Table des matières

Liste des figures iii
Liste des tableaux iv
Liste des abréviations, sigles, acronymes v
Remerciements vi
Introduction 1
Chapitre 1 Sécurité des systèmes d'information 3
1.1. Notions de sécurité 3
1.1.1. Définition et mise en œuvre d'une politique de sécurité 3
1.1.2. Les objectifs d'une politique de sécurité 3
1.1.3. Les outils et technologies utilisés 4
1.2. Attaques : les phases et types d'attaques 7
1.2.1. Les différentes phases d'une attaque 8
1.2.2. Les types d'attaques 8
Chapitre 2 IDS : Concept et classification 13
2.1. Modèles d'architecture des réseaux et encapsulation des données 13
2.1.1. Modèles d'architecture des réseaux 13
2.1.2. Encapsulation des données 16
2.2. Architecture classique d'un IDS 16
2.2.1. Le capteur 17
2.2.2. L'analyseur 18
2.2.3. Le manager 18
2.3. Principe de fonctionnement d'un IDS 18
2.3.1. Méthodes de détection des IDS 18
2.3.2. Les différents types de réponses aux attaques 19
2.4. Classification des IDS 19
2.4.1. IDS basé sur l'hôte (HIDS) 20
2.4.2. IDS basé sur le réseau (NIDS) 20
2.4.3. IDS hybride 20
2.5. Limites et inconvénients d'un IDS 21
2.5.1. La détection d'anomalies 21
2.5.2. Pollution/surcharge 21
2.5.3. Consommation de ressources 21
2.5.4. Perte de paquets 21
2.5.5. Vulnérabilité aux dénis de services 21
2.5.6. Contournement/évasion 21

2.5.7. L'activation du mode promiscuous 22
2.5.8. La base de données de signature 22
Chapitre 3 Intelligence artificielle - Apprentissage automatique - Apprentissage profond 23
3.1. C'est quoi l'IA 23
3.2. Apprentissage automatique 23
3.2.1. Qu'est-ce que l'AA 23
3.2.2. Pourquoi utiliser l'AA 24
3.2.3. Qu'est-ce qu'une donnée 25
3.2.4. Types d'AA 26
3.2.5. Problèmes rencontrés dans un algorithme d'AA 28
3.3. Apprentissage profond 29
3.3.1. Réseau de neurones artificiels 29
3.3.2. Réseaux de neurones à convolution ou convolutifs 31
3.3.3. Réseaux de neurones récurrents 32
3.3.4. Autoencodeurs 32
Chapitre 4 Apprentissage automatique et IDS 36
4.1. Les métriques de performances 36
4.2. Survol de la littérature 37
4.3. Les jeux de données ou datasets 38
Chapitre 5 Développement d'un NIDS avec l'AA 40
5.1. Outils 40
5.1.1. Le langage Python 40
5.1.2. Le gestionnaire de librairies Anaconda 40
5.1.3. Framework d'intelligence artificielle 41
5.2. Méthodologie 44
5.2.1. Statistiques du jeu de données 44
5.2.2. Pré-traitement des données 44
5.2.3. Préparation des données d'entrainement et de validation 45
5.2.4. Création et entrainement de notre modèle 45
5.3. Résultats 46
Conclusion 48
Bibliographie 49
Annexe Codes utilisés 51

Liste des figures

Figure 1 Notions de sécurité 3
Figure 2 Objectifs d'une politique de securite 4
Figure 3 Outil pare-feu 5
Figure 4 Mise en place d'un reseau privé virtuel 5
Figure 5 Mise en place d'un pot de miel 6
Figure 6 Principes de la crytographie 6
Figure 7 IPS/IDS 7
Figure 8 Les differentes phases d'une attaque 8
Figure 9 Attaque par force brute 9
Figure 10 Attaque Botnet 9
Figure 11 Attaque DoS/DDoS 10
Figure 12 Attaque par sniffing 10
Figure 13 Attque par spoofing 11
Figure 14 Attaque Heartbleed 11
Figure 15 Attque SQl Injection 12
Figure 16 Modeles d'architecture réseaux 14
Figure 17 Protocoles TCP/IP 14
Figure 18 Encapsulation des données 16
Figure 19 Architecture classique d'un IDS 17
Figure 20 Classification des IDS 19
Figure 21 Architecture d'un NIDS 20
Figure 22 Etapes d'un algorithme 25
Figure 23 Exemple d'utilisation du K-moyennes 27
Figure 24 Exemple d'apprentissage automatique par renforcement 27
Figure 25 Performance d'un modèle par rapport au nombre de données 28
Figure 26 Architecture d'un RNA 29
Figure 27 Etapes d'un RNA 30
Figure 28 Descente de de gradient 31
Figure 29 Exemples de fonctions d'activation 31
Figure 30 Fonctionnement d'un RNR 32
Figure 31 Architecture d'un autoencodeur 33
Figure 32 Autocencodeur debruiteur 34
Figure 33 Autoencodeur varationnel 34
Figure 34 Autoencodeur contradictoire 35
Figure 35 Comparaison du taux de fausses alarmes des différents algorithmes 37
Figure 36 Comparaison du taux de détection des différents algorithmes 38
Figure 37 Langage python 40
Figure 38 Gestionnaire de librairies Anaconda 40
Figure 39 Framework Pytorch 41
Figure 40 Framework TensorFlow 42
Figure 41 Framework Keras 42
Figure 42 Librairie Pandas 43
Figure 43 Vue globale du modele 45
Figure 44 Evolution de l'erreur de reconstruction du modèle 46

Figure 45 Taux de détection par rapport aux différentes attaques 47

Liste des tableaux

Tableau 1 Exemples de cas d’utilisation de l’apprentissage automatique 24
Tableau 2 Statistiques du jeu de donnees CSE-CIC-IDS2018..... 44

Liste des abréviations, sigles, acronymes

AE: Auto Encoder

ANN: Artificial Neural Network

CPU: Central Processor Unit

DDos: Distributed Deny Of Service

DL: Deep Learning

DoS: Deny Of Service

GHz: Giga Hertz

HIDS: Host Intrusion Detection System

HTTP: HyperText Transfert Protocol

ICMP: Internet Control Message Protocol

IDS: Intrusion Detection System

LSTM: Long Short-Term Memory

ML: Machine Learning

NIDS: Network Intrusion Detection System

RAM: Random Access Memory

ReLU: Rectified Linear Unit

RNC: Réseau de neurones convolutifs

RNR: Réseau de neurones récurrents

RPV : Réseau Privé Virtuel

SNMP: Simple Network Management Protocol

VAE: Variational Auto Encoder

VN: Vrai Négatif

VP: Vrai Positif

Remerciements

Je tiens à exprimer toute ma reconnaissance à mon encadreur de mémoire, Dr Kande DIABY, enseignant au niveau de l'Université Cheikh Anta DIOP, Faculté des Sciences et Techniques pour sa patience, sa disponibilité et surtout ses précieux conseils, qui ont contribué à alimenter ma réflexion et améliorer ce travail.

J'adresse également mes sincères remerciements à Dr Ousmane NDIAYE mais aussi à tous les professeurs du master de TDSI (Transmission de données et Sécurité de l'information) de l'Université Cheikh Anta DIOP, qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et m'ont apporté les connaissances requises afin de mener à bien ce travail.

Je tiens à exprimer toute ma reconnaissance au Pr Omar DIANKHA pour son soutien et ses encouragements ainsi qu'à tous les membres de l'équipe pédagogique du master TDSI.

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Je remercie aussi mes camarades de promotion pour leur aide durant toutes ces années passées ensemble.

Enfin, je remercie ma famille qui sans leur aide, leurs conseils et leurs encouragements, ce travail n'aurait vu le jour.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Introduction

Les progrès rapides des technologies de communication ont abouti à une énorme augmentation de la taille des réseaux informatiques et des données correspondantes. En conséquence, de nouvelles attaques ont vu le jour et se sont trouvées être de réels défis pour les administrateurs réseaux. En effet, dès lors qu'une entreprise se connecte à l'Internet, son réseau risque d'être la cible d'intrusions. Ces intrusions consistent généralement, pour le hacker, à exploiter les failles d'un système d'exploitation, d'un routeur ou d'une application ou bien encore d'une mauvaise gestion des comptes informatiques, afin d'obtenir l'accès à un équipement du site avec des niveaux de privilèges plus ou moins importants. Si, la plupart du temps, les intrusions sont perçues comme un jeu ou une compétition entre hackers et n'ont pas d'impact financier pour l'entreprise, elles peuvent dans certains cas conduire au vol, à la destruction ou à la modification de données de l'entreprise. La menace des intrusions est bien réelle surtout depuis l'apparition des scripts kiddies, c'est-à-dire des personnes sans compétences informatiques particulières, mais qui téléchargent des outils logiciels d'intrusion sur Internet et s'amusent à les tester sur des sites d'entreprises. Aussi, lorsque l'entreprise ouvre son réseau sur Internet, il est fréquent d'observer que les employés l'utilisent de façon abusive à des fins personnelles plus ou moins bien intentionnées. Il peut s'agir d'une simple consultation des réseaux sociaux, de téléchargements de fichiers volumineux par exemple de types audio ou vidéo, etc., mais aussi de communiquer à l'extérieur des informations sensibles de l'entreprise dans un but de malveillance ou lucratif.

Afin de contrôler au mieux le trafic échangé avec le réseau public, les entreprises ont de plus en plus recours à diverses méthodes et outils tels les pare- feux, les logiciels antivirus afin de protéger le réseau privé de l'entreprise, la vie privée de l'utilisateur et les données sensibles. Un système de détection d'intrusion (abréviation SDI, intrusion détection system en anglais) représente une autre méthode de protection parmi ces derniers. Un IDS se concentre sur le trafic réseau ou sur un environnement informatique particulier afin d'analyser et d'identifier des intrusions et signes liés à des activités suspectes.

Cependant celui-ci présente un certain nombre de limites et d'inconvénients. En effet, dû au grand nombre d'attaques réseau découvertes régulièrement, à la difficulté d'analyser rapidement ces attaques, il devient extrêmement difficile pour un IDS de réduire le nombre de fausses alarmes (activités considérées comme attaques alors qu'elles ne le sont pas) tout en augmentant la précision et l'efficacité du modèle de détection.

Aussi récemment, l'intelligence artificielle a suscité un réel engouement et a permis des avancées majeures, entre autres, dans des mécanismes de reconnaissance de formes et de détection d'anomalies, ce qui a encouragé ainsi les chercheurs à adopter l'apprentissage automatique pour la mise en place d'un système de détection d'intrusion. En effet, les algorithmes d'apprentissage automatique peuvent aider à traiter l'immense quantité de données générées par le réseau afin de détecter les attaques, pouvoir décider plus efficacement si une activité représente un comportement malveillant ou est indicatrice d'un comportement légitime mais aussi réduire de façon considérable le taux de fausses alarmes.

Ce mémoire portera sur le développement d'un IDS se basant sur l'apprentissage automatique. Pour ce faire, nous allons aborder quelques notions à savoir :

Qu'est-ce que la sécurité dans les systèmes d'information?

Qu'est-ce qu'un IDS? A quoi sert un IDS?

Qu'est-ce que l'apprentissage automatique?

Comment développer un IDS en se basant sur l'apprentissage automatique?

Chapitre 1 Sécurité des systèmes d'information

1.1. Notions de sécurité

1.1.1. Définition et mise en œuvre d'une politique de sécurité

Figure 1 Notions de sécurité (https://static4.depositphotos.com/1005404/308/i/950/depositphotos_3085251-stock-photo-laptop-and-shield.jpg)

Parce que les risques en matière de sécurité n'ont cessé d'augmenter ces dernières années, il devient primordial pour les entreprises de mettre en place une stratégie qui leur permettra d'être préparé en cas d'incidents et de réduire l'occurrence de ces éventuels incidents. Une politique de sécurité informatique est ainsi une stratégie visant à maximiser la sécurité informatique d'une organisation. Elle peut être matérialisée dans un document qui reprend l'ensemble des enjeux, objectifs, analyses, actions et procédures faisant parti de cette stratégie.
Cependant sa mise en œuvre est assez difficile, du fait de la diversité des aspects à considérer. Une politique de sécurité peut se définir par un certain nombre de caractéristiques : les niveaux où elle intervient, les objectifs de cette politique et enfin les outils utilisés pour assurer cette sécurité. Chaque aspect doit être pris en compte, de façon à atteindre les objectifs de sécurité désirés, en utilisant de façon coordonnée les différents outils à disposition.

1.1.2. Les objectifs d'une politique de sécurité

Figure 2 Objectifs d'une politique de sécurité(https://www.slideshare.net/NouriddinBENZEKRI/la-scurit-informatiques)

La sécurité informatique vise généralement cinq principaux objectifs à savoir :

- L'intégrité : consiste à déterminer si les données n'ont pas été altérées durant la communication de manière fortuite ou intentionnelle.
- La confidentialité : consiste à assurer que seules les personnes autorisées ont accès aux ressources échangées ;
- La disponibilité : permet de maintenir le bon fonctionnement du système d'information;
- La non-répudiation : permet de garantir qu'une transaction ne peut être niée;
- L'authentification : consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être.

Cela signifie que la sécurité doit être abordée dans un contexte global et notamment prendre en compte les aspects suivants :

- La sensibilisation des utilisateurs aux problèmes de sécurité
- La sécurité logique, c'est-à-dire la sécurité au niveau des données, notamment les données de l'entreprise, les applications ou encore les systèmes d'exploitation.
- La sécurité des télécommunications : technologies réseau, serveurs de l'entreprise, réseaux d'accès, etc.
- La sécurité physique, soit la sécurité au niveau des infrastructures matérielles : salles sécurisées, lieux ouverts au public, espaces communs de l'entreprise, postes de travail, etc.

1.1.3. Les outils et technologies utilisés

Afin d'assurer une bonne protection des données d'une organisation ou entreprise, différents outils sont disponibles. Ils sont en général utilisés ensemble, de façon à sécuriser les différentes failles existantes dans un système.

i. Le pare-feu

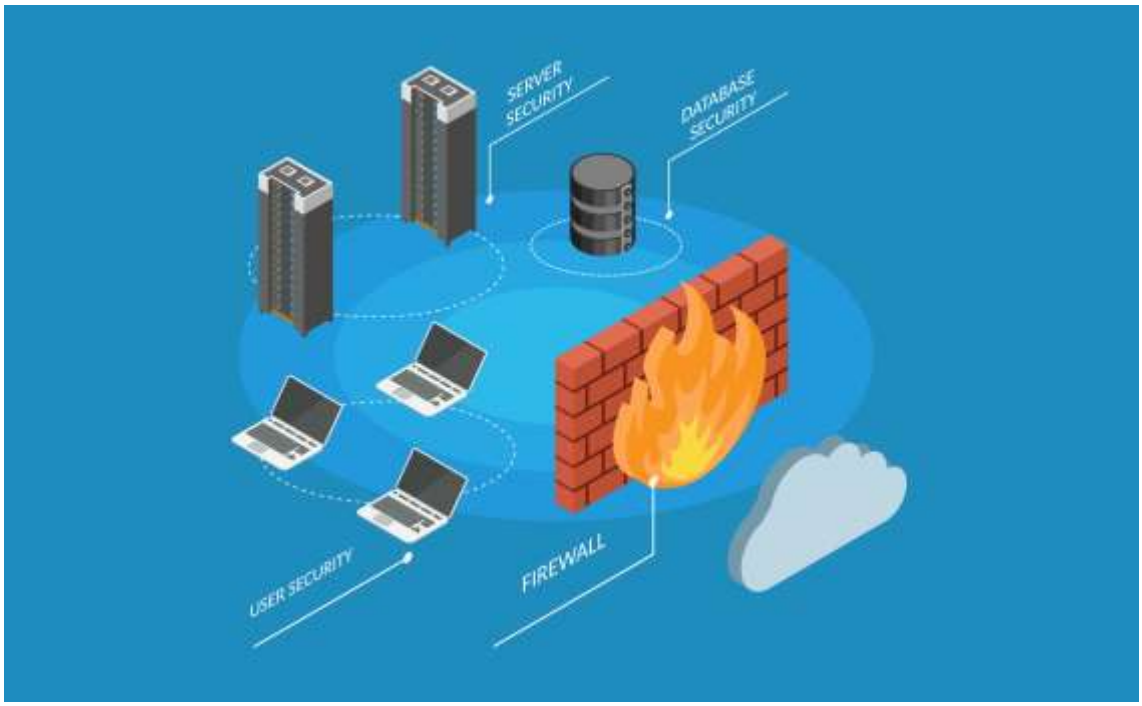


Figure 3 Mise en place d'un pare-fe (https://bluebearsit.com/decouvrir-le-role-dun-firewall/)

Un pare-feu (firewall en anglais) est un outil informatique (matériel et/ou logiciel) conçu pour protéger les données d'un réseau. Il peut consister à la protection d'un ordinateur personnel relié à Internet ou la protection d'un réseau d'entreprise. Il permet d'assurer la sécurité des informations d'un réseau en filtrant les entrées et en contrôlant les sorties selon des règles définies par l'administrateur réseaux.

ii. L'antivirus

Un antivirus est un logiciel informatique destiné à identifier et à supprimer des logiciels malveillants (malwares en anglais), également appelés virus, chevaux de Troie ou vers selon les formes. Les antivirus sont utilisés au niveau des différentes machines du réseau afin de vérifier si des malwares ont pu se propager.

iii. Réseau privé virtuel

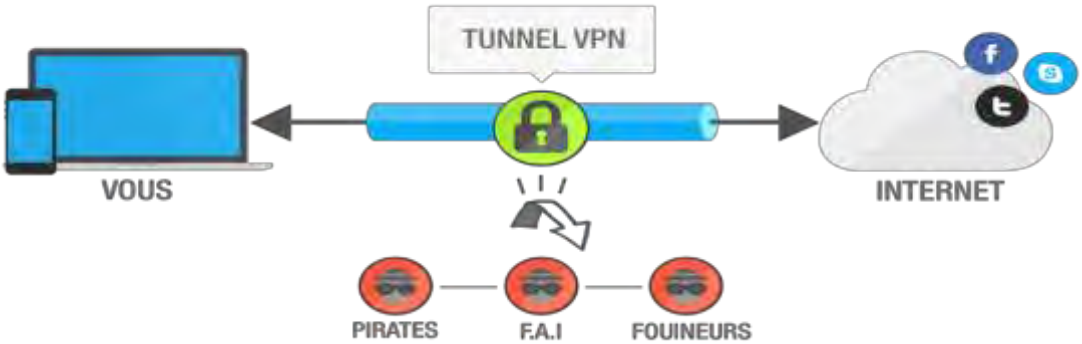


Figure 4 Mise en place d'un réseau privé virtuel (www.purevpn.fr/blog/vpn-tunnel)

Un réseau privé virtuel (Virtual Private Network en anglais, abrégé en VPN) est vu comme une extension des réseaux locaux et préserve la sécurité logique que l'on peut avoir à l'intérieur d'un réseau local. Il correspond en fait à une interconnexion de réseaux locaux via une technique de « tunnel ». La technique consiste à utiliser Internet comme support de transmission en utilisant un protocole de « tunnellisation », c'est-à-dire en encapsulant les données à transmettre de façon chiffrée.

iv. Pot de miel (honeypot)

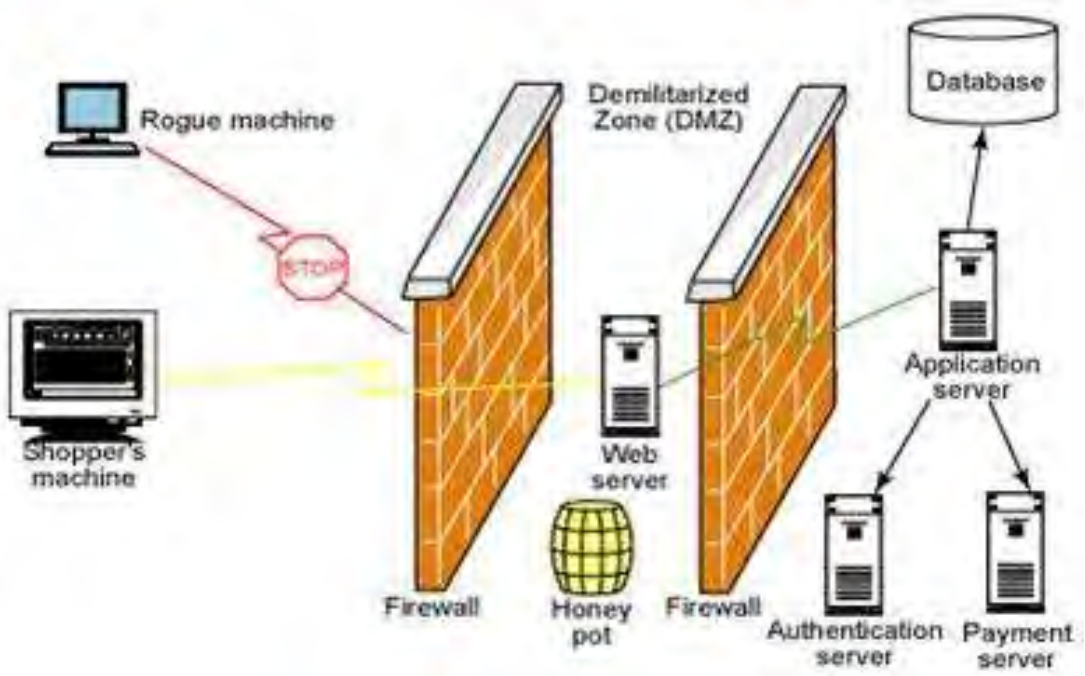


Figure 5 Mise en place d'un pot de miel (www.biyanicolleges.org/honeypot-technology)

Un pot de miel est un outil informatique (système, serveur, programme), volontairement exposé et vulnérable à une ou plusieurs failles connues, destiné à attirer et piéger les pirates, tout en permettant d'observer leur comportement en pleine action et d'enregistrer leurs méthodes d'action afin de mieux les étudier, les comprendre et les anticiper.

v. Cryptographie

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Figure 6 : Principes de la cryptographie (http://igm.univ-mlv.fr)

La Cryptographie est une science qui met en place des outils et techniques pour participer à la sécurisation des données et des infrastructures afin de construire des systèmes sûrs et sécurisés. Elle s'attache à protéger des messages en s'aidant souvent de secrets ou clés.

vi. Système de prévention d'intrusion/ système de détection d'intrusion

Figure 7 IPS / IDS (www.rokacom.com/ids-vs-ips)

Les IPS et IDS sont des outils ayant pour objectifs de détecter des activités malicieuses sur la cible qu'ils surveillent. Ils sont devenus en quelques années des produits indispensables pour les systèmes informatiques. Il convient avant tout de faire la distinction entre ces 2 types de produits :

- Un système de détection des intrusions (IDS) est un ensemble de composants logiciels et/ou matériels dont la fonction principale est de détecter et analyser toute tentative d'effraction volontaire dans un système d'informations

- Un système de prévention des intrusions (IPS) est un ensemble de composants logiciels et/ou matériels dont la fonction principale est d'empêcher toute activité suspecte détectée au sein d'un système.

