Détection d'intrusions dans les objets connectés par des techniques d'apprentissage automatique : étude dans les domaines de l'éducation et des voitures connectées

Arnaud Rosay

COLLEGE DOCTORAL PAYS DE LA LOIRE
EDUCATION, COGNITION, LANGAGES, INTERACTIONS, SANTE

Le Mans Université

THÈSE DE DOCTORAT EN INFORMATIQUE

LE MANS UNIVERSITÉ

ÉCOLE DOCTORALE N° 603
Éducation, Cognition, Langages, Interactions, Santé
Spécialité : *Informatique*

Par
Arnaud ROSAY

Détection d'intrusions dans les objets connectés par des techniques d'apprentissage automatique

Étude dans les domaines de l'éducation et des voitures connectées

Thèse présentée et soutenue à Le Mans, le 5 décembre 2022
Unité de recherche : CREN – Centre de Recherche en Éducation de Nantes
Thèse N° : 2022LEMA1044

Rapporteurs avant soutenance :

Sébastien PILLEMENT	Professeur des Universités, Nantes Université, IETR
Gilles Grimaud	Professeur des Universités, Université de Lille, CRIStAL

Composition du Jury :

Président :	Sébastien PILLEMENT	Professeur des Universités, Nantes Université, IETR
Examinateurs :	Isabelle CHRISMENT	Professeur des Universités, Université de Lorraine, LORIA
	Ludovic MÉ	Habilité à diriger des recherches, Cybersécurité INRIA
Directeur de thèse :	Pascal LEROUX	Professeur des Universités, Le Mans Université
Encadrant de thèse :	Florent CARLIER	Maître de conférences, Le Mans Université

Invité(s) :

Franck GENDROT Directeur adjoint du site STMicroelectronics du Mans

Remerciements

À Léonie et Augustin,
Ne baissez jamais les bras, osez et réalisez vos rêves.

Préparer un doctorat était un rêve de longue date, que j'ai mûri durant des années avant de rendre ce projet concret. Mes motivations résultent d'un savant mélange de curiosité intellectuelle, de volonté de développer de nouvelles compétences complétant mon expérience d'ingénieur mais aussi de montrer qu'à l'approche des cinquante ans, il est encore temps d'oser, de surmonter les obstacles et d'atteindre un objectif personnel. Le chemin n'a pas été des plus simples et je n'aurais pas pu réaliser ce rêve sans l'aide de personnes auxquelles je tiens à exprimer toute ma gratitude.

Tout d'abord, je tiens à remercier Hakim Saadane et Sébastien Le Nours qui m'ont expliqué ce qu'était une thèse, m'ont encouragé à prendre ce chemin et ont nourri ma réflexion jusqu'à ce que je décide de me lancer dans cette aventure.

Je remercie également Pascal Leroux et Florent Carlier pour leur encadrement, les conseils qu'ils m'ont donnés tout au long de ce travail de thèse ainsi que le laboratoire du CREN pour m'avoir accueilli durant trois années.

Je remercie l'entreprise STMicroelectronics, en particulier Franck Gendrot, Antonio Radaelli et Carlo Bagnoli. Mes managers ont adhéré au projet quand je leur ai proposé et m'ont permis de prendre 20 % de mon temps de travail pour mener les travaux de recherche en parallèle de mon rôle d'architecte système sur les produits électroniques pour l'automobile.

Je remercie Eloïse Cheval pour toute l'aide qu'elle a apportée au projet, pour ses contributions de codage et de co-rédaction sur deux publications. Au-delà des aspects techniques, j'ai apprécié nos discussions sur des sujets plus personnels et son soutien moral m'a été d'une grande aide. Ce qui a commencé par un stage se poursuit aujourd'hui par une amitié aussi belle qu'inattendue.

Pour finir, je tiens à remercier mes proches, Agnès, Léonie et Augustin qui ont accepté de sacrifier une importante partie du temps que nous aurions pu passer en famille afin de me permettre de travailler sur ce projet de recherche durant trois ans. Beaucoup de week-ends et de jours de congés se sont transformés en journées de travail intense. Leur soutien a été essentiel pour la réussite de ce projet.

Table des matières

TABLE DES FIGURES

Liste des tableaux

Introduction

« L'histoire humaine n'est qu'un effort incessant d'invention, et la perpétuelle évolution est une perpétuelle création. »

Jean Jaurès
Homme politique (1859 – 1914)

Motivations

L'activité inventive est une caractéristique de l'humanité depuis des millénaires. L'Homme a toujours essayé de trouver des solutions aux problèmes rencontrés à chaque époque. Parmi ces inventions, les technologies du numérique, avec les ordinateurs et l'informatique, ont marqué les XX^{e} et XXI^{e} siècles. Depuis leur création, ces technologies révolutionnent nos vies par de nouveaux moyens de communication, des capacités de calculs accrues pour faire progresser la recherche et la science, de nouvelles façons de produire dans les entreprises. Le domaine de la santé n'est pas en reste avec la possibilité d'opérer des patients à distance ou la mise en place de dispositifs délivrant des substances actives tout au long de la journée en limitant l'impact sur la vie de tous les jours. Enfin, dans les transports, la numérisation nous emmène vers des moyens de déplacement autonome. Ces exemples montrent l'étendue de la révolution permise par les technologies du numérique.

À travers l'histoire, nous avons appris que la mise au point de solutions techniques et technologiques n'apportent pas seulement des solutions, mais crée aussi de nouveaux problèmes. Le domaine du numérique ne fait pas exception. Le premier virus informatique, en fait un ver nommé *Creeper*, a vu le jour en 1971 sous les doigts de Robert Thomas pour se propager de machine en machine. Le premier anti-virus est créé dans la foulée par Ray Tomlinson avec un programme appelé *Reaper*, qui tente d'effacer *Weeper* lorsqu'il le trouve (Perwej et al., 2021). Ceci n'était que le début d'une part d'une longue série de programmes toujours plus sophistiqués pour se répandre, provoquer des dégâts et former les cyberattaques et d'autre part de solutions pour empêcher ces attaques par les techniques de cybersécurité.

L'informatique a évolué avec une généralisation des fonctions de communications grâce à une miniaturisation au niveau matériel. Les circuits intégrés ont utilisé des technologies de gravure de plus en plus fines. Les transistors devenant plus petits, il est devenu possible de faire des circuits avec une puissance de calculs plus importante, tout en réduisant leur consommation électrique. Ces innovations technologiques ont donné naissance aux objets connectés et intelligents, communiquant par un réseau Internet of Things (IoT), qui se retrouvent partout, dans tous les domaines. Il s'agit par exemple de la montre connectée, du téléviseur relié à l'internet pour bénéficier des services de vidéo à la demande, de la

caméra de vidéosurveillance sur la façade d'une maison ou encore des équipements de fabrication dans les usines. La liste peut être déclinée à l'envie. Il existe une catégorie particulière de produits connectés qui ne sont pas tout de suite identifiés comme tels. Un véhicule est un type spécifique d'objet connecté qui embarque un modem cellulaire pour les appels d'urgence et devient ainsi une voiture connectée. Ceci est obligatoire en Europe depuis le 1er avril 2018 pour toutes les voitures de tourisme et les véhicules utilitaires légers (European Parliament, 2015).

Autant les cyberattaques sont connues par beaucoup sur les ordinateurs branchés en réseau, autant les menaces sur les objets connectés le sont moins. Les vulnérabilités sont susceptibles d'être utilisées pour prendre le contrôle d'un appareil, pour le dérouter de ses tâches normales ou encore pour récupérer des informations sensibles. En 2016, une attaque a été lancée simultanément depuis plusieurs dizaines de millions d'objets connectés, préalablement infectés par un logiciel malveillant appelé *Mirai* afin d'inonder Dyn, un fournisseur de service DNS. Cette attaque en déni de service distribué a bloqué l'accès à de nombreux sites webs nord-américains. Le manque de mise en œuvre de mesures de cybersécurité conjugué avec une quantité croissante d'objets connectés conduit les hackeurs à porter une attention particulière à ces objets.

Parmi les avancées de ces dernières années figurent les progrès de l'intelligence artificielle et plus particulièrement de l'apprentissage profond. Si les réseaux de neurones sont connus depuis de nombreuses années, les développements les plus importants ont lieu seulement depuis les vingt dernières années. Ces techniques, qui consistent à apprendre à partir des données, ne sont possibles qu'en ayant de capacités de calculs conséquentes et une grande quantité de données disponibles. Ces deux conditions sont réunies grâce à des circuits intégrés contenant des processeurs de haute performance et des accélérateurs matériels, d'une part, et à une quantité de données générées par l'activité humaine estimée à 59 Zo (1 Zo = 10^{21} octets) pour 2020 (Völske et al., 2021), d'autre part. Les services en ligne permettant le partage d'images ont largement contribué au développement des techniques d'apprentissage profond sur les images et les vidéos. Les fonctionnalités de reconnaissance ou d'analyse d'images assistée par ordinateur se retrouve aujourd'hui dans des processeurs dédiés aux systèmes embarqués et aux objets connectés.

On observe, d'une part, que les objets connectés contiennent de plus en plus souvent des accélérateurs pour des traitements liés à l'apprentissage machine, et d'autre part, que ces objets deviennent des cibles de cyberattaques. L'étude et la mise en œuvre de la cybersécurité dans les objets connectés en utilisant des méthodes d'intelligence artificielle représente un enjeu pertinent.

Contexte de recherche

Ces travaux de recherche s'inscrivent dans le cadre d'un contrat de collaboration entre la société STMicroelectronics et le Centre de Recherche en Éducation de Nantes (CREN).

STMicroelectronics est un des leaders européen des semi-conducteurs et adresse les trois domaines que sont l'internet des objets et la 5G, la puissance et l'énergie, la mobilité intelligente afin de servir quatre marchés : l'automobile, l'industrie, les équipements électroniques personnels et de communication et enfin les ordinateurs et périphériques.

Le CREN, dont les activités sont réparties entre les sites universitaires de Nantes et du Mans, mène des recherches pluridisciplinaires couvrant différents thèmes. L'un deux est intitulé Conception de formation et médiation par le numérique et comprend un volet en informatique.

C'est au travers de ce thème que le centre de recherche et STMicroelectronics ont établi leur collaboration afin d'étudier les thématiques mêlant cybersécurité, objets connectés et apprentissage automatique jusqu'à leur mise en œuvre pratique. J'ai mené ces travaux sous la direction de Pascal Leroux, encadré par Florent Carlier et avec l'aide de stagiaires qui ont notamment contribué à des tâches de développement logiciel. Ce sont les seules personnes impliquées et cette thèse est dans une très large proportion le fruit de mon travail personnel.

