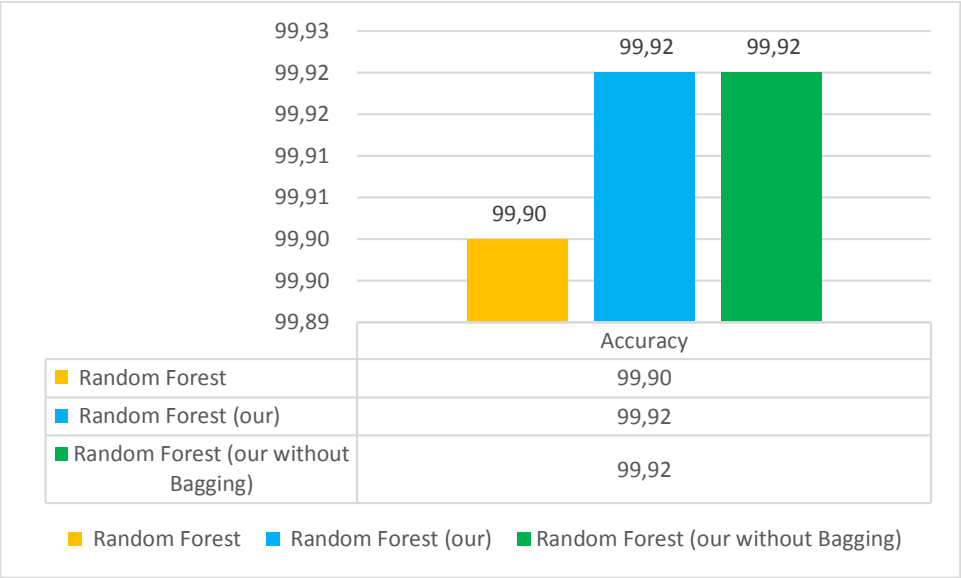
IV.4.4.1 Le Bagging de Random Forest

Les figures 28 et 29 illustrent les différents résultats de test du classificateur Random Forest avec Bagging Standard, le même Bagging comme modèle de sélection dans l'AMN et Random Forest sans Bagging comme modèle de sélection dans L'AMN.



	Normal	Dos	R2L	U2R	Probing
Random Forest	99,98	99,97	95,10	48,08	98,36
Random Forest (our)	99,98	99,98	95,50	59,62	98,78
Random Forest (our without Bagging)	99,98	99,97	95,60	63,46	98,64

Figure 28: La précision (Accuracy Score) de Random Forest en utilisant Bagging

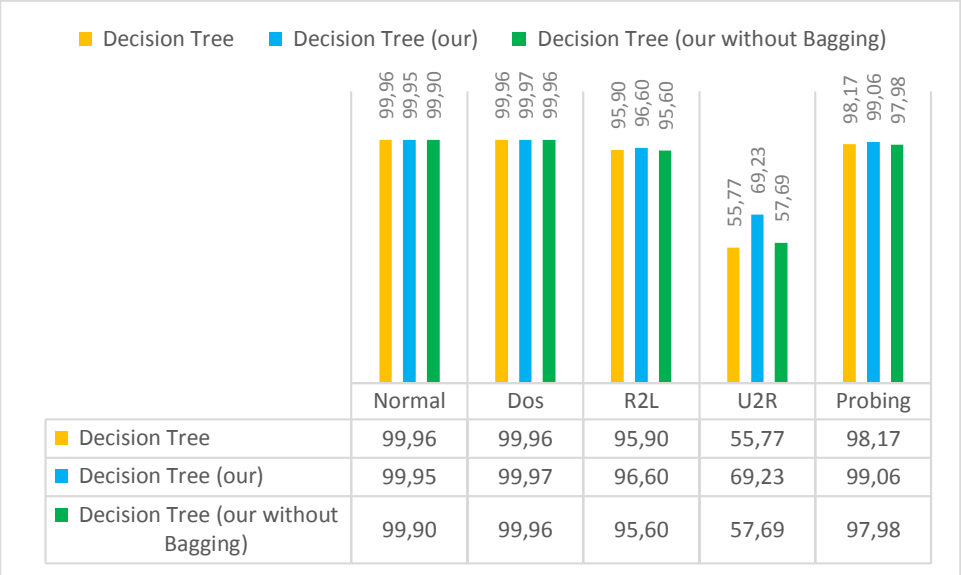


	Accuracy
Random Forest	99,90
Random Forest (our)	99,92
Random Forest (our without Bagging)	99,92

Figure 29: La précision de Random Forest pour chaque catégorie d'attaque en utilisant Bagging.