1.2. Attaques : les phases et types d'attaques

Une attaque informatique représente le type d'actions susceptibles de nuire dans l'absolu à un système informatique. En termes de sécurité informatique les menaces peuvent être le résultat de diverses actions. Nombreux sont les hackers qui profitent de la vulnérabilité des équipements, du manque d'expérience des utilisateurs afin de s'infiltrer dans les réseaux et voler des informations sensibles. Ainsi des banques, des hôpitaux, des entreprises ont été la cible de plusieurs attaques informatiques. De plus en plus sophistiquées, ces dernières ne cessent d'évoluer et de se diversifier, impactant des millions d'entreprise dans le monde. Selon l'Observatoire sur les Systèmes d'Information, les Réseaux et les Inforoutes au Sénégal, le Sénégal est le cinquième pays africain le plus attaqué par les cybercriminels et le 71^{ième} au niveau mondial. Une cyber-attaque peut générer une crise majeure pouvant remettre en cause

la pérennité même de l'entreprise et avoir des conséquences aussi bien financières, juridiques que réputationnelles.
Dans cette section, nous allons étudier les différentes phases d'une attaque et répertorier quelques types d'attaques les plus fréquentes et montrer leurs fonctionnements.

1.2.1. Les différentes phases d'une attaque

La plupart des attaques, de la plus simple à la plus complexe fonctionne selon le schéma suivant :

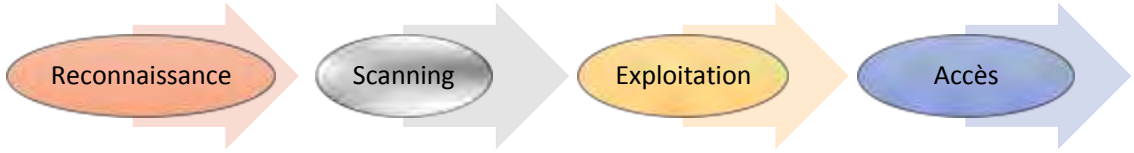


Figure 8 Les différentes phases d'une attaque

i. La reconnaissance

Cette étape est indispensable à toute attaque organisée. Elle permet de récolter le maximum d'informations sur la cible et peut se faire de manière passive (sans interaction avec la cible) ou active (interaction avec la cible).

ii. Le scanning

Elle constitue l'étape préattaque et permet de compléter les informations précédentes réunies sur la cible. Il est ainsi possible de connaitre les ports ouverts, les systèmes d'exploitation, logiciels et équipements utilisés.

iii. L'exploitation

Dans cette étape, le hacker tente à partir des informations recueillies d'exploiter les failles identifiées sur les éléments de la cible, que ce soit au niveau protocolaire, des services et applications ou des systèmes d'exploitation présents sur le réseau.

iv. L'accès

Il est temps pour l'attaquant de réaliser ce pourquoi il a franchi les précédentes étapes. Le but ultime est d'élever ses droits vers root (ou system) sur un système afin de pouvoir y faire tout ce qu'il souhaite comme l'inspection de la machine, récupération d'informations, installation de Backdoor, nettoyage des traces,…

1.2.2. Les types d'attaques

i. Attaque par force brute

L'attaque par force brute est une méthode qui consiste à trouver le mot de passe ou la clé cryptographique d'une personne afin de pouvoir accéder à un service en ligne, à des données personnelles ou un ordinateur. Son principe consiste à essayer toutes les combinaisons possibles jusqu'à trouver la bonne.
La méthode de détection la plus couramment préconisée tient compte des échecs de connexion répétés pour un même utilisateur durant un cours laps de temps.

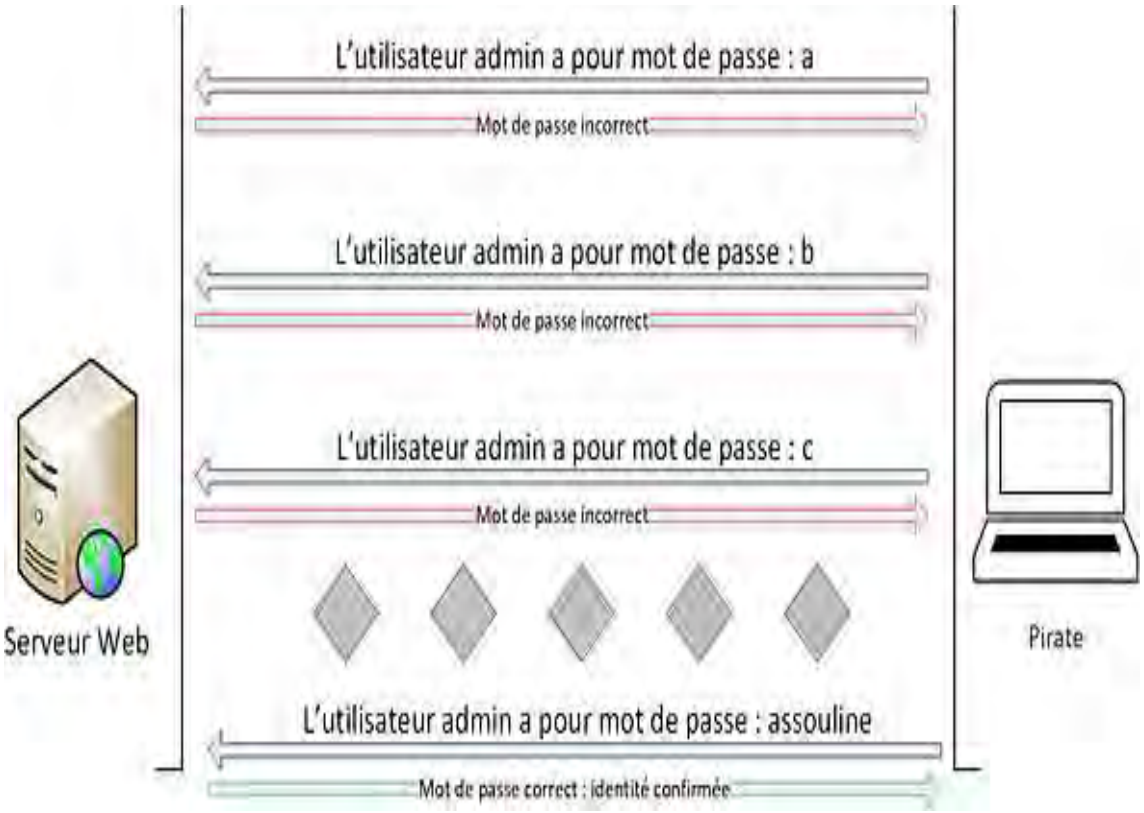


Figure 9 Attaque par force brute (www.editions-eni.fr)

ii. Botnet attack

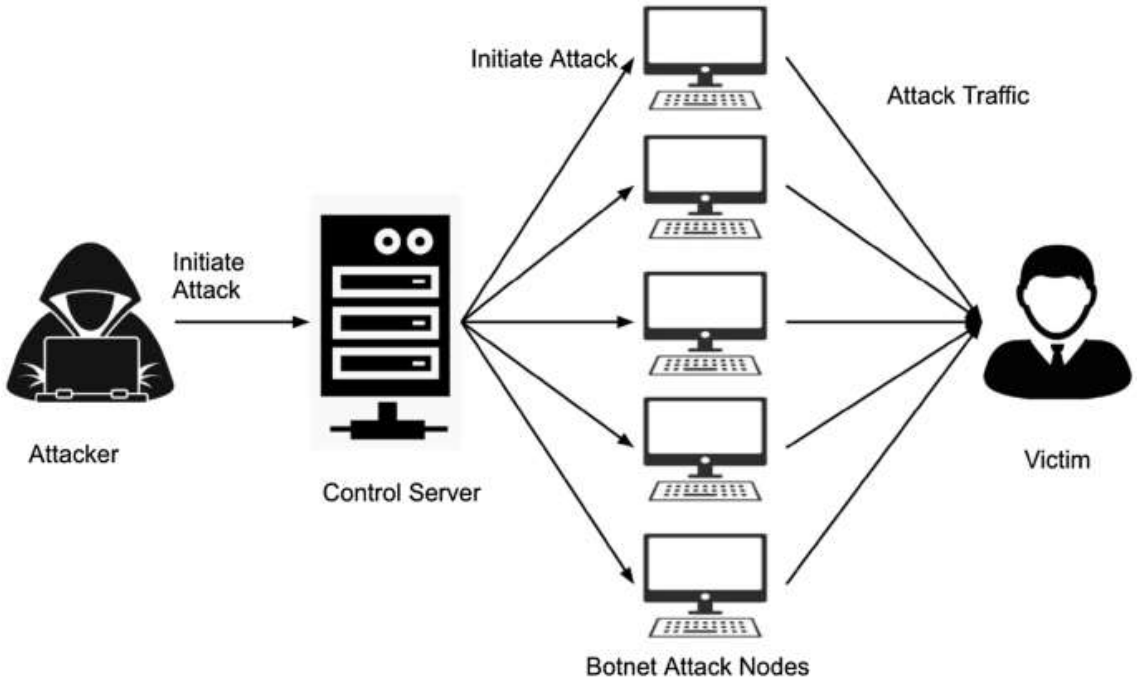


Figure 10 Attaque Botnet (link.springer.com/article/10.1007/)

« Botnet » est une contraction des termes « robot » et « network » (réseau). Les cybercriminels utilisent des chevaux de Troie spéciaux pour violer la sécurité des ordinateurs de différents utilisateurs, prendre le contrôle de chacun de ces ordinateurs infectés et les regrouper au sein d'un réseau de « bots » gérables à distance.

iii. DOS/DDOS attack

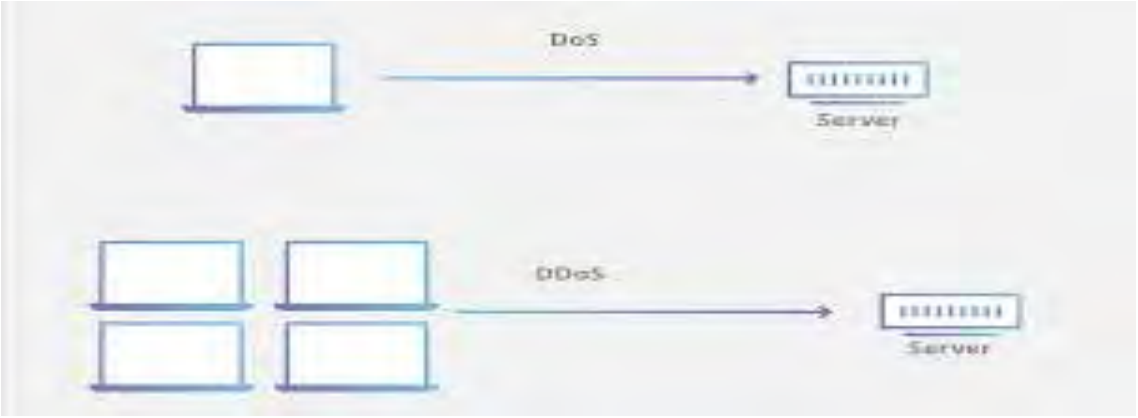


Figure 11 Attque DoS / DDoS (cloudfare.com)

Une attaque par déni de service est une attaque informatique ayant pour but de rendre indisponible un service ainsi d'empêcher les utilisateurs légitimes d'un service de l'utiliser. Une Dos ou DDos consiste ainsi à épuiser toutes les ressources à disposition d'un serveur, le plus souvent celles liées au réseau. Il est également possible de saturer le disque, la RAM ou encore les processeurs des machines visées.

iv. Sniffing attack

Figure 12 Attque par Sniffing (geycampus.com)

Le sniffing consiste à écouter le trafic sur un réseau à l'aide de drones appelés « sniffers » parfois à l'insu des utilisateurs et des administrateurs. Il correspond ainsi à un vol ou une interception de données en capturant le trafic réseau à l'aide d'un sniffer.

v. Le spoofing

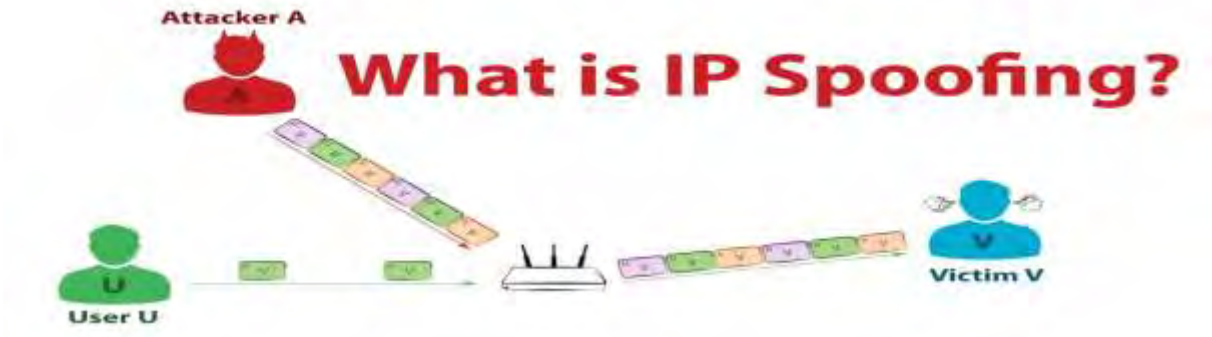


Figure 13 Attaque par Spoofing (https://actualiteinformatique.fr/cybersecurite/definition-spoofing))

L'usurpation d'identité se produit lorsqu'une personne prétend être une autre personne, un organisme ou une entreprise dans le but d'avoir accès au réseau, aux renseignements personnels de nature délicate, y compris les noms d'utilisateur et les mots de passe.

vi. Heartbleed attack

Figure 14 Attaque Heartbleed (www.darkreading.com)

Heartbleed est une vulnérabilité présente dans la bibliothèque de cryptographie open source OpenSSL depuis Mars 2012 et, qui permet à un « attaquant » de lire la mémoire d'un serveur ou d'un client pour récupérer, par exemple, les clés privées utilisées lors d'une communication avec le protocole Transport Layer Security (TLS). Elle concerne de nombreux services Internet.

vii. Injection SQL

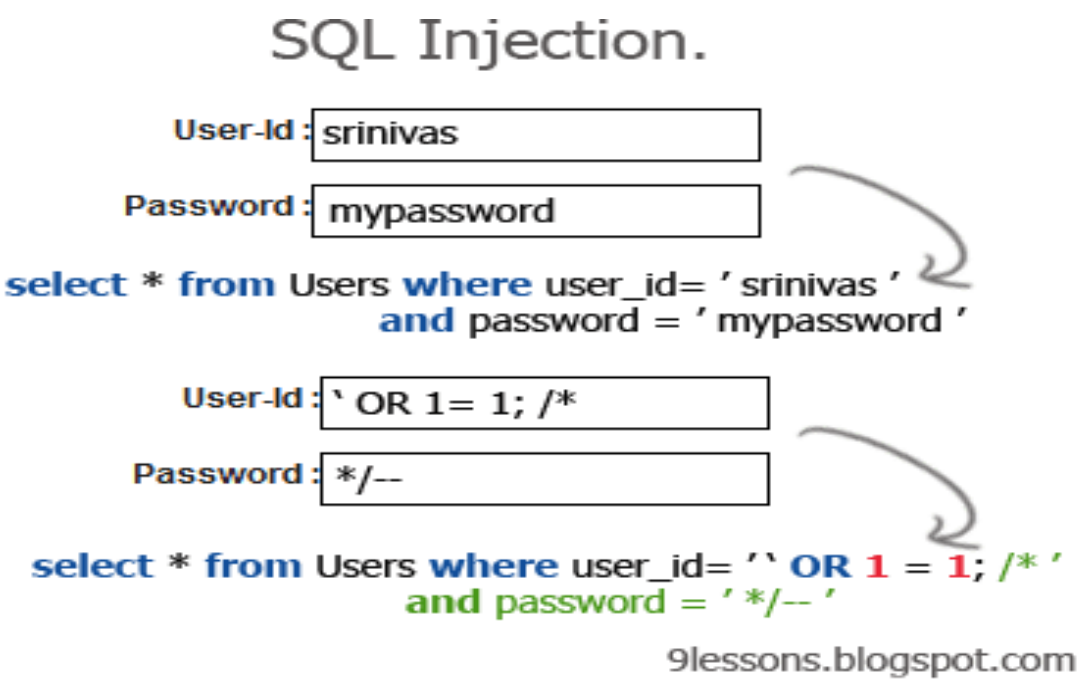


Figure 15 Attaque par SQL Injection (https://gnuit.wordpress.com/2010/06/12/securite-injection-sql/)

L'injection SQL est une technique d'injection de code utilisée pour modifier ou récupérer des données dans des bases de données SQL. En insérant des instructions SQL spécialisées dans un champ de saisie, un pirate est capable d'exécuter des commandes permettant de récupérer des données dans la base de données, de détruire des données sensibles ou d'adopter d'autres comportements de manipulation.

Un IDS permet de détecter et identifier ces menaces.

Chapitre 2 IDS : Concept et classification

Nous avons présenté au chapitre 1 un ensemble d'attaques possibles sur un réseau informatique et quelques différents mécanismes de protection contre ces attaques. Malgré leur grand intérêt, ils présentent des lacunes et failles dues à l'évolution des techniques utilisées par les hackers. Ainsi dans l'impossibilité de détecter la totalité des attaques, le système de détection d'intrusion tente de déceler les attaques passées inaperçues à travers les autres mécanismes de sécurité. Un IDS a pour fonction d'analyser, en temps réel ou différé, le trafic réseau et de détecter les différentes attaques lancées contre le réseau d'une organisation. En effet, IDS est la combinaison de deux mots « intrusion » et « détection system ». L'intrusion fait référence à un accès non autorisé aux informations contenues dans un ordinateur ou des systèmes de réseau afin de compromettre son intégrité, sa confidentialité ou sa disponibilité alors que le système de détection est un mécanisme de sécurité pour la détection d'une telle activité illégale. Ainsi, un IDS est un outil de sécurité (matériel ou logiciel) qui surveille en permanence le trafic hôte et réseau pour détecter tout comportement suspect qui viole la politique de sécurité. L'IDS générera ainsi des alertes sur les virus détectés ou comportements anormaux de l'hôte ou des systèmes réseau. Son objectif principal est d'identifier les incidents possibles, de consigner les informations les concernant dans des rapports.
Les IDS sont devenus un ajout essentiel à l'infrastructure de sécurité de presque toutes les organisations. Différentes méthodes peuvent être utilisées pour détecter les intrusions.
Afin de mieux présenter le concept d'IDS, nous allons étudier les modèles d'architecture réseau à savoir le modèle OSI et le modèle TCP/IP, voir comment sont encapsulées les données, définir le mode de fonctionnement d'un IDS, les différents types d'IDS et présenter les limites et inconvénients d'un IDS.

