République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Seddik Ben Yahia de Jijel

Faculté des Sciences Exactes et Informatique

Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme

Master de Recherche en Informatique

Option : Réseaux et Sécurité

Thème

Proposition d'un modèle hybride multi niveaux basé sur des techniques de l'intelligence artificielle pour la détection des cyber attaques

Présenté par :

Kahlessenane Salim

Benamirouche ibrahim

Encadré par :

Mr.Boulaiche Ammar

Promotion 2022

Remerciement

En premier lieu, nous remercions Dieu le très haut qui nous a donné le courage et la volonté de réaliser ce modeste travail.

Nous remercions nos très chers parents, qui ont toujours été là pour nous.

Nous remercions nos frères et nos sœurs, pour leurs encouragements.

Nos remerciements s'adressent aussi à Mr.Boulaiche Ammar, qui nous fait le grand honneur d'accepter d'examiner notre travail.

Nous tenons à remercier très sincèrement l'ensemble des membres du jury qui nous font le grand honneur d'accepter de juger notre travail.

*E*nfin, nous adressons nos plus sincères remerciements à tous nos amis, qui nous ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

Dédicace

Je dédie ce modeste travail

A ma chère mère et à la mémoire de mon cher père.

A mes Frères et sœurs.

A toute ma famille, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

A ma fiancée.

A tous mes amis.

A tous ceux que j'aime et qui m'aiment je dédie ce travail.

Salim

Dédicace

Je dédie ce modeste travail

A mon cher père et à la mémoire de ma chère mère.

A mes Frères et sœurs.

A tous mes amis et collègues.

Sans oublier tous les professeurs qui ont contribué à ma formation de l'enseignement primaire, moyenne et secondaire jusqu'à l'enseignement supérieur.

Ibrahim

Résumé

Le grand développement technologique, l'accessibilité au réseau internet à grande échelle et l'augmentation des moyens de stockage et d'échange d'informations ont contribué à l'évolution remarquable des cyber-attaques lors de ces dernières années, ce qui rend la sécurisation de toutes ces informations très important. Cette augmentation considérable des attaques a motivé les chercheurs et les spécialistes du domaine de la sécurité informatique à élaborer et à développer des nouveaux outils pour la détection des attaques réseaux, notamment celles qui sont difficilement découvertes par les outils traditionnels, en se basant sur les techniques de l'intelligence artificielle.

Dans ce travail nous avons réalisé un modèle hybride multi niveaux de détection d'intrusions basé sur les algorithmes de classification : machines à vecteurs de support, les réseaux de neurones (perceptron multicouches), K-plus proches voisins, et des arbres de décision, en utilisant l'ensemble de données NSL-KDD pour générer et évaluer ce modèle.

Abstract

The great technological development, the accessibility to the large-scale Internet network and the increase in the means of storing and exchanging information have contributed to a large number of cyber-attacks, which makes the fact of securing these information very important. This increase in attacks makes researchers and specialists in the field of computer security busy developing new tools for detecting attacks not discovered by traditional means based on artificial intelligence techniques.

In this work we realized a multi-level hybrid model of intrusion detection based on classification algorithms: support vector machines, neural networks (multilayer perceptron), K-nearest neighbors, and decision trees, using the NSL-KDD dataset to generate and evaluate this model.

Table des matières

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

L’évolution notable des nouvelles technologies de l'information et de la communication lors de ces dernières années, a augmenté considérablement l’accessibilité au réseau internet par un grand nombre d'utilisateurs avec leurs différentes intentions qui peuvent être parfois destructives. Cette évolution dans l’utilisation des outils informatiques à grande échelle, a malheureusement rendu les données sensibles ainsi que les ressources des utilisateurs et des sociétés de plus en plus vulnérables au vol et même par fois exploitables pour des raisons et intentions malveillantes. Face à toutes ces menaces, la sécurité optimale des systèmes informatiques et des réseaux est devenue un enjeu stratégique, et pour assurer cette sécurité, plusieurs outils différents ont été utilisés, tels que les pare-feu et les anti-virus.

Malheureusement, les anti-virus et les pare-feu sont le plus souvent inefficaces contre ces nouvelles menaces complexes, qui peuvent se propager extrêmement rapidement. Pour pallier cette lacune, de nouvelles solutions de sécurité sont apparues récemment appelées systèmes de détection d'intrusion (IDS (Intrusions Detection System) , en anglais) qui consistent notamment à inspecter le trafic réseau, collecter tous les événements, les analyser et générer des alertes en cas de tentatives malveillantes identifiées. Ces systèmes sont de jour en jour largement utilisés dans les stratégies de sécurité des systèmes informatiques et des réseaux. Cependant, l'enjeu majeur des systèmes de détection d'intrusion réside dans leur capacité à identifier tout comportement malveillant qu'il soit à l'intérieur ou à l'extérieur du système informatique.

En effet, le domaine de la détection d'intrusion est très ouvert à la recherche et au développement, où de véritables produits et solutions logiciels commencent à apparaître de jour en jour et fonctionnent selon deux axes principaux : approche par scénario et approche comportementale, l'approche par scénario, est basée sur la comparaison du comportement d'utilisation du système avec des signatures d'attaques précédemment connues et ne permet pas de détecter de nouvelles attaques sans mettre à jour la base de données de signatures, ce qui peut représenter un inconvénient majeur de cette méthode, et l'approche

comportementale, qui consiste à construire un modèle qui identifie les comportements qui s'écartent du modèle comportemental normal. Lors de ces dernières années, ce type de modèles est souvent construit à partir d'une phase d'apprentissage sur un grand ensemble de données ce qui permet de générer un modèle de classification qui est capable de détecter tout genre d'attaques y compris les nouvelles attaques qui ne sont pas encore découvertes par la communauté de sécurité informatique. Ce dernier point représente un des avantages majeur de cette approche.

Dans ce travail, nous essayons de réaliser un système de détection d'intrusion multi-niveaux comportemental, qui est capable de détecter de nouvelles attaques avec un taux de réussite maximal. Pour cela, nous avons utilisé des machines à vecteurs de support, des réseaux de neurones artificiels, des algorithmes K plus proches voisins et des arbres de décision, qui représentent des outils d'intelligence artificielle pour l'apprentissage et la classification.

CHAPITRE 1

SECURITE INFORMATIQUE ET SYSTEMES DE DETECTION D'INTRUSIONS

Chapitre 1

Sécurité informatique et systèmes de détection d'intrusions

1.1. Introduction

La sécurité des systèmes informatiques consiste à protéger les systèmes, les réseaux et les données informatiques par des mécanismes d'authentification, d'autorisation, de contrôle d'accès et de détection de vulnérabilités et d'intrusions. Toutefois, avec le développement technologique des ces dernières années et notamment avec l'ouverture des entreprises et des particuliers à Internet, l'assurance de la sécurité des systèmes devient très difficile, du fait que, les attaques et les intrusions augmentent de plus en plus et deviennent de jour à l'autre plus complexes et difficiles à éviter. Ainsi, dans ce chapitre introductif, nous présenterons les notions de base de la sécurité des systèmes informatiques et les différentes attaques possibles qui peuvent se produire ainsi que les mécanismes et les outils qui peuvent être mis en place pour assurer la sécurité.

1.2. Sécurité informatique

La sécurité informatique est définie comme étant l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaire pour protéger les systèmes contre les menaces et les attaques accidentelles et intentionnelles. Cette protection doit assurer les trois principaux objectifs : l'intégrité, la disponibilité et la confidentialité des ressources du système informatique hôte.[1]

Confidentialité : Les données ne sont accessibles que par les utilisateurs et processus autorisés.

Intégrité : les données ne seront modifiées qu'à des fins qui ne sont pas malveillantes ou illégitimes.

Disponibilité : Tout utilisateur ou processus autorisé doit avoir les données disponibles chaque fois qu'il en a besoin.

Figure 1.1 : Triade CIA

1.3. Problèmes de sécurité informatique

1.3.1. Vulnérabilités

Une vulnérabilité informatique est une faille de sécurité, le plus souvent cachée, touchant une infrastructure informatique. Ce terme est fréquemment associé aux logiciels mais il regroupe plus généralement toute faiblesse quelle qu'en soit la nature. Une erreur de configuration d'un équipement réseau constitue une vulnérabilité tout comme un mot de passe vide ou trivial.

1.3.1.a. Vulnérabilités réseau

Les vulnérabilités réseaux regroupent toutes les erreurs de conception et d'implémentation des protocoles de la pile tcp/ip (Transmission Control Protocol/Internet Protocol). A titre d'exemple, nous citons :

ARP Spoofing (Address Resolution Protocol) : Une faille de sécurité dans le protocole ARP permettant aux hackers de rediriger le trafic d'une machine vers une autre. Grâce à cette redirection, une personne mal intentionnée peut se faire passer pour une autre. De plus, le pirate peut rerouter les paquets qu'il reçoit vers le véritable destinataire, ainsi l'utilisateur usurpé ne se rendra compte de rien. [2] [3]

DNS spoofing (Domain Name Server) [3] **:** Une faille dans le protocole DNS permettant aux hackers de forcer une entrée DNS pour pointer vers un protocole Internet incorrects. En DNS spoofing, le serveur DNS accepte des informations

incorrectes à partir d'une source non autorisée pour pointer vers une fausse IP. Pour les serveurs DNS vulnérables à l'usurpation, la pratique est une menace de sécurité importante, car les utilisateurs peuvent être redirigés vers des sites Internet mauvais, ou leurs e-mails sortants peuvent être interceptés pour être acheminés vers des serveurs de messagerie non autorisés.

En effet, quand un utilisateur tape une URL (Uniform Resource Locator) dans le navigateur Web, comme Internet Explorer ou Mozilla Firefox, une requête est envoyée à partir du navigateur à un serveur DNS pour récupérer l'adresse IP (Internet Protocol) appropriée pour l'tapé dans l'URL. Dès l'obtention de l'adresse IP du serveur DNS, le navigateur est capable de se connecter à l'adresse IP du site Web, puis télécharger et afficher le contenu de la page. Dans le détournement de DNS, lorsqu'un utilisateur tape l'URL, le serveur DNS renvoie une adresse IP inexactes, et le navigateur est obligé de charger un site contrefait.

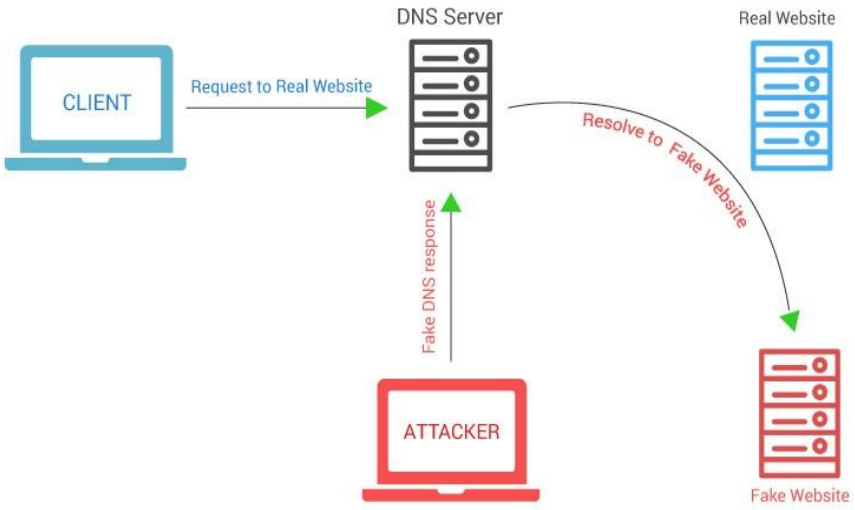

Figure 1.2 : Exemple d'attaque DNS Spoofing

1.3.1.b. Vulnérabilités systèmes

Les vulnérabilités systèmes regroupent toutes les erreurs de conception et d'implémentation des fonctions systèmes et des applications utilisateurs. A titre d'exemple, nous citons :

Les injections SQL (Structured Query Language) **:** Une faille dans la plupart des serveurs SQL qui ne vérifient pas les paramètres d'entrée des formules des pages clientes. En exploitant cette vulnérabilité le pirate peut facilement injecter du code

SQL dans une requête de base de données. Ainsi, il est possible de récupérer des informations se trouvant dans la base (exemple : des mots de passe) ou encore de détruire des données.