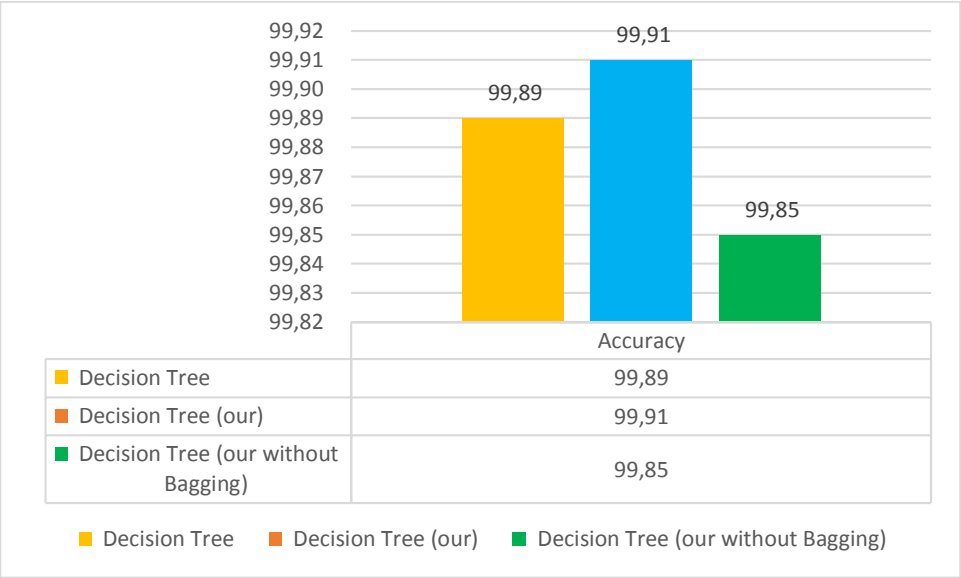
IV.4.4.2 Le Bagging de Decision Tree

Les figures 30 et 31 illustrent les différents résultats de test du classificateur Decision Tree avec Bagging Standard, le même Bagging comme modèle de sélection dans notre technique et Decision Tree sans Bagging comme modèle de sélection dans notre technique.



	Normal	Dos	R2L	U2R	Probing
Decision Tree	99,96	99,96	95,90	55,77	98,17
Decision Tree (our)	99,95	99,97	96,60	69,23	99,06
Decision Tree (our without Bagging)	99,90	99,96	95,60	57,69	97,98

Figure 30 : La précision de Decision Tree pour chaque catégorie d'attaque en utilisant Bagging.

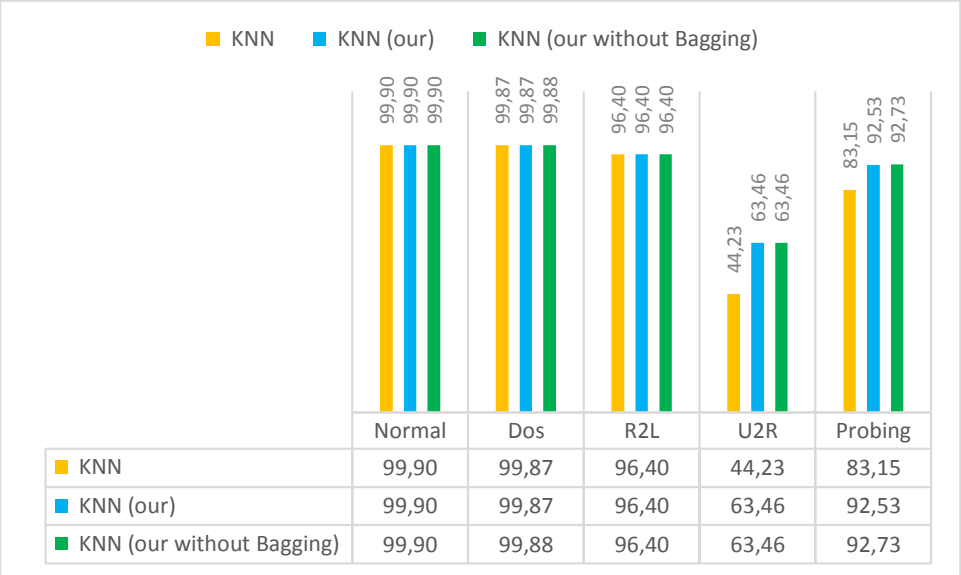


	Accuracy
Decision Tree	99,89
Decision Tree (our)	99,91
Decision Tree (our without Bagging)	99,85