2.1. Modèles d'architecture des réseaux et encapsulation des données

2.1.1. Modèles d'architecture des réseaux

Un protocole est un ensemble de règles qui définit comment différents systèmes communiquent entre eux. Dans le cas de réseaux, les protocoles définissent comment les données sont transférées d'un système à un autre. Il est possible théoriquement d'utiliser un unique protocole qui prend les données d'une application informatique et les envoie à une application sur un autre ordinateur. Le problème avec cette approche est qu'elle est très rigide, car tout changement nécessite de modifier l'ensemble du protocole. Ainsi pour ajouter de la souplesse et de l'efficacité, des modèles en couches ont été définis. Chaque protocole va appartenir à une couche précise et chaque couche va avoir une fonction différente des autres. Le grand intérêt des modèles en couches réside dans la séparation des fonctions : on va pouvoir modifier des protocoles ou utiliser un protocole d'une couche plutôt qu'un autre sans affecter les autres. Un modèle en couches est ainsi une norme adoptée par plusieurs constructeurs en vue de faire communiquer l'ensemble des protocoles adoptés.
Aujourd'hui, dans le monde réseau, il existe deux modèles largement dominants : le modèle OSI qui définit 7 couches et le modèle TCP/IP (« modèle Internet ») qui définit 4 couches. Le modèle OSI est un modèle de réseau idéalisé, tandis que le modèle TCP/IP est une implémentation pratique.

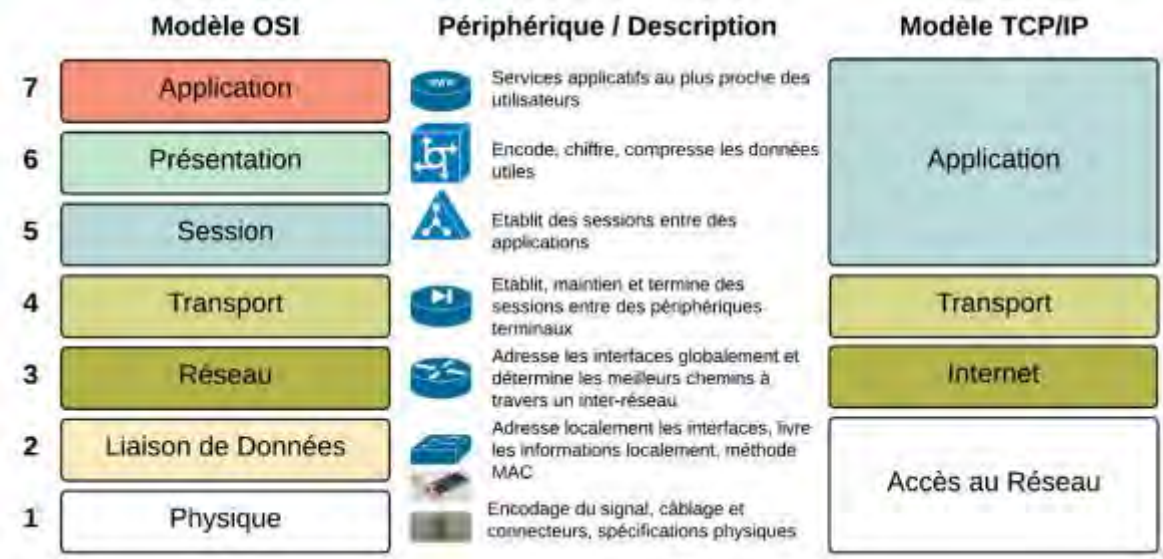


Figure 16 Modeles d'architecture des réseaux (cisco.goffinet.org)

Dans Figure 17 , nous pouvons aussi voir quelques protocoles et les couches associées à ces derniers.

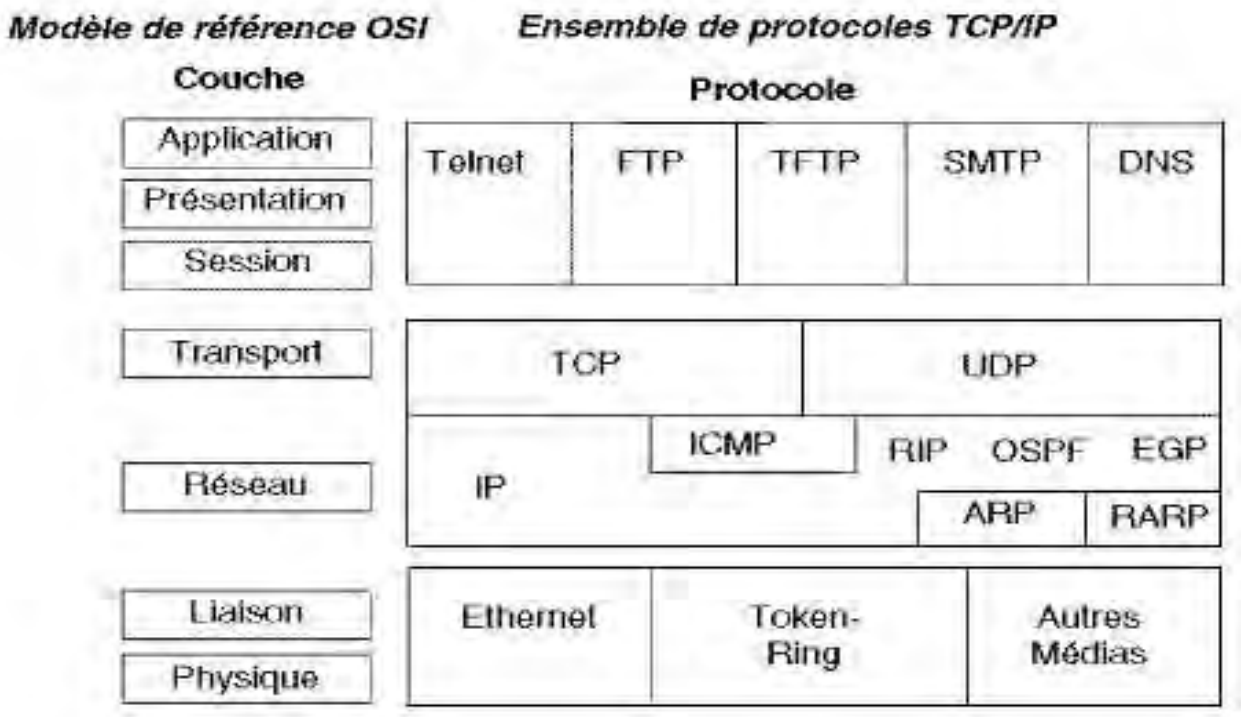


Figure 17 Protocoles TCP/IP (linux-france.org)

i. Modèle OSI

Le modèle OSI comporte sept couches, chaque couche avec ses caractéristiques et fonctions.

- La couche physique est chargée de la transmission effective des signaux entre les interlocuteurs. Son service est limité à l'émission et la réception d'un bit ou d'un train de bit continu.
- La couche liaison de données gère les communications entre 2 machines directement connectées entre elles, ou connectées à un équipement qui émule une connexion

directe (commutateur). Un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique.

- La couche réseau gère les communications entre machines (sous-réseaux, routage et adressage des paquets).
- La couche transport c'est l'une des couches les plus importantes, car elle gère les communications de bout en bout entre processus (programmes en cours d'exécution). Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.
- La couche session gère la synchronisation des échanges et les transactions, permet l'ouverture et la fermeture de session. Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue.
- La couche présentation est chargée du codage des données applicatives, précisément de la conversion entre données manipulées au niveau applicatif et chaînes d'octets effectivement transmises. Elle peut aussi reformater, crypter et compresser les données.
- La couche application est le point d'accès aux services réseaux, elle n'a pas de service propre spécifique.

ii. Modèle TCP/IP

TCP/IP est l'ensemble des protocoles utilisés pour le transfert des données sur internet. Elle est souvent appelée TCP/IP, d'après le nom de ses deux premiers protocoles : TCP (Transmission Control Protocol) et IP (Internet Protocol). La couche IP (Internet Protocol) s'occupe principalement de gérer l'adressage logique et d'assurer l'acheminement des paquets d'un nœud à l'autre. La couche TCP (Transmission Control Protocol), s'occupe de gérer les erreurs et de contrôler le flux en mettant en place des mécanismes de répétition de paquets et d'ajustage de fenêtre de réception (c'est-à-dire le nombre de paquets qu'il est possible de recevoir avant de les valider/invalider). Le modèle TCP/IP comporte quatre couches : la couche application, la couche transport, la couche Internet et la couche d'accès au réseau. Certaines couches du modèle TCP/IP portent le même nom que des couches du modèle OSI, mais elles comportent des fonctions différentes qu'on cite au-dessous.

- Couche application : Elle est située au sommet des couches du protocole TCP/IP. Elle contient des applications réseau permettant de communiquer grâce aux couches inférieures. La couche application gère les protocoles de haut niveau : représentation codage et contrôle du dialogue.
- Couche transport : Elle assure tout d'abord une communication de bout en bout en faisant abstraction des machines intermédiaires entre l'émetteur et le destinataire. La couche transport contient deux protocoles permettant à deux applications d'échanger des données ; dans le cas de TCP (Transmission Control Protocol), elle s'occupe de réguler le flux de données et assure un transport fiable (données transmises sans erreur et reçues dans l'ordre de leur émission) et non fiable dans le cas d'UDP (User Datagramme Protocol). Il n'est pas garanti qu'un paquet (appelé dans ce cas datagramme) arrive à bon port, c'est à la couche application de s'en assurer.

- Couche internet : Le rôle de la couche Internet consiste à envoyer des paquets source à partir d'un réseau quelconque de l'inter réseau et à les faire parvenir à destination, indépendamment du trajet et des réseaux traversés pour y arriver. Le protocole qui régit cette couche est appelé protocole IP (Internet Protocol). L'identification du meilleur chemin et la commutation de paquets ont lieu au niveau de cette couche.
- Couche accès réseau : C'est la première couche de la pile TCP/IP, elle se charge de tout ce dont un paquet IP a besoin pour établir une liaison physique. Elle prend en charge les notions suivantes :
  - Acheminement des données sur la liaison.
  - Coordination de la transmission de données (synchronisation).
  - Conversion des signaux (analogique-numérique).
  - Contrôle des erreurs à l'arrivée

2.1.2. Encapsulation des données

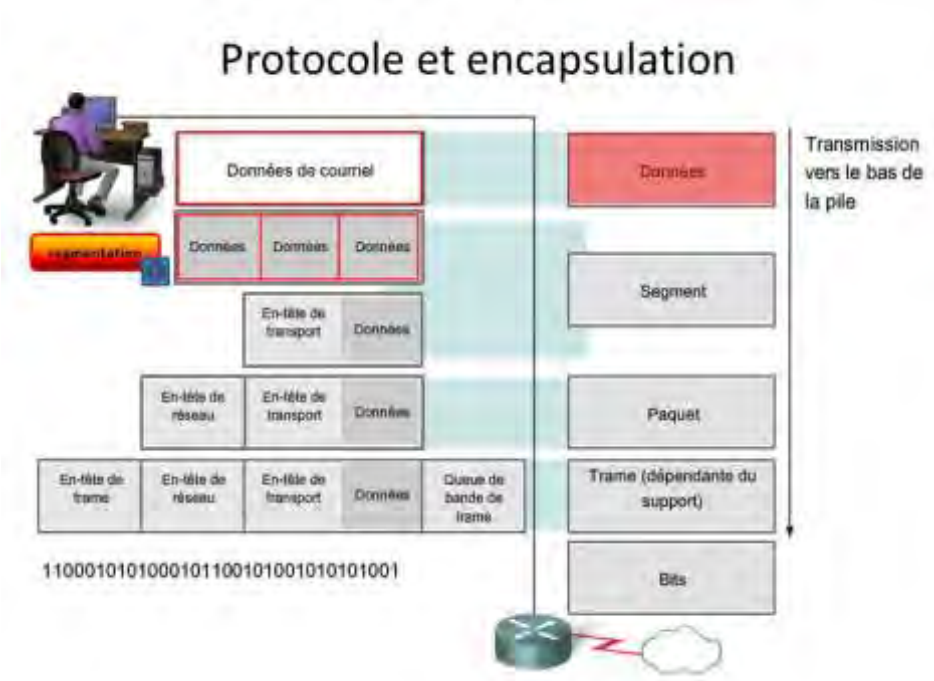


Figure 18 Encapsulation des données (https://slideplayer.fr/slide/1652779/)

Lors d'une transmission, les données traversent chacune des couches au niveau de la machine émettrice. A chaque couche, une information est ajoutée au paquet de données, il s'agit d'une entête, ensemble d'informations qui garantissent la transmission. Au niveau de la machine réceptrice, lors du passage dans chaque couche, l'entête est lu, puis supprimé. Ainsi, à la réception, le message est dans son état original.

2.2. Architecture classique d'un IDS

Nous décrivons dans cette sous-section les trois composants qui constituent classiquement un système de détection d'intrusions. La figure suivante illustre les interactions entre ces trois composants. Un capteur est chargé de collecter des informations sur l'évolution de l'état du système et de fournir une séquence d'événements qui traduit l'évolution de l'état du système. Un analyseur détermine si un sous-ensemble des événements produits par le capteur est caractéristique d'une activité malveillante. Un manager collecte les alertes produites par le

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

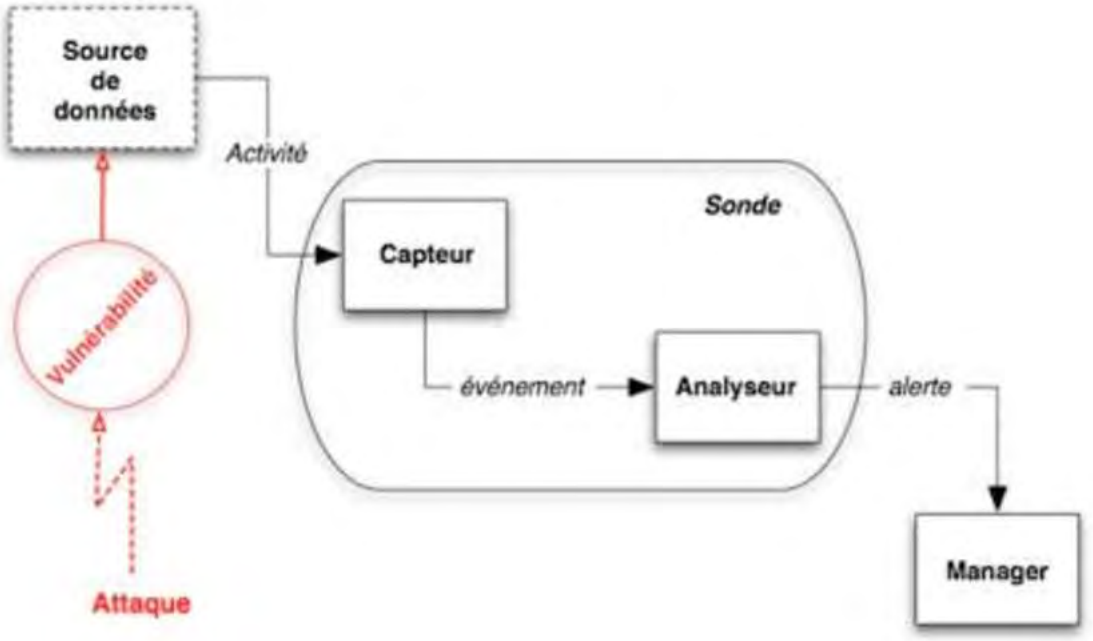


Figure 19 Architecture classique d'un IDS (https://www.researchgate.net/figure/Architecture-classique-dun-IDS_fig2_30514332)

2.2.1. Le capteur

Le capteur observe l'activité du système par le biais d'une source de données et fournit à l'analyseur une séquence d'événements qui informe de l'évolution de l'état du système. Le capteur peut se contenter de transmettre directement ces données brutes, mais en général un prétraitement est effectué. Ce traitement permet, par exemple, de filtrer un certain nombre de données considérées comme non pertinentes afin de limiter la quantité d'information à analyser par la suite. De plus, le capteur réalise généralement une mise en forme des données brutes acquises afin de présenter à l'analyseur des données utilisant un certain format d'événements.

On distingue classiquement trois types de capteurs en fonction des sources de données utilisées pour observer l'activité du système :

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

2.2.2. L'analyseur

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

i. Dans l'approche comportementale, une attaque est qualifiée par la mesure d'une déviation sensible du système surveillé par rapport à un comportement de référence, réputé sain et défini auparavant.
ii. Dans l'approche par signatures, le système de détection possède une base de signatures qui modélisent les différentes attaques connues. L'analyse consiste à rechercher l'occurrence d'un motif caractéristique d'une attaque dans le flux d'événements.

2.2.3. Le manager

Le manager est responsable de la présentation des alertes à l'opérateur (fonction de console de management. Il peut aussi assurer le traitement de l'incident, par exemple au travers des fonctions suivantes :

i. Confinement de l'attaque, qui a pour but de limiter les effets de l'attaque ;
ii. Éradication de l'attaque, qui tente d'arrêter l'attaque ;
iii. Recouvrement, qui est l'étape de restauration du système dans un état sain ;
iv. Diagnostic, qui est la phase d'identification du problème, de ses causes et qui peut éventuellement être suivi d'actions contre l'attaquant (fonction de réaction).

Du fait du manque de fiabilité des systèmes de détection d'intrusions actuels, les réactions sont rarement automatisées, car elles peuvent se traduire par un déni de service en cas de réaction à des fausses alertes.

2.3. Principe de fonctionnement d'un IDS

Un IDS peut analyser les couches Réseau (IP, ICMP), Transport (TCP, UDP) et Application (HTTP, Telnet). Nous pouvons classer les IDS selon la méthode de détection et selon le type de réponse apporté à l'attaque.

2.3.1. Méthodes de détection des IDS

i. Approche par scénario ou par signature

Ce type d'IDS analyse les paquets du réseau ou les données d'un système particulier afin de trouver des signatures, signes de comportements d'intrusion. Il intervient dans le cas où les attaques sont connues et conservées dans une base de données. Cette technique s'appuie ainsi sur la connaissance des techniques utilisées par les attaquants pour en déduire des scénarios typiques. Elle ne tient pas compte des actions passées effectuées par l'utilisateur mais utilise des signatures d'attaques existantes (ensemble de caractéristiques permettant d'identifier une activité intrusive : une chaîne alphanumérique, une taille de paquet inhabituelle, une trame formatée de manière suspecte, …).

ii. Approche comportementale

Ce type d'IDS analyse les données afin de reconnaitre ses situations anormales qui diffèrent du comportement normal du système réseau. Cette capacité peut être atteinte en se basant sur des données et informations précédemment fournies qui ont été utilisés afin de former un algorithme. Cette méthode est très prometteuse car contrairement à la méthode précédente,

elle permet de trouver des attaques zero-day mais aussi permet une personnalisation plus robuste pour un système ou réseau particulier.

iii. Approche hybride

Les IDS hybrides sont des combinaisons d'IDS basés sur la signature et d'IDS basés sur les anomalies. Il permet de combiner les avantages des deux solutions précédentes pour minimiser les fausses alarmes mais aussi augmenter l'efficacité de la détection d'attaques connues.

2.3.2. Les différents types de réponses aux attaques

Il existe deux types de réponses, suivant les IDS utilisés à savoir la réponse passive et la réponse active.

i. La réponse passive

La réponse passive est disponible pour tous les IDS. Elle consiste à enregistrer les intrusions détectées dans un fichier de log qui sera analysé par le responsable de sécurité. Certains IDS permettent d'enregistrer l'ensemble des connexions identifiées comme malveillantes. Ceci permet ainsi de remédier aux failles de sécurité pour empêcher les attaques enregistrées de se reproduire, mais elle n'empêche pas directement une attaque de se produire.

ii. La réponse active

Elle doit être implémentée au niveau de l'IDS. Contrairement à la réponse passive, elle a pour but de stopper une attaque au moment de sa détection. Pour cela on dispose de deux techniques : la reconfiguration du firewall et l'interruption d'une connexion TCP. La reconfiguration du firewall permet de bloquer le trafic malveillant au niveau du firewall, en fermant le port utilisé ou en interdisant l'adresse de l'attaquant. Cette fonctionnalité dépend du modèle de firewall utilisé, tous les modèles ne permettant pas la reconfiguration par un IDS. De plus, cette reconfiguration ne peut se faire qu'en fonction des capacités du firewall.
L'IDS peut également interrompre une session établie entre un attaquant et sa machine cible, de façon à empêcher le transfert de données ou la modification du système attaqué. Pour cela l'IDS envoie un paquet TCP reset aux deux extrémités de la connexion (cible et attaquant). Un paquet TCP reset a le flag RST de positionner, ce qui indique une déconnexion de la part de l'autre extrémité de la connexion. Chaque extrémité en étant destinataire, la cible et l'attaquant pensent que l'autre extrémité s'est déconnectée et l'attaque est interrompue. Dans le cas d'une réponse active, il faut être sûr que le trafic détecté comme malveillant l'est réellement, sous peine de déconnecter des utilisateurs normaux

2.4. Classification des IDS

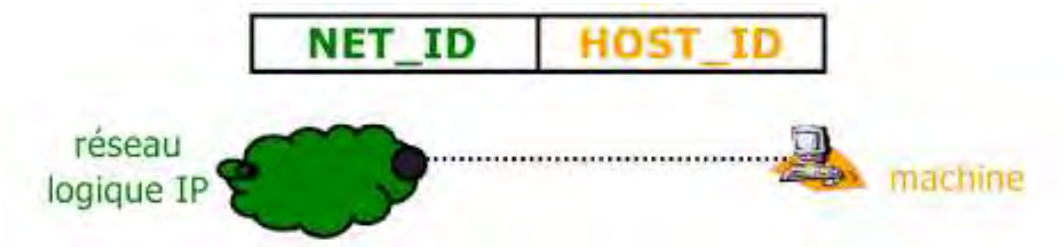


Figure 20 Classification des IDS (wikipedia.org)

Les IDS bases sur le domaine de surveillance dépendent d'où l'activité suspecte a été observée.
Elle peut donc se situer au niveau d'une machine cliente ou au niveau du réseau.