Positionnement des travaux

Deux aspects sont abordés. Tout d'abord, j'évoque les objets connectés et les problématiques de cybersécurité. Ensuite, j'aborde le domaine de l'apprentissage automatique et sa possible utilisation pour détecter des cyberattaques vers les objets connectés.

Objets connectés et cybersécurité

Comme le montre la Figure 1, la croissance rapide ainsi que les progrès technologiques des objets connectés ont considérablement augmenté la tendance à connecter en réseau toute sorte de dispositifs intelligents de la vie de tous les jours.

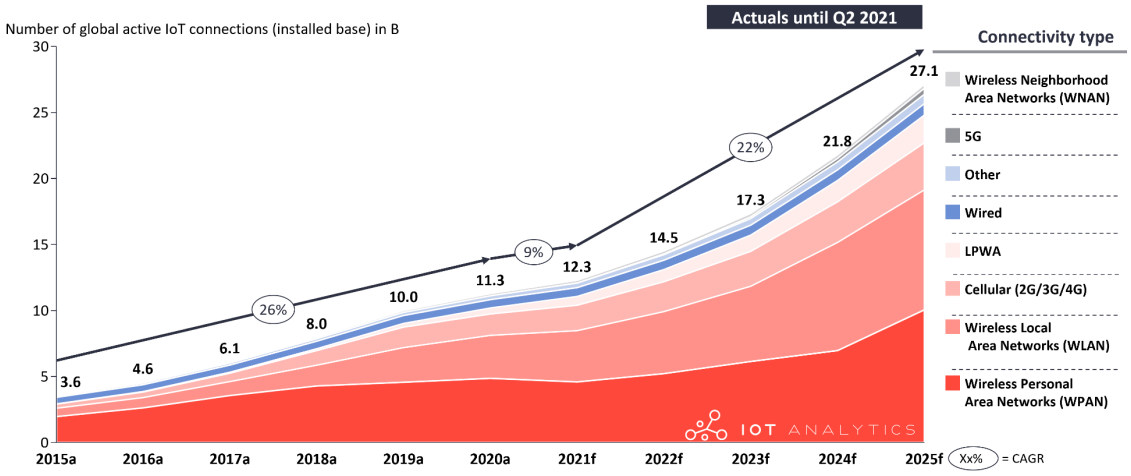


Figure 1 – Évolution du marché de l'IoT (Van, 2021).

L'internet des objets est un réseau composé de ces objets intelligents, adressables de manière unique et qui fonctionne avec une interaction humaine minimale. Une telle connectivité permet aux systèmes intelligents de collecter des informations du monde physique, de les traiter et de prendre des décisions agissant sur ce monde réel. L'IoT permet une gestion efficace des ressources, une productivité accrue et une meilleure qualité de vie pour les populations humaines au travers d'environnements intelligents.

La FIGURE 2 illustre différents objets intelligents contribuant à de tels environnements.

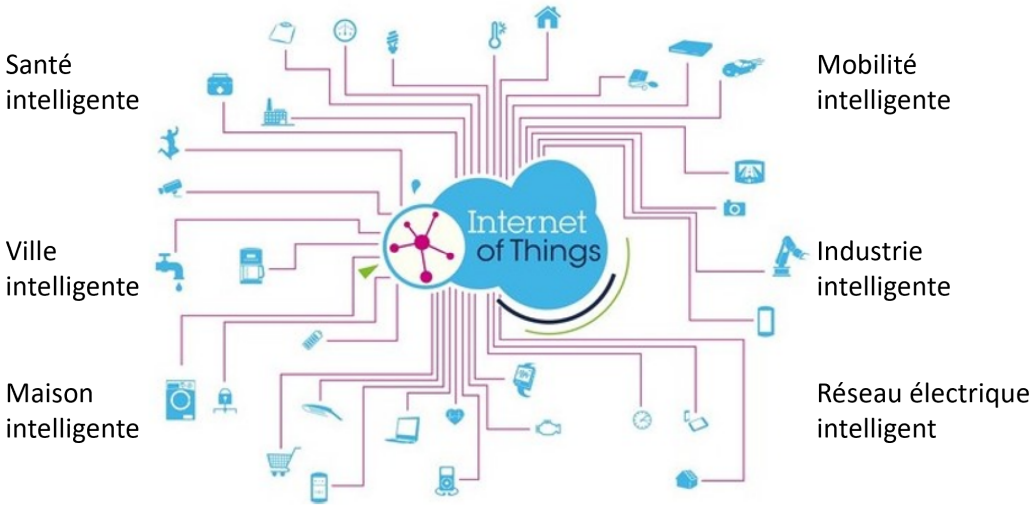


FIGURE 2 – Exemple d'environnements intelligents.

La pluralité de ces environnements est la source d'une grande diversité d'appareils connectés, avec des contraintes variables. Beaucoup d'objets connectés de la maison intelligente sont connectés à l'internet via une ou plusieurs passerelles, filaire ou sans fil, par exemple en Ethernet ou en Wifi. Leur capacité de calcul dépend de la tâche réalisée : de très réduite pour un pèse-personne à très grande pour une télévision connectée qui doit décoder un flux vidéo 4K. Une montre connectée utilise un nombre limité de capteurs, réalise peu de calculs et peut se contenter d'un lien de communication sans fil à courte portée comme le Bluetooth. *A contrario*, un robot mobile autonome tel que ceux développés pour les entrepôts contient une multitude de capteurs et d'actionneurs pour ranger ou récupérer des produits. La réalisation de ces tâches nécessite l'exécution d'algorithmes complexes avec une puissance de calcul important. La surface des bâtiments requiert un lien de communication avec une portée plus importante. Comme dernier exemple, je cite des appareils de suivis de ruches dans le monde apicole afin de connaitre l'état de santé de la colonie d'abeilles ou encore l'état des récoltes de nectars. Les ruches pouvant se situer en pleine nature, ces appareils doivent être autonomes en énergie et privilégient les liens de communication à faible consommation de type LPWAN (LoRa, Sigfox, NB-IoT).

Les contraintes propres à chaque type d'objet connecté impactent les possibilités de

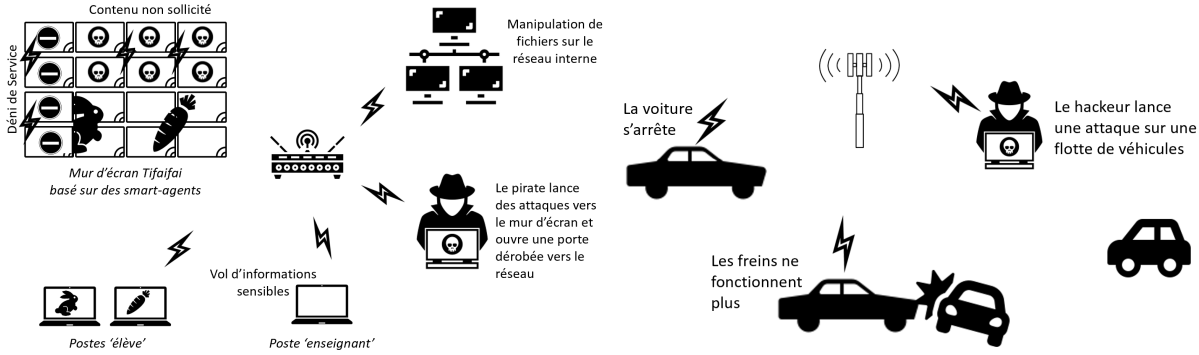
cyberattaques sur ces mêmes appareils selon la technologie de communication utilisée. En accord avec le contexte de recherche, je borne le champ de nos recherches aux technologies applicables aux deux domaines que sont l'éducation et les voitures connectées. Les appareils technologiques sont de plus en plus nombreux dans le monde de l'éducation. Un exemple de ce type de dispositif est le mur d'écrans reconfigurable capable d'interagir avec des systèmes mobiles (Renault et al., 2014) où chaque écran est piloté par un système embarqué sur lequel sont implémentés des agents de type IoT-A (Carlier & Renault, 2016). La distribution de contenus sur le mur d'écrans se fait par une liaison Ethernet ou Wifi. Les architectures électriques/électroniques dans le domaine automobile, quant à elles, migrent vers une utilisation d'une connexion cellulaire pour la connectivité extra-véhiculaire, de plusieurs sous-réseaux Ethernet organisés en domaine ou en zone et connectés par des passerelles au sein du véhicule (Bandur et al., 2021), et dans une moindre mesure du Wifi. Les architectures électroniques des véhicules doivent être capable de transporter des flux de données de plus en plus volumineux, contenant les images ou les données des radars captées autour du véhicule, alors que le Wifi est plutôt le point de connexion pour les passagers du véhicule dans le but d'améliorer leur confort au cours du voyage. Pour le mur d'écrans comme pour la voiture connectée, une protection par un pare-feu n'est pas envisageable, car n'importe quel élève ou passager montant dans la voiture doit pouvoir se connecter au réseau.

Les techniques de cyberattaques habituellement utilisées sur les réseaux d'ordinateurs connectés par Ethernet ou Wifi sont tout à fait applicables aux objets connectés que sont le mur d'écrans d'une part, la voiture connectée d'autre part. La Figure 3a illustre une attaque vers le mur d'écrans. Le hackeur peut chercher une vulnérabilité à exploiter sur les écrans et ainsi empêcher l'utilisation d'une partie d'entre eux ou afficher du contenu non sollicité. En se faisant passer pour un élément de confiance du réseau, il peut chercher à voler des informations sensibles ou manipuler des fichiers sur le réseau interne. La Figure 3b montre un hackeur qui attaque une flotte de véhicules et peut déclencher des dysfonctionnements graves comme l'arrêt de la voiture ou la désactivation des freins. Comme ces attaques s'appuient sur la pile de protocoles IP, tous les objets connectés utilisant cette même pile de protocoles deviennent des cibles potentielles.

Sur la première moitié de 2021, les attaques vers les IoT ont augmenté de plus de 100% selon une analyse de Kaspersky (« IoT Attacks Skyrocket, Doubling in 6 Months », 2021), passant de 639 millions sur la dernière moitié de 2020 à 1.5 milliard d'attaques. La majorité de ces attaques ont utilisé telnet, SSH et des connexions web, des techniques qui se basent sur le protocole IP. Selon la Figure 1, ces objets représenteront plus de 46% du marché des IoT en 2025, c'est-à-dire plus de douze milliards de dispositifs.

Apprentissage automatique

Les objets connectés, par nature, interagissent avec le monde physique, réel et produisent des données numériques. Ces données peuvent par exemple être le résultat d'une



(a) Cas du mur d'écrans. (b) Cas de la voiture connectée.

FIGURE 3 – Exemples de cyberattaques.

mesure effectuée, mise en forme avant un envoi sur le réseau de communication, ou encore le partage d'une action décidée par l'objet.