Figure 31: La précision (Accuracy Score) de Decision Tree en utilisant Bagging

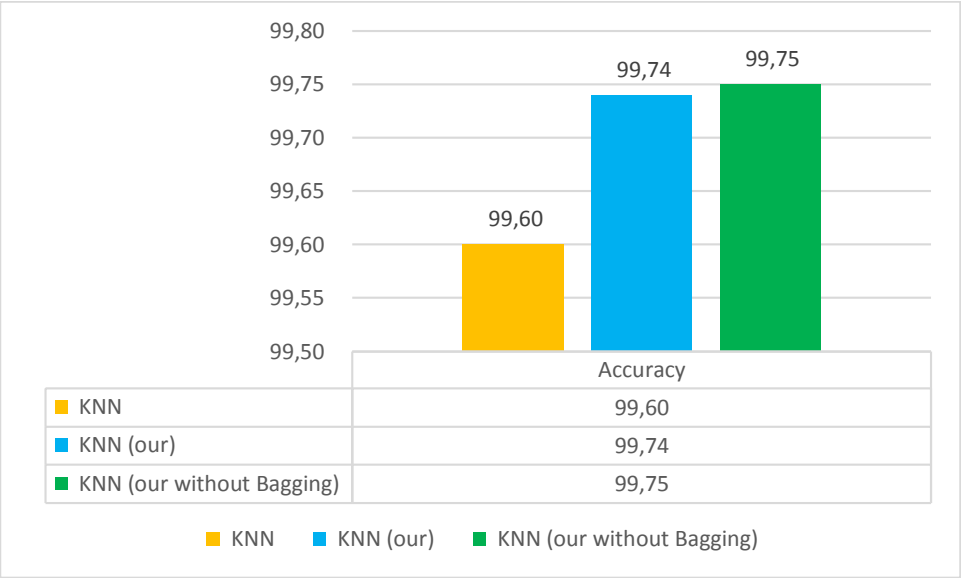
IV.4.4.3 Le Bagging de KNN

Les figures 32 et 33 illustrent les différents résultats de test du classificateur KNN avec Bagging Standard, le même Bagging comme modèle de sélection dans l'AMN et KNN sans Bagging comme modèle de sélection dans l'AMN.



	Normal	Dos	R2L	U2R	Probing
KNN	99,90	99,87	96,40	44,23	83,15
KNN (our)	99,90	99,87	96,40	63,46	92,53
KNN (our without Bagging)	99,90	99,88	96,40	63,46	92,73

Figure 32: La précision de KNN pour chaque catégorie d'attaque en utilisant Bagging



	Accuracy
KNN	99,60
KNN (our)	99,74
KNN (our without Bagging)	99,75

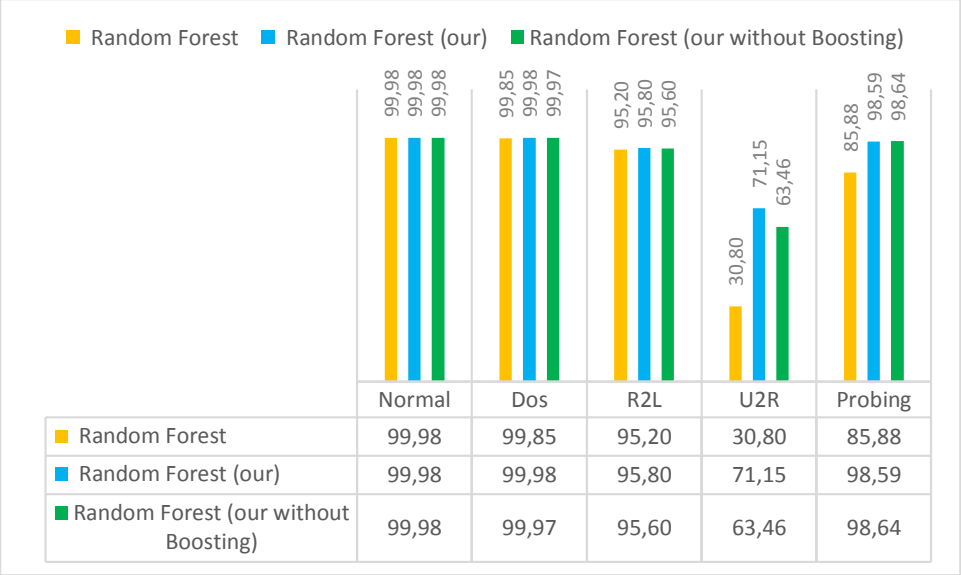
Figure 33: La précision (Accuracy Score) de KNN en utilisant Bagging