2.4.1. IDS basé sur l'hôte (HIDS)

Les HIDS analysent le fonctionnement et l'état des machines sur lesquelles ils ont installés, afin de détecter des attaques. Ils ont pour objectif d'analyser les journaux système, de contrôler l'accès au système, de vérifier l'intégrité des systèmes de fichiers etc. Ils sont en général placés sur des machines sensibles, susceptibles de subir des attaques car possédant des données sensibles de l'entreprise. Ce type d'IDS possède un certain nombre d'avantages : il est possible de constater immédiatement l'impact d'une attaque et donc de mieux réagir. Grâce à la quantité des informations étudiées, il est possible d'observer les activités se déroulant sur l'hôte avec précision et d'optimiser le système en fonction des activités observées. Pour finir, voici quelques HIDS connus:

i. Dragon Squire,
ii. Security Manager

2.4.2. IDS basé sur le réseau (NIDS)

Le rôle essentiel d'un IDS réseau 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.

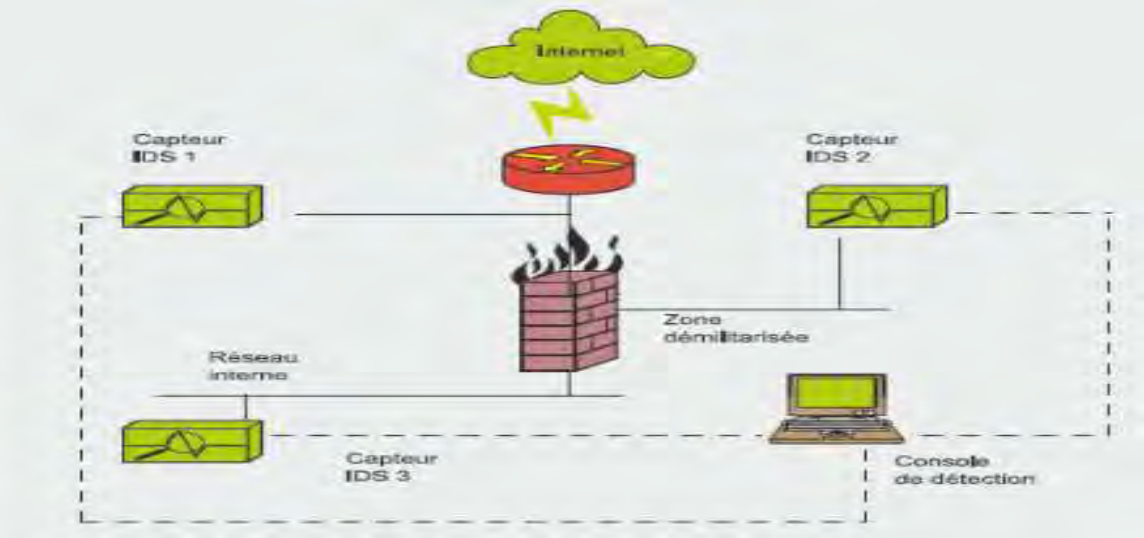


Figure 21 Architecture d'un NIDS (https://www.memoireonline.com/ Mise-en-place-dun-IDS)

Les capteurs placés sur le réseau sont placés en mode furtif, 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. Ainsi les alertes sont envoyées à une console sécurisée, qui les analyse et les traite éventuellement. Cette console est généralement située sur un réseau isolé, qui relie uniquement les capteurs et la console. Voici quelques outils open source :

i. SNORT
ii. Prélude

2.4.3. IDS hybride

Les systèmes de détection d'intrusion hybride rassemblent les caractéristiques de HIDS et NIDS. Ils permettent, en un seul outil, de surveiller le réseau et l'hôte grâce à des sondes. Les sondes sont placées dans des points stratégiques, et agissent comme NIDS et /ou HIDS

suivant leurs emplacements. Toutes les sondes remontent alors les alertes à une machine qui va centraliser, agréger et lier les informations d'origines multiples.

2.5. Limites et inconvénients d'un IDS

Aucun système d'information n'est sûr à 100%. Pour une entreprise connectée à Internet, le problème aujourd'hui n'est plus de savoir si elle va se faire attaquer mais plutôt, quand sera-t-elle attaquée, d'où une mise en œuvre de divers moyens destinés à augmenter le niveau de sécurité et prévenir ces attaques. Ainsi les entreprises se tournent de plus en plus vers les solutions d'IDS, dont les possibilités et capacités ne sont plus à prouver. Cependant, ces dernières présentent un certain nombre de limites et d'inconvénients parmi lesquels :

2.5.1. La détection d'anomalies

L'IDS utilisant la détection d'anomalies génère énormément d'alertes qui ne correspondent pas forcement à des attaques.

2.5.2. Pollution/surcharge

Les IDS peuvent être pollués ou surchargés, par exemple par la génération d'un trafic important. Une quantité importante d'attaques bénignes peut également être envoyée afin de surcharger les alertes de l'IDS. Des conséquences possibles de cette surcharge peuvent être la saturation de ressources (disque, CPU, mémoire), la perte de paquets, le déni de service partiel ou total.

2.5.3. Consommation de ressources

Outre la taille des fichiers de journalisation (log), la détection d'intrusion est fortement gourmande en ressources.

2.5.4. Perte de paquets

Les vitesses de transmission sont parfois tellement rapides qu'elles dépassent largement la vitesse d'écriture des disques durs les plus rapides du marché, ou même la vitesse des processeurs. De ce fait, des paquets peuvent ne pas être reçus par l'IDS mais reçus par les machines destinataires.

2.5.5. Vulnérabilité aux dénis de services

Un attaquant peut essayer de provoquer un déni de services au niveau du système de détection d'intrusion ou au pire au niveau du système d'exploitation de la machine supportant l'IDS. Ainsi une fois l'IDS désactivé, le hacker pourra tenter d'accéder au réseau et faire ce qui lui plait.

2.5.6. Contournement/évasion

Les IDS peuvent être contournes et outrepassés.
Dans le cas d'une attaque par évasion, le système de détection d'intrusion rejette un paquet qui sera pourtant accepté par la machine destinataire. D'autres attaques peuvent être mises en place afin d'outrepasser l'IDS.
Aussi, la source d'une attaque n'est pas toujours identifiable. On peut citer le cas de l'attaque <<Scan Zombie>> qui permet de s'appuyer sur une autre machine afin de cacher la provenance de l'attaque à l'IDS. En effet, l'outil nmap dispose de l'option <<-sl>> qui permet

de forger des paquets adéquats et de se faire passer pour une autre machine afin d'analyser le comportement à adopter en fonction de la réaction de la cible.

2.5.7. L'activation du mode promiscuous

i. Réponse involontaire du système

En effet, les IDS doivent mettre leurs cartes réseau sur ce mode afin de pouvoir recevoir l'intégralité des trames circulant sur le réseau. L'IDS ne génèrera aucun trafic et se contentera d'analyser les paquets.
Cependant ce mode affranchit les machines du filtrage MAC. Il se peut ainsi que la machine réponde à certaines requêtes (ICMP ECHO REQUEST).

ii. Mise en évidence de la présence d'un IDS

En effet, ce mode génère des accès mémoire et processeurs assez importants et ainsi en comparant les temps de réponse, il est possible de détecter la présence d'un IDS derrière.

2.5.8. La base de données de signature

Toutes les attaques ne sont pas détectées d'où le besoin de mettre à jour au fil du temps sa base de données de signatures.

En définitive, IDS comporte de nombreux avantages pour les administrateurs réseau mais présente aussi un certain nombre d'inconvénients et de limites à considérer pour une meilleure efficacité de détection d'intrusion.

Chapitre 3 Intelligence artificielle - Apprentissage automatique - Apprentissage profond

3.1. C'est quoi l'IA

Le terme intelligence artificielle (IA) désigne le domaine d'étude ayant pour objet la reproduction artificielle des facultés cognitives de l'intelligence humaine comme la perception, le raisonnement, l'apprentissage et la résolution de problèmes.

On attribue les origines de l'IA aux années 50, lorsque des chercheurs se sont intéressés à la question si un ordinateur peut être programmé de sorte qu'il « réfléchisse ». L'IA n'est pas une technologie unique associée à une application particulière, mais un ensemble de technologies possédant des applications générales dans beaucoup de domaines. Pendant assez longtemps, de nombreux experts croyaient que l'intelligence artificielle pouvait être réalisée en programmant un ensemble suffisamment grand de règles explicites. Cette approche, connue sous le nom d'IA symbolique, était le paradigme dominant au début de l'IA. C'est ainsi qu'ont été construit des programmes sachant jouer aux échecs. Cependant l'IA symbolique est difficile à maintenir et est peu performante pour des taches complexes comme la reconnaissance de la parole. Ces deux difficultés peuvent être gérées par l'apprentissage automatique qui est actuellement l'approche dominante en intelligence artificielle.

3.2. Apprentissage automatique

3.2.1. Qu'est-ce que l'AA

Bien qu'il soit de plus en plus populaire, le ML n'est pourtant pas une discipline récente. Nous commençons avec deux définitions proposées.

« L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés. » [16].

Une autre définition plus technique.
« Étant donné une tache T et une mesure de performance P, on dit qu'un programme informatique apprend à partir d'une expérience E si les résultats obtenus sur T, mesurés par P, s'améliorent avec l'expérience E. » [17].

L'idée générale qui en sort est que l'apprentissage automatique est la science de programmer une machine de sorte qu'elle puisse apprendre (une tache) à partir de données. La phase d'apprentissage est aussi appelée entrainement. Après cette phase, nous obtenons un modèle capable de faire des prédictions sur des nouvelles données sans instructions explicites.

Remarque : Un modèle d'apprentissage automatique est la sortie d'un algorithme de ML déjà entrainé. Cependant les termes algorithme d'apprentissage automatique et modèle d'apprentissage automatique sont parfois utilisés de façon interchangeable.

Cas d'application	Jeu de dames	Reconnaissance de caractères manuscrits
Taches	Jouer aux dames	Reconnaitre et classer les caractères manuscrits présents dans des images
Mesure de performance	Pourcentage de matchs gagnés contre des adversaires	Pourcentage de mots correctement classifies
Entrainement	Pratiquer le jeu en jouant contre soi-même	Une base de données disponible de mots classifiés

Tableau 1 Exemples de cas d'utilisation de l'apprentissage automatique

3.2.2. Pourquoi utiliser l'AA

Nous allons répondre à cette question à l'aide d'un exemple Supposons qu'on veuille construire de façon traditionnelle un algorithme qui classifie des avis de spectateurs d'un film. Nous aurions essentiellement ces étapes :

- Examiner ce qui caractérise en général un avis positif ou négatif. Nous remarquerons que les avis positifs comportent souvent des mots comme « aimer », « bon », « extraordinaire ». Tandis que les avis négatifs s'expriment avec des mots comme « affreux », « mauvais », …

- Écrire un algorithme qui détecte ces caractéristiques et dira qu'un avis est positif dépendamment du nombre de caractéristiques positifs rencontrés

- Ajouter autant de caractéristiques que possibles et répéter les étapes ci-haut jusqu'à avoir un résultant satisfaisant

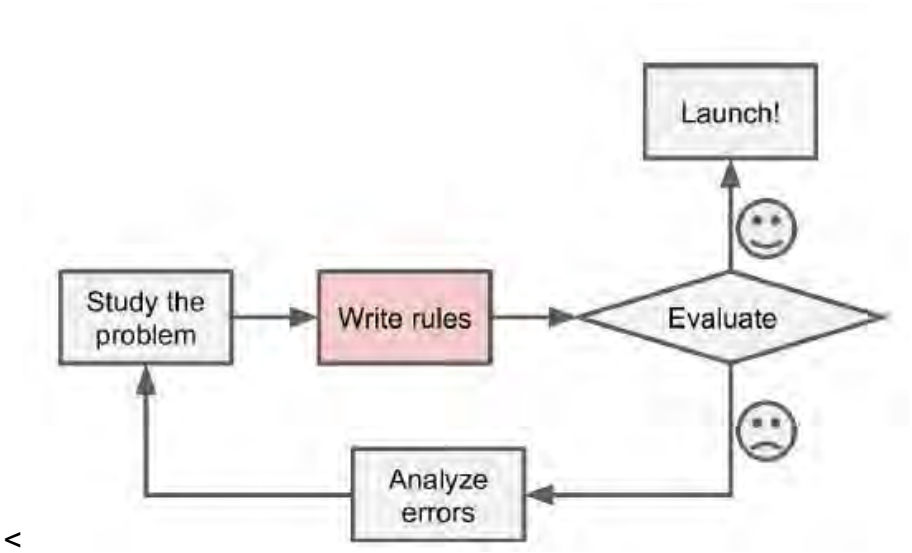


Figure 22 [1] Etapes d'un algorithme

Le problème avec cette approche est que nous devrons continuellement ajouter de nouvelles règles afin d'avoir un résultat satisfaisant. Par exemple, s'il y a des abonnés qui utilisent des anglicismes, il faudra ajouter des règles pour des mots en anglais. Cet ajout de règles va complexifier notre algorithme et le rendre difficile à maintenir. De plus une liste de règles aura du mal à classifier des avis complexes comme « j'adore l'actrice principale, mais ce film est deguelasse. ».
Contrairement à cet algorithme qui utilise une liste de règles de décision, un algorithme de ML apprend automatiquement quels sont les caractéristiques (mots ou phrase) qui expriment un sentiment positif ou négatif. Ceci le rend donc plus facile à maintenir. De plus le l'algorithme peut aider à découvrir d'autres caractéristiques qui influencent sur le sentiment exprimé comme l'utilisation d'expression « Waouh », « Oh my god ».
Un algorithme de ML peut aussi résoudre certains problèmes pour lesquels il n'existe pas de solution connue ou qui sont très complexes. Considérons par exemple, la tache de retranscrire la voix en écrit. Il serait très dur d'écrire des règles qui s'adaptent à la façon dont toutes les personnes parlent dépendamment de l'intonation de leur voix, leur accent etc.

3.2.3. Qu'est-ce qu'une donnée

En statistique, une donnée est le résultat d'une observation faite sur une population. Une donnée constitue un élément d'information dans un contexte donné. Les données constituent la nourriture des algorithmes d'apprentissage automatique.

i. Type de données
- Une donnée quantitative décrit une quantité mesurable, sous la forme de nombres sur lesquels on peut appliquer des opérations mathématiques
- Une donnée qualitative décrit quant à elle des catégories ou des types. Nous ne pouvons pas la mesurer ou la décompter

ii. Niveaux de structuration des données
- Les données structurées sont des données que nous pouvons organiser sous formes de tableaux dont les lignes indiquent les observations et les colonnes les variables.

Elles proviennent souvent de bases de données relationnelles. Ex : base de données de stock d'articles disponibles au niveau de différents magasins.

Id_magasin	Date	Stock Lait	Stock Œufs
1204	2020-12-12	400	76
679	2020-12-31	80	456

- Les données non structurées font référence aux pixels d'une image, aux différents mots d'un texte, aux différentes valeurs d'un signal. Elles ne sont pas stockées sens sous forme de tableau.
  Image onde

3.2.4. Types d'AA

Nous rencontrons principalement 3 types d'apprentissage automatique dépendant de l'importance et la nature de la supervision humaine requise.

i. Apprentissage supervisé

Dans ce type d'apprentissage, les données d'entrainement ou caractéristiques (features en anglais) qui sont fournies à l'algorithme comportent les solutions désirées, appelées étiquettes ou labels.
Une classification est un type d'apprentissage automatique ayant pour objectif de prédire la classe d'une observation. Ex : un filtre anti-spam
Une régression est un type d'apprentissage automatique qui consiste à prédire une valeur numérique continue. Ex : Prédire le prix d'une voiture
Voici quelques exemples d'algorithmes d'apprentissage supervisé

- K plus proches voisins
- Régression linéaire
- Régression logistique
- Machines à vecteurs de support
- Arbres de décision

ii. Apprentissage non supervisé

Dans ce type d'apprentissage, les données d'entrainement ne comportent pas d'étiquettes. L'algorithme essaie d'apprendre sans étiquette.
Voici quelques exemples d'algorithmes d'apprentissage non supervisé

- K-moyennes (K-means)
- Analyse en composantes principales (PCA)
- Méthode t-SNE (t distributed stochastic neighbor embedding)

Un algorithme K-moyennes est utile pour faire du partitionnement. Par exemple, cet algorithme peut segmenter les utilisateurs d'un site de vente de livres qui ont des comportements similaires.

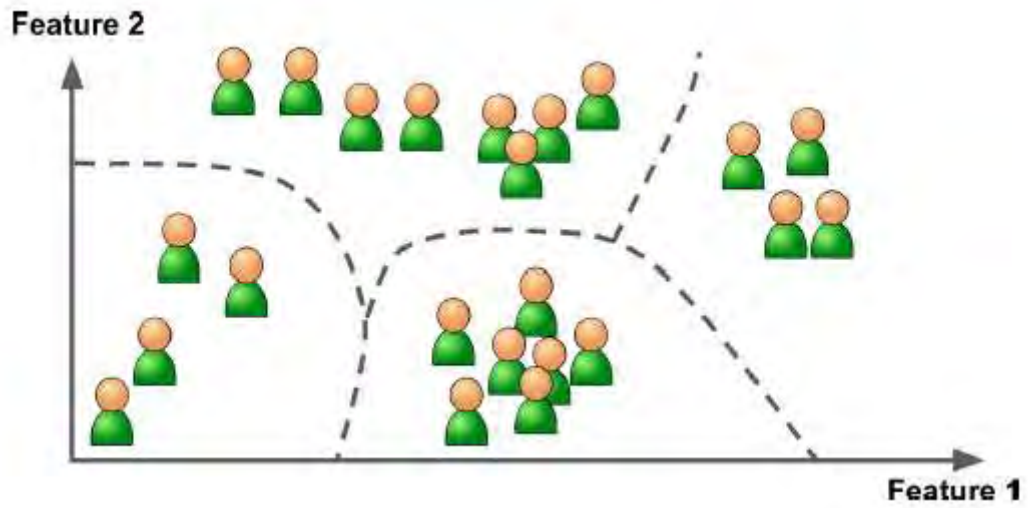


Figure 23 [1] Exemple d'utilisation du K-moyennes

Les algorithmes PCA ou t-SNE permettent de réduire la dimensionnalité d'un jeu de données en gardant les informations les plus importantes. Ceci peut être intéressant lorsqu'on travaille avec des signaux.

iii. Apprentissage par renforcement

L'apprentissage par renforcement est très différent des deux types d'apprentissage déjà rencontrés. Ici on utilise les notions : agent, action, récompense, pénalité, politique.
L'agent observe le contexte (variables), sélectionne et accomplit des actions. L'agent obtient en retour des récompenses ou pénalités. Une politique définit quelle action l'agent doit choisir dans un contexte donné. L'agent doit donc apprendre par lui-même la meilleure politique pour obtenir autant de récompenses que possible.

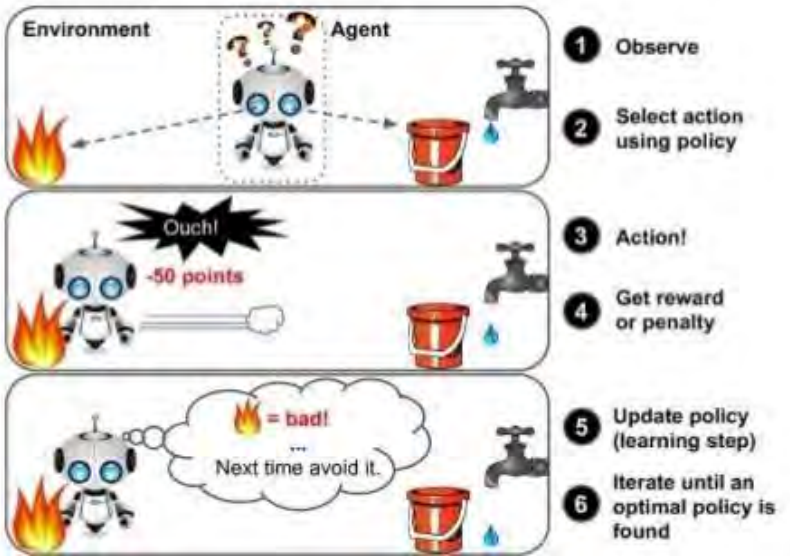


Figure 24 [1] Exemple d'apprentissage automatique par renforcement

Pour comprendre l'apprentissage par renforcement, pensons au dressage d'un animal afin qu'il agisse de telle ou telle façon en fonction des récompenses qu'on lui donne. Si un chien reçoit une friandise chaque fois que son maître lui demande de s'asseoir, cette action deviendra alors chez lui automatique.

L'apprentissage par renforcement est plus présent en robotique. De nombreux robots utilisent ce type d'apprentissage pour apprendre à marcher.