La disponibilité de données ouvre la porte à des traitements dits d'intelligence artificielle et en particulier à l'utilisation d'algorithmes d'apprentissage automatique, aussi appelé machine learning en anglais (ML). Ces algorithmes permettent de faire des prédictions à partir de l'observation de données au lieu de chercher à décrire explicitement comment interpréter les données.

Un système de détection d'intrusions réseau peut être vu comme un programme de machine learning qui apprend à identifier les tentatives d'intrusions à partir d'un jeu de données contenant des exemples de telles attaques. Le système de détection d'intrusions (IDS) basé sur des techniques de machine learning apprend de manière automatique quelles caractéristiques parmi celles du jeu de données d'entraînement sont les meilleurs prédicteurs des tentatives d'intrusions.

Différentes catégories d'algorithmes de ML existent (KANG & JAMESON, 2018). Seules quelques-unes sont illustrées dans cette section en regard de la détection d'intrusions.

L'apprentissage supervisé utilise un jeu de données contenant le résultat escompté sous forme d'un label qui représentera une classe pour un exercice de classification ou une valeur numérique pour un exercice de régression. Détecter des attaques d'intrusion correspond à une classification et les échantillons auront chacun un label indiquant s'il s'agit d'un exemple d'attaque ou bien d'un trafic normal.

Dans le cas de l'apprentissage non supervisé, le jeu de données ne contient pas de label. Ce type d'apprentissage sert à faire des regroupements ou encore à réduire la dimension des données. Cette technique est utilisable pour un IDS dans lequel on pourrait avoir un groupe de données correspondant à un trafic normal et d'autres groupes correspondant aux différents types d'intrusions.

L'apprentissage semi-supervisé se trouve à mi-chemin de l'apprentissage supervisé et non-supervisé. Dans ce cas de figure, seule une partie du jeu de données contient des labels. La détection d'anomalies est un bon exemple d'apprentissage semi-supervisé. Il est alors possible d'entraîner un algorithme pour classer comme attaques des données qui

s'éloigneraient des échantillons correspondant à un trafic réseau légitime.

Le machine learning représente une grande famille d'algorithmes. Certains d'entre eux sont anciens, Cramer rapporte que les origines de la régression logistique remontent au XIX^{e} siècle (Cramer, 2002). Au cours des dernières années, Une branche du machine learning s'est développée, utilisant des réseaux de neurones et forme un sous-domaine appelé l'apprentissage profond.

Définition de la problématique

Je constate que, d'une part, les cyberattaques vers les objets connectés sont en augmentation et utilisent majoritairement les communications par protocole IP et, d'autre part, ce même lien de communication sert à partager des données ou des informations lors d'une utilisation normale. Ces échanges, qu'ils soient normaux ou liés à des cyberattaques sont susceptibles de servir à des algorithmes d'apprentissage automatique. Je considère la possibilité de joindre les domaines de l'objet connecté, de la cybersécurité et du machine learning au travers de la problématique suivante :

Problématique

Les objets connectés utilisant des communications par protocole IP peuvent-ils être protégés des tentatives d'intrusions réseau grâce à des techniques d'apprentissage automatique supervisé ?

Comme présenté précédemment, les objets connectés représentent des appareils d'une grande variété. Dans la suite de ce document, j'appelle objet connecté tout appareil correspondant à la définition 1.

Définition 1: Objet connecté

Un objet connecté est un appareil électronique interagissant avec le monde réel et communiquant en réseau avec d'autres appareils électroniques en utilisant le protocole IP.

Cette problématique s'inscrit parfaitement dans le contexte de recherche du CREN avec le mur d'écrans et de STMicroelectronics avec les voitures connectées.

Contributions visées

À travers la problématique, j'évoque des dispositifs qui sont contraints en termes de ressources, par exemple en capacité de calcul ou en capacité mémoire. Les possibilités de détection d'intrusions doivent prendre en compte cette dimension, à la fois dans la

sélection des algorithmes de machine learning, mais plus généralement dans une approche globale, matérielle et logicielle, sur tous les aspects, depuis la réception de trames réseau à la classification du trafic.

La première hypothèse que j'avance est la suivante :

Hypothèse 1

Un réseau de neurones, potentiellement exécutable sur un accélérateur matériel modeste, peut atteindre d'aussi bonnes performances de détection d'intrusions réseau que les algorithmes classiques d'apprentissage automatique.

Cette hypothèse porte sur le choix des algorithmes de machine learning. La tendance dans le monde des microcontrôleurs est d'embarquer des accélérateurs neuronaux compatibles avec une utilisation dans les objets connectés (Desoli et al., 2017). Ces accélérateurs à l'architecture spécifique aux traitements neuronaux présentent deux intérêts. D'une part, ils sont optimisés en termes de temps de calcul et de consommation d'énergie. Ceci a typiquement été démontré en comparant les performances d'un processeur, d'une unité de traitement graphique et d'une unité de traitement de tenseurs (Patterson, 2018). D'autre part, les calculs complexes sont déportés hors du processeur principal, lui permettant de réaliser d'autres traitements en parallèle. Il existe néanmoins un inconvénient pour une application à la détection d'intrusions. En effet, ils sont très spécialisés et peuvent difficilement être exploités par d'autres types d'algorithmes d'apprentissage automatique comme les arbres de décision ou les machines à vecteurs de support.

Certaines publications (Sharafaldin et al., 2018 ; Tang et al., 2016) tendent à montrer que les réseaux de neurones sont moins performants pour détection des cyberattaques d'intrusions réseau. S'il est possible de montrer qu'un réseau de neurones pouvant être accéléré de façon matérielle atteint d'aussi bonne performance de détection que les algorithmes classiques, alors la charge du processeur due à la détection d'intrusions pourra être plus faible qu'avec d'autres algorithmes ne tirant pas parti d'accélérateur matériel. Ainsi, la détection d'intrusions deviendrait plus efficace en utilisant moins de ressources du processeur principal et donc en consommant moins d'énergie. Ces avantages sont importants dans les objets connectés, souvent très contraints en ressources.

Dans tout système utilisant l'apprentissage automatique, la qualité des données du corpus numérique, selon la définition 2, est capitale pour entrainer un modèle de détection d'intrusions et obtenir de bons résultats.

Définition 2: Corpus numérique

Un corpus numérique est ensemble de données pouvant servir à l'entrainement, aux réglages et au test de modèles d'apprentissage automatique.

Pour ma deuxième hypothèse, je me focalise sur le couple performance et corpus numérique. L'objectif est toujours tourné vers la nécessité de consommer le moins de

ressources possible dans les objets connectés, sans pour autant sacrifier les performances de détection d'intrusions.

La qualité d'un corpus numérique est essentielle pour obtenir de bonnes performances avec des algorithmes d'apprentissage automatique. L'importance de cette qualité est connue depuis longtemps. Le niveau de qualité peut être le facteur limitant aux capacités de classification d'algorithmes de machine learning, comme établi dans le domaine des télécommunications (Cortes et al., 1994). Dans le domaine médical (Miller et al., 2019), ces algorithmes n'ont pas montré d'avantage significatif par rapport à d'autres familles d'algorithmes, évoquant un problème de qualité des données.

Pour traiter ce type de difficulté, des propositions existent sur la documentation et les pratiques nécessaires au développement de corpus numériques afin de leur assurer un bon niveau de qualité (Paullada et al., 2021 ; Picard et al., 2020), mais restent très générales. Lorsqu'un nouveau corpus est disponible, des chercheurs l'utilisent parfois sans se questionner sur la qualité de son contenu.

Il n'existe pas de solution universelle pour fiabiliser un corpus numérique existant. Dans le domaine spécifique de la détection d'intrusions réseau, je souhaite étudier comment la fiabilisation d'un corpus existant peut influencer les performances et la complexité du modèle de détection. Je vise en particulier, par l'amélioration du corpus grâce à une méthodologie spécifique à la détection d'intrusions réseau, une optimisation des performances et du modèle de détection.

La deuxième hypothèse que je formule est la suivante :

Hypothèse 2

La fiabilisation d'un corpus de détection d'intrusions conduit à améliorer la performance de détection tout en garantissant un modèle plus simple.

Dans la suite logique des hypothèses précédentes, la troisième hypothèse porte sur la mise en œuvre d'un système complet de détection d'intrusions réseau sur un objet connecté. En effet, un système de détection d'intrusions ne se résume pas à un modèle de détection. Il est nécessaire de prendre en compte d'autres activités comme l'acquisition et la transformation d'informations.

Au fil du temps, les débits sur les réseaux ont tendance à augmenter, notamment avec du transfert de flux vidéo. Ce point s'applique à la fois dans le contexte de recherche avec du partage de vidéos sur un mur d'écrans, mais aussi avec des flux venant de plusieurs caméras pour l'aide au stationnement dans les voitures connectées. La détection d'intrusions à des débits élevés est connue pour engendrer une charge importante sur les processeurs (Dreger et al., 2004). C'est *a fortiori* encore plus vrai sur des cibles matérielles contraintes généralement utilisées dans les objets connectés.

Les algorithmes de classification pour la détection d'intrusions sont régulièrement revisités et améliorés. Toutefois, l'étude du système de détection dans sa globalité est nécessaire pour s'assurer qu'il peut fonctionner à un débit suffisant et qu'il n'existe pas

un goulet d'étranglement ailleurs que dans l'algorithme de classification. C'est donc une nécessité de considérer l'ensemble des traitements depuis la réception des trames réseau jusqu'à la classification.

J'énonce ma troisième hypothèse comme suit :

Hypothèse 3

Il est possible d'implémenter, sur un système embarqué, un système de détection d'intrusions incluant l'extraction des caractéristiques des paquets réseau et l'algorithme de détection d'intrusions.

Structure du mémoire

Après avoir introduit le contexte de nos recherches, la problématique et les trois hypothèses, la suite de ce manuscrit de thèse est composée de trois grandes parties.

Dans la partie I, j'aborde les domaines propres à nos recherches. Après une analyse des objets connectés dans le chapitre 1, je détaille la détection d'intrusions réseau sous l'angle des cyberattaques, des corpus numériques spécifiques à ce domaine et des techniques de détection dans le chapitre 2. Le chapitre 3 couvre les spécificités de l'apprentissage automatique, à la fois par différentes formes d'apprentissage et par le type de tâches du machine learning. Différents algorithmes, souvent utilisés dans la détection d'intrusions, sont présentés ainsi que la méthode d'entraînement basée sur la descente du gradient.