IV.4.5 Comparaison des résultats des classificateurs et l'AMN avec utilisation de 'Boosting'

Pour le Boosting nous envisageons l'utilisation des classificateurs suivants : Random Forest et Decision Tree. Ici notre but consiste à tester l'AMN, le Boosting comme Modèle de sélection dans la technique proposée et le Boosting Standard avec comparaison des trois résultats obtenus.

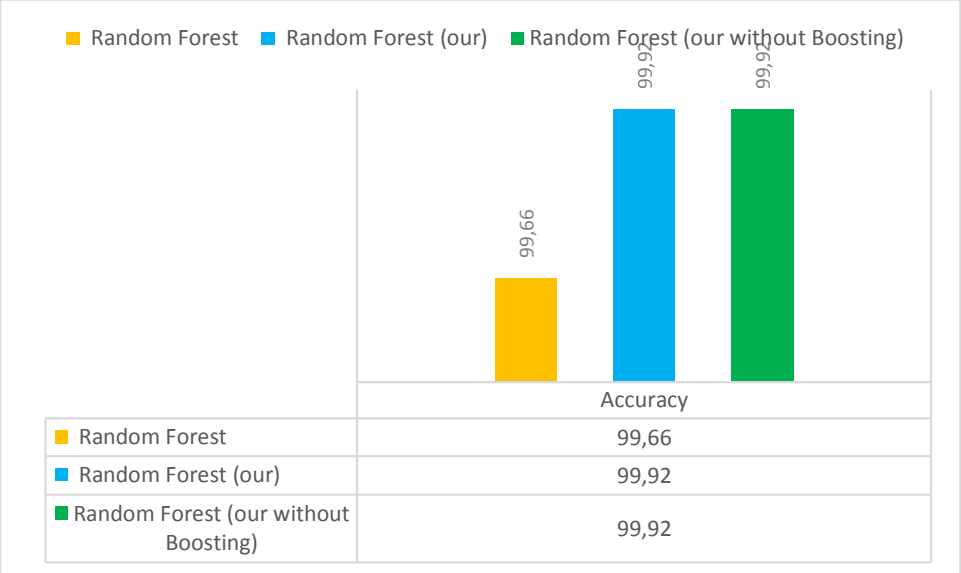
IV.4.5.1 Le Boosting de Random Forest

Les figures 34 et 35 illustrent les différents résultats de test du classificateur Random Forest avec Boosting Standard, le même Boosting comme modèle de sélection dans l'AMN et Random Forest standard comme modèle de sélection dans l'AMN.



	Normal	Dos	R2L	U2R	Probing
Random Forest	99,98	99,85	95,20	30,80	85,88
Random Forest (our)	99,98	99,98	95,80	71,15	98,59
Random Forest (our without Boosting)	99,98	99,97	95,60	63,46	98,64

Figure 34: La précision de Random Forest pour chaque catégorie d'attaque en utilisant Boosting