3.2.5. Problèmes rencontrés dans un algorithme d'AA

i. Données insuffisantes

La plupart des algorithmes d'apprentissage automatique ont besoin d'un grand nombre de données. La quantité des données est un facteur important afin d'avoir un modèle performant. Ce résultat est renforcé par les travaux de Banko et al [9].
Dans cet article, les auteurs montrent l'augmentation de la performance d'un modèle de traitement du langage naturel lorsqu'ils ajoutent plus de mots à leur corpus (ensemble de mots de données d'entrainement).

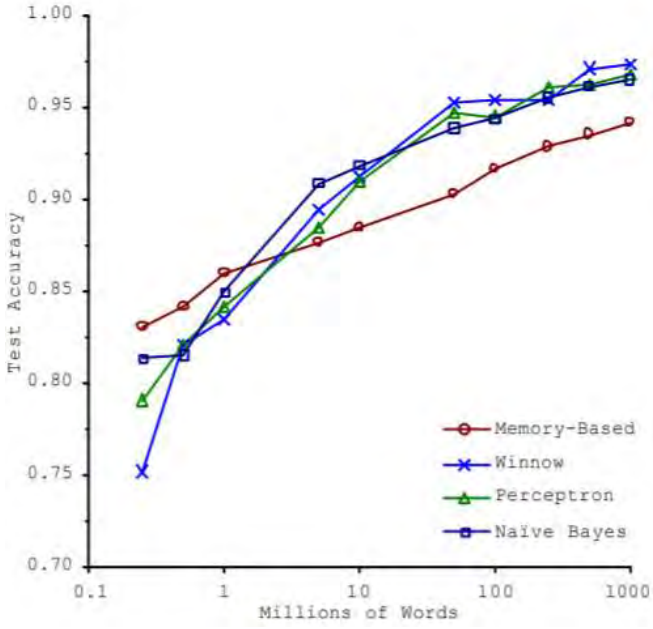


Figure 25 [9] Performance d'un modele par rapport aux données

Halevy et al [2] vont plus loin en prônant l'idée que les données sont plus importantes que les algorithmes.
Cependant, nous assistons de plus en plus à des techniques pour corriger le manque de données comme le transfert d'apprentissage.

ii. Surajustement ou surapprentissage

Le surajustement se produit lorsque l'algorithme d'apprentissage ou modèle est trop complexe par rapport à la quantité de données d'apprentissage et au bruit qu'elles contiennent. En plus d'apprendre les propriétés nécessaires des données d'entrainement, le modèle capture des propriétés non nécessaires. Ce phénomène cause un manque de généralisation de la performance du modèle.

Compromis Biais - Variance
Lors du surajustement, le modèle est très bon pour les données d'entrainement et commet peu d'erreurs. Il présente un haut biais. Cependant, le modèle varie fortement pour un changement des données. Le modèle présente une haute variance.

Pour résoudre le problème de surentrainement, nous pouvons entre autres :

- Simplifier le modèle : réduire le nombre de paramètres du modèle, imposer des contraintes aux paramètres
- Augmenter le nombre de données d'entrainement
- Réduire le bruit dans les données (supprimer les données aberrantes)

iii. Sous-ajustement

Le sous-ajustement est le phénomène inverse. Il se produit lorsque le modèle est trop simple pour capturer l'information voulue à partir des données. Pour corriger ce problème, nous pouvons

- Choisir un algorithme d'apprentissage plus puissant
- Fournir de meilleures variables à l'algorithme d'apprentissage

3.3. Apprentissage profond

3.3.1. Réseau de neurones artificiels

L'apprentissage profond est basé sur les réseaux de neurones artificiels (RNA). Par essence, les RNA imitent le cerveau humain, tout comme les oiseaux ont inspiré la création des avions. Les RNA sont progressivement devenus semblables au fonctionnement des neurones biologiques. Certains chercheurs utilisent même le terme unité au lieu de neurones pour marquer la différence.

Haykin [14] définit un RNA comme « une combinaison parallèle de simples unités de calcul qui peuvent acquérir un savoir à partir d'un environnement à travers un processus d'entrainement et garder ce savoir dans ses connections ».

Un RNA est généralement constitué

- D'une couche d'entrée qui distribue les entrées aux neurones,
- D'une ou plusieurs couches cachées
- D'une couche de sortie

Dans les couches cachées et la couche de sortie, les neurones effectuent des sommes pondérées de leurs entrées, puis appliquent des transformations appelées fonctions d'activation.

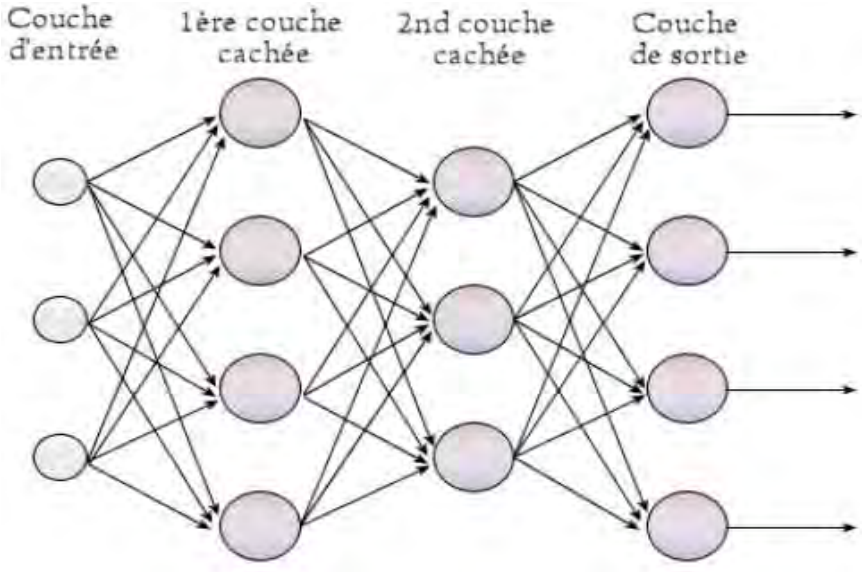


Figure 26 Architecture d'un RNA (wikipedia.org)

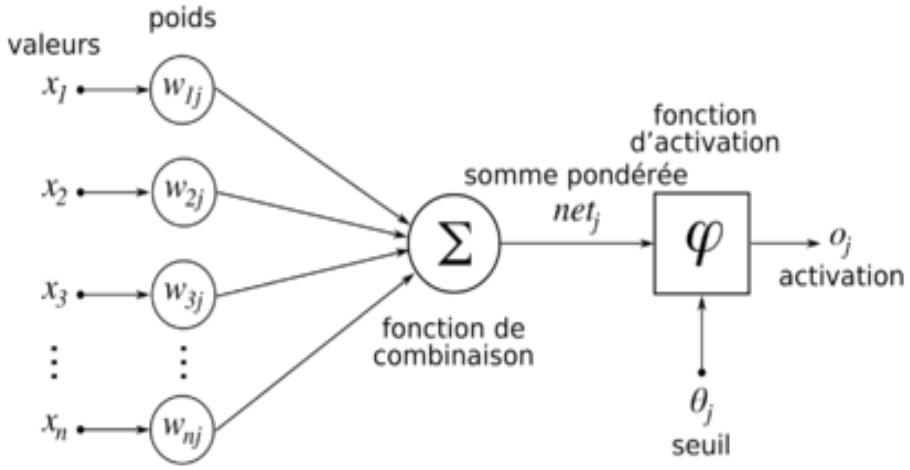


Figure 27 [12] Etapes d'un RNA

Lorsqu'un RNA possède plus de deux couches cachées, on parle de réseau de neurones profond (RNP).
Entrainer un RNA a été un grand défi pendant longtemps. A un moment donné, il a même été abandonné par certains chercheurs au profit des machines à vecteurs de supports. Cependant Rumelhart et al [4] ont publié un article révolutionnaire dans lequel ils introduisent un algorithme d'entrainement à rétropropagation, appelée aujourd'hui descente de gradient.

i. Descente de gradient

Cet algorithme fournit chaque observation d'entrainement au réseau et calcule la sortie de chaque neurone dans chaque couche consécutive : il s'agit de la passe en avant.
Puis l'algorithme mesure l'erreur de sortie du réseau et détermine dans quelle mesure chaque neurone de la dernière couche cachée a contribué à l'erreur de chaque neurone de sortie. Par la suite, l'algorithme poursuit en mesurant la portion de ces contributions à l'erreur qui revient à chaque neurone de la couche cachée précédente. Le processus se poursuit jusqu'à ce que l'algorithme atteigne la couche d'entrée : il s'agit de la passe en arrière. Enfin l'algorithme modifie les poids des connexions entre les neurones du RNA afin de réduire l'erreur

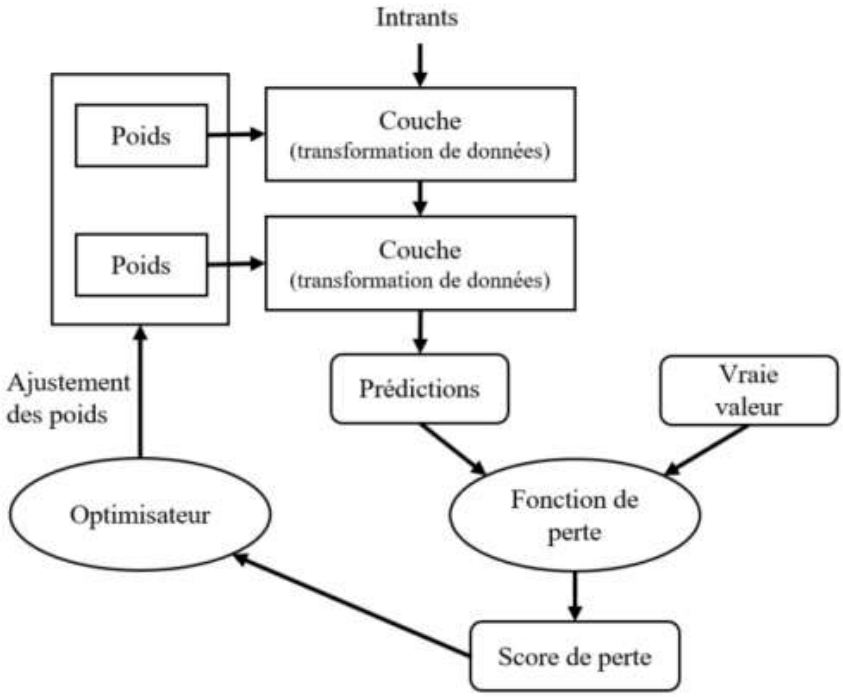


Figure 28 Descente de gradient
(https://savoirs.usherbrooke.ca/bitstream/handle/11143/13397/Cornioley_Paul_MSc_2018.pdf?sequence=1&isAllowed=y)

ii. Fonction d'activation

Les fonctions d'activation permettent aux RNA de pouvoir apprendre à partir de distributions de données complexes. Ne pas utiliser de fonctions d'activation reviendrait à réaliser un modèle linéaire. Les fonctions d'activation les plus utilisées dans la littérature sont :

- La fonction sigmoïde : $\sigma(x) = 1/(1+\exp(-x))$
- La fonction tangente hyperbolique : $\tanh(x) = 2\sigma(2x) - 1$
- La fonction ReLu : $\mathrm{Relu}(x) = \max(0, x))$

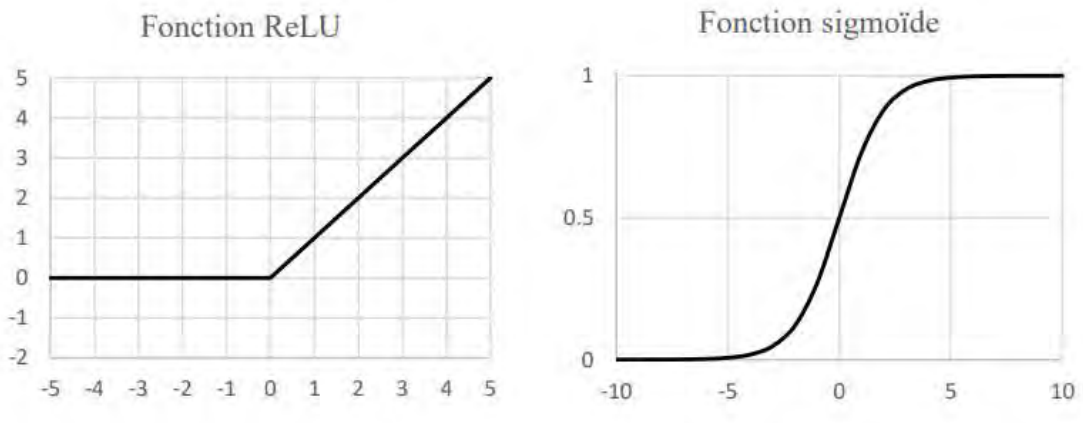


Figure 29 [12] Exemples de fonctions d'activation

3.3.2. Réseaux de neurones à convolution ou convolutifs

Les réseaux de neurones convolutifs (CNN) sont un type de réseaux de neurones particulièrement adaptés pour le traitement d'images. Ils ont été développés à la suite d'études du cortex visuel du cerveau. En effet, en 1958, Hubel et al [5] ont effectué des observations intéressantes. D'après [1], ces observations peuvent être résumées comme suit

- Certains neurones du cortex visuel ont un petit champ récepteur local et ils réagissent uniquement à un stimulus visuel qui se trouve dans une région limitée du champ visuel.
- Certains neurones réagissent uniquement aux images de lignes horizontales, tandis que d'autres réagissent uniquement aux images ayant d'autres orientations.
- Certains neurones réagissent à des motifs plus complexes correspondant à une combinaison de motifs de bas niveau.

Les neurones d'un CNN sont construits en s'inspirant de ces observations. Dans chaque couche, les neurones sont connectés uniquement aux neurones situés à l'intérieur d'un petit rectangle de la couche précédente. Ceci permet au réseau d'apprendre des caractéristiques de bas niveau au niveau des premières couches, puis de les assembler en caractéristiques de haut niveau dans les couches cachées. De plus, si une caractéristique est apprise à un endroit de l'image, elle sera reconnue à n'importe quel autre endroit de l'image, le réseau n'a pas besoin de l'apprendre à nouveau.

3.3.3. Réseaux de neurones récurrents

Les réseaux de neurones récurrents (RNR) constituent une extension des RNA. L'architecture d'un RNR est basée sur l'utilisation de connexions neuronales cycliques qui permettent de considérer les états précédents des neurones. Ainsi cette technique procure aux neurones un concept de « mémoire ».
Ils sont pratiques pour modéliser des données séquentielles. Ils sont très performants pour modéliser des données physiologiques comme un électrocardiogramme, réaliser une conversion de voix en texte…

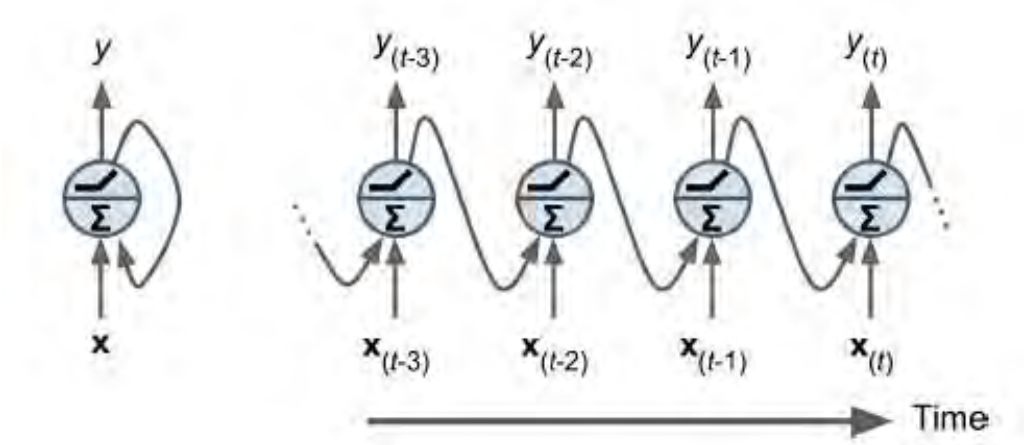


Figure 30 [1] Shéma d'un RNR

3.3.4. Autoencodeurs

Un autoencodeur est un RNP utilisé en apprentissage non supervisé. L'objectif d'un autoencodeur est de reconstruire son entrée de la meilleure façon en apprenant une représentation de cette entrée appelée codage. L'espace des codages est appelée espace latent. Ainsi un autoencodeur apprend simplement à copier son entrée. Cependant cette tâche est rendue challengeant en imposant des contraintes au réseau de neurones autoencodeur, comme la compression de la représentation des observations. Ce type de réseau est généralement construit de façon symétrique avec deux parties appelées Encodeur et décodeur.

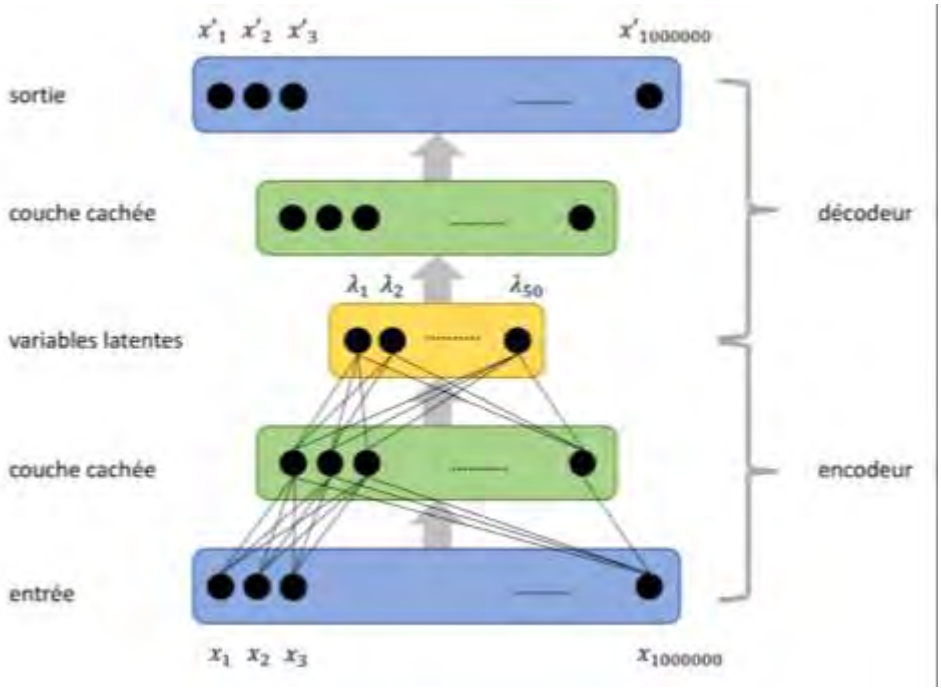


Figure 31 Architecture d'un autoencodeur (https://weave.eu/)

Il peut être utilisé indirectement en apprentissage supervisé. En effet, nous pouvons passer à un autoencodeur beaucoup d'images de pommes. A la fin de l'entrainement, nous nous attendons qu'il reconstruise correctement des images de pommes. Cependant si nous lui donnons une image contenant un autre fruit, l'erreur de reconstruction sera supérieure à ce que nous avions jusqu'à présent. L'autoencodeur devient ainsi un classificateur pomme vs autre chose. Cette technique est très utilisée dans la détection d'anomalies.
D'autres types d'autoencodeur existent

i. Autoencodeur débruiteur

Pour imposer plus de contraintes à l'autoencodeur, nous pouvons corrompre les entrées en leur ajoutant du bruit et entrainer l'autoencodeur pour qu'il retrouve ces entrées sans le bruit.
Vincent et al [13] présentent la phase d'ajout de bruit comme suit : pour chaque entrée x, un nombre constant donné de ses composantes est choisi au hasard, et les valeurs dans ces composantes sont remplacées par 0. Les valeurs dans les autres composantes restent inchangées.

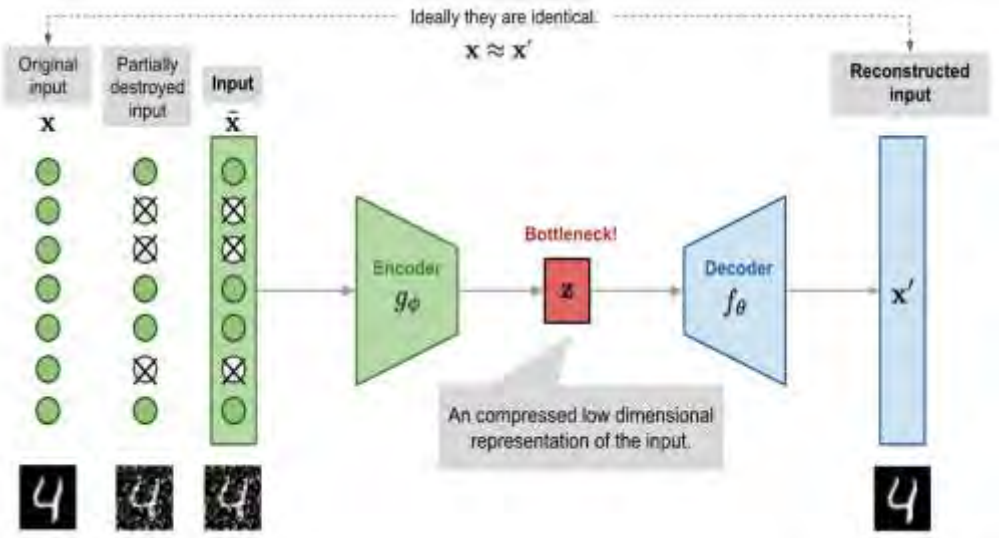


Figure 32 Autoencodeur debruiteur (lilianweng.github.io/)

ii. Autoencodeur variationnel ou VAE

Ce type d'autoencodeur a été proposé par Kingma et al [6]. C'est devenu l'un des autoencodeurs les plus populaires.