Les buffers overflow : En fait, **les** tampons (buffers) sont des régions de stockage de mémoire qui contiennent temporairement des données pendant leur transfert d'un emplacement à un autre. Un buffer overflow se produit lorsque le volume de données dépasse la capacité de stockage de la mémoire tampon. En conséquence, le programme tentant d'écrire les données dans le tampon écrase les emplacements de mémoire adjacents. [4]

Les débordements de tampon peuvent affecter tous les types de logiciels. Ils résultent généralement d'entrées mal formées ou d'un échec à allouer suffisamment d'espace pour le tampon. Si la transaction écrase le code exécutable, le programme peut se comporter de manière imprévisible et générer des résultats incorrects, des erreurs d'accès à la mémoire ou des plantages.

1.3.2. Attaques informatiques

Une attaque informatique est une action malveillante qui essaye d'exploiter une vulnérabilité dans un système informatique afin d'obtenir un accès frauduleux au système, voler des informations, tels que des secrets industriels ou des propriétés intellectuelles, ou perturber le fonctionnement de ce dernier.

1.3.2.a. Cycle de vie d'une attaque

Les attaques informatiques passent généralement par trois étapes différentes :

Renseignement : collecter le maximum d'informations techniques sur la victime et tracer une cartographie plus ou moins détaillée sur celle-ci.

Exploitation : exploiter les vulnérabilités découvertes dans la première étape en essayant d'usurper une des trois objectifs de la sécurité informatique cités plus haut.

Progression : dans cette étape le pirate prépare d'autres attaques, en même temps il essaye d'effacer tout la trace de l'attaque déjà réalisée.

1.3.2.b. Classes d'attaques informatiques

Les attaques informatiques sont généralement classées dans l'une des classes suivantes :

Le déni de service (DOS) : il s'agit d'une classe d'attaques visant à rendre indisponible un service, un système, une machine ou un réseau informatique. Ceci peut s'effectuer de plusieurs manières, par le biais d'une surcharge réseau rendant ainsi le système visé totalement injoignable ou bien de manière applicative en crashant l'application à distance. Grâce à quelques instructions malicieuses et suite à une erreur de programmation, une personne mal intentionnée peut rendre indisponible un service (serveur web, serveur de messagerie) voire un système complet. Des attaque comme le Syn fooding, le udp flooding, le packet fragment et le smurfling sont des exemples d'attaques réseaux qui permettent de rendre indisponible un service.[5][7]

Probing : regroupe les attaques dont l'objectif est de collecter des informations sur des machines, des systèmes et des réseaux. Cette opération peut s'effectuer de plusieurs manières différentes, par exemple un scan de ports grâce au programme Nmap pour déterminer la version des logiciels utilisés, ou encore un scan de vulnérabilités à l'aide du programme Nessus. Pour découvrir les problèmes de sécurité d'un serveur web, les pirates peuvent utiliser des outils. Ils peuvent également utiliser des outils comme firewalk, hping ou SNMP Walk qui permettent quant à eux de découvrir la nature d'un réseau.[6][7]

Remote to Local (R2L) : c'est une catégorie d'attaques dans laquelle l'attaquant exploite les vulnérabilités d'une machine distante comme les bugs des applications, les mauvaises configurations des systèmes d'exploitation et d'autres afin d'obtenir un accès illégal à celle-ci en bénéficiant des privilèges d'un utilisateur local. Plusieurs attaques se trouvent dans cette catégorie telles que : xlock, guest, xnsnoop, phf, warezmaster, spy, warezclient...etc.[6][7]

User to Root (U2R) : il s'agit d'une classe d'attaques dans laquelle le pirate tente d'abuser des vulnérabilités du système afin d'obtenir des privilèges de super utilisateur partant d'un compte utilisateur normal [6]. En d'autre terme, l'objectif

de cette attaque est d'obtenir les privilèges de l'administrateur système (Root) en allant d'un simple compte utilisateur (User) et cela en exploitant des failles dans le système comme le débordement de tampon, les erreurs de programmation..etc. Il existe plusieurs attaques de ce type comme Eject, Ffbconfig, Fdformat, Load module, Perl, Xterm...etc.[6][7]

Attaque par force brute : il s'agit d'une méthode ancienne de cassage de mot de passe qui est très répandue chez la communauté des pirates. Elle consiste à tester, l'une après l'autre, chaque combinaison possible d'un mot de passe ou d'une clé pour un identifiant donné afin de se connecter au service ciblé au nom de celui-ci.

Le temps nécessaire à celle-ci dépend du nombre de possibilités, de la vitesse que met l'attaquant pour tester chaque combinaison et des défenses qui lui sont opposées.

Ce type d'attaque étant relativement simple, ainsi pour se protéger de ce type de comportement, il suffit de bloquer le compte après un nombre limité d'échecs d'authentification pour un même identifiant.

Man In The Middle : L'attaque man-in-the-middle (Homme au milieu) est une technique de piratage informatique permettant d'intercepter des échanges cryptés entre deux personnes ou deux ordinateurs A et B dans le but de décrypter les messages de ces derniers. L'attaquant doit donc être capable de recevoir les messages des deux parties et d'envoyer des réponses à une partie en se faisant passer pour l'autre.

1.3.2.c. Attaques par programmes malveillants

Un programme malveillant peut être décrit comme un programme indésirable installé dans un système informatique à l'insu de son propriétaire. Il peut s'attacher à un code légitime et se propager, se cacher dans des applications utiles ou se reproduire sur Internet.

Voici les types de programmes malveillants les plus courants :

Le virus : est un code malveillant qui est capable de s'auto-reproduire en s'attachant dans d'autres programmes légaux dont l'objectif est de perturber le fonctionnement normal du système infecté.

Il existe différents types de virus comme les virus macro, les virus de boot, les virus d'exécutables, les virus polymorphiques, métamorphiques, etc. Les virus peuvent s'infiltrer dans un système informatique par l'ouverture d'un message (mail, MMS, chat), d'une pièce jointe ou d'un clic sur un lien frauduleux.[8]

Les vers : un ver informatique est un programme malveillant autonome (ne se cache pas dans un autre programme légal) qui se propage automatiquement sur le réseau en essayant d'infecter le maximum de systèmes. Il permet d'espionner l'activité d'un poste, de détruire ou de corrompre des données, d'ouvrir une porte dérobée aux hackers, etc. Le ver peut également être employé pour réaliser une attaque par déni de service. C'est-à-dire, saturer un réseau ou un site Web cible par génération massive de données afin de le rendre inaccessible.[8] [9]

Bombe logique Il s'agit d'un type de programmes malveillants qui se déclenchent par un événement spécifique, comme une condition logique ou une date et une heure spécifiques.

Les chevaux de Troie : Un cheval de Troie est une forme de logiciel malveillant déguisé en logiciel utile. Son but : se faire exécuter par l'utilisateur, ce qui lui permet de contrôler l'ordinateur et de s'en servir pour ses propres fins. Généralement d'autres programmes malveillants seront installés sur votre ordinateur, dans le but de collecter frauduleusement des données, de les falsifier ou de les détruire.

1.4. Mécanismes et outils de protection

a) Chiffrement

Lorsque les données circulent sur Internet en clair sans aucune mesure pour les protéger, elles courent le risque d'être interceptées, compromises ou volées par des personnes malveillantes. Pour éviter cette menace, les données sensibles traversant le réseau Internet doivent être chiffrées avant d'être envoyées ver le réseau.

Le chiffrement désigne la conversion d'un texte brut lisible par l'être humain en un texte incompréhensible, appelé texte chiffré. Cette conversion correspond à la capture de données lisibles et à leur modification en données

apparemment aléatoires. Le chiffrement implique l'utilisation d'une clé cryptographique, c'est-à-dire un ensemble de valeurs mathématiques convenues par l'expéditeur et le destinataire. Le destinataire utilise la clé pour déchiffrer les données et les transformer en texte brut lisible.[10]

Les deux principales techniques de chiffrement sont le chiffrement symétrique et asymétrique. Ces noms font référence à la clé, qui peut être la même ou non pour le chiffrement et le déchiffrement.

Chiffrement symétrique : La clé utilisée pour encoder le texte source est la même que celle utilisée pour décoder celui-ci. DES est un exemple d'un algorithme de chiffrement symétrique.

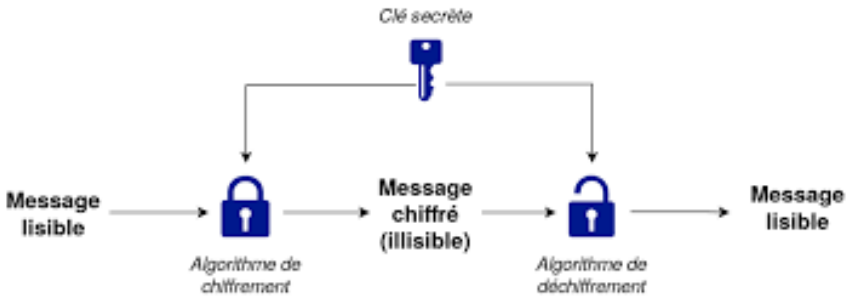

Figure 1.3 : Chiffrement symétrique.

Chiffrement asymétrique : cette méthode utilise deux clés différentes (publique et privée), la clé privée est tenue secrète par le propriétaire pour le chiffrement et la clé publique sera partagée publiquement entre les destinataires autorisés pour le déchiffrement. RSA est un exemple d'un algorithme de chiffrement asymétrique.

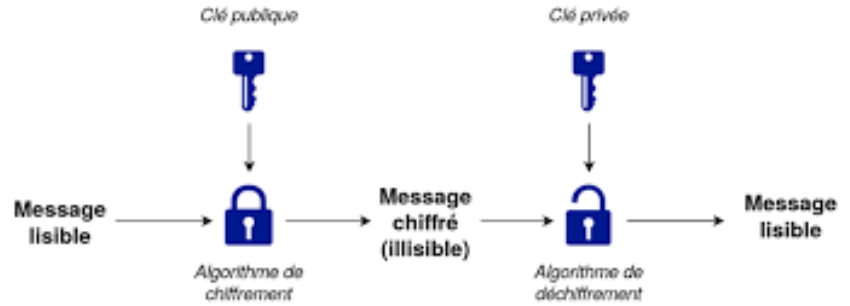

Figure 1.4 : Chiffrement asymétrique.

b) Par feu

Un pare-feu est un équipement matériel ou logiciel ou une combinaison des deux qui est destiné à protéger un système ou un réseau en surveillant le trafic entrant et sortant de celui-ci en décidant d'autoriser ou de bloquer toute ou une partie de ce trafic en fonction d'un ensemble de règles de sécurité prédéfinies [11]. Les pare-feux constituent la première ligne de défense des réseaux depuis plus de 25 ans. Ils établissent une barrière entre les réseaux internes sécurisés et contrôlés qui sont dignes de confiance et les réseaux externes non fiables tels qu'internet.

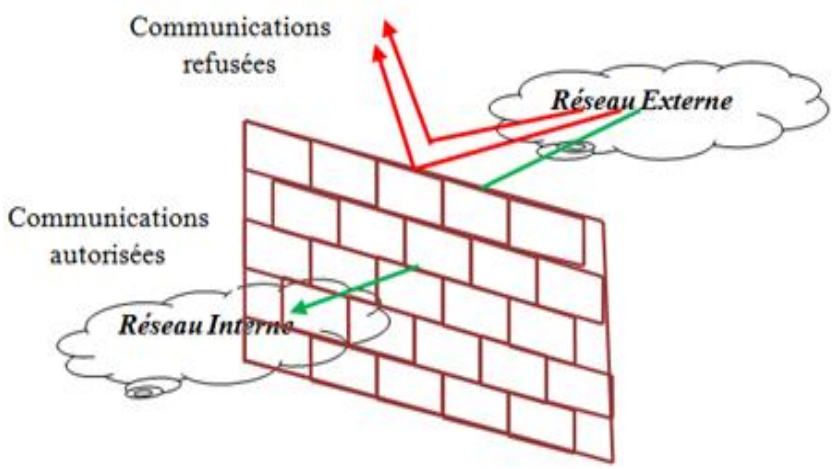

Figure 1.5 : Pare-feu

c) Antivirus

Un antivirus est un programme de sécurité qui scanne les fichiers sauvegardés dans une machine ou dans un support amovibles (clefs USB, CD, DVD, etc.), les courriers électroniques, les secteurs de démarrage, les programmes et les données chargés dans la mémoire vive de l'ordinateur, les données qui transitent sur les éventuels réseaux (dont internet), etc. dans le but de détecter les fichiers infectés par des virus.