La partie II décrit nos contributions, les expérimentations menées ainsi que les résultats obtenus dans trois chapitres, chacun adressant une des hypothèses liées à la problématique. Tout d'abord, le chapitre 4 montre que l'utilisation de réseaux de neurones pour la détection d'intrusions est une approche viable. Ensuite, j'étudie, dans le chapitre 5, le lien entre la fiabilisation du corpus et la nécessaire optimisation du modèle de détection pour un système embarqué contraint. Pour finir cette partie, une approche globale du système de détection d'intrusions décrite dans le chapitre 6 montre qu'il est possible de réaliser un tel système dans un objet connecté, notamment grâce à l'apport d'une solution originale réduisant la charge du processeur, une innovation brevetée au cours de ce projet de recherche.

Enfin, je termine ce manuscrit par une synthèse de nos contributions. L'analyse des résultats obtenus amène des éléments à mettre en regard des hypothèses formulées dans l'introduction afin de répondre à la problématique. Pour finir, une ouverture vers des travaux ultérieurs est proposée pour, d'une part, approfondir les recherches que j'ai menées, et d'autre part, mentionner de possibles extensions connexes aux travaux réalisés durant cette thèse.

Références

Bandur, V., Selim, G., Pantelic, V. & Lawford, M., (2021), Making the Case for Centralized Automotive E/E Architectures, *IEEE Transactions on Vehicular Technology*, *70 2*, 1230-1245, https://doi.org/10.1109/TVT.2021.3054934

Carlier, F. & Renault, V., (2016), Iot-a, embedded agents for smart internet of things : Application on a display wall., *In 2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA), IEEE Computer Society*, 80-83.

Cortes, C., Jackel, L. D. & Chiang, W.-P., (1994), Limits on Learning Machine Accuracy Imposed by Data Quality, In G. Tesauro, D. Touretzky & T. Leen (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/1994/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf

Cramer, J., (2002), *The Origins of Logistic Regression* (Tinbergen Institute Discussion Papers N° 02-119/4), Tinbergen Institute, https://doi.org/10.2139/ssrn.360300

Desoli, G., Chawla, N., Boesch, T., Singh, S.-p., Guidetti, E., De Ambroggi, F., Majo, T., Zambotti, P., Ayodhyawasi, M., Singh, H. & Aggarwal, N., (2017), 14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems, *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 238-239, https://doi.org/10.1109/ISSCC.2017.7870349

Dreger, H., Feldmann, A., Paxson, V. & Sommer, R., (2004), Operational Experiences with High-Volume Network Intrusion Detection, 2-11, https://doi.org/10.1145/1030083.1030086

European Parliament, (2015), Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 April 2015 Concerning Type-Approval Requirements for the Deployment of the eCall in-Vehicle System Based on the 112 Service and Amending Directive 2007/46/EC, *Official Journal of the European Union.*

IoT Attacks Skyrocket, Doubling in 6 Months, (2021), Threatpost. Récupérée 6 septembre 2021, à partir de https://threatpost.com/iot-attacks-doubling/169224/

Kang, M. & Jameson, N. J., (2018), Machine Learning : Fundamentals. *Prognostics and Health Management of Electronics* (p. 85-109), https://doi.org/10.1002/9781119515326.ch4

Miller, P. E., Pawar, S., Vaccaro, B., McCullough, M., Rao, P., Ghosh, R., Warier, P., Desai, N. R. & Ahmad, T., (2019), Predictive Abilities of Machine Learning Techniques May Be Limited by Dataset Characteristics : Insights From the UNOS Database, *Journal of Cardiac Failure*, *25 6*, 479-483, https://doi.org/https://doi.org/10.1016/j.cardfail.2019.01.018

Patterson, D., (2018), 50 Years of computer architecture : From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set, *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 27-31, https://doi.org/10.1109/ISSCC.2018.8310168

Paullada, A., Raji, I. D., Bender, E. M., Denton, E. & Hanna, A., (2021), Data and its (dis)contents : A survey of dataset development and use in machine learning research, *Patterns*, *2 11*, 100336, https://doi.org/https://doi.org/10.1016/j.patter .2021.100336

Perwej, D., Qamar Abbas, S., Pratap Dixit, J., Akhtar, D. N. & Kumar Jaiswal, A., (2021), A Systematic Literature Review on the Cyber Security, *International Journal of scientific research and management*, *9 12*, 669-710, https://doi.org/10 .18535/ijsrm/v9i12.ec04

Picard, S., Chapdelaine, C., Cappi, C., Gardes, L., Jenn, E., Lefevre, B. & Soumarmon, T., (2020), Ensuring Dataset Quality for Machine Learning Certification, *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 275-282, https://doi.org/10.1109/ISSREW51248.2020.000 85

Renault, V., Carlier, F. & Bourdon, P., (2014), TIFAIFAI : Conception de Nouveaux Espaces d'Interactions pour Apprendre [(Accessed Mar 14, 2022)], *Distances et médiations des savoirs*, https://doi.org/10.4000/dms.588

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., (2018), Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, 108-116, https://doi.org/10.5220/0006639801080116

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R. & Ghogho, M., (2016), Deep learning approach for Network Intrusion Detection in Software Defined Networking, *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 258-263, https://doi.org/10.1109/WINCOM.2016.7777224

Van, H. P., (2021), *State of IoT - summer report*. Récupérée 26 août 2022, à partir de https://iot-analytics.com/product/state-of-iot-summer-2021/

Völske, M., Bevendorff, J., Kiesel, J., Stein, B., Fröbe, M., Hagen, M. & Potthast, M., (2021), Web Archive Analytics, In R. H. Reussner, A. Koziolek & R. Heinrich (Éd.), *INFORMATIK 2020* (p. 61-72), Gesellschaft für Informatik, Bonn, https://doi.org/10.18420/inf2020_05

Première partie

État de l'art

Chapitre 1

Objets connectés

« *Maintenant, nous sommes tous connectés par Internet, comme des neurones dans un cerveau géant.* »

Stephen Hawking
Physicien théoricien et cosmologiste (1942 – 2018)

Le nombre d'êtres humains connecté à l'internet ne cessent de croître et est estimé pour 2021 à 4.9 milliards, soit environ 63 % de la population mondiale (ITU, 2021). La même année, le nombre d'objets connectés est estimé à 14.5 milliards (Van, 2021). Une telle quantité de connexions peut être comparée à un cerveau géant avec tous ses neurones et synapses. Toutefois, si ce cerveau est grand par son étalement autour du globe, il n'en reste pas moins incomparable à un cerveau humain et ses quatre-vingt-six milliards de neurones. À titre de comparaison, le nombre d'objets connectés en 2021 est du même ordre de grandeur que le nombre de neurones d'un babouin (Herculano-Houzel, 2020).

La comparaison des connexions à l'internet avec un cerveau atteint rapidement ses limites. Néanmoins, les objets connectés et en particulier les systèmes embarqués ont en commun avec le cerveau d'avoir la faculté de percevoir et d'agir avec leur environnement, proche ou éloigné, grâce à des systèmes de communication.

Pour faciliter la compréhension, commençons par clarifier quelques termes et lever certaines ambigüités. Une multitude de définitions existe pour définir l'objet connecté ou l'IoT. Quelques fois considéré comme l'objet connecté lui-même, l'IoT est d'autres fois considéré comme un réseau d'objets connectés (Madakam et al., 2015 ; Sharma et al., 2019). Dans ce document, nous différencions les objets connectés ou objets intelligents de l'IoT en lui laissant son sens littéral d'internet des objets. IoT est utilisé pour se référer au réseau utilisé par les objets connectés.

Dans ce chapitre, nous présentons ce qu'est un objet connecté et quelques-unes de ses caractéristiques dans la section 1.1. Ensuite, la section 1.2 met l'accent sur les communications et, en particulier, celles en rapport avec notre contexte de recherche.

1.1 Systèmes embarqués communicants

Le concept d'objet connecté est déjà ancien avec une apparition probable à la fin des années 1980 ou au début des années 1990. Un toaster mis en fonctionnement par l'internet est inventé par John Romkey (Romkey, 2017) et d'autres inventions ont rapidement

suivi (Suresh et al., 2014). Quarante ans plus tard, la plupart des équipements que nous utilisons sont pilotés ou surveillés à travers l'IoT.

Les objets connectés sont déclinés dans de nombreux domaines. Signe de leur ubiquité, des études sont menées sur des utilisations très diverses, notamment dans l'industrie (H. Xu et al., 2018), le système médical (Dang et al., 2019) ou encore le monde agricole (Friha et al., 2021). Ces appareils intelligents se retrouvent aussi dans les villes (Ejaz & Anpalagan, 2019) et nos habitations (Marikyan et al., 2019).

Une telle variété dans leurs usages conduit à des définitions variées des objets connectés et des systèmes embarqués. Une définition très générale consiste à décrire les systèmes embarqués comme des systèmes traitant des informations au sein d'un produit (Marwedel, 2021). La description des systèmes cyber-physiques (CPS) comme des intégrations de traitements ou calculs avec des processus physiques (Lee, 2007) adjoint une dimension à la définition précédente en ajoutant le lien avec le monde physique. Dans une autre étude, un système embarqué est défini comme un artefact d'ingénierie impliquant des traitements ou des calculs soumis à des contraintes physiques qui sont liées, d'une part, à leur environnement physique, et d'autre part, à l'exécution sur une plateforme physique (Henzinger & Sifakis, 2007). Cette définition présente l'avantage d'être générique et couvre les systèmes embarqués et les CPS. Les objets connectés se différencient d'une partie des systèmes embarqués par la présence d'au moins une interface de communication pour partager de l'information en réseau.

Face à ces descriptions générales, nous commençons par présenter des contraintes s'appliquant aux objets connectés. Nous déclinons certaines d'entre elles, particulièrement utiles pour notre domaine de recherche. Nous étudions ensuite certains aspects matériels et logiciels liés à leur implémentation. Enfin, nous abordons la cybersécurité sous l'angle de ses propriétés principales, de ses menaces et vulnérabilités et des contre-mesures possibles.

1.1.1 Généralités sur les contraintes

Une qualité importante des objets connectés est la fiabilité. Par leur interaction avec l'environnement physique, ces objets peuvent avoir un impact immédiat sur le monde réel. Cette notion de fiabilité (Marwedel, 2021) se décline sous différentes contraintes de performances ou de réactivité, de cybersécurité, de sûreté de fonctionnement, de robustesse et d'efficacité. Toutefois, ces contraintes ne sont pas toutes applicables à l'ensemble des objets connectés et dépendent de leurs usages.

Dans un système dit temps-réel, les contraintes de réactivité sont classées selon deux catégories. Le temps-réel dur implique un respect strict des contraintes de temps. Leurs non-respects entrainent des conséquences majeures. Au contraire, dans un système temps-réel souple, le non-respect de ce type de contraintes ne génère pas de défaut critique.

La sécurité est un élément primordial pour éviter les vols d'informations sensibles d'un dispositif embarqué ou encore pour empêcher son fonctionnement normal. Selon son usage, les qualités d'intégrité, de confidentialité et de disponibilité peuvent être essentielles à un objet connecté.