Selon les auteurs, une VAE « apprend une correspondance entre l'espace des observations x, dont la distribution est compliquée et l'espace latent des codages z dont la distribution peut être simple (comme une loi normale) ».

Ainsi au lieu de produire directement un codage pour une entrée donnée, l'encodeur produit un codage moyen μ et un écart typeσ. Le codage réel est ensuite échantillonné aléatoirement avec une loi normale de moyenne μ et d'écart type σ. Enfin le décodeur décode normalement le codage réel.

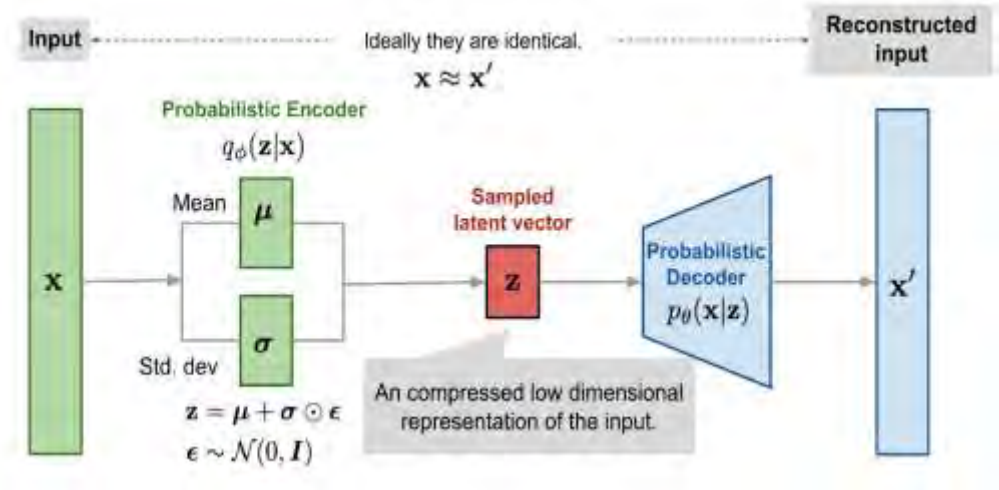


Figure 33 Autoencodeur varationnel (lilianweng.github.io/)

iii. Autoencodeur contradictoire

Un premier RNA, appelé le discriminant est entrainé à distinguer les vraies données, des fausses produites par un second réseau appelé le générateur. Pendant l'entrainement le générateur apprend à tromper le discriminant, tandis que le discriminant apprend à éviter les

pièges du générateur. Cette compétition aboutit à des fausses données de plus en plus réalistes et des codages de plus en plus robustes.

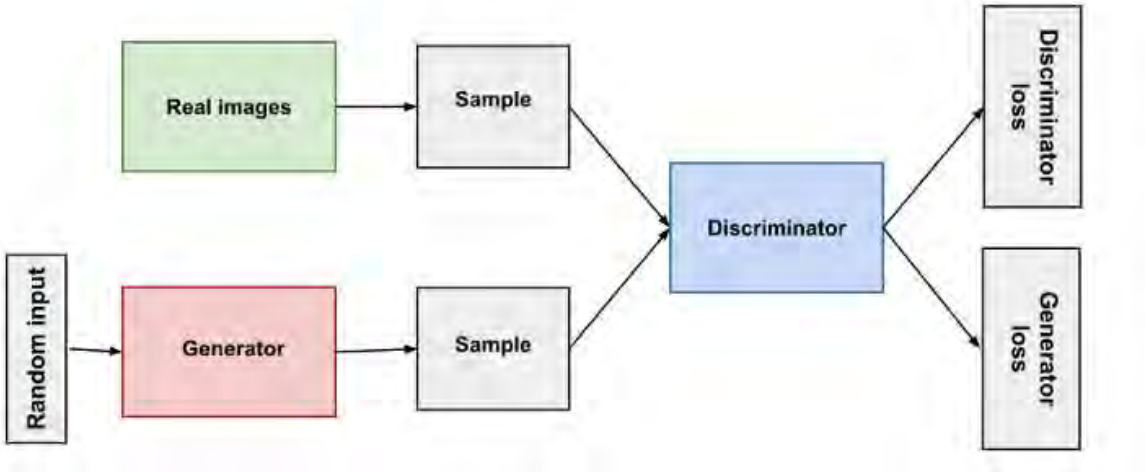


Figure 34 Autoencodeur contradictoire (https://wandb.ai/authors/alae/reports/Adversarial-Latent-Autoencoders--VmlldzoxNDA2MDY)

Chapitre 4 Apprentissage automatique et IDS

Les NIDS sont devenus des outils essentiels pour une infrastructure informatique moderne. Cependant les NIDS traditionnels rencontrent de nombreux inconvénients dus à leur efficacité limitée dans la détection d'intrusion mais également au niveau du taux élevé de fausses alertes. Prenons l'exemple de Snort (qui est un NIDS open source très répandu), la fréquence de certains faux positifs (activités considérées comme attaques alors qu'elles ne le sont pas) était telle que les signatures elles-mêmes ont dû être désactivées, les rendant ainsi inutiles. Cette situation a encouragé ainsi les chercheurs à adopter l'apprentissage automatique. En effet, cette nouvelle approche décrit un mécanisme de détection d'intrusion capable de détecter les intrusions avec plus de précision et de réduire considérablement le taux de fausses alertes.
Dans ce sens, de nombreuses études connexes ont été réalisées dans le but de traiter des techniques d'apprentissage automatique pour les systèmes de détection d'intrusion.

4.1. Les métriques de performances

Cette section décrit les métriques d'évaluation les plus couramment utilisées afin de mesurer les performances des méthodes de l'apprentissage automatique et de l'apprentissage profond. Toutes les métriques d'évaluation sont basées sur les différents attributs utilisés dans la matrice de confusion. Une matrice de confusion est une matrice bidimensionnelle fournissant des informations sur la classe réelle et prédite. Elle comprend :

i. Vrai positif : les instances de données correctement prédites comme une attaque
ii. Faux négatif : les instances de données prédites à tort comme des instances normales.
iii. Faux positif : les instances de données classées à tort comme une attaque.
iv. Vrai négatif : les instances correctement classées comme instances normales.

La diagonale de matrice de confusion indique les bonnes prédictions tandis que les éléments à l'extérieure de la diagonale sont les mauvaises prédictions pour un algorithme donné.
Dans les études récentes portant sur le développement d'un IDS basé sur l'apprentissage automatique, six (6) métriques d'évaluation ont été définies à savoir :

i. La précision : C'est le rapport entre les attaques correctement prédites et tous les échantillons étiquetés comme des attaques.
ii. Le rappel : C'est un rapport de tous les échantillons correctement classés comme des attaques à tous les échantillons qui sont en fait des attaques. C'est aussi appelé un taux de détection.
iii. Le taux de fausses alarmes : Il est également appelé le taux de faux positifs et est défini comme le rapport d'échantillons d'attaques mal prédits à tous les échantillons normaux.
iv. Le taux de vrai négatif : Il est défini comme le rapport du nombre d'échantillons normaux correctement classés à tous les échantillons qui sont normaux.
v. La précision de détection : Il s'agit du rapport entre les instances correctement classées et le nombre total d'instances.
vi. Mesure F : Il est défini comme la moyenne harmonique de la précision et du rappel. En d'autres termes, c'est une technique statistique pour examiner la précision d'un système en tenant compte à la fois de la précision et du rappel du système.

4.2. Survol de la littérature

Gao et al [18] ont testé cette approche en choisissant un modèle d'ensemble utilisant plusieurs classificateurs de base tels que l'arbre de décision, les forêts aléatoires, la K-moyennes, les RNP. Leur modèle a été vérifié à l'aide du jeu de données *NSL-KDD* (voir plus loin) et les résultats expérimentaux ont démontré son efficacité par rapport à des modèles proposés précédemment. Cependant, elle n'a pas donné de résultats satisfaisant pour certaines classes d'attaques plus rares. Shone et al [11] ont eux aussi eu à utiliser un autoencodeur comportant de multiples couches cachées non symétriques. Leur étude s'est basée sur les ensembles de données *KDD Cup '99* et *NSL-KDD* et a pris en compte cinq (5) métriques de performance à savoir l'exactitude, la précision, le rappel, les fausses alarmes et le F1 score. Les résultats de l'évaluation sur l'ensemble de données *KDD Cup '99* montrent que le modèle proposé est en mesure d'offrir une précision de 97,85% alors que ceux sur l'ensemble de données *NSL-KDD* offrent un taux d'exactitude total de 85,42%. Un IDS utilisant les RNR a aussi été étudié par Yin et al [3] dans le contexte de la classification binaire et multi-classes sur l'ensemble de données *NSL- KDD*.

Le modèle a été testé un utilisant un nombre différent de nœuds cachés et des taux d'apprentissage différents. Les résultats ont démontré que ces 2 paramètres affectent considérablement la précision du modèle et la meilleure précision a été obtenue en utilisant 80 nœuds cachés et 0,1 comme taux d'apprentissage pour les scenarios binaires et 0,5 pour ceux multi-classes. Leur modèle a eu un plus grand succès par rapport à ceux effectués avec les algorithmes d'apprentissage automatique classiques mais présente des limites du fait du temps de calcul mais aussi et principalement du fait d'un taux de détection plus faible pour les attaques R2L (Remote to local) et U2R (User to root). Ces attaques permettant respectivement d'obtenir un accès non autorisé à une machine du réseau et de posséder illégalement les privilèges root lors de l'accès légal à une machine locale respectivement.

Ferrag et al [8] ont effectué une étude comparative d'une grande partie des approches de détection d'intrusions utilisant l'AP. Ils se sont intéressés à sept types de modèles dont les RNR, CNN, autoencodeurs. Les performances ont été jugées sur deux jeux de données dont *CSE-CIC-IDS 2018*. Il revient de cette recherche que les CNN et les autoencodeurs possèdent le plus grand taux de détection et le plus petit taux de fausses alarmes.

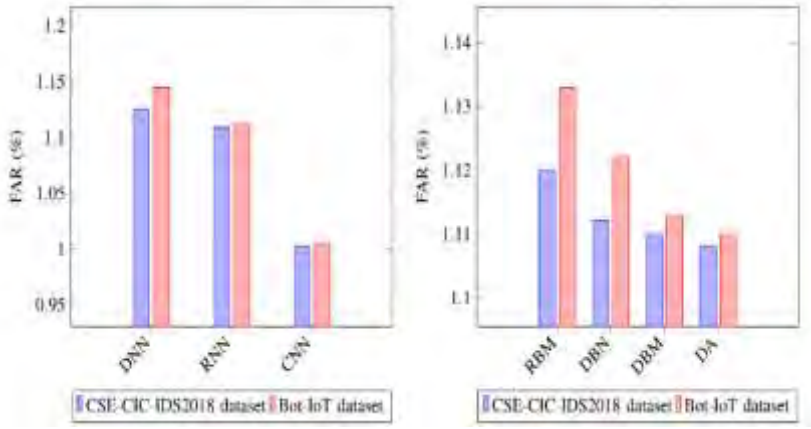


Figure 35 [8] Comparaison du taux de fausses des différents algorithmes

De plus leur étude démontre que les modèles d'apprentissage profond sont plus performants que ceux d'AA classique sur le taux de détection.

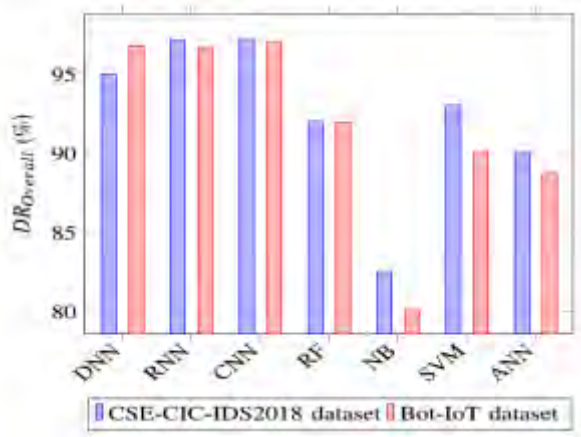


Figure 36 [8] Comparaison du taux de détection des différents algorithmes

Ainsi, les recherches faites ont démontré que les algorithmes d'apprentissage profond sont préférés à ceux d'apprentissage automatique standards. Le vaste ensemble de données et la nature profonde des algorithmes d'apprentissage profond rendront le processus d'apprentissage plus gourmand en termes de ressources de calcul et de consommation de temps mais seront plus efficaces en raison de leur capacité à apprendre à partir de grands ensembles de données sous forme brute. C'est pourquoi, au cours des trois dernières années, les chercheurs se sont concentrés sur les outils de l'apprentissage profond pour la conception du système IDS. On remarque que 60% des méthodes proposées sont purement basées sur les approches DL, 20% des solutions utilisent une approche hybride impliquant la combinaison d'algorithmes basés sur ML et DL alors que seulement 20% des solutions proposées sont basées sur des méthodes ML. L'idée clé des méthodes hybride est de tirer profit de différentes méthodes de classification de l'AA et de l'apprentissage profond. Puisque chaque algorithme a ses forces et faiblesses, certains peuvent bien fonctionner pour détecter un type spécifique d'attaques et montre des performances médiocres sur d'autres types d'attaques.
Étant donné que les algorithmes d'apprentissage profond nécessitent des ressources de calcul étendues, l'avènement des GPU et des plates-formes Cloud a facilité la mise en œuvre de ses méthodes. Nous avons observé que dans la plupart des solutions proposées, les modèles sont testés à l'aide d'anciens jeux de données. Nous avons également observé que dans certaines solutions proposées, les performances du modèle affichant des résultats extrêmement excellents pour les anciens ensembles de données sont diminuées pour les ensembles de données plus récents et récemment proposés. De plus, nous avons observé que certaines des méthodologies sont assez complexes et qu'elles nécessitent en fin de compte plus de temps d'apprentissage du modèle. Nous observons un compromis entre la complexité du modèle et la structure profonde des méthodes DL. Plus l'algorithme est profond, plus le modèle sera complexe et par conséquent, il consommera plus de temps et de ressources informatiques. Ainsi, la sélection intelligente de fonctionnalités utiles pour la formation du modèle améliorera finalement cet inconvénient.

4.3. Les jeux de données ou datasets

Les systèmes d'information sont aujourd'hui de plus en plus ouverts sur Internet. Cette ouverture, a priori bénéfique, pose néanmoins un problème majeur : il en découle un nombre croissant d'attaques. La mise en place des outils de surveillance pour auditer ces systèmes et détecter d'éventuelles intrusions, afin de renforcer la politique de sécurité, nécessite l'utilisation d'un ensemble de données représentant ces attaques. Ces bases de données nous

seront d’une aide capitale afin d’entrainer notre modèle. En effet, l’’efficacité d’un IDS alimenté par l'IA dépend fortement de la formation à l'aide d'un ensemble de données approprié.
Cette section fournit des détails sur les ensembles de données populaires utilisés dans les recherches pour tester les performances de la méthodologie proposée.

i. KDD Cup 99

C'est l'un des ensembles de données les plus populaires et les plus utilisés pour l'IDS.Il contient environ cinq et deux millions d'enregistrements pour la formation et les tests respectivement. Chaque enregistrement contient 41 caractéristiques ou attributs différents et est étiqueté comme normal ou attaque.

ii. NLS-KDD

Cet ensemble de données a été proposée par Tavallae et al [10]. Il a été conçu à partir du jeu de données KDD et apporte des améliorations par rapport ce jeu de données tels que la suppression des enregistrements en double.

iii. Kyoto 2006+

Cet ensemble de données est créé à partir des enregistrements de trafic réseau, obtenus en déployant des pots de miel, des capteurs darknet, serveurs de messagerie, robot d'exploration Web et autres mesures de sécurité réseau de l'Université de Kyoto. L’ensemble de données le plus récent comprend l'enregistrement de trafic de 2006 à 2015.

iv. UNSW-NB 15

Cet ensemble de données est créé par le Centre australien pour la cybersécurité. Il contient environ deux millions d'enregistrements avec un total de 49 fonctionnalités, qui sont extraits à l'aide de Bro-IDS, des outils Argus et de certains algorithmes récemment développés. Cet ensemble de données contient les types d'attaques nommées comme Reconnaissance, Port Scans, Backdoor, Dos,…

v. CIC-IDS 2017

Cet ensemble de données a été créé par l'Institut canadien de la cyber sécurité (CIC) en 2017. Il contient les mises à jour des flux normaux et attaques du monde réel. Le trafic réseau est analysé par CICFlowMeter à l'aide des informations basé sur les horodatages, les adresses IP source et de destination, les protocoles et les attaques. De plus, CICIDS2017 comprend des scénarios d'attaque courants tels que l'attaque par force brute, l'attaque HeartBleed, le Botnet, l'attaque par déni de service (DoS) et DDOS, attaque Web et attaque d'infiltration.

vi. CSE-CIC-IDS 2018

C’est un projet collaboratif entre le Centre de la sécurité des télécommunications (CST) et l'Institut canadien pour la cybersécurité (CIC). Il s’agit du jeu de données de détection le plus récent accessible au public et qui couvre un large éventail des types d’attaques. Il comprend sept scénarios d'attaque différents: Brute-force, Heartbleed, Botnet, Dos, DDos, attaques Web et infiltration du réseau de l'intérieur.

Chapitre 5 Développement d'un NIDS avec l'AA

Dans cette section, nous implémenterons un IDS grâce à un autoencodeur. Comme mentionné dans [8], cette architecture offre de meilleurs résultats dans la détection d'attaques et possède un plus faible taux de fausses alarmes. De allons rendre notre modèle plus complexe en utilisant l'autoencodeur débruiteur. Nous allons utiliser le jeu de données CSE-CIC-IDS2018 afin d'entrainer notre modèle.

5.1. Outils

5.1.1. Le langage Python

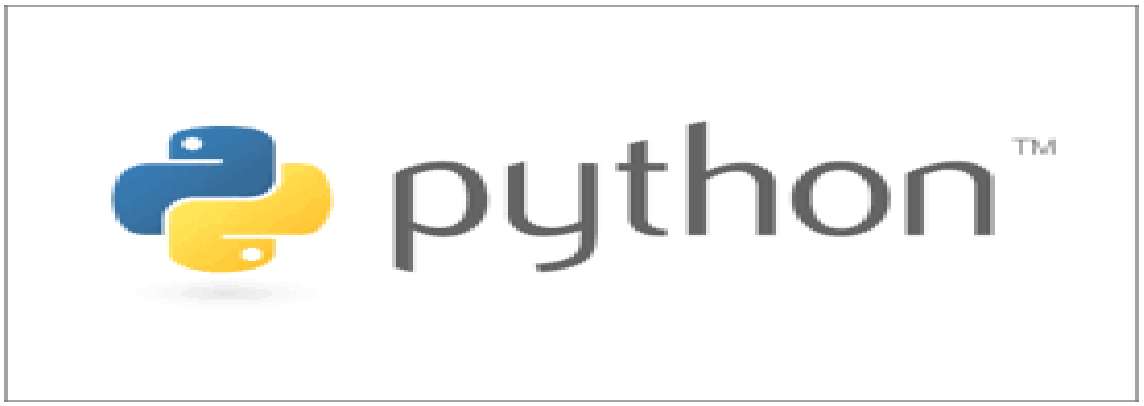


Figure 37 Langage python (python.org)

Python est un langage de programmation interprété et multiplateformes. Il favorise la programmation structurée, fonctionnelle et orientée objet. C'est un langage très apprécié du fait de sa simplicité permettant ainsi une initiation aisée aux concepts de base de la programmation.

5.1.2. Le gestionnaire de librairies Anaconda

Figure 38 gestionnaire de librairies Anaconda (anaconda.com)

Anaconda est une distribution libre et open-source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique). Ainsi en installant Anaconda, vous installerez Python, Numpy, Jupyter Notebook et des dizaines de packages scientifiques, dont certains indispensables à l'analyse de données. Pour installer Anaconda aussi bien sur Windows que sur Linux, téléchargez la distribution Anaconda correspondant à votre système d'exploitation, en Python version 3 : https://www.anaconda.com/distribution/.

Sur Windows ou sur Mac :

- Téléchargez l'installeur Windows ou macOS, et double-cliquez pour lancer l'installation.
- Répondez aux différentes questions (les options par défaut suffisent !). Une fois l'installation terminée, vous pouvez vérifier que celle-ci s'est bien passée en lançant l'application Jupyter Notebook.