Le fonctionnement des principaux antivirus du marché se concentre essentiellement sur des fichiers en comparant alors la signature virale du virus aux codes à vérifier. L'analyse de forme repose sur le principe de filtrage à base de règles, qui sont mises dans un fichier junk. Cette dernière méthode peut être très efficace pour les serveurs de messagerie électronique supportant les regexp type postfixe puisqu'elle ne repose pas sur un fichier de signatures.

Les antivirus peuvent balayer le contenu d'un disque dur, mais également celui de la mémoire vive de l'ordinateur.

d) Audit de sécurité

La plupart des systèmes d'exploitation et des applications sont équipés de mécanismes pour enregistrer les événements significatifs dans ce que l'on appelle des "journaux" (log). Ces enregistrements se trouvent dans des journaux qui peuvent être analysés en temps réel ou hors ligne, pour détecter les activités malveillantes et autres activités qui violent les politiques de sécurité. Une autre technique d'audit avancée consiste à installer un système de détection d'intrusion sur la machine ou le réseau cible.

e) Détection d'intrusions

En fait, la détection des activités malveillantes permettent aux administrateurs du réseau de réagir au bon moment pour faire face et pour arrêter une attaque en cours. Vu l'importance de la détection des intrusions, une variété de systèmes de détection d'intrusion (IDS) a été conçue et développée au fil des années.[12]

1.5. Systèmes de détection d'intrusions (IDS)

1.5.1. Définition

La détection d'intrusion est définie comme étant un mécanisme de sécurité permettant d'écouter le trafic réseau de manière furtive, afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une stratégie de prévention sur les risques d'attaques.[13]

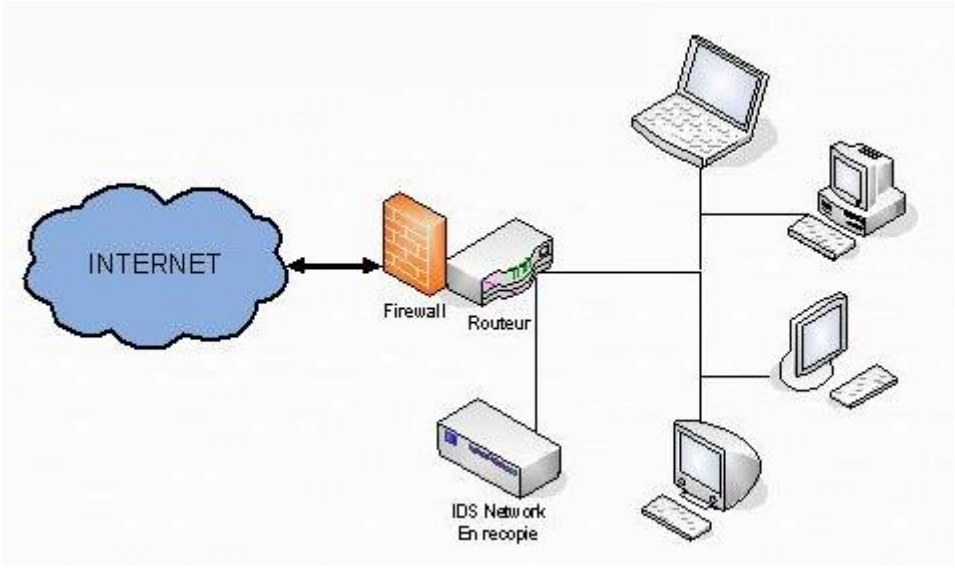

Figure 1.6 : Exemple IDS réseau.

1.5.2. Types des systèmes de détection d'intrusions

Selon son placement, sa position et la nature du système à protéger, nous pouvons distinguer deux types différents de systèmes de détection d'intrusions : les systèmes de détection d'intrusions hôte (H-IDS : Host-Based IDS) et les systèmes de détection d'intrusion réseau (N-IDS : Network-Besed IDS). Pour le premier type, le système de détection d'intrusions est installé dans une machine hôte spécifique et ne détecte donc que les activités malicieuses destinées à cette machine. A l'opposé des systèmes de détection d'intrusions hôte, les systèmes de détection d'intrusions réseau sont destinés à protéger la totalité du réseau et non seulement la machine où l'IDS est installé, pour atteindre cet objectif, le NIDS doit être placé dans un endroit bien défini pour pouvoir capturer et analyser le trafic de toutes les machines du réseau.

1.5.3. Approches des systèmes de détection d'intrusions

Pour détecter une intrusion, le système de détection d'intrusion se base sur une des deux approches suivantes :

a) Approche par signatures

Dans cette approche, la détection des intrusions est basée sur la comparaison du trafic capturé ou des activités observées avec des signatures d'attaques qui sont sauvegardées dans une base de signatures contenant les

signatures de toutes les attaques connues. Cette approche a l'avantage de détecter avec efficacité toutes les attaques connues dont leurs signatures sont sauvegardées dans la base de signatures, mais elle a aussi quelques inconvénients, telles que la difficulté de construction des bases de signatures qui sont généralement construits manuellement, ainsi que la difficulté de détection des nouvelles attaques (0-day attacks) et celles qui sont inconnues pour la base de signature.

b) Approche comportementale

Contrairement aux systèmes de détection d'intrusions par signatures, les systèmes de détection d'intrusion à base de comportement ne se reposent pas sur des signatures décrivant les attaques, mais plutôt sur la déviation des activités du système surveillé du comportement normal que ce dernier doit respecter. Pour ce faire, le système va reposer sur deux phases :

- Une phase d'apprentissage, au cours de laquelle le système va définir et apprendre le comportement normal de la machine ou du réseau.

- Une phase de détection, dans laquelle le système analyse le trafic ou les activités de la machine ou du réseau surveillé en cherchant à identifier les événements anormaux en se basant sur les connaissances déjà acquises dans la première phase.

Cette approche a l'avantage de pouvoir détecter les attaques inconnues ainsi que les nouvelles attaques (0-day attacks), mais elles souffrent aussi de quelques inconvénients tels que le nombre élevé de fausses alertes qu'elles génèrent, où une légère déviation normale du comportement sera considérée comme une attaque.

1.5.4. Réaction des systèmes de détection d'intrusions

Après avoir détecté une attaque, le système de détection d'intrusion va réagir de différentes manières, telles que :

- Reconfiguration d'équipement tiers (firewall, ACL (Access Control List) sur routeurs) : ordre envoyé par l'IDS à un équipement tiers (filtreur de paquets, pare-feu) pour une reconfiguration immédiate dans le but de bloquer le trafic de l'intrusion détectée. Cette reconfiguration est possible par passage des informations détaillant une alerte (en tête(s) de paquet(s)).

- Envoi d'une trap SNMP à un hyperviseur tiers : envoi de l'alerte avec toutes les informations nécessaires sous forme d'un datagramme SNMP à une console tierce comme HP OpenView, Tivoli, Cabletron Spectrum, etc.

- Envoi d'un e-mail à un ou plusieurs utilisateurs : pour notifier la détection d'une intrusion sérieuse.

- Journalisation (log) de l'attaque : en sauvegardant le détail de l'intrusion détectée dans un fichier log, tel que : le timestamp, @IP de l'intrus, @IP de la cible, protocole utilisé, payload, etc.

- Sauvegarde des paquets suspicieux suspects (raw packets) qui ont déclenchés une alerte.

- Lancement d'un programme extérieur pour exécuter une action spécifique (envoi d'un message sms, émission d'une alerte auditive, etc.).

- Construction et envoi d'un paquet TCP FIN pour forcer la fin d'une connexion (uniquement valable sur des techniques d'intrusions utilisant le protocole de transport TCP).

- Notification visuelle de l'alerte en affichant celle-ci dans une ou plusieurs console(s) de management.

1.5.5. Limites des IDS

Les IDS ont, comme tout système informatique, des limites, dont on peut citer [14] :

Surcharge : L'IDS peut être surchargé, par exemple, un grand nombre d'attaques peuvent également être envoyées afin de surcharger les alertes de l'IDS, des conséquences possibles de cette surcharge peuvent être la saturation des ressources (disque, CPU, mémoire).

Perte de paquets : à cause de limitation des performances, il est possible que des paquets ne soient pas traités par l'IDS, parce que les vitesses de transmission dépassent parfois largement la vitesse de traitement des processeurs.

Vulnérabilité aux dénis de service : un attaquant peut essayer de provoquer un déni de service au niveau du système de détection d'intrusion, ou au niveau du

système d'exploitation de la machine accueillant l'IDS. Une fois l’IDS désactivé, l'attaquant peut tenter tout ce qui lui convient.

1.6. Conclusion

Dans ce premier chapitre, nous avons présenté une vue générale sur les principes de la sécurité informatique, en précisant les différentes étapes d’une attaque ainsi que les différentes techniques utilisées par les attaquants avec les différents mécanismes de protection qui peuvent être employés pour faire face à ces attaques. Puis, nous avons finalisé ce chapitre par la présentation des systèmes de détection d’intrusions qui représentent le noyau de notre projet de fin d’étude.

CHAPITRE 2

INTELLIGENCE ARTIFICIELLE ET METHODES DE CLASSIFICATION

Chapitre 2

Intelligence Artificielle et Méthodes de Classification

2.1. Introduction

En 1950, le mathématicien Alan Turing se posait une question : ” les machines peuvent-elles penser ? “. Pour l’affirmer, Alan a fait un test dans lequel, il a mis un humain en conversation à l’aveugle avec un ordinateur et un autre humain, c'est-à-dire, tous les deux sont cachés l'un de l'autre et de l’ordinateur . La conversation a été préparée par écrit uniquement afin que l'incapacité de l'ordinateur à parler ne soit pas un obstacle à l'épreuve. L’ordinateur réussit le test si l'observateur ne peut pas le différencier d'un humain. Il n'est pas nécessaire que l'ordinateur donne des réponses correctes, mais il suffit de simuler ce qu'une personne pourrait dire. Cette expérience a donné naissance de l'intelligence artificielle. C’est quoi donc l’intelligence artificielle ?

2.2. Intelligence artificielle

2.2.1. Définition

Le concept de l'intelligence artificielle (IA) est un concept récent qui vise à faire penser les machines « comme des humains », en d'autres termes, effectuer des tâches telles que raisonner, planifier, apprendre et comprendre notre langage.

Le but de l’intelligence artificielle (IA) est double. D’une part, l’IA s’attache à résoudre des problèmes qui relèvent d’activités humaines ou animales de nature variée : perception, planification, interprétation de données, diagnostic, prise de décision, compréhension du langage, conception. D’autre part, l’IA cherche à mieux comprendre et modéliser l’intelligence. Elle se rapproche ainsi des sciences cognitives dont elle s’inspire par ailleurs pour la conception de modèles (mémoire, raisonnement, apprentissage).

2.2.2. Utilisation de l'intelligence artificielle

L'IA est de plus en plus présente dans notre quotidien. Elle est par exemple utilisée dans le web et les réseaux pour améliorer les performances des services offerts et pour l'amélioration de la sécurité des systèmes et des réseaux, elle est utilisée aussi par les services de détection des fraudes des établissements financiers, pour la prévision des intentions d'achat et dans les interactions avec les services clients en ligne, et les services : automobile, logistique, énergie, industrie, etc. Aucun secteur d'activité n'est épargné par la montée en puissance de l'intelligence artificielle. Voici quelques exemples :

- Détection des fraudes : Dans le secteur financier, l'intelligence artificielle est utilisée de deux manières. Les applications qui évaluent les demandes de crédit utilisent l'IA pour évaluer la solvabilité des consommateurs. Des outils d'IA plus avancés sont chargés de surveiller et de détecter en temps réel les paiements frauduleux effectués avec des cartes de crédit.