Certains objets intelligents tels que les voitures connectées ont des contraintes fortes en termes de sûreté de fonctionnement. Les blessures physiques ou les atteintes à la santé d'une personne sont des risques inacceptables. Dans le domaine automobile, la sûreté de fonctionnement est adressée par le respect d'un processus de développement standardisé, l'ISO 26262 (Debouk et al., 2018). La sûreté de fonctionnement est souvent indissociable de la sécurité (Serpanos, 2019). Une très bonne illustration est le cas de la prise de contrôle à distance d'une voiture connectée (Miller, 2019). Les failles de sécurité exploitées dans cet exemple auraient pu conduire à la mise en danger des passagers du véhicule.

Pour des raisons d'efficacité, des contraintes supplémentaires doivent être prises en compte (Barkalov et al., 2019). Les objets comme une montre connectée ou un appareil de mesure du glucose ne peuvent pas être reliés en continu à une source d'énergie et sont contraints en termes de consommation d'énergie. Ces mêmes objets sont également soumis à des contraintes de poids et d'encombrement. Enfin, une contrainte de coût, commune à tous les produits fabriqués en grande quantité, doit être respectée.

1.1.2 Matériel et logiciel

La variété des objets connectés conduit à des implémentations très différentes, allant de solutions complètements matérielles à une combinaison de matériel et de logiciel.

1.1.2.1 Élements matériels de base

CPLD Un composant à logique programmable complexe (CPLD) est un composant électronique embarquant un ensemble de fonctions logiques de base et de blocs permettant de les combiner afin de réaliser des fonctions logiques complexes, combinatoires ou séquentielles. Grâce à ces capacités de programmation, un produit peut être changé rapidement et autant de fois que nécessaire. Ses limitations viennent de la quantité de portes logiques disponibles et des combinaisons possibles.

Dans les systèmes embarqués, les CPLD peuvent être utilisés pour rendre la fonction principale (Shah et al., 2015) ou bien comme solution apportant une fonction spécialisée à un système plus complexe (Kong et al., 2012).

FPGA Les réseau de portes programmables in situ (FGPA) sont des versions plus évoluées des CPLD afin d'implémenter des fonctions beaucoup plus complexes. Ils rendent possible la programmation d'une puce de silicium grâce à une matrice de fonctions logiques et des blocs d'entrées/sorties. Comme les CPLD, leur reconfiguration les rend très pratiques pour changer rapidement le contenu d'un produit, permettant ainsi des cycles de modifications très courts.

Une fois encore, selon la complexité des objets connectés à réaliser, les FGPA sont utilisés comme élément principal ou bien sont combinés avec d'autres éléments de traitements (Elnawawy et al., 2019).

CPU Les processeurs (CPU) sont construits autour de trois éléments : une unité arithmétique et logique, une unité de contrôle et de la mémoire sous forme de registres. Les CPU exécutent les instructions selon une séquence définie dans un logiciel stocké dans une mémoire. L'architecture des processeurs dépend du jeu d'instructions supporté.

Au cours de leur évolution, la complexité des CPU a augmenté afin d'améliorer leurs performances avec l'ajout de blocs tels que des unités de calcul à virgule flottante, ou encore d'unité de calcul vectoriel NEON sur les processeurs ARM (Smith, 2020), MMX ou SSE sur les processeurs Intel (Conte et al., 2000). La miniaturisation grâce aux gravures de plus en plus fines des semi-conducteurs a permis une augmentation de la complexité en plaçant des unités en parallèle sur les architectures superscalaires, en découpant les traitements en sous-étapes grâce à des pipelines à plusieurs niveaux ou encore en introduisant des unités de branchements prédictifs.

Outre les fonctions rendant les processeurs plus rapides, la consommation d'énergie est un critère important pour les systèmes fonctionnant sur batterie. Les ingénieurs de la société ARM ont réalisé un travail impressionnant dans ce domaine depuis l'arrivée du processeur ARM7TDMI (Segars, 1997) jusqu'aux versions plus récentes (Akram & Sawalha, 2019). Ce critère est une des raisons de la prédominance des processeurs ARM dans les systèmes embarqués.

DSP Les processeurs de traitement du signal (DSP) sont des processeurs dédiés au traitement numérique du signal. Grâce à une architecture spécialisée, les DSP traitent de manière optimale les opérations telles que :

— le filtrage ;
— la compression/décompression ;
— l'encodage/décodage ;
— la modulation/démodulation.

Parmi les objets connectés, les DSP se retrouvent couramment dans les systèmes de télécommunications tels que les modems et le très populaire téléphone mobile, mais aussi dans les appareils multimédia.

GPU Les processeurs graphiques (GPU) sont des processeurs optimisés pour le traitement d'affichage. Leur architecture est hautement parallèle afin d'optimiser le traitement d'images ou de vidéo et le rendu 3D. Les calculs vectoriels et matriciels sont également accélérés par ce type d'architecture. Pour cette raison, les GPU sont aussi utilisés dans les objets connectés mettant en œuvre des réseaux de neurones (J. Xu et al., 2017).

1.1.2.2 Intégration matérielle

Avec une quantité toujours plus grande de systèmes embarqués et d'objets connectés, des besoins d'intégration matérielle sont apparus. Parmi ce mode d'intégration, nous étudions les cas particuliers des microcontrôleurs et des systèmes sur puce.

Microcontrôleur Un microcontrôleur (MCU) est un micro-ordinateur avec tous ses éléments dans un circuit intégré. Il contient un processeur, de la mémoire non-volatile, de la mémoire vive, des périphériques variés tels que des timers et des interfaces de communication, filaire ou non, ainsi que des entrées et sorties, analogiques ou numériques.

Les MCU se déclinent en une multitude de familles afin d'adresser des besoins différents, qu'ils soient d'ordre économique ou technique. Les performances peuvent être extrêmement réduites avec des MCU 4 bits, longtemps utilisés dans les calculatrices, ou bien plus puissantes avec des MCU 16 et 32 bits. Les MCU actuels peuvent fonctionner à une fréquence allant de quelques méga-hertz à un giga-hertz (NXP, 2021). Les tailles de mémoires embarquées varient également de quelques kilo-octets à quelques méga-octets de façon à sélectionner la version proposant des performances de calculs et une taille suffisante pour le logiciel du produit visé.

Dans le but d'illustrer la différence qu'il peut exister entre différents MCU, la Figure 1.1 présente le schéma fonctionnel du MCU 68HC11 conçu par Motorola en 1984 tandis que la Figure 1.2 illustre celui de la dernière génération de MCU Stellar (STMicroelectronics, 2022) pour l'automobile conçu par STMicroelectronics.

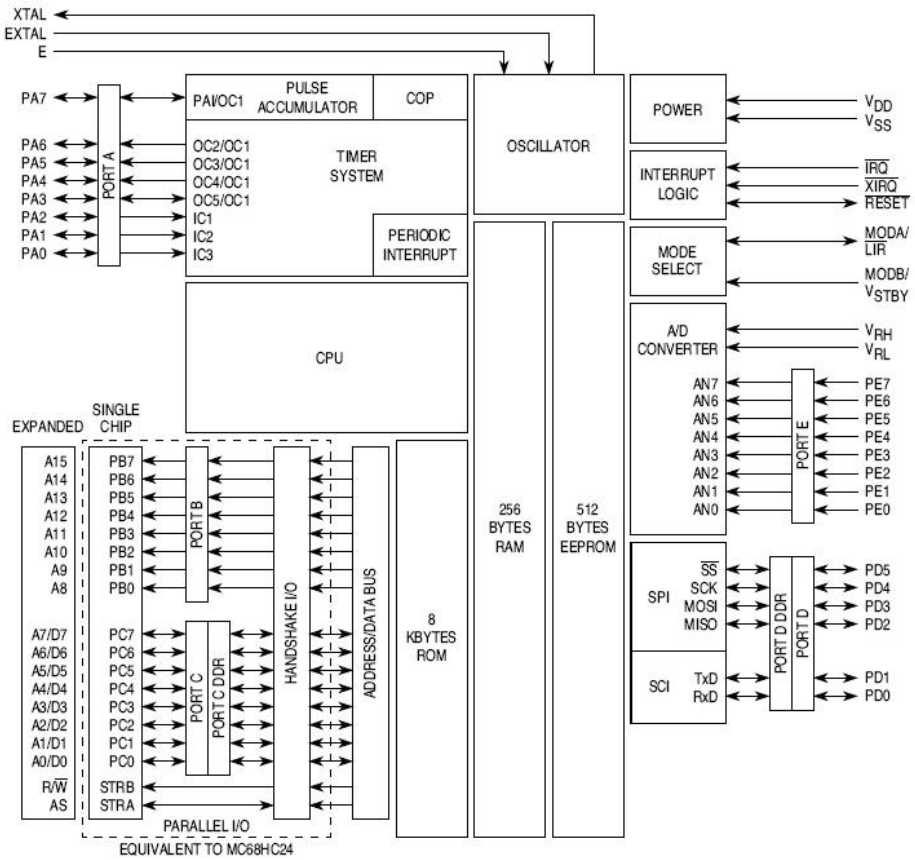


Figure 1.1 – Schéma fonctionnel du MCU 68HC11 (source : wikipedia)

Le 68HC11, dans la version du diagramme, est conçu autour d'un processeur 8 bits avec 8 kilo-octets de mémoire non-volatile, 256 octets de RAM et 512 octets de mémoire effaçable électriquement. Les ports d'entrées et sorties sont configurables en convertisseur analogique-numérique, en liens de communication série synchrone et asynchrone. D'autres blocs apportent un oscillateur intégré, un gestionnaire de temps et des interruptions.