Sur Linux :

- Téléchargez l'installeur pour Linux.
- Entrez l'instruction suivante dans la commande :

```
bash ~/Downloads/Anaconda3-5.3.0-Linux-x86_64.h
```

5.1.3. Framework d'intelligence artificielle

Avec l'apparition de l'intelligence artificielle, plusieurs Framework ont vu le jour et se sont trouvés être des atouts considérables pour s'affranchir de certaines taches fastidieuses lors de l'expérimentation, de l'optimisation ou de la mise en production d'un modèle d'apprentissage profond.

i. Pytorch

Figure 39 Framework PyTorch (pytorch.org)

PyTorch est une bibliothèque logicielle Python open source d'apprentissage machine qui s'appuie sur Torch développée par Facebook. PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond. PyTorch est une bonne option si vous :

- Faites des recherches ou vos exigences de production ne sont pas très exigeantes

- Vous voulez une meilleure expérience de développement et de débogage

ii. TensorFlow

Figure 40 Framework TensorFlow (tensorflow.org)

Originaire de la famille Google, Tensorflow s'avère être un framework open-source robuste qui prend en charge l'apprentissage profond. Tensorflow fonctionne sur presque toutes les principales plates-formes utilisées aujourd'hui, des mobiles aux ordinateurs de bureau, aux clusters distribués de serveurs sur le cloud ou sur site. Dans sa version 2.0, elle inclut une version de Keras permettant l'accès à de nombreuses techniques ayant trait aux réseaux neuronaux.

iii. Keras

Figure 41 Framework Keras (keras.io)

Keras est une bibliothèque open-source de haut niveau écrite en python et construite sur Theano ou Tensorflow. Elle fournit une API pour la construction de réseaux de neurones et sa première version a été publiée le 28 mars 2015.Son objectif est de permettre la construction rapide des réseaux neuronaux. Elle permet aussi l'accès aux fonctions proposées par plusieurs librairies de machine Learning telles que Tensorflow.
Dans notre mémoire, nous utiliserons le framework Tensorflow avec son API de plus haut niveau Keras. Pour l'installation :

- Aller sur Anaconda prompt
- Créer un environnement
- Exécuter la commande ' conda activate nom_environnement'
- Exécuter la commande ' pip install tensorflow

iv. La librairie Pandas

Figure 42 Librairie Pandas (pandas.pydata.org)

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pour installer Pandas :

- Aller sur Anaconda prompt
- Exécuter la commande ‘ conda activate nom_environnement’
- Exécuter la commande ‘ pip install pandas

v. Google Colabs

Google Colaboratory, souvent raccourci en "Colab", permet d'écrire et d'exécuter du code Python dans votre navigateur. Il offre les avantages suivants :

- Aucune configuration requise
- Accès gratuit aux GPU
- Partage facile
- Une ram de 12 Go

Colab permet aussi de tirer pleinement parti des bibliothèques populaires Python pour analyser et visualiser des données.

vi. Autres outils

- Numpy est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.
- Matplotlib est une bibliothèque Python capable de produire des graphes de qualité.

5.2. Méthodologie

Un IDS plus particulièrement un NIDS développé à l'aide des méthodes ML et DL implique généralement de suivre trois étapes majeures, à savoir la phase de prétraitement des données, la phase d'entrainement et la phase de test. Pour toutes les solutions proposées, l'ensemble de données est d'abord prétraité pour le transformer dans le format adapté à être utilisé par l'algorithme. Cette étape implique généralement l'encodage, la standardisation et la normalisation. Les données prétraitées sont ensuite divisées au hasard en deux parties : l'ensemble de données d'entraînement et l'ensemble de données de test. En règle générale, l'ensemble de données d'entraînement comprend près de 80% de la taille de l'ensemble de données d'origine et les 20% restants constituent l'ensemble de données de test. L'algorithme ML ou DL est ensuite entraîné à l'aide de l'ensemble de données d'apprentissage dans la phase d'apprentissage.

5.2.1. Statistiques du jeu de données

Notre jeu de données comprend ainsi 15 scenarios possibles pouvant être regroupés en 7 classes.

Catégorie	Type d'attaque	Nombre
Benign	/	13484863
Bot	/	286191
DoS	DoS attacks-Hulk	461912
	DoS-attacksSlowHTTPTest	139890
	DoS attacks-GoldenEye	41508
	DoS attacks-Slowloris	10990
DDoS	DDoS attack-HOIC	686012
	DDoS attack-LOIC-UDP	1730
	DDoS attacks-LOIC-HTTP	912035
Brute Force	Brute Force -Web	611
	Brute Force -XSS	230
	FTP-BruteForce	193360
	SSH-BruteForce	187589
Infiltration	/	161934
SQL Injection	/	87

Tableau 2 Statistiques du jeu de données CIC-CSE-IDS2018

5.2.2. Pré-traitement des données

- Supprimer les colonnes inutiles à savoir les colonnes Timestamp, Src IP, Src Port, Dst IP
- Supprimer les observations avec des données manquantes et aberrantes
- Encoder les variables catégoriques avec la méthode OneHotEncoder (sklearn)

En effet, il est nécessaire de convertir les caractéristiques catégorielles en une représentation numérique. Dans notre cas, nous observons 2 variables catégoriques à savoir les variables Dst Port et Protocol

- Normaliser les variables continues avec la technique MinMaxScaler (sklearn)
  L'objectif de la normalisation est de modifier les valeurs des colonnes numériques du jeu de données pour utiliser une échelle commune, sans que les différences de plages de valeurs ne soient faussées et sans perte d'informations.

A la fin de ce prétraitement, nous obtenons un jeu de données avec 103 colonnes.

5.2.3. Préparation des données d'entrainement et de validation

- Diviser notre jeu de données en données d'entrainement et de test
- Filtrer les paquets normaux de notre jeu de données
  En effet, puisque nous travaillerons avec l'autoencodeur, il nous faudra filtrer les observations bénignes du reste du jeu de données afin qu'il puisse apprendre, de ces observations, à distinguer une activité normale d'une attaque.
  De plus, face aux problèmes de puissances de calcul, nous prendrons un échantillon de 1.5 millions de trafics normaux pour entrainer et valider le modèle.
- Entrainer notre modèle avec les données d'entrainement

5.2.4. Création et entrainement de notre modèle

- Pour imposer plus de contraintes à notre modèle, nous utiliserons un autoencodeur débruiteur. Pour cela, nous allons fixer un dropout=0,1 et l'appliquer à l'entrée. Le dropout est une technique ou des neurones sélectionnés au hasard sont ignorés pendant l'entrainement sur une probabilité t donnée. Ceci permettra à notre modèle d'être capable d'une meilleure généralisation et d'être moins susceptible à surajuster les données d'entrainement.
  Dans Figure 43, nous avons une vue globale du modèle crée.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 102)]             0
_________________________________________________________________
dropout (Dropout)            (None, 102)               0
_________________________________________________________________
dense (Dense)                (None, 50)                5150
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 50)                550
_________________________________________________________________
dense_3 (Dense)              (None, 102)               5202
=================================================================
```

Figure 43 Vue globale du modèle

Après l'entrainement, nous évaluons le taux de fausses alarmes sur de nouveaux paquets normaux puis calculons le taux de détection pour chaque type d'attaque

5.3. Résultats

L'évolution de l'erreur de reconstruction associée à notre modèle est présentée dans Figure 44. Le seuil de l'erreur de reconstruction est de 0.0007 à la fin de l'entrainement.

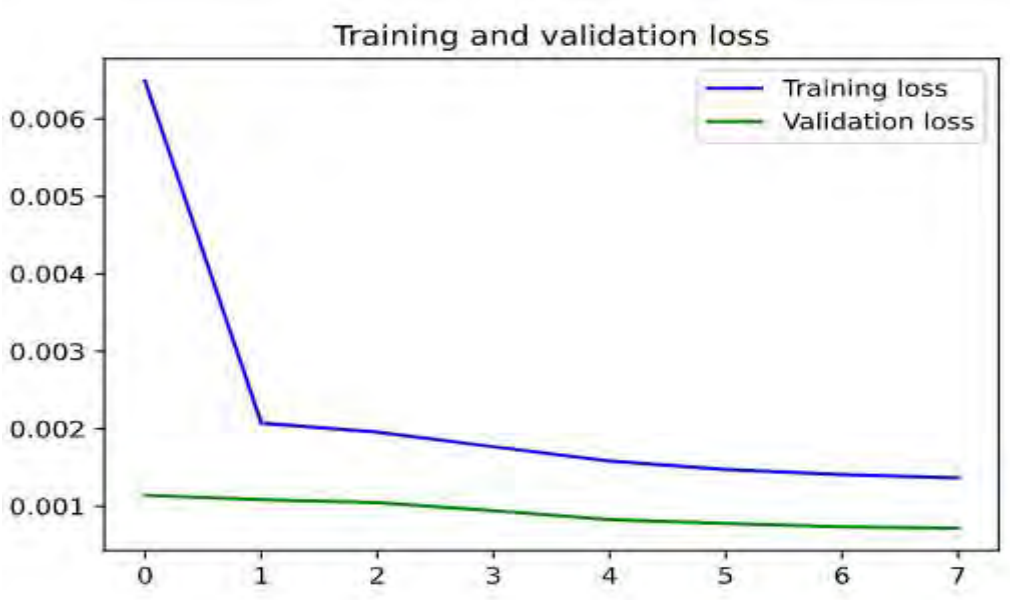


Figure 44 Évolution de l'erreur de reconstruction

- Taux de fausses alarmes

Afin d'évaluer notre modèle, nous lui fournissons les données de test (données sur lesquels il ne s'est jamais entrainé) pour déterminer s'il arrive à reconnaitre les observations normales (appartenant à la classe Benign). Le nombre d'observations fourni est de 1 million. Notre modèle entrainé obtient un taux de fausses alarmes de 11,39%.

- Taux de détection pour les attaques

Âpres avoir évalué notre modèle au niveau de la partie précédente, nous pouvons ainsi l'utiliser comme IDS afin de détecter les attaques sur le réseau.
En effet, notre encodeur sait avec une bonne précision reconnaitre une activité normale donc parallèlement reconnaitre une attaque. D'après les résultats dans Figure 45, l'algorithme développé est très efficace dans la détection de la majeure partie des types d'attaques. Les attaques telles que Bot, SSH-BruteForce, FTP-BruteForce ont été totalement détectées. Les performances sont moyennes pour les attaques BruteForce-XSS et DDoS-LOIC-HTTP. Cependant les attaques de type Infiltration et DDoS HULK ne sont pas bien détectées.

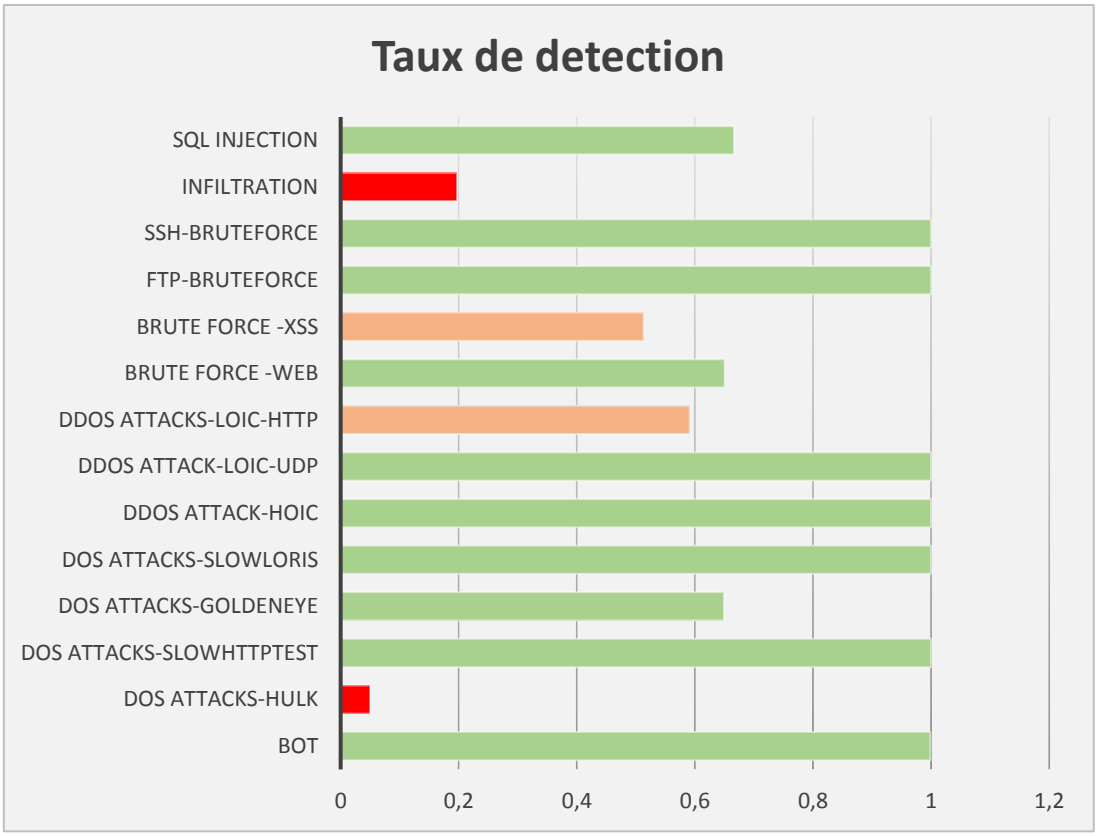


Figure 45 Taux de détection du modèle par rapport aux différentes attaques

Conclusion

La sécurité réseau reste l'un des problèmes les plus sérieux dans le domaine de l'entreprise. En effet, il ne sera jamais possible de sécuriser totalement un système d'informations, car il y aura toujours des hackers pour découvrir des nouvelles failles, mais nous pouvons toujours rendre une intrusion rapidement détectable et voire quasi impossible en exploitant de nouvelles approches. C'est dans ce cadre que les IDS ont été déployés et sont devenus en quelques années des outils essentiels pour la sécurité réseau. Cependant, ils possèdent un certain nombre de limites et inconvénients. Dans ce mémoire, nous abordons une tendance récente visant à adopter l'apprentissage automatique afin d'améliorer les performances et l'efficacité d'un IDS en termes de précision et de réduction du taux de fausses alarmes. L'apprentissage automatique permet la reconnaissance du comportement normal ou attaque. Les recherches et travaux effectués ont montré que les autoencodeurs et les réseaux de neurones convolutifs constituent les deux (2) algorithmes les plus performants. Ainsi, dans notre travail, nous avons exploré un type d'autoencodeur sur le récent jeu de données CSE-CIC-IDS2018. L'objectif principal était d'apprendre au modèle à distinguer les flux de paquets anormaux des flux normaux, en se référant à l'erreur de reconstruction. Le modèle développé donne des résultats satisfaisants et montre un véritable potentiel par rapport aux systèmes de détection d'intrusion utilisant l'approche comportementale et l'approche par signatures. En effet, il permet de détecter des attaques inconnues et comble la principale lacune de la détection d'anomalies à savoir le taux de fausses alarmes. Cependant, pour que la détection d'intrusion à l'aide de l'AA se développe, nous avons besoin d'un grand volume de données catégorisées et d'exemples d'attaques. Or, la plupart des jeux de données disponibles publiquement sont soit trop vieux, soit trop restreints. À l'heure actuelle, la disponibilité de données d'apprentissage reste le plus grand frein au développement de cette technologie. En définitive, nous avons pu constater qu'il ne s'agit pas d'une simple application de l'apprentissage automatique à la sécurité informatique, mais que ce domaine nécessite une compréhension approfondie des diverses attaques d'un point de vue technique, c'est à-dire définir les caractéristiques des jeux de données pour chaque attaque spécifique mais aussi des modèles de ML. Ainsi une forte coopération entre les experts en sécurité et en science des données est donc nécessaire afin de développer le potentiel de ces méthodes de détection.

Bibliographie

[1] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems, O'Reilly Media, 2009, p. 2019.

[2] A. Halevy, P. Norvig and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems,* pp. 8-12, 2009.

[3] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access,* Vols. 21954-21961, p. 5, 2017.

[4] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* pp. 533-536, 1986.

[5] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology.*

[6] D. P. Kingma and M. Welling, "Stochastic gradient VB and the variational auto-encoder," *In Second International Conference on Learning Representations,* 2014.

[7] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint,* 2014.

[8] M. A. Ferrag, L. Maglaras, S. Moschoyiannis and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications,* 2020.

[9] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," *In Proceedings of the 39th annual meeting of the Association for Computational Linguistics,* pp. 26-33, 2001.

[10] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *In 2009 IEEE symposium on computational intelligence for security and defense applications,* pp. 1-6, 2009.

[11] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence,* pp. 41-50, 2018.

[12] P. Cornioley, savoirs.usherbrooke.ca/bitstream/handle/11143/13397

[13] P. Vincent, L. H., Y. Bengio and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *In Proceedings of the 25th international conference on Machine learning,* pp. 1096-1103, 2008.

[14] S. Haykin and R. Lippmann, "Neural networks, a comprehensive foundation," *International journal of neural systems,* pp. 363-364, 1994.

[15] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," *In Advances in neural information processing systems,* pp. 473-479, 1997.

[16] Samuel, A. L., «Some studies in machine learning using the game of checkers,» *IBM Journal of research and development,* pp. 210-229, 1959.

[17] T. M. Mitchell, «Does machine learning really work?,» *AI magazine,* pp. 11-11, 1997.

[18] X. Gao, C. Shan, C. Hu, Z. Niu and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access,* vol. 7, pp. 82512-82521, 2019.

Annexe Codes utilisés

In [1]:
```
import os
import pickle
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras import regularizers
import matplotlib.pyplot as plt
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [3]:
```
main_path = r"../donnees"
csv_files = os.listdir(main_path)
csv_files
```

Out[3]:
```
['Friday-02-03-2018_TrafficForML_CICFlowMeter.csv',
 'Friday-16-02-2018_TrafficForML_CICFlowMeter.csv',
 'Friday-23-02-2018_TrafficForML_CICFlowMeter.csv',
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_0.csv',
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_7.csv',
 'Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv',
 'Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv',
 'Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv']
```

In [4]:
```
train_valid_files = np.random.choice(csv_files, size=8, replace=False)
test_files = [f for f in csv_files if f not in train_valid_files]
print('TRAIN :\n', train_valid_files)
print('TEST :\n', test_files)
```

```
TRAIN :
 ['Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv'
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_0.csv'
 'Friday-02-03-2018_TrafficForML_CICFlowMeter.csv'
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_7.csv'
 'Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv'
 'Friday-16-02-2018_TrafficForML_CICFlowMeter.csv'
 'Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv'
 'Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv']
TEST :
 ['Friday-23-02-2018_TrafficForML_CICFlowMeter.csv', 'Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv', 'Wednesday-14-02-2018
_TrafficForML_CICFlowMeter.csv']
```

Entrainement

In [5]:
```
list_df = []
for i, csv_file in enumerate(train_valid_files):
    path = os.path.join(main_path, csv_file)
    df = pd.read_csv(path)
    list_df.append(df)
```

In [6]:
```
df_tot = pd.concat(list_df, axis=0, ignore_index=True)
del list_df
```

In [7]:
```
# df_tot.head(5)
```

In [8]:
```
df_tot.shape
```

Out[8]:
```
(5716888, 103)
```

In [9]:
```
df_tot["Label"].unique()
```

Out[9]:
```
array(['Benign', 'DDOS attack-LOIC-UDP', 'DDOS attack-HOIC',
       'DDoS attacks-LOIC-HTTP', 'Bot', 'Infilteration',
       'DoS attacks-SlowHTTPTest', 'DoS attacks-Hulk',
       'DoS attacks-GoldenEye', 'DoS attacks-Slowloris',
       'Brute Force -Web', 'Brute Force -XSS', 'SQL Injection'],
      dtype=object)
```

filtrer les benign et les attaques

De plus, face aux problemes de puissances de calcul, nous prenons un echantillon de 1.5 millions de traffics normaux pour entrainer et valider le modele

In [10]:
```
n_observations = int(1.5e6)
df_benign = df_tot[df_tot['Label']== 'Benign']
del df_tot
```

In [11]:
```
df_benign.replace(np.inf, np.nan, inplace=True)
df_benign.dropna(inplace=True)
df_benign = df_benign.sample(n_observations)
```

In [12]:
```
df_benign.pop('Label')
training_df, testing_df = train_test_split(df_benign,test_size = 0.3)
del df_benign
```

In [13]:
```
training_df.columns
```

Out[13]:
```
Index(['Flow Duration', 'Tot Fwd Pkts', 'Tot Bwd Pkts', 'TotLen Fwd Pkts',
       'TotLen Bwd Pkts', 'Fwd Pkt Len Max', 'Fwd Pkt Len Min',
       'Fwd Pkt Len Mean', 'Fwd Pkt Len Std', 'Bwd Pkt Len Max',
       ...
       'Port_139', 'Port_443', 'Port_445', 'Port_500', 'Port_3128',
       'Port_3389', 'Port_5355', 'Port_8080', 'Port_27017', 'Port_49152'],
      dtype='object', length=102)
```

In [14]:
```
def minmax_scale_values(training_df,testing_df,col_name):
    scaler = MinMaxScaler()
    scaler = scaler.fit(training_df[col_name].values.reshape(-1, 1))
    train_values_standardized = scaler.transform(training_df[col_name].values.reshape(-1, 1))
    training_df[col_name] = train_values_standardized
    test_values_standardized = scaler.transform(testing_df[col_name].values.reshape(-1, 1))
    testing_df[col_name] = test_values_standardized
    return scaler

cat_columns = [column for column in list(training_df.columns) if column.startswith('Port') or column.startswith('Protocol')]
cont_columns = [column for column in list(training_df.columns) if column not in cat_columns]
scalers = {}
for column in cont_columns:
    scaler = minmax_scale_values(training_df,testing_df,column)
    scalers[column] = scaler
```