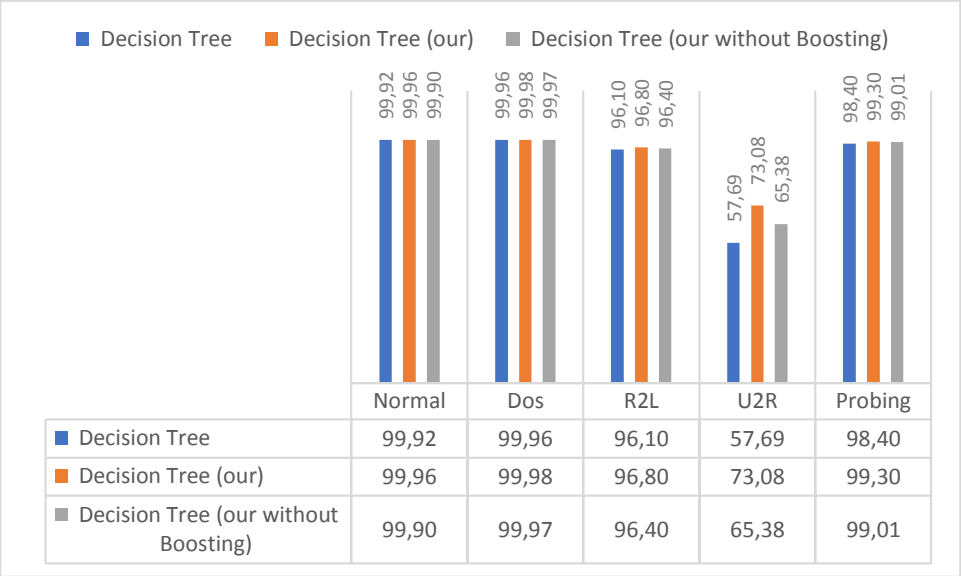


	Accuracy
Random Forest	99,66
Random Forest (our)	99,92
Random Forest (our without Boosting)	99,92

Figure 35: La précision (Accuracy Score) de Random Forest pour chaque catégorie d'attaque en utilisant Boosting

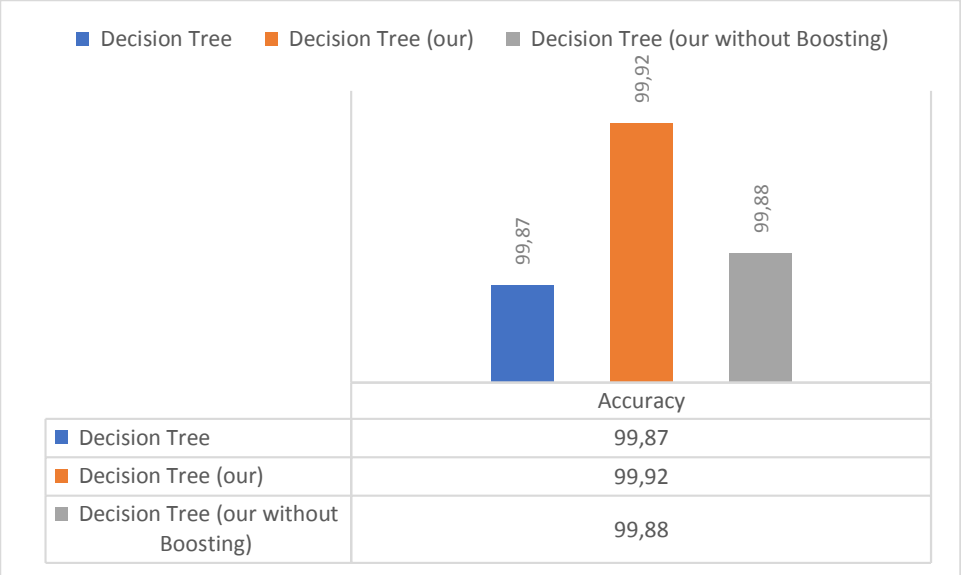
IV.4.5.2 Le Boosting de Decision Tree

Les figures 36 et 37 illustrent les différents résultats de test du classificateur Decision Tree avec Boosting Standard, le même Boosting comme modèle de sélection dans l'AMN et Decision Tree standard comme modèle de sélection dans l'AMN.



	Normal	Dos	R2L	U2R	Probing
Decision Tree	99,92	99,96	96,10	57,69	98,40
Decision Tree (our)	99,96	99,98	96,80	73,08	99,30
Decision Tree (our without Boosting)	99,90	99,97	96,40	65,38	99,01

Figure 36: La précision de Decision Tree pour chaque catégorie d'attaque en utilisant Boosting



	Accuracy
Decision Tree	99,87
Decision Tree (our)	99,92
Decision Tree (our without Boosting)	99,88

Figure 37: La précision (Accuracy Score) de Decision Tree pour chaque catégorie d'attaque en utilisant Boosting

IV.5 Discussion Générale

Nous remarquons que notre approche (AMN) a vraiment amélioré les résultats par rapport aux classificateurs que se soit qu'avec méthode traditionnelle ou celles d'ensemble, grâce au regroupement des types d'attaques vers catégories d'attaques. Puisqu'avec les 24 types 'standard', le classificateur peut classer par exemple une attaque DOS comme attaque DOS mais en vérité, il lui assigne le type d'attaque Smurf au lieu de Teadrop. Aussi il fait l'apprentissage des types un par un ce qui affaiblit la précision de la classification, spécialement l'attaque U2R qui a un nombre réduit de 52 instances et cinq types (Buffer_overflow 30, Rootkit 10, Loadmodule 9, perl 3, Httptunel 0), comparé au type Normal qui contient 87832 instances, la grande différence va créer un sur-apprentissage. De plus les connexions d'attaques ressemblent aux connexions normales cela engendre l'erreur de classification.