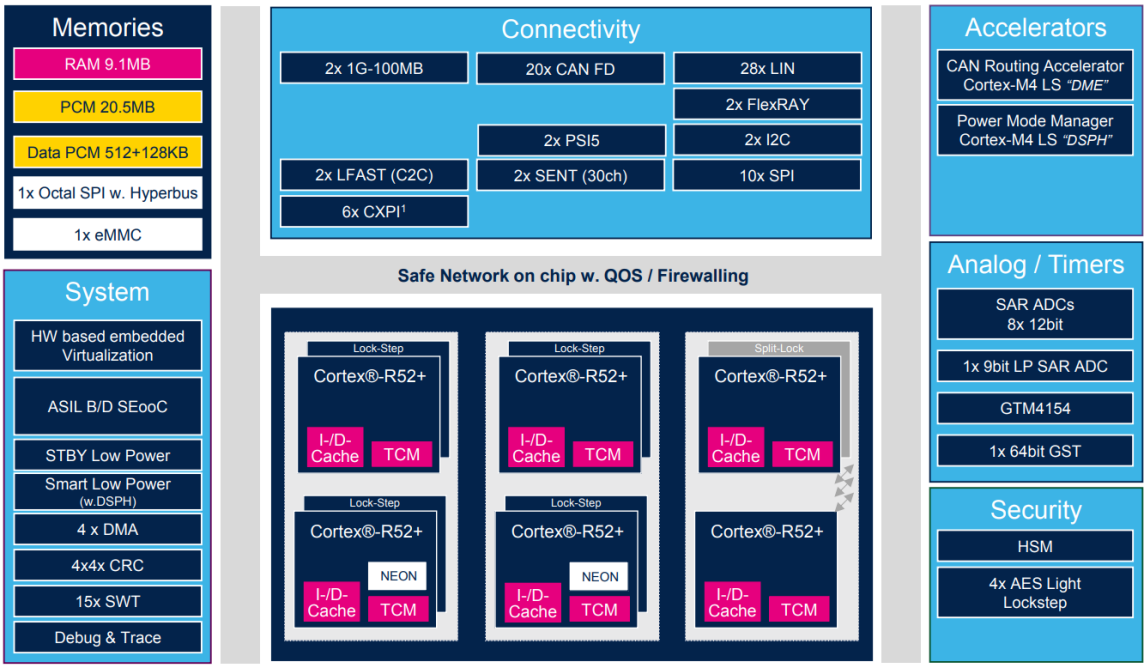


FIGURE 1.2 – Schéma fonctionnel du MCU Stellar.

La famille Stellar possèdent des processeurs beaucoup plus puissants contenant des Cortex-R52 à 400 MHz ainsi que deux processeurs moins puissants, des Cortex-M4 pour accélérer des tâches spécifiques. Les tailles mémoires permettent d'implémenter des logiciels complexes dans les 20 méga-octets de mémoire non-volatile à base de mémoire à changement de phase et pouvant utiliser 9 méga-octets de RAM. Les périphériques sont plus nombreux. En particulier, les blocs de connectivité rendent possible la connexion de deux interfaces Ethernet et vingt bus CAN.

La famille Stellar adresse la sûreté de fonctionnement avec des blocs spécifiques et la détection de dysfonctionnement sur processeurs ARM avec la technologie Lock-step.

Enfin, les services de cybersécurité sont assurés par un sous-système dédié, appelé HSM pour Hardware Security Module.

Les comparaisons de ces deux MCU montrent à quel point le niveau de performance et d'intégration en fonction du domaine d'application.

Système sur puce Les fabricants de semi-conducteurs poussent l'intégration plus loin avec la création de système sur puce (SoC pour System-on-Chip). Des fonctions plus

complexes, nécessitant davantage de blocs logiques ou analogiques, sont placées sur une unique puce.

La FIGURE 1.3 présente un exemple de SoC (STMICROELECTRONICS, 2020). Il contient deux processeurs applicatifs Cortex-A7 à 800 MHz avec deux niveaux de mémoire cache, accompagné d'un microcontrôleur à base de Cortex-M4 et d'un GPU. Un ensemble de périphériques permet un usage polyvalent et des communications, par exemple via USB ou Ethernet.

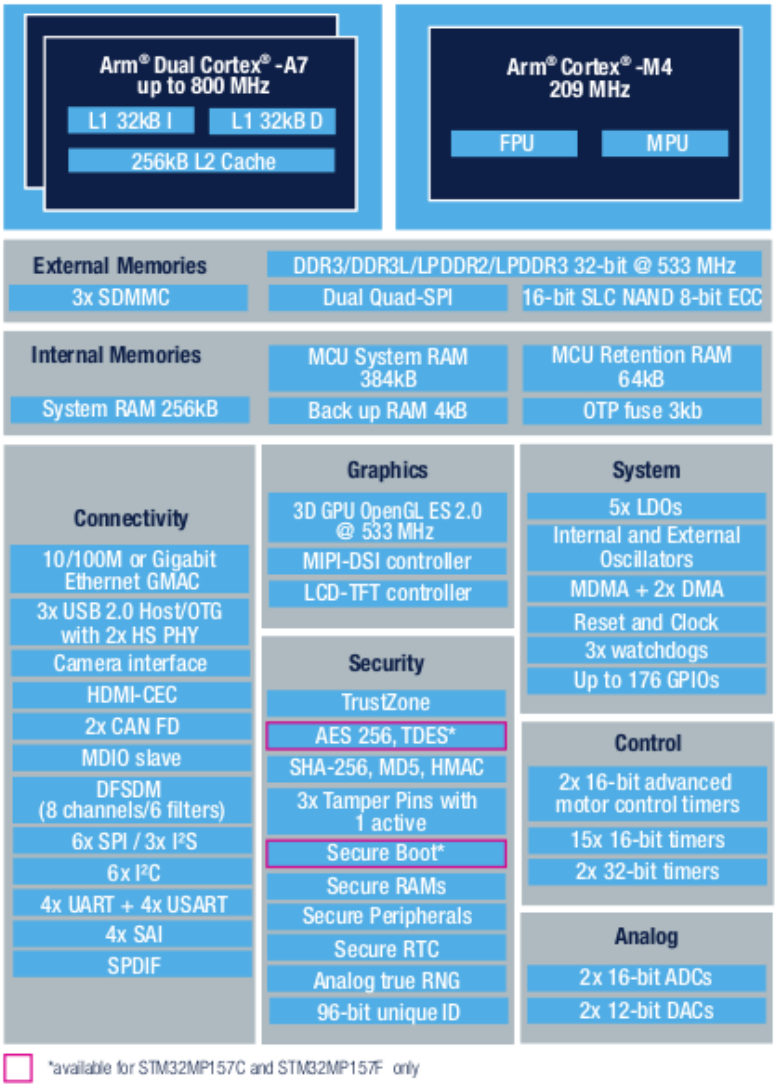


FIGURE 1.3 – Schéma fonctionnel du SoC STM32MP157.

La création de micro-ordinateur sur de petites cartes électroniques devient possible avec ce type de SoC comme le démontre les cartes Raspberry Pi pour en faire des objets connectés (COPPOLA et al., 2021 ; JABBAR et al., 2021).

1.1.3 Cybersécurité

La cybersécurité est une préoccupation importante pour les objets connectés. Certains d'entre-eux peuvent contenir ou partager des informations sensibles tandis que d'autres prennent des décisions potentiellement inadaptées ou néfastes s'ils subissent des attaques. Enfin, leurs usages peuvent être détournés dans le but de réaliser des fonctions utiles aux hackeurs.

Propriétés de sécurité Lors de l'étude de la sécurité d'un objet, plusieurs propriétés sont considérées. Les quatre suivantes sont particulièrement importantes :

- la confidentialité assure que les données ne sont disponibles que pour les utilisateurs autorisés ;
- l'intégrité garantit que les données n'ont pas été modifiées ;
- la disponibilité fait en sorte que l'objet et les données soient disponibles lorsqu'ils doivent être utilisés ;
- l'authentification assure que les données ou les applications proviennent de tiers de confiance.

Menaces, vulnérabilités et attaques Bien qu'il n'existe pas un cadre largement accepté sur l'architecture des objets connectés, différents modèles existent en trois, quatre ou cinq couches (Krishna et al., 2021 ; Lin et al., 2017). Dans le modèle le plus simple, celle de perception, souvent appelée couche physique, collecte les données depuis les capteurs et les transfère à la couche réseau qui gère les communications et fait la connexion avec la couche d'application, responsable de rendre le service prévu par l'objet connecté en prenant en compte les données collectées en local et à distance. Dans le modèle le plus évolué, la couche d'application est scindée en trois : service, opération et application. Les différents étages du modèle adopté permettent de séparer les menaces et les vulnérabilités en catégories. Krishna et al. propose une taxonomie des menaces et des attaques.

Une autre étude complète l'analyse par architecture avec des vecteurs de menaces (Rizvi et al., 2018) de trois types :

- les attaques de communication menée par le réseau, par exemple le déni de service, l'usurpation d'identité, l'homme du milieu et les injections SQL ;
- les attaques physiques comme l'ingénierie inverse, la falsification de données et les attaques par canaux cachés ;
- les attaques logicielles avec, par exemple, l'exploitation de faille dans les configurations des objets connectés, des fonctions logicielles permettant de l'injection de code malveillant ou bien détournant les fonctionnalités de sécurité en remplaçant une partie du code (injection de shell, par exemple par débordement de buffer).

Les attaques liées à la détection d'intrusions réseau sont détaillées dans la section 2.1.

Contre-mesures Les objets connectés étant souvent produit en grand volume, l'aspect économique est une dimension importante à prendre en compte. D'un côté, une mauvaise

sécurisation peut conduire à des vols de données ou des dysfonctionnements et impactera l'image de la société qui produit ces objets connectés. D'un autre côté, une sur-protection conduit à l'intégration dans les MCU ou les SoC de fonctions inutiles. Ces blocs occupent une surface de silicium et impacte directement le coût du composant électronique. Le niveau de sécurité pris en compte dans les objets connectés doit donc être choisi en fonction de la valeur de ce qui doit être protégé. L'idée est bien de forcer un potentiel hackeur à dépenser plus d'argent à casser la sécurité d'un objet connecté que la valeur de ce qu'il pourrait y récupérer.

Pour cela, les composants électroniques incorporent des technologies plus ou moins pointues. Si l'objet connecté ne contient aucun bien de valeur, les fonctions de sécurité seront absentes. Pour un niveau intermédiaire, des services de cryptographies, avec ou sans accélération matérielle, seront prévus.

Citons quelques exemples concrets. Le processeur d'un MCU exécutera le programme de la mémoire non-volatile seulement après avoir vérifié son authenticité. Un fabricant d'un produit voulant empêcher une société mal intentionnée de copier son matériel et son logiciel le protègera en chiffrant le programme avec une clef unique au composant électronique.

Pour cela, les composants intègrent des algorithmes cryptographiques de hachage, de chiffrement à clefs symétrique et asymétrique, de signature, associés à des moyens de protéger les clefs secrètes ou privées. Les algorithmes à utiliser sont spécifiés par des organismes nationaux comme le *National Institute of Standards and Technology* (NIST) aux États-Unis d'Amérique, l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) en France ou encore le *Bundesamt für Sicherheit in der Informationstechnik* (BSI) en Allemagne. Ces fonctionnalités peuvent être intégrées dans un bloc spécifique comme le HSM mentionné sur la Figure 1.2.

Enfin, si le niveau de sécurité est très critique, les MCU ou les SoC peuvent être connectés à des dispositifs électroniques ultra-protégés, du même type que ceux utilisés dans les cartes bancaires ou encore les documents d'identité. Ces dispositifs répondent à des critères très stricts comme les critères communs du standard ISO/IEC 15408 (ISO, 2009).