In [15]:
```
def getModel():

    inp = layers.Input(shape= (training_df.shape[1],) )
    #probabilite pour une colonne dune observation detre nulle
    d0= layers.Dropout(0.1)(inp)

    #couche hidden (unit etant le nombre de colonnes ou neurones dansla procahine couche)
    hidden1 = layers.Dense(50, activation='elu', kernel_regularizer=keras.regularizers.l2(0.001))(d0)

    hidden2= layers.Dense(10,activation='elu')(hidden1)
    #hidden3=Layers.Dense(10,activation='elu')(hidden2)


    decoded1 =layers.Dense(50, activation='elu')(hidden2)
    decoded2 =layers.Dense(training_df.shape[1], activation=None)(decoded1)
   # decoded3 =Layers.Dense(training_df.shape[1], activation=None)(decoded2)

    #dessiner le model inp representant lentree et decode2 le inp decode par lautoencodeur
    autoencoder = Model(inputs=inp, outputs=decoded2)
    optimizer = keras.optimizers.Adam(0.0001)
    autoencoder.compile(optimizer=optimizer, loss='mean_squared_error')
    return autoencoder
```

In [16]:
```
autoencodeur = getModel()
```

In [17]:
```
autoencodeur.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 102)]             0
_________________________________________________________________
dropout (Dropout)            (None, 102)               0
_________________________________________________________________
dense (Dense)                (None, 50)                5150
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 50)                550
_________________________________________________________________
dense_3 (Dense)              (None, 102)               5202
=================================================================
Total params: 11,412
Trainable params: 11,412
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```
x_train = training_df.values
x_test = testing_df.values
del training_df
del testing_df
```

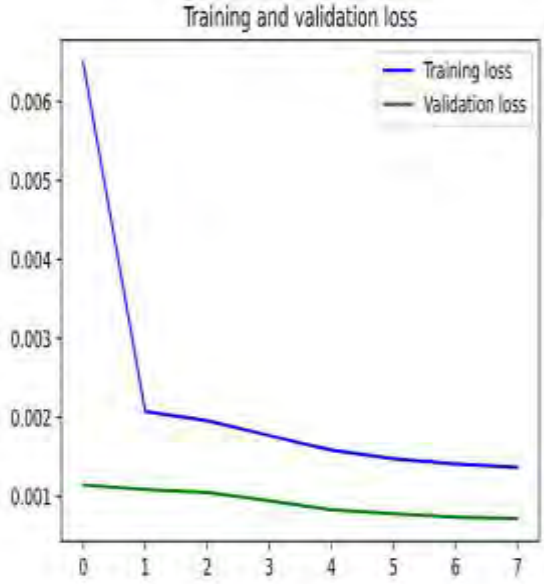
entrainons notre modele

In [19]:
```
history=autoencodeur.fit(x_train,x_train,
                epochs=8,
                batch_size=50,
                shuffle=False,
                validation_data=(x_test,x_test),
                verbose=1)
```

```
Epoch 1/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0065 - val_loss: 0.0011
Epoch 2/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0021 - val_loss: 0.0011
Epoch 3/8
21000/21000 [==============================] - 33s 2ms/step - loss: 0.0020 - val_loss: 0.0010
Epoch 4/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0018 - val_loss: 9.4340e-04
Epoch 5/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0016 - val_loss: 8.2913e-04
Epoch 6/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0015 - val_loss: 7.7636e-04
Epoch 7/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0014 - val_loss: 7.3455e-04
Epoch 8/8
21000/21000 [==============================] - 34s 2ms/step - loss: 0.0014 - val_loss: 7.1701e-04
```

In [20]:
```
def plot_train_history(history, title):
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  epochs = range(len(loss))
  plt.figure()
  plt.plot(epochs, loss, 'b', label='Training loss')
  plt.plot(epochs, val_loss, 'g', label='Validation loss')
  plt.title(title)
  plt.legend()
  plt.show()

plot_train_history(history,
                   'Training and validation loss')
```



In [21]:
```
threshold = history.history['val_loss'][-1]
print(threshold)
```

```
0.0007170071476139128
```

In [22]:
```
del x_train
del x_test
```

Evaluation : Calculons le taux de fausses alarmes

nous allons utiliser des traffics normaux jamais vu par le modele jusqu'a present

In [23]:
```
list_df = []
main_path = '../donnees'
for i, csv_file in enumerate(test_files):
    path = os.path.join(main_path, csv_file)
    df = pd.read_csv(path)
    list_df.append(df)
```

In [24]:
```
df_tot = pd.concat(list_df, axis=0, ignore_index=True)
del list_df
```

In [25]:
```
df_benign = df_tot[df_tot['Label']== 'Benign']
del df_tot
```

In [26]:
```
df_benign.replace(np.inf, np.nan, inplace=True)
df_benign.dropna(inplace=True)
n_observations_eval = int(1e6)
if df_benign.shape[0] > n_observations_eval:
    df_benign = df_benign.sample(n_observations_eval)
```

In [27]:
```
def apply_scaling(dataframe, scalers):
    for col_name, scaler in scalers.items():
        values_standardized = scaler.transform(dataframe[col_name].values.reshape(-1, 1))
        dataframe[col_name] = values_standardized
    return dataframe
```

In [28]:
```
def calculate_losses(X,preds):
    losses=np.zeros(len(X))
    for i in range(len(X)):
        losses[i]=((preds[i] - X[i]) ** 2).mean(axis=None)
    return losses
```

In [29]:
```
df_benign = apply_scaling(df_benign, scalers)
y = df_benign.pop('Label')
X=df_benign.values
y0 = np.zeros(len(y))
```

In [30]:
```
testing_set_predictions=autoencodeur.predict(X)
test_losses=calculate_losses(X,testing_set_predictions)
testing_set_predictions =np.zeros(len(test_losses))
testing_set_predictions[np.where(test_losses>threshold)]=1
```

In [31]:
```
test_losses
```

Out[31]:
```
array([2.43255003e-05, 5.02333242e-05, 1.71471272e-05, ...,
       7.12727284e-05, 2.61566442e-04, 3.74081725e-05])
```

In [32]:
```
false_alarm_rate = len(np.where(testing_set_predictions==1)[0]) / len(testing_set_predictions)
```

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

In [33]:
```
false_alarm_rate
```

Out[33]:
```
0.113897
```

In [34]:
```
df_benign.shape
```

Out[34]:
```
(1000000, 102)
```

Exportons le modeles et les normaliseurs afin de mesurer la performance sur les attaques

In [35]:
```
import os
from datetime import datetime
suffix = datetime.now().strftime("%m_%d_%Y_%H_%M")
folder_name = 'resultats_' + suffix
# cree un repertoire pour le model actuel
print(folder_name)
#folder_name = os.path.join('/content/drive/MyDrive/codes/models', folder_name)
os.makedirs(folder_name)
# sauvegarde du modele
autoencodeur.save(os.path.join(folder_name, 'autoencodeur'))
# sauvegarde des normaliseurs
with open(os.path.join(folder_name, 'scalers.pkl'), 'wb') as f:
    pickle.dump(scalers, f)
```

```
resultats_01_27_2021_21_38
WARNING:tensorflow:From C:\Users\BiexLab\anaconda3\envs\training2\lib\site-packages\tensorflow\python\ops\resource_variable_op
s.py:1817: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecat
ed and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: resultats_01_27_2021_21_38\autoencodeur\assets
```

In [1]:
```
import os
import pickle
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras import regularizers
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,accuracy_score,recall_score,precision_score,f1_score
```

In [10]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

In [2]:
```
def calculate_losses(x,preds):
    losses=np.zeros(len(x))
    for i in range(len(x)):
        losses[i]=((preds[i] - x[i]) ** 2).mean(axis=None)

    return losses


threshold=0.000717
```

In [3]:
```
def apply_scaling(dataframe, scalers):
    for col_name, scaler in scalers.items():
        values_standardized = scaler.transform(dataframe[col_name].values.reshape(-1, 1))
        dataframe[col_name] = values_standardized
    return dataframe
```

A - Chargement du modele et des scalers

In [4]:
```
folder_name = '/content/drive/MyDrive/codes/models/resultats_01_27_2021_16_13'
folder_name = 'resultats_01_27_2021_21_38'
autoencodeur = keras.models.load_model(os.path.join(folder_name, 'autoencodeur'))
with open(os.path.join(folder_name, 'scalers.pkl'), 'rb') as f:
  scalers = pickle.load(f)
```

Dossier des donnees

In [5]:
```
main_path = r"/content/drive/MyDrive/codes/donnees"
main_path = r'./donnees'
csv_files = os.listdir(main_path)
csv_files
```

Out[5]:
```
['Friday-02-03-2018_TrafficForML_CICFlowMeter.csv',
 'Friday-16-02-2018_TrafficForML_CICFlowMeter.csv',
 'Friday-23-02-2018_TrafficForML_CICFlowMeter.csv',
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_0.csv',
 'Thuesday-20-02-2018_TrafficForMl_CICFlowMeter_7.csv',
 'Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv',
 'Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv',
 'Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv',
 'Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv']
```

In [ ]:

B - Evaluation sur les attaques

In [6]:
```
list_df = []
for i, csv_file in enumerate(csv_files):
    path = os.path.join(main_path, csv_file)
    df = pd.read_csv(path)
    list_df.append(df)
```

In [7]:
```
df_tot = pd.concat(list_df, axis=0, ignore_index=True)
del list_df
```

In [37]:
```
df_tot['Label'].unique()
```

Out[37]:
```
array(['Benign', 'Bot', 'DoS attacks-SlowHTTPTest', 'DoS attacks-Hulk',
       'Brute Force -Web', 'Brute Force -XSS', 'SQL Injection',
       'DDoS attacks-LOIC-HTTP', 'Infilteration', 'DoS attacks-GoldenEye',
       'DoS attacks-Slowloris', 'FTP-BruteForce', 'SSH-Bruteforce',
       'DDOS attack-LOIC-UDP', 'DDOS attack-HOIC'], dtype=object)
```

1 Attaques Bot

In [18]:
```
df_bot = df_tot[df_tot['Label']== 'Bot']
df_bot.replace(np.inf, np.nan, inplace=True)
df_bot.dropna(inplace=True)
```

In [19]:
```
df_bot = apply_scaling(df_bot,scalers)
y = df_bot.pop('Label')
x=df_bot.values
y0 = np.ones(len(y))
testing_set_predictions_bot=autoencodeur.predict(x)
test_losses=calculate_losses(x,testing_set_predictions_bot)
testing_set_predictions_bot =np.zeros(len(test_losses))
testing_set_predictions_bot[np.where(test_losses>threshold)]=1
```

Calcul du score

In [20]:
```
rate= len(np.where(testing_set_predictions_bot==1)[0]) / len(testing_set_predictions_bot)
print("metriques de performance sur le testing_df \n")
print(" Detection rate (Precision) : {} ".format(rate))
```

metriques de performance sur le testing_df

Detection rate (Precision) : 0.9990847558639478

In [21]:
```
print(f"{len(np.where(testing_set_predictions_bot==0)[0])} attaques classees comme normales parmi {len(testing_set_predictions_b
ot)} attaques")
```

258 attaques classees comme normales parmi 281892 attaques

2 Attaques DoS attacks-SlowHTTPTest

In [22]:
```
df_dosslowhttp = df_tot[df_tot['Label']== 'DoS attacks-SlowHTTPTest']
df_dosslowhttp.replace(np.inf, np.nan, inplace=True)
df_dosslowhttp.dropna(inplace=True)
#del df_tot
```

In [23]:
```
df_dosslowhttp = apply_scaling(df_dosslowhttp,scalers)

y_dosslowhttp = df_dosslowhttp.pop('Label')
x_dosslowhttp=df_dosslowhttp.values
y0_dosslowhttp = np.ones(len(y_dosslowhttp))

testing_set_predictions_dosslowhttp=autoencodeur.predict(x_dosslowhttp)
test_losses_dosslowhttp=calculate_losses(x_dosslowhttp,testing_set_predictions_dosslowhttp)
testing_set_predictions_dosslowhttp =np.zeros(len(test_losses_dosslowhttp))
testing_set_predictions_dosslowhttp[np.where(test_losses_dosslowhttp>threshold)]=1
```

CALCUL DU SCORE

In [24]:
```
rate= len(np.where(testing_set_predictions_dosslowhttp==1)[0]) / len(testing_set_predictions_dosslowhttp)
print("metriques de performance sur le testing_df \n")
print(" Detection rate (Precision) : {} ".format(rate))
```

```
metriques de performance sur le testing_df

 Detection rate (Precision) : 1.0
```

3 Attaques DoS attacks-Hulk

In [25]:
```
df_doshulk = df_tot[df_tot['Label']== 'DoS attacks-Hulk']
df_doshulk.replace(np.inf, np.nan, inplace=True)
df_dosnulk.dropna(inplace=True)
#del df_tot
```

In [26]:
```
df_doshulk = apply_scaling(df_doshulk,scalers)

y_doshulk = df_doshulk.pop('Label')
x_doshulk=df_doshulk.values
y0_doshulk = np.ones(len(y_doshulk))

testing_set_predictions_doshulk=autoencodeur.predict(x_doshulk)
test_losses_doshulk=calculate_losses(x_doshulk,testing_set_predictions_doshulk)
testing_set_predictions_doshulk =np.zeros(len(test_losses_doshulk))
testing_set_predictions_doshulk[np.where(test_losses_doshulk>threshold)]=1
```

CALCUL DE LA PRECISION

In [27]:
```
rate= len(np.where(testing_set_predictions_doshulk==1)[0]) / len(testing_set_predictions_doshulk)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
 Detection rate (Precision) : 0.05543047160498103
```

In [35]:
```
len(np.where(testing_set_predictions_doshulk==1)[0]), len(testing_set_predictions_doshulk)
```

Out[35]:
```
(25604, 461912)
```

4 Attaques FTP-BruteForce

In [28]:
```
df_ftpbrut = df_tot[df_tot['Label']== 'FTP-BruteForce']
df_ftpbrut.replace(np.inf, np.nan, inplace=True)
df_ftpbrut.dropna(inplace=True)
```

In [29]:
```
df_ftpbrut = apply_scaling(df_ftpbrut,scalers)

y_ftpbrut = df_ftpbrut.pop('Label')
x_ftpbrut=df_ftpbrut.values
y0_ftpbrut = np.ones(len(y_ftpbrut))

testing_set_predictions_ftpbrut=autoencodeur.predict(x_ftpbrut)
test_losses_ftpbrut=calculate_losses(x_ftpbrut,testing_set_predictions_ftpbrut)
testing_set_predictions_ftpbrut =np.zeros(len(test_losses_ftpbrut))
testing_set_predictions_ftpbrut[np.where(test_losses_ftpbrut>threshold)]=1
```

In [30]:
```
rate= len(np.where(testing_set_predictions_ftpbrut==1)[0]) / len(testing_set_predictions_ftpbrut)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 1.0
```

5 Attaques SSH-Bruteforce

In [40]:
```
df_sshbrut = df_tot[df_tot['Label']== 'SSH-Bruteforce']
df_sshbrut.replace(np.inf, np.nan, inplace=True)
df_sshbrut.dropna(inplace=True)
#del df_tot
```

In [41]:
```
df_sshbrut = apply_scaling(df_sshbrut,scalers)

y_sshbrut = df_sshbrut.pop('Label')
x_sshbrut=df_sshbrut.values
y0_sshbrut = np.ones(len(y_sshbrut))

testing_set_predictions_sshbrut=autoencodeur.predict(x_sshbrut)
test_losses_sshbrut=calculate_losses(x_sshbrut,testing_set_predictions_sshbrut)
testing_set_predictions_sshbrut =np.zeros(len(test_losses_sshbrut))
testing_set_predictions_sshbrut[np.where(test_losses_sshbrut>threshold)]=1
```

In [42]:
```
rate= len(np.where(testing_set_predictions_sshbrut==1)[0]) / len(testing_set_predictions_sshbrut)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 1.0
```

6 Attaques DDOS attack-LOIC-UDP

In [44]:
```
df_dosudp = df_tot[df_tot['Label']== 'DDOS attack-LOIC-UDP']
df_dosudp.replace(np.inf, np.nan, inplace=True)
df_dosudp.dropna(inplace=True)
```

In [45]:
```
df_dosudp = apply_scaling(df_dosudp,scalers)

y_dosudp = df_dosudp.pop('Label')
x_dosudp=df_dosudp.values
y0_dosudp = np.ones(len(y_dosudp))

testing_set_predictions_dosudp=autoencodeur.predict(x_dosudp)
test_losses_dosudp=calculate_losses(x_dosudp,testing_set_predictions_dosudp)
testing_set_predictions_dosudp =np.zeros(len(test_losses_dosudp))
testing_set_predictions_dosudp[np.where(test_losses_dosudp>threshold)]=1
```

In [46]:
```
rate= len(np.where(testing_set_predictions_dosudp==1)[0]) / len(testing_set_predictions_dosudp)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 1.0
```

7 Attaques DDOS attack-HOIC

In [47]:
```
df_doshoic = df_tot[df_tot['Label']== 'DDOS attack-HOIC']
df_doshoic.replace(np.inf, np.nan, inplace=True)
df_doshoic.dropna(inplace=True)
#del df_tot
```

In [48]:
```
df_doshoic = apply_scaling(df_doshoic,scalers)

y_doshoic = df_doshoic.pop('Label')
x_doshoic=df_doshoic.values
y0_doshoic = np.ones(len(y_doshoic))

testing_set_predictions_doshoic=autoencodeur.predict(x_doshoic)
test_losses_doshoic=calculate_losses(x_doshoic,testing_set_predictions_doshoic)
testing_set_predictions_doshoic=np.zeros(len(test_losses_doshoic))
testing_set_predictions_doshoic[np.where(test_losses_doshoic>threshold)]=1
```

In [49]:
```
rate= len(np.where(testing_set_predictions_doshoic==1)[0]) / len(testing_set_predictions_doshoic)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 1.0
```

8 Attaques DDoS attacks-LOIC-HTTP

In [50]:
```
df_doshoichttp = df_tot[df_tot['Label']== 'DDoS attacks-LOIC-HTTP']
df_doshoichttp.replace(np.inf, np.nan, inplace=True)
df_doshoichttp.dropna(inplace=True)
```

In [51]:
```
df_doshoichttp = apply_scaling(df_doshoichttp,scalers)

y_doshoichttp = df_doshoichttp.pop('Label')
x_doshoichttp=df_doshoichttp.values
y0_doshoichttp = np.ones(len(y_doshoichttp))

testing_set_predictions_doshoichttp=autoencodeur.predict(x_doshoichttp)
test_losses_doshoichttp=calculate_losses(x_doshoichttp,testing_set_predictions_doshoichttp)
testing_set_predictions_doshoichttp=np.zeros(len(test_losses_doshoichttp))
testing_set_predictions_doshoichttp[np.where(test_losses_doshoichttp>threshold)]=1
```

In [52]:
```
rate= len(np.where(testing_set_predictions_doshoichttp==1)[0]) / len(testing_set_predictions_doshoichttp)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.5916757580575307
```

9 Attaques Infilteration

In [53]:
```
df_infiltration = df_tot[df_tot['Label']== 'Infilteration']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)
#del df_tot
```

In [54]:
```
df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [55]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.19690324651013533
```

10 SQL Injection

10 SQL Injection

In [8]:
```
df_infiltration = df_tot[df_tot['Label']== 'SQL Injection']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)

df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [9]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.6666666666666666
```

11 Brute Force -Web

In [10]:
```
df_infiltration = df_tot[df_tot['Label']== 'Brute Force -Web']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)

df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [11]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.6508196721311476
```

12 Brute Force -XSS

In [12]:
```
df_infiltration = df_tot[df_tot['Label']== 'Brute Force -XSS']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)

df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [13]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.5130434782608696
```

13 DoS attacks-GoldenEye

In [14]:
```
df_infiltration = df_tot[df_tot['Label']== 'DoS attacks-GoldenEye']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)

df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [15]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 0.6498265394622723
```

14 DoS attacks-Slowloris

In [16]:
```
df_infiltration = df_tot[df_tot['Label']== 'DoS attacks-Slowloris']
df_infiltration.replace(np.inf, np.nan, inplace=True)
df_infiltration.dropna(inplace=True)

df_infiltration = apply_scaling(df_infiltration,scalers)

y_infiltration = df_infiltration.pop('Label')
x_infiltration=df_infiltration.values
y0_infiltration= np.ones(len(y_infiltration))

testing_set_predictions_infiltration=autoencodeur.predict(x_infiltration)
test_losses_infiltration=calculate_losses(x_infiltration,testing_set_predictions_infiltration)
testing_set_predictions_infiltration=np.zeros(len(test_losses_infiltration))
testing_set_predictions_infiltration[np.where(test_losses_infiltration>threshold)]=1
```

In [17]:
```
rate= len(np.where(testing_set_predictions_infiltration==1)[0]) / len(testing_set_predictions_infiltration)
print(" Detection rate (Precision) : {} ".format(rate))
```

```
Detection rate (Precision) : 1.0
```