- Service client virtuel (SCV) : les centres d'appels utilisent le SCV pour anticiper et répondre aux demandes des clients sans intervention humaine. Les simulateurs de reconnaissance vocale et de dialogue humain sont le premier point d'interaction avec le service client. Les exigences les plus complexes nécessitent une intervention humaine.

- Lorsque les internautes ouvrent une fenêtre de dialogue sur un site web (chatbot), leur interlocuteur est généralement un ordinateur exécutant un formulaire d'IA spécialisé. Si le chatbot ne peut pas expliquer la question ou résoudre le problème, un agent humain prend le relais. Ces erreurs d'interprétation sont envoyées au système d'apprentissage automatique pour améliorer les interactions futures des applications d'IA.

2.2.3. Apprentissage

L'apprentissage artificiel, ou Machine Learning en anglais, c'est le processus de construire un modèle général à partir de données (observations) particulières du monde réel.

Objectif : extraire et exploiter automatiquement l'information présente dans un jeu de données, généralement, pour construire des modèles ou des systèmes dédiés à la détection, la recherche, la localisation ou la reconnaissance/identification des classes d'objets (par exemple, voitures, documents, animaux, lire des panneaux publicitaires, etc.).

L'apprentissage automatique (littéralement l'apprentissage machine) ou apprentissage statistique est un domaine d'étude de l'intelligence artificielle qui s'appuie sur des méthodes statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, cela implique la conception, l'analyse, le développement et la mise en œuvre de telles méthodes.

2.3. Classification

Apprendre à classer des choses ou des objets est un problème central de l'intelligence, naturelle comme artificielle. En fait, une règle de classification est un acte ou une procédure cognitive permettant d'affecter à un objet à quelle famille il appartient, autrement dit, de le reconnaître. C'est ainsi qu'une personne apprend à classer les choses, par exemple : animaux en « moutons » ou « vaches », etc.

Généralement, il y en a deux approches différentes : la classification supervisée et non supervisée.

2.3.1. Classification non-supervisée

Dans la classification non supervisée, parfois appelée partitionnement, segmentation ou regroupement (Clustering en anglais), on vise à regrouper/classer dans une même catégorie les individus qui se ressemblent le plus, ceux qui ont des caractéristiques semblables.

De manière générale, la classification non supervisée est souvent caractérisée par :

- on ne connaît pas le nombre de groupes qui existent dans l'ensemble de données.

- on ne connaît pas le groupe auquel appartient chaque observation de l'ensemble de données.
- Classer les observations dans des groupes homogènes à partir de différents attributs.

Il existe plusieurs méthodes de classification non supervisée. Les plus communes sont:

- la classification hiérarchique.
- la classification non hiérarchique, par exemple la méthode des k-moyennes (k-means).
- la classification basée sur une densité.
- la classification basée sur des modèles statistiques/probabilistes, par exemple un mélange de lois normales.

2.3.2. Classification supervisée

En contrepartie, si les classes sont prédéterminées et les exemples connus, le système apprend à classer selon un modèle de classification ou de classement, on parle alors d'apprentissage supervisé. Dans ce type de classification, les données d'apprentissage doivent préalablement être étiquetées par un expert. Le processus se passe en deux phases. Lors de la première phase, il s'agit de déterminer un modèle à partir des données étiquetées. La seconde phase (en ligne, dite de test) consiste à prédire l'étiquette d'une nouvelle donnée, connaissant le modèle préalablement appris. Parfois il est préférable d'associer une donnée non pas à une classe unique, mais une probabilité d'appartenance à chacune des classes prédéterminées (on parle alors d'apprentissage supervisé probabiliste) [15].

La classification non supervisée est généralement caractérisée par :

- La connaissance du nombre de groupes qui existent dans la population.
- La connaissance du groupe auquel appartient chaque observation.
- Classer les observations dans les bons groupes à partir de différentes variables.

2.3.3. Quelques Algorithmes de Classification supervisée

2.3.3.1. Machines à vecteurs de support

Les machines à vecteur de support (SVM : Support Vector Machine en anglais) sont un des techniques d'apprentissage supervisé les plus puissantes qui sont généralement destinées à résoudre des problèmes de discrimination et de régression. Elles sont considérées comme une généralisation des classifieurs linéaires qui peuvent classifier des données autant linéairement séparables que non linéairement séparables grâce à une astuce appelée astuce de noyau [16].

Principe : les SVM ont pour but de séparer les données en classes à l'aide d'une frontière aussi simple que possible, de sorte que la distance entre les différentes classes de données et la frontière qui les sépare soit maximale. Cette distance est aussi appelée « marge » et les SVM sont ainsi qualifiés de « séparateurs à vaste marge », les « vecteurs de support » étant les données les plus proches de la frontière.

Dans cette méthode de classification, il existe un unique hyperplan optimal défini comme l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur [17], la recherche de cet hyperplan revient à résoudre un problème d'optimisation à partir des données d'une base d'apprentissage. Comme le montre la figure 2.1 (B), seuls les points situés sur les hyperplans de marges maximales, appelés vecteurs support, participent à la définition de l'hyperplan optimal.

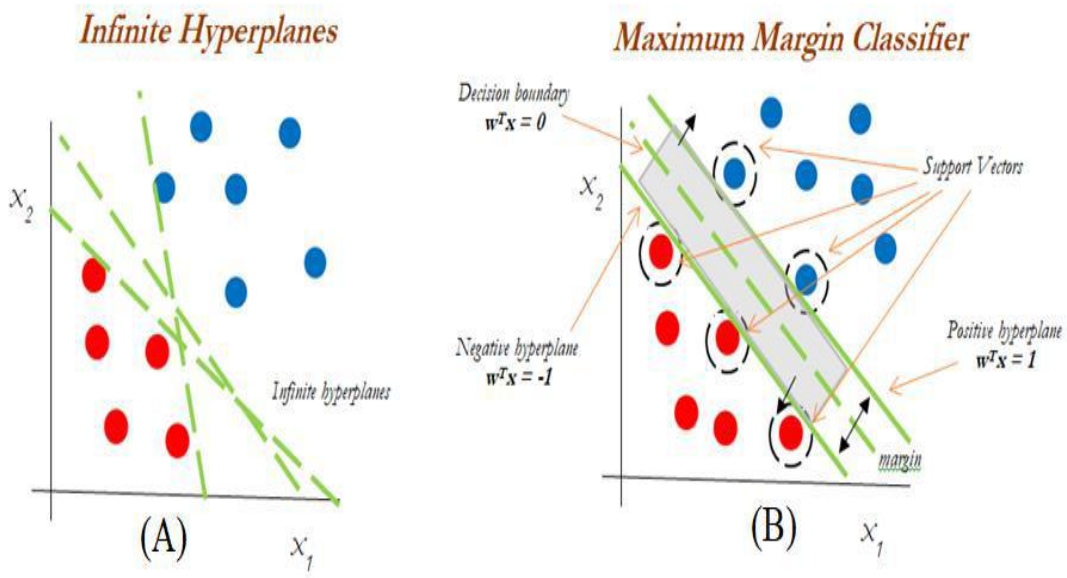

Figure 2.1. Notion de marge maximale.

(A) : représente un ensemble des points linéairement séparables, il existe une infinité d'hyperplans séparateurs.

(B) : lignes en vert représentent l'hyperplan optimal avec la marge maximale, les échantillons entourés représentent les vecteurs supports.

Dans le cas des données non linéairement séparables où il n'existe pas d'hyperplan capable de séparer les données en classes, le SVM utilise une astuce très efficace appelée astuce du noyau ou kernel trick en anglais. L'idée de cette astuce est basée sur l'utilisation d'une transformation non linéaire pour reconstruire les données d'apprentissage dans un espace de dimension supérieure. Les données sont non linéairement séparables dans l'espace d'origine et donc plus facilement séparables dans l'espace de grande dimension, voir Figure 2.2, (trouver une frontière de séparation linéaire dans un espace de grande dimension revient à trouver une frontière non-linéaire dans l'espace de départ). Le but est alors de définir dans le nouvel espace, appelé espace de re-description, un hyperplan permettant de découper les données d'apprentissage de manière optimale, c'est la notion de marge maximale.[18]

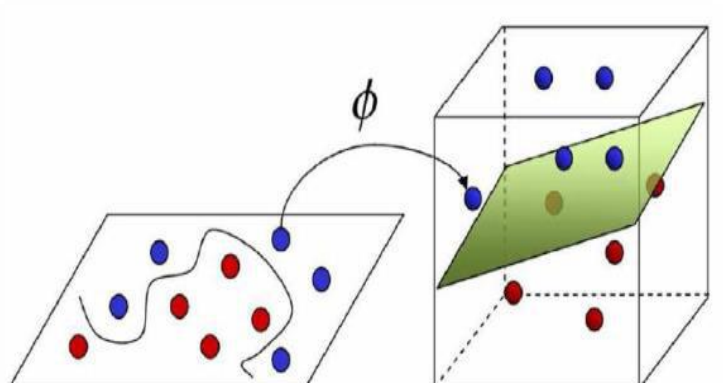

Figure 2.2 : Séparation linéaire dans un espace de grande dimension.

2.3.3.2. Réseaux de Neurones

Le réseau de neurones artificiels est basé sur la modélisation des neurones biologiques (Figure 2.3). L'objectif initial de ce modèle était de reproduire la capacité du cerveau humain à interpoler ou à classifier les informations.

Ainsi, l'axe principal de recherche dans les réseaux de neurones est la classification ou l'interpolation : deux points sont très importants pour modéliser des systèmes complexes dont le comportement ne peut être déterminé mathématiquement.

Les réseaux de neurones artificiels sont aujourd'hui considérés comme des simples et efficaces outils de classification et de raisonnement informatiques.

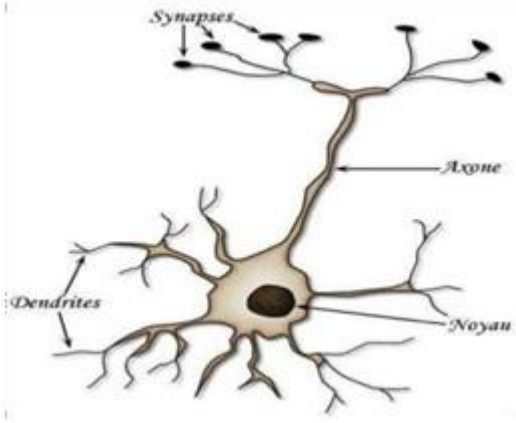

Figure 2.3 : Exemple de neurone biologique.

Le premier modèle de base d'un neurone artificiel (figure 2.4) est proposé par McCulloch et Pitts [19] en 1943. Bien que très simple, Il compose l'élément de base des réseaux de neurones.

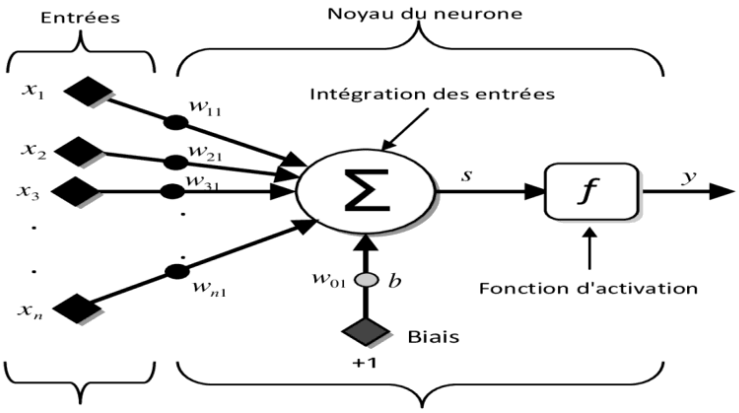

Figure 2.4 : Structure d'un neurone artificiel.