Pour l'Apprentissage Multi Niveau quand il classe l'attaque DOS pour la première fois au niveau de Modèle de sélection il ne va pas classer le type du DOS (smurf, Teadrop, etc.) il va l'envoyer au niveau du modèle Classificateur de Types qui est plus précis à la classification des types une fois la catégorie spécifiée. En ce qui concerne U2R, au lieu d'apprendre les types séparés il les apprend comme un seul type dans Modèle de Sélection qui lui donne plus de précision lors de leurs classements comme U2R, en suite au niveau du Classificateur de Types, les types d'attaques de U2R vont se classer facilement étant donné que le classificateur est plus exact à la catégorie U2R. Maintenant comme les connexions d'attaques ressemblent aux connexions normales le regroupement diminue le classement des normales comme attaque.

Conclusion Générale

Parmi les différents outils de sécurité informatique, on trouve le system de détection d'intrusion. Cet outil est devenu très indispensable pour tout réseau informatique, il nous permet de connaitre toutes activités anormales qui peuvent présenter un danger pour notre réseau. Les systèmes de détection d'intrusions ont deux types. Le premier type est basé sur signature, ce type a montré beaucoup de limites avec la rapidité et l'augmentation du trafic réseau mais il ne peut pas détecter les nouvelles formes des attaques. Le deuxième type basé sur les anomalies a été proposé afin de traiter les problèmes du premier type, ou les techniques d'apprentissage automatique ont été utilisés. Malgré la puissance et l'efficacité des techniques de l'apprentissage automatique, les systèmes de détection d'intrusion de ce dernier souffrent de certaines limites comme la nécessité de faire une mise à jour régulière, la nécessité de préparer les données d'entrainement, la difficulté de détecter les nouvelles formes d'attaques etc.

Dans notre étude nous avons proposé une nouvelle approche surnommée Apprentissage Multi Niveau, cette technique est basée sur le regroupement des attaques par catégorie grâce à la structure hiérarchique des attaques. Le but de ce travail est de diminuer le sur-apprentissage, Tout en utilisant deux étapes, la première consiste à sélectionner où bien classer la catégorie d'attaque avant de classer son type a l'aide du Modèle de sélection, qui a été implémenté par le k-NN, Decision Tree, Random Forest, Bagging, Boosting, Stacking et Voting Classifier. Concernant la deuxième étape, elle contient quatre modèles nommés Classificateur de Type, chaque catégorie (DOS, Probe, U2R et R2L) a son Classificateur de Types spécifique qui est offert par sa spécialisation avec une précision élevée, ces classificateurs utilisent Random Forest.

Pour vérifier l'efficacité de notre méthode, nous avons mené des expériences sur les données de trafic réseau KDD99 Train 10%.

Les résultats de notre expérimentation montrent que notre approche a prouvé l'efficacité d'amélioration de la précision, surtout celle concernant U2R.

Références

[1] J. Kim. Integrating Artificial Immune Algorithms for Intrusion Detection.

[2] InfoSec Reading Room. Intrusion detection systems: definition, need and challenges. SANS Institut, 2001.

[3] Slimane TAJNI. Mise en place d'une sonde SNORT monitoring network 2006-01-03.

[4] Ghenima BOURKACHE. Un IDS réparti basé sur une société d'agents intelligents, mémoire de magister informatique, université de Boumerdes. Algérie 2007.

[5] Lehmann Guillaum, cours de sécurité informatique 2003-04-13.

[6] Dorothy Denning. An intrusion detection models. IEEE, transaction on software engineering, 13(2) :222–232, 1987.

[7] Stephen E. Smaha. Haystack: An intrusion detection system.21th National Computer Security Conference,. :37–44, 1988.

[8] Steven R. Snapp, Stephen E. Smaha. Haystack Laboratories, Daniel M. Tim Grance. The DIDS (Distributed Intrusion Detection System) Prototype, United States, June 8 1992

[9] Mary E. Hanna Michael M. Sebring, Eric Shellhouse and R. Alan Whitehurst. Experts systems in intrusion detection: A case study. Proceeding of the 11th National Computer Security Conference, Baltimore, Maryland, . :74–81, 1988.