Des contre-mesures logicielles sont nécessaires pour toute partie de programme sensible. Des règles de codage strictes limitent les risques (Kurachi et al., 2018). Dans le domaine automobile, le processus de développement ISO/SAE 21434 (ISO, 2021) est l'équivalent de l'ISO 26262 (Debouk et al., 2018) pour la sûreté de fonctionnement. Il couvre la gestion de la cybersécurité durant toutes les phases d'un produit, depuis la conception, le développement, la production, l'exploitation, la maintenance, jusqu'à la mise hors service des systèmes électriques et électroniques des véhicules.

Pour toutes les attaques passant par les communications réseau, la détection d'intrusions fait partie des techniques à mettre en œuvre. Ces systèmes sont couverts en détail dans le chapitre 2.

1.2 Communications

Les objets connectés collectent de l'information et l'échangent dans le but de réaliser des comportements plus ou moins complexes. Le partage d'information est une fonction primordiale de ces appareils. Cette capacité à échanger de l'information est assurée par l'interopérabilité des moyens de communication. Celle-ci est assurée par des standards produits par différents groupes dont quelques uns sont listés :

— l'*Institute of Electrical and Electronics Engineers* (IEEE) ;
— l'*International Telecommunication Union* (ITU) ;
— l'*International Organization for Standardization* (ISO) ;
— l'*European Telecommunications Standards Institute* (ETSI) ;
— le *3rd Generation Partnership Project* (3GPP) ;
— l'Internet Engineering Task Force (IETF).

Rappelons maintenant notre définition d'un objet connecté donnée dans le chapitre Introduction : *un objet connecté est un appareil électronique interagissant avec le monde réel et communiquant en réseau avec d'autres appareils électroniques en utilisant le protocole IP.*

Dans ce contexte, nous nous appuyons principalement sur les spécifications de l'IEEE et de l'IETF. Les standards de l'internet sont définis dans des documents appelés *Request for Comments* (RFC). À ces documents s'ajoutent des protocoles propriétaires, définis par des entreprises majeures du secteur des communications.

Les spécifications des systèmes de communications sont organisées en couches. Le modèle le plus commun est le modèle OSI (Kumar et al., 2014), publié en 1984 par l'ISO et l'actuel ITU.

1.2.1 Modèles OSI et TCP/IP

Le modèle OSI est basé sur une stratification en couches. Chacune d'elles exécute un ensemble de fonctions similaires, en utilisant et en enrichissant les services fournis par la couche immédiatement inférieure. La superposition de couches fournit une décomposition logique d'un réseau de communication complexe en parties plus petites, plus compréhensibles et plus faciles à gérer.

Les données de la couche la plus haute sont complétées par des informations permettant leur acheminement dans le réseau de communication. En traversant les couches des plus hautes vers les plus basses, des informations supplémentaires s'ajoutent avant ou parfois après les données.

Le modèle OSI, composé de sept couches, est représenté par la Figure 1.4 avec deux dispositifs échangeant de l'information à travers un routeur.

Couche physique La couche physique définit les spécifications électriques et physiques de la connexion. Elle couvre en particulier la transformation de données binaires pour un transfert sur un support de transmission physique ainsi que les aspects mécaniques comme

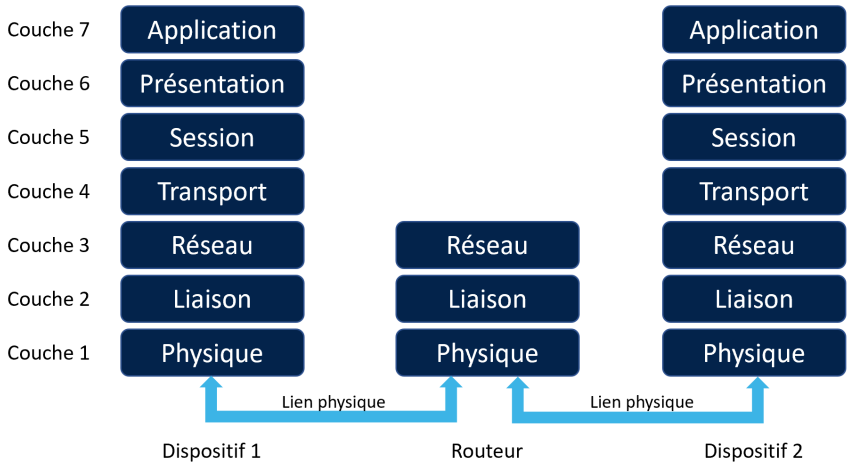


FIGURE 1.4 – Modèle OSI.

les connecteurs ou l'arrangement des broches de connexion.

Cette conversion permet par exemple l'adaptation des données au système de communications sans fil, filaire, cuivrée ou optique.

Couche liaison de données La couche de liaison de données assure plusieurs fonctions non supportées par la couche physique. Elle transfère, entre autres, des données sous forme de trames, avec des marqueurs de début et de fin. Elle joue également un rôle important dans la fiabilité du transfert de données avec des mécanismes de détection et/ou de correction d'erreur, généralement mis en œuvre avec un contrôle de redondance cyclique.

Couche réseau La couche réseau possède un rôle essentiel dans le routage des paquets de données de bout en bout. Lorsque les données doivent être acheminées à des dispositifs par des nœuds intermédiaires, il est nécessaire de fournir un système d'adressage pour déterminer le chemin approprié.

Couche transport Cette couche intervient dans l'acheminement des données et un programmeur s'interface souvent à ce niveau pour faire des transferts réseau. De plus, la couche transport peut adresser les déficiences de service des couches inférieures. Si le réseau n'est pas fiable, la couche transport peut fournir les mécanismes qui vont fiabiliser la communication.

Couche session L'objectif principal de cette couche est de fournir la capacité à la couche immédiatement supérieure d'organiser la communication pour de multiples sessions ayant lieu en même temps. Les applications à chaque extrémité peuvent échanger des données ou envoyer des paquets à l'autre partie aussi longtemps que dure la session.

La couche gère son établissement, les échanges de données ou de messages, et sa destruction lorsqu'elle se termine. Un mécanisme d'identification et de contrôle d'accès aux informations assure que seules les parties désignées puissent participer à la session.

Couche présentation La couche de présentation est responsable de la manière dont les données sont présentées à l'application. À l'établissement de la communication, la couche de présentation négocie le format des données à transférer avec l'autre dispositif sur le réseau et la syntaxe de transfert. Après cette phase de négociation, des services supplémentaires peuvent être utilisés par l'application, tels que la compression, le chiffrement et la traduction.

Couche application La couche application est celle avec laquelle l'utilisateur interagit directement via les logiciels applicatifs nécessitant une communication. Cette couche fait typiquement le lien entre le navigateur internet, les logiciels de partage d'écrans, les systèmes de messagerie, email ou chat, et les données échangées avec d'autres machines sur le réseau. Nous y trouvons une grande quantité de protocoles permettant des usages très variés.

Au moment de la publication du modèle OSI, une suite de protocoles pour l'internet, appelée TCP/IP, était déjà en cours de développement. La décomposition de cette suite ne suit pas exactement le modèle OSI et la terminologie diffère légèrement (Rayes & Salam, 2017). Une démarcation importante entre les couches OSI se situe entre les couches quatre et cinq. Cette frontière distingue les questions liées aux applications des télécommunications. Une comparaison entre les deux modèles est illustrée par la Figure 1.5.

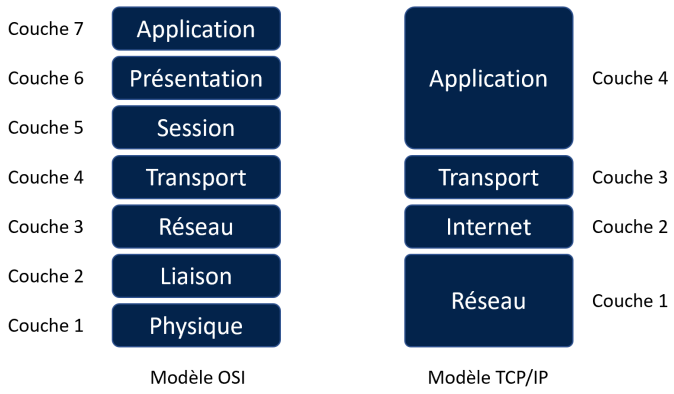


Figure 1.5 – Correspondance des modèles OSI et TCP/IP.

Dans le modèle TCP/IP, les couches OSI cinq, six et sept sont regroupées en une seule. De même, les couches OSI un et deux sont réunies.

1.2.2 Protocoles

La description des quelques protocoles, dont la plupart sont très utiles dans notre contexte de recherche, suivent la décomposition selon le modèle OSI, à l'exception des couches les plus hautes regroupées en une couche application comme dans le modèle TCP/IP.

1.2.2.1 Couches physique et liaison de données

Pour plusieurs technologies, les standards spécifient à la fois la couche physique et la couche liaison de données. C'est pourquoi nous regroupons ces deux couches dans cette section.

Ethernet Un protocole de réseau local filaire appelé Ethernet a été développé au début des années 1970. Le standard IEEE 802.3 (« IEEE Standard for Ethernet », 2018) définit à la fois la couche physique et la couche liaison de données. Ce protocole, ayant un débit à l'origine de 10 Mbps, continue encore aujourd'hui à évoluer. Des versions plus récentes ont été développées pour supporter jusqu'à 10 Gbps.

Le format de la trame Ethernet est donné par la Figure 1.6, extraite du document « IEEE Standard for Ethernet ».

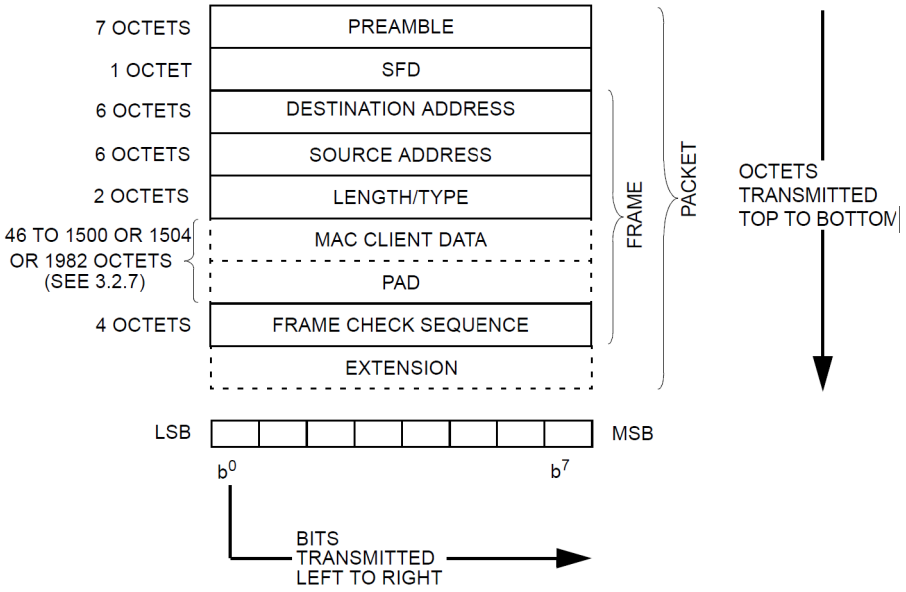


Figure 1.6 – Format de la trame Ethernet (« IEEE Standard for Ethernet », 2018).