Les x_i de ce modèle représentent les vecteurs d'entrées, ils proviennent soit des sorties d'autres neurones, soit de stimuli sensoriels (capteur visuel, sonore, etc.). Les w_{ij} sont les poids synaptiques du neurone j. Ils correspondent à l'efficacité synaptique dans les neurones biologiques ($w_{ij} > 0$: synapse excitatrice, $w_{ij} < 0$: synapse inhibitrice). Ces poids pondèrent les entrées et peuvent être modifiés par apprentissage. En plus des vecteurs d'entrées et des poids synaptiques, un neurone artificiel est constitué aussi d'un biais et d'un noyau. Le Biais est une entrée qui prend souvent les valeurs -1 ou +1 et qui permet d'ajouter de la flexibilité au modèle en permettant de varier le seuil de déclenchement du neurone par l'ajustement des poids et du biais lors de l'apprentissage. Tandis que le noyau est essentiellement constitué d'un intégrateur qui effectue la somme pondérée « s » des entrées et des biais et qui calcule la sortie « y » du neurone selon une fonction d'activation qui est souvent non linéaire pour donner une plus grande flexibilité d'apprentissage. [19]

En suivant les notations présentées à la section précédente, les n entrées du neurone correspondent au vecteur $x = [x_1, x_2, x_3, \cdots x_n]^T$, alors que $w = [w_{11}, w_{21}, w_{31}, \ldots, w_{n1}]$ représente le vecteur des poids du neurone. La sortie s de l'intégrateur est donnée par l'équation suivante :

$$s = b + \sum_i wi * xi$$

On peut représenter la sortie du modèle par l'équation suivante (avec f la fonction d'activation) :

$$y = f_{activation}(s)$$

$$y = f_{activation}(b + \sum_i wi * xi)$$

En effet, les réseaux de neurones peuvent prendre différentes formes selon les objets de données qu'ils traitent, la complexité et la méthode de traitement des données. De plus, les architectures de réseaux neuronaux peuvent être divisées en 4 grandes familles : [20]

- Réseaux de neurones Feedfowarded
- Réseaux de neurones récurrent (RNN)
- Réseaux de neurones à résonance
- Réseaux de neurones auto-organisés

Ces architectures ont bien évidemment leurs forces et leurs faiblesses, mais elles peuvent bien sûr être combinées pour optimiser leurs résultats. Le choix de l'architecture s'avère ainsi crucial et il est déterminé principalement par l'objectif du modèle.

2.3.3.3. Arbre de décision

Les arbres de décision (AD) sont une catégorie d'arbres utilisée dans l'exploration de données et en informatique décisionnelle. Ils emploient une représentation hiérarchique de la structure des données sous forme des séquences de tests et de décisions en vue de la prédiction d'un résultat ou d'une classe. Chaque observation, qui doit être attribuée à une classe, est décrite par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prise dans les nœuds feuille [21].

Un arbre de décision peut donc être considéré comme un ensemble de règles de classification dont leur décision est essentiellement basée sur des tests

associés aux attributs, organisés de manière arborescente. Il s'agit, en effet, d'une méthode simple, supervisée, et très connue de classification et de prédiction.

a. Construction d'un arbre de décision

La construction d'un arbre de décision consiste tout simplement à définir une suite de nœud, dont chaque nœud permettra de faire une partition des objets en 2 groupes sur la base d'une des variables explicatives [22].

Donc, pour construire un arbre, il faut :

- définir un critère permettant de sélectionner le meilleur nœud possible à une étape donnée,
- définir quand s'arrête le découpage, en définissant un nœud terminal (une feuille de fin),
- attribuer au nœud terminal la classe ou la valeur la plus probable,
- élaguer l'arbre quand le nombre de nœuds devient trop important en sélectionnant un sous arbre optimal à partir de l'arbre maximal,
- valider l'arbre à partir d'une validation croisée ou d'autres techniques.

b. Critère de sélection d'un nœud

La construction d'un nœud doit minimiser l'encombrement des objets de manière optimale. Pour mesurer ce trouble, on définit l'entropie d'une variable qualitative Y à q modalités par :

$$H(Y) = -\sum_{k=1}^{q} P(Y = k) * \log(P(Y = k))$$

avec la convention 0 log (0)=0

On peut ensuite définir l'entropie de Y conditionnée par une variable qualitative X ayant q' modalités.

$$H(Y|X) = -\sum_{k}^{k'} P(Y = k, X = k') * \log\big(P(Y = k|X = k')\big)$$

$$= \sum_{k} P(X = k')H(Y|X = k')$$

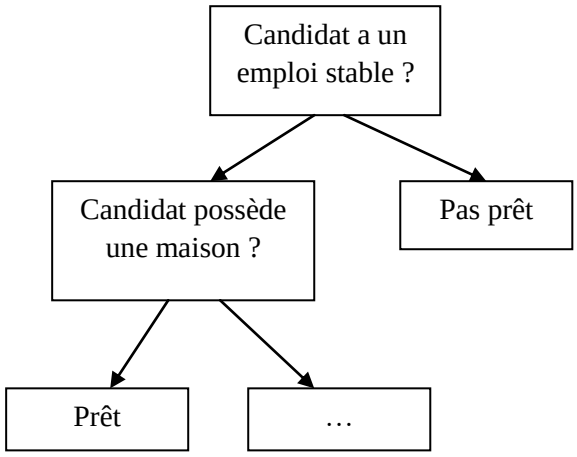

Figure 2.5 : Exemple d'un arbre de décision.

2.3.3.4. K-plus Proches Voisins

L'algorithme de classification des k plus proches voisins ou k-ppv (k-Nearest Neighbor ou k-NN, en anglais) est l'un des algorithmes de classification supervisé les plus simples. Contrairement aux autres algorithmes de classification supervisée, l'algorithme de KNN ne génère pas de modèle et la classification des nouvelles observations se fait directement à partir de la base d'apprentissage.

Une nouvelle observation est classifiée par vote majoritaire de ses k "voisins" (par mesure de distance), c'est-à-dire qu'elle est prédite de classe C si la classe la plus représentée parmi ses k voisins est la classe C. Un cas particulier est le cas où k = 1, la nouvelle observation est alors affectée à la classe de son plus proche voisin.
Cette méthode a l'avantage d'être facile à mettre en œuvre et de donner de bons résultats. Sa principale limitation est liée à la faible vitesse de classification due au grand nombre de distances à calculer.

Exemple, sur la figure 2.6, on peut voir l'effet du choix de k sur le résultat de la classification. En effet, si k=3; l'exemple à prédire (noté "?") serait classifié comme étant de la classe "rouge", mais si k=5, il serait classifié comme étant de la classe "bleu".

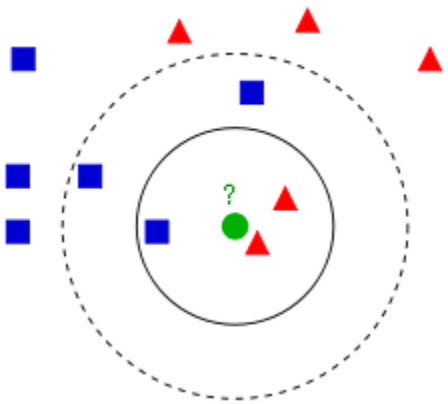

Figure 2.6 : Schéma d'une classification par la méthode k-NN.

2.4. Conclusion

Dans ce deuxième chapitre, nous avons parlé d'abord de l'intelligence artificielle, ensuite nous avons présenté l'apprentissage automatique et la classification des données, où nous avons abordé les concepts de base du domaine, les différents types de classification. Puis, nous avons finalisé le chapitre par une présentation détaillée des quatre algorithmes de classification sur lesquels va se baser notre modèle de classification hybride multi-niveaux.

CHAPITRE 3

IMPLEMENTATION ET RESULTATS

Chapitre 3

Implémentation et résultats

3.1. Introduction

Un système de détection d'intrusions réseau est un système de sécurité qui vise à protéger un réseau informatique contre toute sorte de manipulations non autorisées et d'activités malveillantes. Pour un système de détection d'intrusions comportemental, la phase la plus importante dans le processus de réalisation du système est la phase d'apprentissage qui consiste à construire un classificateur (c'est-à-dire un modèle prédictif) capable de classifier, avec précision, les connexions TCP/IP en connexions normales et connexions malveillantes. La génération de ce modèle prédictif est basée sur un ensemble de données d'apprentissage appelé benchmark d'apprentissage (benchmark NSL-KDD dans notre cas). Une fois généré, le modèle prédictif doit d'abord être évalué avant d'être mis en marche. Pour l'évaluer, nous devons utiliser un autre ensemble de données totalement différent du premier ensemble qui a déjà été utilisé pour l'apprentissage. Ce deuxième ensemble est appelé benchmark de test ou base de test. Ainsi, dans ce dernier chapitre, nous allons détailler la démarche que nous avons suivie pour réaliser notre modèle hybride multi-niveau en utilisant le benchmark NSL-KDD ainsi que les résultats que nous avons obtenus.

3.2. Présentation de l'ensemble de données NSL-KDD

3.2.1. Historique

L'ensemble de données NSL-KDD (**N**etwork **S**ecurity **L**ayer-**K**nowledge **D**iscovery in **D**atabases, en anglais) est un sous-ensemble de l'ensemble de donnée KDD'99 qui a été construit en 2008 dans le but de résoudre certains des problèmes inhérents à l'ensemble de données KDD'99 tels que le nombre très élevé de connexions redondantes. L'ensemble de donnée KDD'99, lui aussi a été créé par la DARPA en 1999 à partir du trafic réseau d'un ensemble d'attaques réseaux et systèmes effectuées par DARPA en 1998 [23]. Ce dernier est constitué

de 4 898 430 enregistrements (chaque enregistrement correspond à une connexion réseau) qui ont été réduits dans NSL-KDD à uniquement 148517 enregistrements.[24]

3.2.2. Description du NSL-KDD

L'ensemble de données NSL-KDD contient 125973 enregistrements pour l'apprentissage et 22544 enregistrements pour le test. Les enregistrements de NSL-KDD sont constitués de 42 attributs dont les 41 premiers attributs représentent les caractéristiques des différentes connexions, tandis que le 42éme attribut indique la nature de la connexion s'il s'agit d'une attaque ou non. Pour les attaques de NSL-KDD, nous distinguons quatre classes différentes d'attaques : DoS (déni de service), R2L (accès à une machine local à partir d'une machine distante), U2R (accès aux privilèges root (administrateur) à partir d'un compte utilisateur simple) et Probe (sonde le réseau), dont chaque classe regroupe plusieurs types d'attaques comme le montre le tableau 1 suivant. La distribution des connexions normales et des attaques des quatre classes sur les deux ensembles de données (d'apprentissage et de test) est montrée dans le tableau 2 [25].

Classe	Attaques
Dos	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

Tableau 3.1 : Répartition des attaques dans NSL-KDD

Apprentissage		**Test**	
Classe	**Nombre d'enregistrements**	**Classe**	**Nombre d'enregistrements**
Normal	67343	Normal	9711
Dos	45927	Dos	7458
Probe	11656	Probe	2421
R2L	995	R2L	2754
U2R	52	U2R	200
Total	125973	**Total**	22544

Tableau 3.2 : Distribution des connexions de l'ensemble de données NSL-KDD.

3.2.3. Attributs de la base NSL-KDD

Les attributs la base NSL-KDD sont classés en trois groupes :

- **Les attributs de base:** ces attributs décrivent les informations de base d'une connexion, telles que la durée, les hôtes source et destination, port et flag.

- **Les attributs du trafic:** ces attributs contiennent des informations statistiques des différentes connexions, tels que le nombre d'octets envoyés et reçus dans chaque connexion.

- **Les attributs du contenu:** ces attributs sont construits à partir de la charge utile des paquets du trafic tels que le nombre d'échec de connexion et le nombre d'accès aux fichiers de contrôle.

La description des attributs de la base NSL-KDD est détaillée dans l'annexe de ce mémoire.