[10] H. Yao, X. Sun, Z. Zhou, L. Tang, and L. Shi, Joint optimization of sub channel selection and spectrum sensing time for multiband cognitive radio networks, in International Symposium on Communications & Information Technologies, 2010.

[11] TRW Defense System Groupe. Intrusion detection expert system feasibility study. Technical report, Final report 46761, 1986.

[12] Yiming Yang. An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1 :67–88, 1999.

[13] H S Vaccaro and G E Liepins. Detection of anomalous computer session activity. Proceeding of the 1989 IEEE Symposium on Security and Privacy. Oakland, California, . :280–289, 1-3 May 1989.

[14] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Departement of Computer Engineering Chalmers University of Technology, Goteborg, Sweden. :15–23, 14 March 2000.

[15] Karl Levitt Biswanath Mukherjee Jeff Wood Todd Heberlein, Gihan Dias and David Wolber. A network security monitor. Proceeding of the 1990 IEEE Symposium on Research in Security and Privacy. Soc Press, Los Alamitos, CA: USA,1990.

[16] L Todd Heberlein Biswanath Mukherjee and Karl Levitt. Network intrusion detection. IEEE Network, 8(3) :26-41, May/June 1994.

[17] Monique Becker Hervé Debar and Didier Siboni. A neural network component for an intrusion detection system. Proceeding of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA,. :240–25L0, May 1992.

[18] Daniel M Teal Steven R Snapp, Stephen E. Smaha and Tim Garance. The dids (distributed intrusion detection system) prototype. Proceeding of the Summer USENIX Conference, San Antonio, Texas. :227–233, 8-12 June 1992.

[19] Sandeep Kumar and Eugene H. Spafford. A software architecture to support misuse intrusion detection. Technical report, The COAST Project, Department of Computer Sciences, Purdu University, West Lafayette, IN, 47907-1398, USA, 17March 1995.

[20] Sandeep kumar. classification and detection of computer Intrusions. PhD thesis, Purdu University, West Lafayette, Indiana, USA, August 1995.

[20] Sandeep kumar and Eugene H. Spafford. A pattern matching model of misuse intrusion detection. Proceeding of the 17th National Computer Security Conference, Baltimore MD, USA, .:11-21-1994.

[21] Sandeep kumar and Eugene H. Spafford. An application of pattern matching in intrusion detection. technical report csd-tr-94-013, the coast project. Technical report, dept of computer Sciences, Purdue University, West Lafatette, IN, USA, June 1994.

[23] Todd Ellis Ivan Krsul Mark Ceosbie, Bryn Dole and Eugene Spafford. Idiot user guide. Technical report The COAST Project, Dept of Computer Science, Purdu University, West Lafayette, IN, USA, 4 September 1996.

[24] T. Frivold D. Anderson and A. Valdes. Next-generation intrusion-detection expert system (nides). Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA 94025-3493, USA, May 1995.

[25] Harold Javitz Ann Tamaru Debra Anderson, Teresa F. Lunt and Alfonso Valdes. Detecting unusual progragram behavior using the statistical component of the next-generation intrusion detection system (nides). Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, May 1995.

[26] R. Crawford M. Dilger J. Frank J. Hoagland K Levitt C. Wee R. Yip S. StaniFord Chen, S. Cheung and D. Zerkle. Grids-a graph-based intrusion detection system for large networks. Proceeding of the 19th National Information Systems Security Conference, 1996.

[27] Philip A Porras and Peter G Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. Proceeding of the 20th National Information Systems Security Conference, Baltimore, Maryland, USA, . :353–365,7-10 October 1997.

[28] Philip A Porras and Alfonso Valdes. Live traffic analisys of tcp/ip gateways. Proceeding of the 1998 ISOC Symposium on Network and Distributed System Security, San Diego: California, 11-13 March 1998.

[29] Jajodia S.-Popyack L. Barbara D., Julia Couto J. and Wu N. Adam: detecting intrusions by data mining. Proceedings of the 2001 IEEE workshop on information assurance and security, NY, USA :310 – 18, 2001.

[30] Mining audit data to build intrusion detection models. wenke lee and salvatore j. stolfo and kui w. mok. 1998.