3.3. Processus de génération de système

Pour réaliser notre modèle de détection d'intrusions hybride multi-niveaux, nous avons passé par trois étapes différentes :

- Dans la première étape, nous avons réalisé un prétraitement de la base NSL-KDD afin de préparer et d'adapter ses données avec les exigences de notre modèle.
- Dans la deuxième étape, nous avons généré et testé quatre modèles différents (SVM, KNN, MLP et arbre de décision) pour chaque classe d'attaque (DOS, U2R, R2L et Probe).
- Dans la troisième étape, nous avons généré et testé notre modèle hybride multi-niveau, où chaque niveau est conçu pour détecter une classe d'attaque précise (DOS, U2R, R2L ou Probe) en utilisant le modèle de classification le plus adapté en se basant sur les résultats de la deuxième étape.

Ces trois étapes sont résumées dans le schéma de la figure suivante.

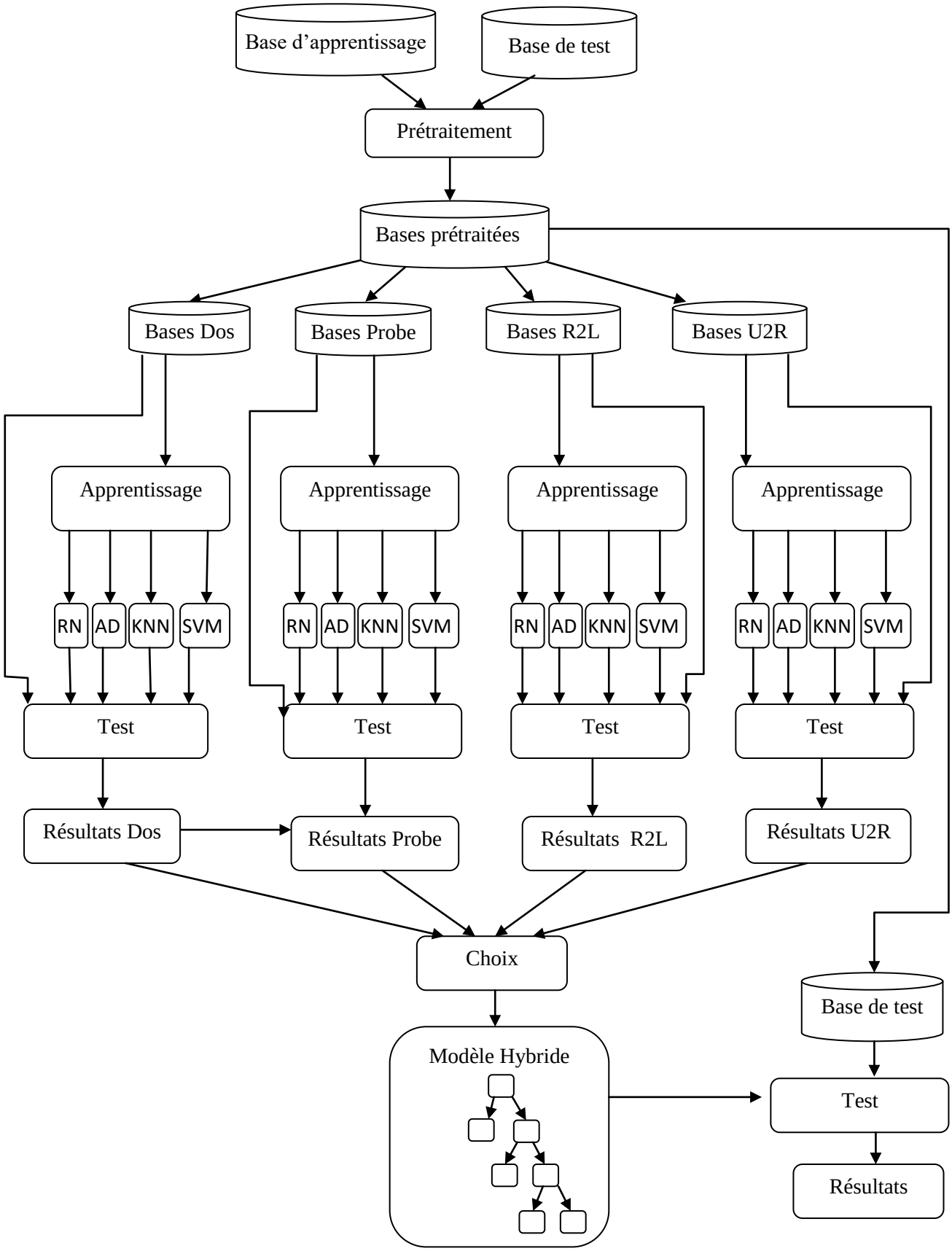

Figure 3.1 : Processus de génération de notre modèle hybride.

3.3.1. Prétraitement des données

Le prétraitement des données est la phase la plus importante du processus de génération des modèles de classification. Il vise à modifier l'ensemble de données pour le rendre plus lisible et plus adapté aux algorithmes de classification. Dans notre cas, les opérations de prétraitement que nous avons réalisées sont résumées dans les points suivants :

a) Changement des noms des attaques

Puisque le modèle que nous visons à construire est un modèle hybride multi niveaux, dont chaque niveau correspond à une classe d'attaque différente (Dos, Probe, R2L, U2R), les noms des attaques de la base NSL-KDD sont remplacés par les noms des classes qui leur correspondent.

b) Création d'un ensemble de données séparé pour chaque classe

Après le changement des noms des attaques de l'ensemble de données, nous avons créé quatre copies différentes de l'ensemble de données, une copie par classe d'attaques. Pour chaque copie, nous avons gardé uniquement une seule classe d'attaques, les autres classes sont remplacées par le mot « normal ».

c) Numérisation

Pour pouvoir utiliser l'ensemble de données avec les quatre modèles de classification (SVM, KNN, MLP et arbre de décision), les valeurs des attributs de la base NSL-KDD doivent être toutes numériques. Parmi les 41 attributs de notre ensemble de données, trois attributs sont alphabétique, qui sont : protocol_type, service et flag, il est donc nécessaire de rendre toutes les valeurs de ces attributs numériques. Dans notre cas, pour chaque attribut de N valeurs alphabétiques, ses valeurs sont remplacées par des valeurs entières comprises entre 0 et N-1, y compris l'attribut des classes d'attaques (variable cible) dont ses valeurs sont remplacées par les valeurs numériques 0, 1, 2, 3 et 4 (pour Normal, Dos, Probe, R2L et U2R respectivement).

d) Normalisation

Afin d'améliorer et d'augmenter les performances de notre modèle de classification, il est très important d'ajuster et d'amender les valeurs des attributs pour les rendre tous dans le même intervalle. Pour ce faire, il suffit d'appliquer

certaines modifications sur les données de la base d'apprentissage et de test, en utilisant un des fonctions de normalisation des données. Dans notre cas, nous avons utilisé la fonction min-max qui est donnée par la formule suivante :

$$valeur_{norm} = \frac{valeur - Min_{intv}}{Max_{intv} - Min_{intv}}$$

Où :

- *valeur* : la valeur à normaliser.
- $valeur_{norm}$: la valeur après la normalisation.
- Min_{intv} : la valeur la plus petite de l'intervalle à qui *valeur* appartient.
- Max_{intv} : la valeur la plus grande de l'intervalle à qui *valeur* appartient.

e) Sélection des meilleurs attributs

L'ensemble de données NSL-KDD se constitue de 41 attributs, comme nous avons vu précédemment. L'utilisation de tous ces attributs pour générer notre modèle de classification peut affecter considérablement les performances de notre système en terme de temps de réponse ainsi que en terme de ressources utilisées. Pour éviter tous ces problèmes et pour améliorer les performances ainsi que l'efficacité de notre système, il est très important de choisir et d'utiliser uniquement les attributs les plus significatives et les plus pertinents. Pour ce faire, nous avons utilisé l'algorithme d'élimination récursive d'attributs (Recursive Feature Elimination, en anglais), ou RFE, en abrégé.

L'objectif de RFE est de sélectionner des attributs en considérant de manière récursive des ensembles des attributs de plus en plus petits. Tout d'abord, l'estimateur est formé sur l'ensemble initial des attributs et l'importance de chaque attributs est obtenue soit par n'importe quel attribut spécifique, soit appelable. Ensuite, les attributs les moins importantes sont supprimés de l'ensemble actuel des attributs. Cette procédure est répétée de manière récursive sur l'ensemble élagué jusqu'à ce que les attributs qui sont plus ou plus pertinentes pour prédire la variable cible soit finalement atteint.

f) Equilibrage des données

Comme le montre le tableau 2 précédent, il y a un grand déséquilibre dans la distribution des connexions de l'ensemble de données NSL-KDD. Par exemple,

le nombre de connexions correspondant au trafic normal représente plus de 50% du nombre total des connexions, alors que celui correspondant aux attaques R2L et U2R ne représente que 1% du nombre total des connexions. Malheureusement, ce déséquilibre va conduire à une classification qui sera biaisée beaucoup plus en faveur du trafic normal au détriment du trafic correspondant aux attaques R2L et U2R, ce qui va affectera considérablement la performance de notre système de détection d'intrusions [26].

Pour résoudre ce problème, nous avons utilisé deux algorithmes de rééquilibrage des données d'entrainement, un pour le problème de sous-échantillonnage (undersampling, en anglais), et l'autre pour le problème de sur-échantillonnage (oversampling, en anglais). Pour le problème de sous-échantillonnage, nous avons utilisé la technique SMOTE **(**Synthetic Minority Oversampling Technique, en anglais**).** Le principe de cette technique consiste à sélectionner un exemple aléatoire de la classe minoritaire, ensuite, les voisins les plus proches pour cet exemple sont trouvés, puis un voisin sélectionné au hasard est choisi pour créer un nouvel exemple synthétique dans l'espace des caractéristiques [27]. De même pour le sur-échantillonnage, nous avons utilisé une technique qui cherche à sélectionner et à supprimer des échantillons de la classe majoritaire pour réduire le nombre de connexions de la classe majoritaire [27]. Nous avons combiné les deux techniques pour former un ensemble de données bien équilibré.

3.3.2. Apprentissage et génération du modèle de classification

L'objectif de notre travail est d'établir un modèle hybride multi niveaux pour la détection d'intrusions, dont chaque niveau représente un sous-modèle spécialisé dans la détection d'une classe précise d'attaque, nous aurons donc quatre niveaux différents, un niveau par classe d'attaque (DOS, U2R, R2L et Probe), avec quatre sous-modèles différents, un sous-modèle par niveau (SVM, MLP, KNN et DT). C'est-à-dire, dans chaque niveau de notre modèle de classification hybride, nous allons implémenter un sous-modèle de classification d'une classe d'attaque bien précise en se basant sur un des algorithmes de classification suivantes : les machines à vecteurs de support (SVM), le perceptron multicouche (MLP), les K plus proches voisins (KNN pour K-Nearest Neighbors,

en anglais), et les arbres de décision (DT pour Decision Tree, en anglais). Le problème qui se pose lors de la génération de notre modèle hybride est comment choisir le bon algorithme de classification que nous devons mettre dans chaque niveau. Pour résoudre ce problème, nous avons généré et testé 4 sous-modèles différents (SVM, MLP, KNN et DT) pour chaque classe d'attaque, ça fait au total 16 sous-modèles différents. Puis, à partir des résultats de ces sous modèles, nous choisi le sous-modèle le plus performant pour chaque niveau (classe d'attaque).

3.3.3. Test et évaluation du modèle généré

Pour pouvoir évaluer les performances de notre modèle de détection d'intrusions, nous devons calculer quelques métriques lors de la phase de test telles que la matrice de confusion, la précision, le rappel...etc. Ces métriques sont définies comme suit : [26]

- la matrice de confusion : est une matrice qui rassemble en lignes les observations et en colonnes les prédictions. Les éléments de la matrice représentent le nombre d'exemples correspondant à chaque cas.

		Classe prédite	
		Négatif (Normal)	Positif (attaque)
Classe réelle	Négatif (Normal)	**VN** (Vrai négatif)	**FP** (Faux positif)
	Positif (attaque)	**FN** (Faux négatif)	**VP** (Vrai positif)

Tableau 3.3 : Matrice de confusion.

Où :

- VN : représente les vrais négatifs, c'est-à-dire, les activités normales correctement classées comme normal.
- FN : représente les faux négatifs, c'est-à-dire, les attaques considérées comme trafic normal.
- VP : représente les vrais positifs, c'est-à-dire, les attaques correctement détectées.
- FP : représente les faux positifs, c'est-à-dire, les activités normales considérées comme attaques.

- Taux de réussite : le ratio des enregistrements correctement classés.

$$\boldsymbol{Taux\ de\ réssite} = \frac{VN + VP}{VN + FN + VP + FP} * 100\ \%$$

- **Précision :** la proportion des résultats prédits positivement et qui sont réellement positifs.

$$\boldsymbol{Précision} = \frac{VP}{VP + FP} * 100\ \%$$

- **Rappel :** le taux des positifs prédits correctement parmi tous les positifs.

$$\boldsymbol{Rappel} = \frac{VP}{VP + FN} * 100\ \%$$

- **F1-Score :** la moyenne harmonique de la précision et le rappel.

$$\boldsymbol{F1 - Score} = 2 * \frac{Précision * Rappel}{Précision + rappel} * 100\ \%$$

3.4. Implémentation

3.4.1 Environnement de programmation

Nous avons choisi l'environnement de programmation IDE PyCharm pour Windows afin d'implémenter notre modules d'apprentissage et de test de notre modèle de détection d'intrusions. Notre choix du langage Python a été argumenté par les avantages offerts par la programmation et l'implémentation des modèles d'apprentissage dans le domaine de l'intelligence artificielle. Il offre plusieurs bibliothèques et des outils prédéfinis pour l'utilisation et l'implémentation comme la bibliothèque Sklearn, et la bibliothèque PyQt5 qui est parfaite pour le désigne des interfaces des applications. L'utilisation de ces bibliothèques facilite grandement le travail du programmeur lors de la construction d'applications complexes. De plus, cet IDE est disponible gratuitement.

Les spécifications techniques du micro-ordinateur que nous avons utilisé pour l'implémentation de notre modèle sont répertoriées dans le tableau suivant :

N°	Composants du matériel	Spécifications techniques
1	Processeur	Intel(R) Core(TM) i3-6100U
2	Vitesse de processeur	2.30GHz
3	Mémoire	8,00 Go
4	Système d'exploitation	Windows 10 64 bits

Tableau 3.4 : Spécifications techniques de notre ordinateur

3.4.2. Résultats Expérimentaux

a) Résultats des 16 sous-modèles préliminaires

Les résultats obtenus pour chaque classe d'attaque par classifieur sont détaillé dans le Tableau 4 ci-dessous.

Classe	Modèle	11 Attributs	
		Taux de réussite	Précision
Dos	SVM	90 %	73 %
	MLP	94 %	85 %
	KNN	93 %	82 %
	DT	92 %	80 %
R2l	SVM	92 %	41 %
	MLP	93 %	46%
	KNN	92 %	41 %
	DT	93 %	49 %
Probe	SVM	93 %	71 %
	MLP	95 %	76 %
	KNN	95 %	70 %
	DT	94 %	82 %
U2R	SVM	93 %	73 %
	MLP	96 %	60 %
	KNN	95 %	64 %
	DT	96 %	49s %

Tableau 3.5 : Résultats des 16 sous-modèles préliminaires.

A partir de ces résultats, nous pouvons facilement déduire que le meilleur algorithme de classification pour détecter les attaques de la classe DOS est le MLP, le DT et le MLP pour la classe R2L, le KNN et le MLP pour la classe Probe et DT et MLP pour la classe U2R.

b) Choix des sous-modèles de chaque niveau et résultats de test

A partir des résultats des 16 sous-modèles préliminaires précédents, nous avons choisi le sous-modèle MLP pour le premier niveau (classe DOS) de notre modèle hybride, le sous-modèle KNN pour le deuxième niveau (classe Probe), le

sous modèle SVM pour le troisième niveau (classe R2L) et le sous-modèle DT pour le dernier niveau (classe U2R). Notre modèle hybride sera donc schématisé comme suite (figure 2).

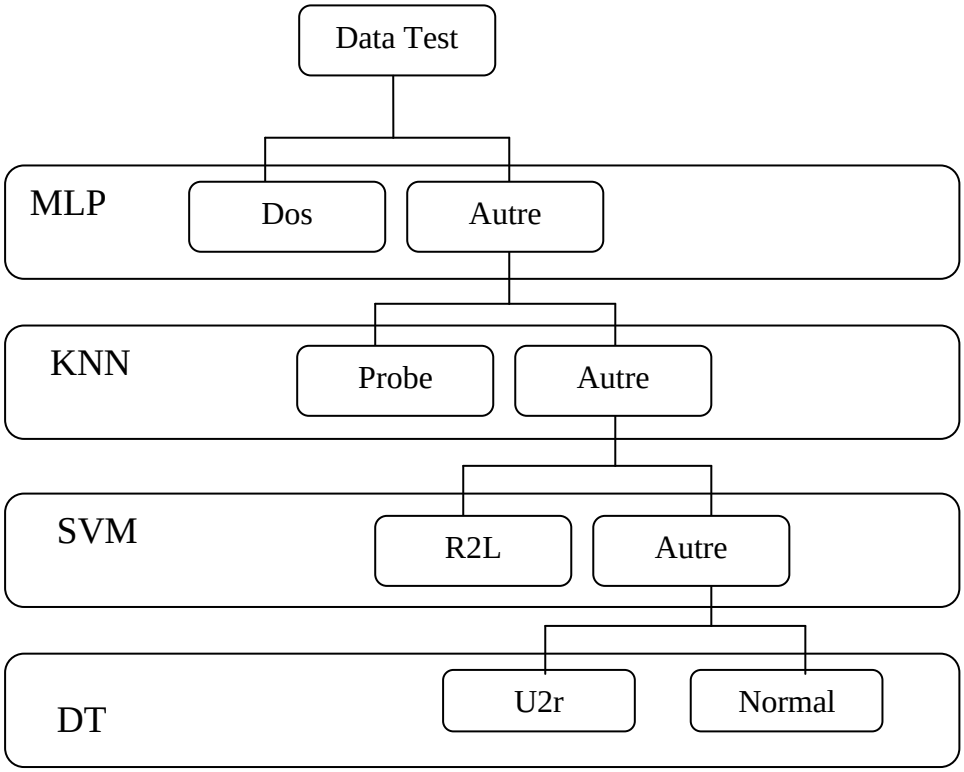

Figure 3.2 : Le modèle hybride proposé

c) Résultats et comparaison

Afin de montrer l'efficacité du modèle hybride proposé, nous avons implémenté et testé un autre modèle simple basé sur les réseaux de neurones MLP dans le but de comparer les résultats des deux modèles. Les résultats de test de ces deux modèles sont montrés dans les tableaux et les graphes suivants :

- Résultats du modèle MLP simple :

		Classe prédite				
		Normal	**Attaques**			
Classe réelle	**Normal**	9177	136	254	129	15
	Attaques	1718	5351	166	223	0
		398	167	1747	108	1
		936	2	180	1567	69
		154	0	6	7	33

Tableau 3.6 : Matrice de confusion du modèle MLP simple.

	Taux de réussite	Précision	Rappel	F1_Score
Modèle simple (MLP)	83 %	94.74 %	75.02 %	83.7 %

Tableau 3.7 : Performances du modèle MLP simple.

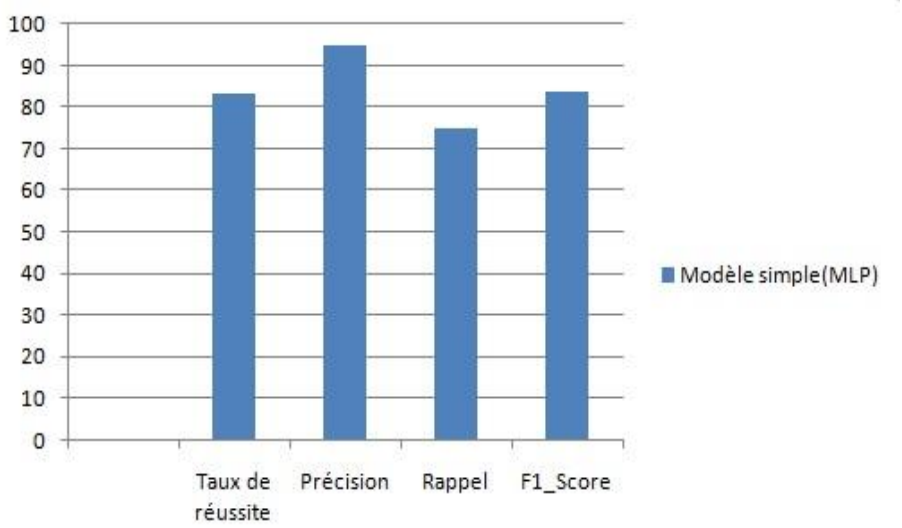

Figure 3.3 : Performances du modèle MLP simple.

- Modèle hybride :

		Classe prédites				
		Normal	**Attaques**			
Classe réels	**Normal**	9275	69	168	186	13
	Attaques	982	6228	130	118	0
		454	166	1787	14	0
		926	219	44	1558	7
		78	2	3	43	74

Tableau 3.8 : Matrice de confusion de notre modèle hybride.

	Taux de réussite	Précision	Rappel	F1_Score
Modèle hybride	87.24 %	95.97 %	80.99 %	87.85 %

Tableau 3.9 : Performances de notre modèle hybride.

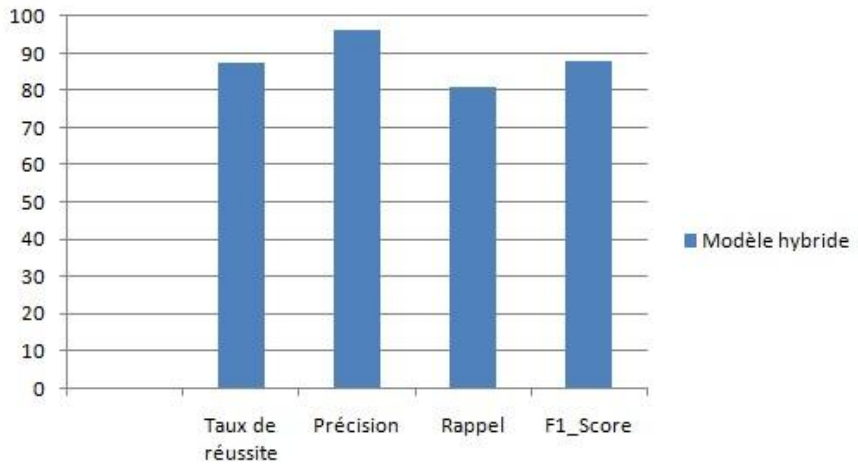

Figure 3.4 : Performances de notre modèle hybride.

Les résultats présentés ci-dessus montrent clairement l'efficacité de notre modèle hybride de détection d'intrusions qui donne des résultats dépassant largement ceux donnés par le modèle MLP simple, notamment en terme de taux de réussite. Il détecte les intrusions avec un taux de réussite (précision) arrivant jusqu'à 87 % (96 %) contre 83% (95%) pour le modèle MLP simple.

Conclusion

Dans ce dernier chapitre, nous avons abordé la partie implémentation, génération et test de notre modèle hybride multi-niveau de détection d'intrusions, où nous avons présenté l'ensemble de données NSL-KDD qui a été utilisé pour l'apprentissage et le test du modèle, les différentes opérations de prétraitement effectuées, la démarche suivie pour réaliser notre modèle, ainsi que les différents tests réalisés pour évaluer le modèle généré. Les résultats obtenus montrent clairement l'efficacité du modèle proposé ainsi que l'importance de la combinaison de plusieurs sous-modèles pour générer un modèle plus performant.

CONCLUSION GENERALE

Conclusion générale

Le nombre des attaques informatiques a considérablement augmenté au cours des dernières années et elles représentent désormais un risque réel pour les systèmes d'information, les logiciels et les infrastructures réseau des entreprises. Cela nous a conduit dans ce mémoire au développement d'un modèle hybride du système de détection d'intrusion comme moyen d'identifier les attaques dans le but de minimiser leurs dommages. En d'autres termes, la découverte régulière et continue de ces attaques peut, le cas échéant, contribuer à les réduire en mettant en place les mesures de protection nécessaires.