[31] Kenaza Tayeb.Détection d'intrusion coopérative basée sur la fusion de données. Master's thesis, Institut National de formation en Informatique / ALGER,2006.

[32] Tadeusz Pietrazek. Alert classification to reduce false positives in intrusion detection. PhD thesis, Albert-Ludwigs-University of Freibury, Germany,2006.

[33] Nist. icat metabase. web page at http://icat.nist.gov/20002004.

[34] Thomas H. Ptacek and Timothy N Newsham. Insertion, evasion and denial of service: eluding network intrusion detection. Technical report, Secure Networks Inc, 1998.

[35] Umesh Shankar and Vern Paxson. Active mapping: Resisting nids evasion without altering traffic. In Proceedings of the 2003 IEEE Symposium on Security and Privacy, Oakland, CA: 4462, 2001.

[36] Robin Sommer and Vern Paxson. Enhancing byte-level network intrusion detection signatures with context. In Proceeding of the 10th ACM Conference on Computer and Communication Security, Washington, DC :262-271,2003.

[37] Seth Webster Richard Lippmann and Dauglas Stetson. The effect of identifying vulnerabilities and patching software on the utility of network intrusion detection. In Recent Advances in Intrusion Detection (RAID2002), Spring-Verlag, 2516 LNCS: 307-326, 2002.

[38] C. Kruegel F. Valeur, G. Vigna and R. Kemmmerer. Acomprehensive approch to Dependable and Secure Computing, 1(3) :146-169,2004.

[39] Julisch K. Using Root Cause Analysis to handle Intrusion Detection Alarms. PhD thesis, University of Dortmund, Germany, 2003.

[40] Wenke Lee and Salvator J. Stolfo. Data mining approaches for intrusion detection. Proceeding of the 7th USENIX Security Symposium, san Antonio, 1998.

[41] Schwartzbard A. Ghosh A. A study in using neural networks for anomaly and misuse detection. In the eighth USENIX security symposium, Washington, USA.14151,1999.

[42]Janoski G.H. Mukkamala S. Intrusion detection: support vector machines and neural networks In the IEEE international joint conference on neural networks, Honolulu USA.,2002.

[43] M^{elle} MAHDJANE Karima. Détection d'anomalies sur des données biologiques par SVM. Université Mouloud Mammeri de Tizi Ouzou, 14 Octobre 2012.

[44] Nicolas La Roux. Avancées théoriques sur la représentation et l'optimisation des réseaux de neurones, Université de Montréal, Mars, 2008.

[45] Le, Machine Learning Engineer, The 10 Machine Algorithms. Learning Engineers Need to Know. Technical report, https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html, August, 2019.

[46] Younes Benzaki, website Mr.Mint, Naive Bayes Classifier**.** https://mrmint.fr/naive-bayes-classifier, 26 Juillet 2017.

[47] Carlos Gershenson, Artificial Neural Networks for Beginners, Universidad Nacional Autónoma de México, September 2003.

[48] Heutte, L. Combinasion de Classificateur : pourquoi et comment les combiner ?.université de Rouen .2005

[49] breiman, L. Bagging Predictors. Machine Learning Journal.Vol.24, No.2. pp.123-140.1996.

[50] Freund, Y.et Schapire, R.E. Experiments with a new boosting algorithm. The 13th International Conference on Machine Learning.pp.148-156.1996.

[51] Wolpert, D.H. Stacked generalization. Neural Networks, Pergamon Press. Vol. 5. No 2. pp.241-259.1992.

[52] Pellerin, E. Méta-apprentissage des algorithmes génétique. Thèse d'exigence partielle de la maitrise en mathématique et informatique appliquées, Université du Québec. Décembre 2005.

[53] Larkey, L.S. et Croft W.B. Combining classifiers in text categorization. The 19th annual international ACM SIGIR conference on Research and development in information retrieval, Zurich, Suisse. PP. 289-297.1996.

[54] Merz, C.J. Using correspondence analysis to combine classifiers. Machine Learning. Vol. 36, No 1-2. pp. 33-58.1999.

[55] Opitz, D. et Maclin, R. Popular ensemble méthode: an empirical study. Journal of AI Research. Vol. 11. pp. 169-198. 1999.

[56] Atilla.Özgür, Hamit. Erdem. A Review of KDD99 Dataset Usage in Intrusion

Detection and Machine Learning between 2010 and 2015. Baskent University. 14 Apr 2016.

[57] University of California Irvine, KDD 99 Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 28 2019