En raison de la capacité des machines à vecteurs de support, des réseaux de neurones artificiels, des algorithmes de classification K plus proches voisins et des arbres de décision, nous nous sommes appuyés sur ces quatre modèles pour créer notre modèle hybride multi niveaux.

Pour réaliser notre modèle, nous avons utilisé l'ensemble de données du benchmark NSL-KDD comme source de données, et nous avons créé des modèles de classification binaire (deux classes : normale et attaque) pour catégoriser les connexions TCP/IP en deux catégories : attaque ou normale à l'aide des machines à vecteurs de support (SVM), du perceptron multicouches (MLP), des K-plus proches voisins (KNN), et des arbres de décision (DT), qui sont efficaces dans la classification des données. Afin d'obtenir des bons résultats, nous avons effectué plusieurs expériences afin de choisir les meilleures méthodes de classification pour notre modèle hybride multi niveaux. Nous avons aussi utilisé dans notre application une méthode de balancement des données et une méthode de sélection des meilleurs attributs pour améliorer encore plus les résultats de classification de notre modèle.

En conclusion, la majorité des objectifs identifiés dans ce travail ont été atteints, mais il reste encore quelques opportunités et améliorations qui peuvent être apportées à l'avenir, telles que la génération de notre modèle hybride en utilisant les données de benchmarks récents comme celles de UNSW-NB15 et de

CICIDS2017 afin de tester la capacité de ce modèle à détecter les attaques récentes qui ne figurent pas dans la base NSL-KDD.

ANNEXE

Les attributs de l'ensemble de données NSL KDD : Les détails des attributs sont répertoriés dans le tableau suivant : [28]

N°	Nom de l'attribut	Description
1	duration	La durée de connexion
2	protocol_type	Protocole utilisé dans la connexion (tcp, udp, icmp)
3	service	Service réseau de destination,(http,telnet, ftp_data, etc.)
4	flag	Statut de la connexion –Normal ou Erreur (SF, REJ, S0, S1, etc.)
5	src_bytes	Nombre d'octets de donnée transférés de la source à la destination (491, etc.)
6	dst_bytes	Nombre d'octets de données transférés de destination à la source (0, etc.)
7	land	**Si** l'adresse IP de source et destination et le nombre de port sont les mêmes **alors**, *land*=1 **sinon** *land*=0
8	wrong_fragment	Nombre total de fragments erronés dans cette connexion
9	urgent	Nombre de paquets urgents
10	Hot	Nombre d'indicateurs « Hot »
11	num_failed_logins	Nombre de tentatives de connexion échouées
12	logged_in	**Si** connecté avec succès **alors** *logged_in*=1 **sinon** *logged_in*=0
13	num_compromised	Nombre de conditions compromises
14	root_shell	1 si le root shell est obtenu, 0autrement
15	su_attempted	1 **si** la commande "su root" a été tentée ou utilisée, **sinon** 0
16	num_root	Nombre d‟accès " root " ou nombre d‟opérations effectuées comme racine dans la connexion
17	num_file_creations	Nombre d'opérations de création de fichiers
18	num_shells	Nombre d'invites du shell
19	num_access_files	Nombre d'opérations sur les fichiers de contrôle d'accès

20	num_outbound_cmds	Nombre de commandes sortantes dans une session FTP
21	is_host_login	1 **si** la connexion appartient à la liste du « hot » (root ou admin);**sinon** 0
22	is_guest_login	1 **si** le login est un login «guest »;**sinon** 0
23	count	Nombre de connexions vers le même hôte de destination que la connexion en cours dans les deux dernières secondes
24	srv_count	Nombre de connexions vers le même service (N° Port) que la connexion en cours dans les deux dernières secondes
25	serror_rate	Le pourcentage de connexions qui ont activé le *flag* s0, s1, s2 ou s3,parmi les connexions agrégées dans *count*
26	srv_serror_rate	Le pourcentage de connexions qui ont activé le *flag* s0, s1, s2 ou s3,parmi les connexions agrégées dans *srv_count*
27	rerror_rate	Le pourcentage de connexions qui ont activé le *flag* REJ, parmi les connexions agrégées dans *count*
28	srv_rerror_rate	Le pourcentage de connexions qui ont activé le *flag* REJ, parmi les connexions agrégées dans srv_*count*
29	same_srv_rate	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans *count*
30	diff_srv_rate	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans *count*
31	srv_diff_host_rate	Le pourcentage de connexions qui sont à différentes machines de destination, parmi les connexions agrégées dans srv_*count*
32	dst_host_count	Nombre de connexions ayant la même adresse IP de l'hôte de destination
33	dst_host_srv_count	Nombre de connexions ayant le même numéro de port
34	dst_host_same_srv_rate	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans *dst_host_count*
35	dst_host_diff_srv_rate	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans *dst_host_count*
36	dst_host_same_src_port_rate	Le pourcentage de connexions qui sont au même port de source, parmi les connexions agrégées dans*dst_host_srv_count*
37	dst_host_srv_diff_host_rate	Le pourcentage de connexions qui sont à différentes machines de destination, parmi les connexions agrégées dans *dst_host_srv_count*
38	dst_host_serror_rate	Le pourcentage de connexions qui ont activé le *flag* s0, s1, s2 ou s3,parmi les connexions agrégées dans *dst_host_count*

39	dst_host_srv_serror_rate	Le pourcentage de connexions quiont activé le *flag* s0, s1, s2 ou s3, parmi les connexions agrégées dans *dst_host_srv_count*
40	dst_host_rerror_rate	Le pourcentage de connexions qui ont activé le *flag* REJ, parmi les connexions agrégées dans *dst_host_count*
41	dst_host_srv_rerror_rate	Le pourcentage de connexions qui ont activé le *flag* REJ, parmi les connexions agrégées dans *dst_host_srv_count*

Tableau (1) : Les 41 attributs de la base NSL-KDD

Bibliographie

[1] S.Natkin , « Les protocoles de sécurité d'Internet », Dunod science sup, 2002.

[2] A. Viardin, « Un petit guide pour la sécurité », Novembre 2003.

[3] SecuriteInfo, « Le grand livre de la sécurité informatique », Editions du 6 nouvembre 2006

[4] O. Aleph, article écrit dans le magazine Phrack.

[5] J.O.Gerphagnon, M. P. de Albuquerque, and M. P. de Albuquerque, « Attaques informatique » CBPFNT007/00, Centre brésilien de recherche physique, Rio de Janeiro – RJ – Brazil, 2004.

[6] M. S. Hoque, M. Mukit, M. Bikas, A. Naser, et al., « An implementation of intrusion detection system using genetic algorithm » arXiv preprint arXiv :1204.1336, 2012.

[7] S. Paliwal and R. Gupta, « Denialofservice, probing & remote to user (r2l) attack detection using genetic algorithm» *International Journal of Computer Applications*, vol. 60, no. 19, pp. 57–62, 2012.

[8] L. Poinsot, «Introduction à la sécurité informatique», support de cours, Université Paris 13.

[9] R. Philippe, « Virus, vers, chevaux de Troie… mieux connaître les codes malveillants »,2010

[10] W. Stallings, *Network security essentials : applications and standards*. Pearson, 2016.

[11] K. Salah, K. Sattar, Z. Baig, M. Sqalli, and P. Calyam, « Resiliency of opensource firewalls against remote discovery of lastmatching rules » in *Proceedings of the 2nd International Conference on Security of Information and Networks*, SIN '09, (New York, NY, USA), p. 186–192, Association for Computing Machinery, 2009.

[12] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, « Evaluating computer intrusion detection systems : A survey of common practices » ACM Comput. Surv., vol. 48, Sept. 2015.

[13] P. Biondi, « Architecture expérimentale pour la détection d'intrusions dans un système informatique » Article de recherche,(AvrilSptembre 2001), 2001.

[14] R. Graham, « Faq : Network intrusion detection systems, » http ://www. robertgraham.com/pubs/networkintrusiondetection.html, 2000.

[15] Revue scientifique, « Machine Learning », « Editorial Board of the Kluwer Journal, Machine Learning: Resignation Letter », *SIGIR Forum*, vol. 35, n^{o} 2, 2001

[16] https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support

[17] C. Cortes, V. Vapnik. Support-Vector Networks. Machine Leaning, 20, pp. 273 – 297, 1995.

[18] M. Cheriet, N. Kharma, C. lin Liu, and C. Suen. Character Systems, « A Guide for Students and Practitioners.» Wiley- Interscience,2007.

[19] W.S. McCulloch et Pitts W., « A Logical Calculus of the Ideas Immanent in Nervous Activity ». Bulletin of Mathematical Biophysics Vol. 5, pp. 115-133Proc. Of the 5 the ACM Conf. on Computational Learning Theory (COLT'92), pp. 144- 152, Pittsburgh, PA, USA, July, 1992.

[20] B. Djamel, Doctorant en droit et Responsable du développement de Juri Predis, article « Démystifier le Machine Learning, Partie 2 : les Réseaux de Neurones artificiels **».**

[21] F. Marin ,Cours « Arbres de décision », Département Informatique, Conservatoire National des Arts & Métiers, Paris, France.

[22] R. Rakotomalala, L. ERIC, Université Lumière Lyon 2

[23] S. Stolfo, W. Fan, W. Lee, *et al.*, « Kddcup99 task description, » *http //KDD. ics. uci. edu/databases/kddcup99/task.html*, 1999.

[24] Tavallaee, E. Bagheri, W. Lu et A. Ghorbani, « Une analyse détaillée de l'ensemble de données KDD CUP 99 », présenté au deuxième symposium IEEE sur l'intelligence informatique pour les applications de sécurité et de défense (CISDA), 2009.

[25] L.Dhanabal& Dr. S.P. Shantharajah, «A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms», International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 6, June 2015.

[26] L. Maxime, O. Alexis, Z. Djamal « Automatisation du processus d'entraînement d'un ensemble d'algorithmes de machine learning optimisés pour la détection d'intrusion », Institut LIST, CEA, F-91120, Palaiseau, France, Université Paris-Saclay, France, Institut Télécom Télécom SudParis, Évry, France.

[27] C. Nitesh, Al, technique was described in their 2002 paper named for the technique titled «SMOTE: Synthetic Minority Over-sampling Technique.». Page 47, Imbalanced Learning: Foundations, Algorithms, and Applications, 2013. Page 45, Imbalanced Learning: Foundations, Algorithms, and Applications, 2013

[28] L. Dhanabal and S. Shantharajah, «A study on nslkdd dataset for intrusion detection system based on classification algorithms,» International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 446–452, 2015.

Résumé

Le grand développement technologique, l'accessibilité au réseau internet à grande échelle et l'augmentation des moyens de stockage et d'échange d'informations ont contribué à l'évolution remarquable des cyber-attaques lors de ces dernières années, ce qui rend la sécurisation de toutes ces informations très important. Cette augmentation considérable des attaques a motivé les chercheurs et les spécialistes du domaine de la sécurité informatique à élaborer et à développer des nouveaux outils pour la détection des attaques réseaux, notamment celles qui sont difficilement découvertes par les outils traditionnels, en se basant sur les techniques de l'intelligence artificielle.

Dans ce travail nous avons réalisé un modèle hybride multi niveaux de détection d'intrusions basé sur les algorithmes de classification : machines à vecteurs de support, les réseaux de neurones (perceptron multicouches), K-plus proches voisins, et des arbres de décision, en utilisant l'ensemble de données NSL-KDD pour générer et évaluer ce modèle.

Abstract

The great technological development, the accessibility to the large-scale Internet network and the increase in the means of storing and exchanging information have contributed to a large number of cyber-attacks, which makes the fact of securing these information very important. This increase in attacks makes researchers and specialists in the field of computer security busy developing new tools for detecting attacks not discovered by traditional means based on artificial intelligence techniques.

In this work we realized a multi-level hybrid model of intrusion detection based on classification algorithms: support vector machines, neural networks (multilayer perceptron), K-nearest neighbors, and decision trees, using the NSL-KDD dataset to generate and evaluate this model.