La trame est initiée par un préambule, suivi d'un délimiteur de trame (SFD). Les données sont précédées par les adresses MAC de l'émetteur et du destinataire et d'un champ indiquant soit la longueur de la trame, soit un type appelé *Ethertype* et dont les valeurs (Eastlake & Abley, 2013) indiquent le protocole utilisé par la couche réseau.

WIFI Le système de communication sans fil WIFI possède des couches physique et de liaison de données spécifiques, définies par le standard IEEE 802.11 (« IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Redline », 2021). La partie basse de la couche de liaison de données incluse dans ce standard correspond au contrôle d'accès au médium (MAC).

PPP Le protocole point à point (PPP) connecte deux dispositifs sans aucun intermédiaire. Il est utilisé avec un large éventail de support physique, notamment en téléphonie cellulaire. La RFC 1332 (McGregor, 1992) spécifie ce protocole.

LLDP Le protocole de découverte des topologies réseau (LLDP) permet la découverte des topologies réseau de proche en proche et l'échange d'information entre éléments présents sur le réseau. Il tend à remplacer des protocoles propriétaires équivalents. Ce protocole est spécifié par les standards IEEE 802.1ab (« IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery », 2016) et 802.3 (« IEEE Standard for Ethernet », 2018).

CDP Le protocole de découverte Cisco (CDP) révèle les périphériques présents sur un réseau. Il a été développé par l'entreprise Cisco Systems fournissant des appareils réseaux. CDP est un protocole propriétaire procurant les mêmes services que LLDP.

1.2.2.2 Couche réseau

Parmi les protocoles de la couche réseau, nous détaillons les protocoles ARP, RARP, IP, ICMP et IGMP.

ARP et RARP Le protocole de résolution d'adresse (ARP) est utilisé pour obtenir l'adresse de protocole de couche réseau d'un dispositif distant à partir de son adresse de protocole de couche de liaison de données tandis que la fonction équivalente déterminant l'adresse de la couche liaison de données à partir de celle de la couche réseau est réalisée par RARP. ARP est notamment utilisé pour un protocole réseau de type IP sur Ethernet pour identifier l'adresse MAC à partir de l'adresse IP, quand RARP obtient l'adresse IP à partir de l'adresse MAC.

ARP est spécifié dans les RFC 826, 5227 et 5494 (Arkko & Pignataro, 2009 ; Cheshire, 2008 ; Plummer, 1982) tandis que RARP l'est dans la RFC 903 (Finlayson et al., 1984).

IP Le protocole internet (IP) définit un mécanisme d'acheminement de données au mieux, sans établissement de connexion, sur un réseau dont la fiabilité n'est pas assurée. Après plusieurs versions expérimentales, la quatrième version, appelée IPv4, est établie

en tant que standard décrit dans la RFC 791 (Postel, 1981b). Ce protocole fournit trois éléments essentiels :

— le format de toutes les données circulant sur l'internet, composé d'un en-tête suivi des données ;
— la fonction d'adressage ;
— la fragmentation/défragmentation des données lorsque leur taille le nécessite.

Le format de l'en-tête IPv4 est donné par la Figure 1.7. La signification de chaque champ est détaillée dans la RFC 791.

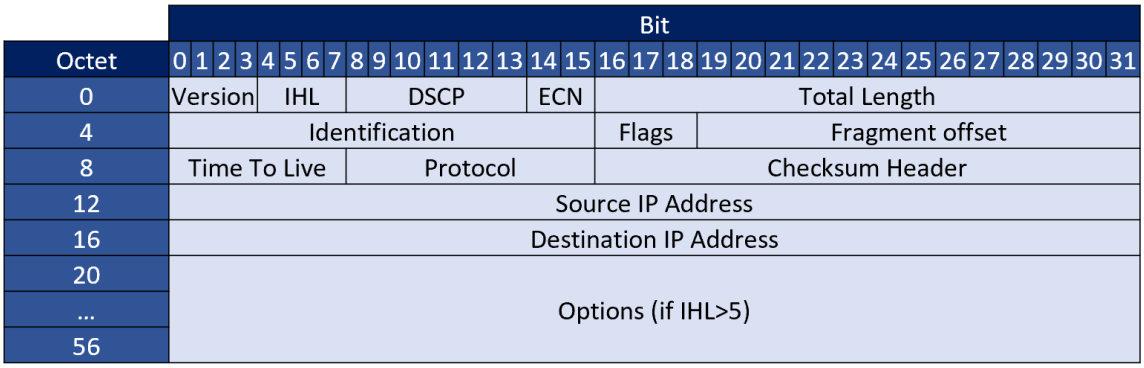


Figure 1.7 – Format de l'en-tête IPv4.

La fragmentation, c'est-à-dire le découpage des données, est gérée avec les champs *Identification*, *Flags* et *Fragment offset*.

Ce format indique que seul l'en-tête est protégé en termes de fiabilité par une somme de contrôle. S'il est nécessaire de protéger les données, un mécanisme doit être mis en place au niveau de la couche transport.

La fonction d'adressage utilise les adresses IP source et destination, encodées sur 32 bits.

Le nombre d'adresses IP arrive à saturation et l'IPv6, avec des adresses sur 128 bits, remplace peu à peu l'IPv4. Le format de l'en-tête obligatoire IPv6 est donné par la Figure 1.8.

Son contenu est simplifié par rapport à l'en-tête IPv4. Toutefois, le champ *Next header* peut indiquer la présence d'en-têtes optionnels pouvant être chaînés. Par défaut, ce champ indique le protocole de transport, mais certaines valeurs indiquent la présence d'information de différents types comme :

— *Hop-by-Hop options* : options devant être examinées et traitées à chaque nœud du réseau, indiquant entre autres les Jumbogrammes pouvant atteindre des tailles de $2^{32} - 1$ octets ;
— *Routing* : méthode pour spécifier le chemin en passant par un ou plusieurs nœuds spécifiques ;
— *Fragment* : paramètres de gestion de la fragmentation afin que le récepteur puisse réassembler les paquets ;

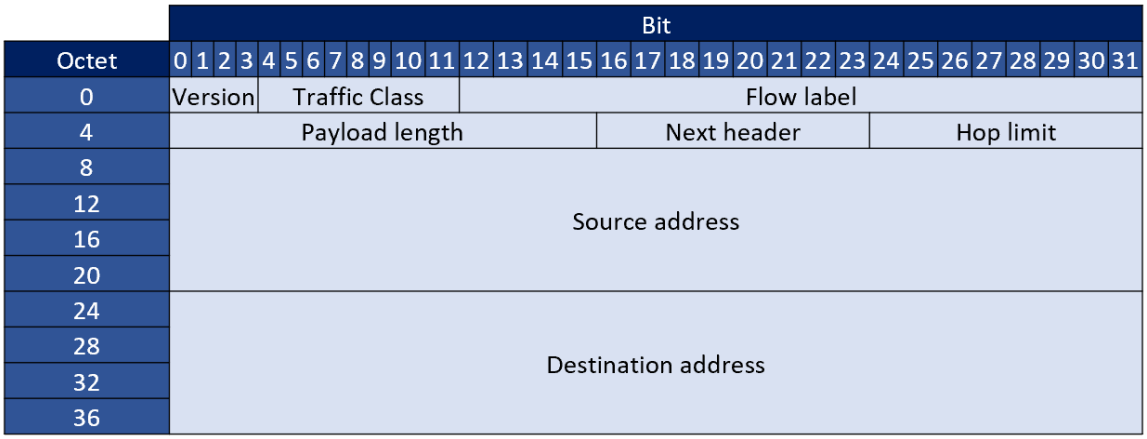


FIGURE 1.8 – Format de l'en-tête IPv6.

— *Authentication header* : information permettant la vérification de l'authenticité d'un paquet pour une communication sécurisée de type IPsec ;
— *Encapsulating Security Payload* : données chiffrées pour une communication sécurisée de type IPsec ;
— *Destination options* : information à destination du récepteur du paquet ;
— *No next header* : indication qu'il n'y a aucune charge utile à suivre.

Un premier niveau d'information sur le contenu des en-têtes est disponible dans la RFC 8200 (DEERING & HINDEN, 2017). Pour une meilleure compréhension du contenu et du rôle exact des en-têtes optionnels, il est nécessaire de se référer aux multiples RFC correspondantes, par exemple les RFC 4302 et 4303 (KENT, 2005a, 2005b) couvrant l'authentification et le chiffrement de la spécification IPsec (KENT & SEO, 2005).

ICMP Le protocole de message de contrôle est un moyen essentiel de remonter des erreurs sur un réseau utilisant le protocole IP. Bien qu'ICMP soit encapsulé dans un datagramme IP, il fait partie intégrante de la couche réseau. IP et ICMP sont indissociables. Le premier se charge du transport des données, mais ne permet pas l'envoi de message d'erreur.

Ce point est notamment clairement mentionné dans la RFC 792 (POSTEL, 1981a) définissant le protocole ICMP.

IGMP Le protocole IP supporte la diffusion à un groupe grâce aux adresses comprises entre 224.0.0.0 et 239.255.255.255. Pour participer à une diffusion IP sur un réseau local, un hôte doit être capable d'envoyer et de recevoir des datagrammes de diffusion. Pour une extension sur plusieurs réseaux, l'hôte doit informer les routeurs locaux qui contactent à leur tour d'autres routeurs, leur transmettent les informations de groupe et établissent les routes nécessaires (COMER, 2000, p. 328). Tout ceci est possible grâce au protocole IGMP. Tout comme ICMP, ce protocole est encapsulé dans un datagramme IP. Toutefois,

son support n'est pas obligatoire.

Les différentes versions d'IGMP sont définies dans les RFC 1112, 2236 et 3376 (Cain et al., 2002 ; Deering, 1989 ; Fenner, 1997).

1.2.2.3 Couche transport

Les adresses IP identifient des ordinateurs ou plus généralement les dispositifs attachés au réseau. L'étape ultime de l'acheminement se fait par la couche transport en spécifiant par un numéro de port l'application concernée par les données échangées. Les numéros de port sont assignés et gérés conformément à la RFC 6335 (Cotton et al., 2011). La Figure 1.9 illustre comment les données des applications sont encapsulées dans les couches de niveaux inférieurs sur un lien Ethernet.

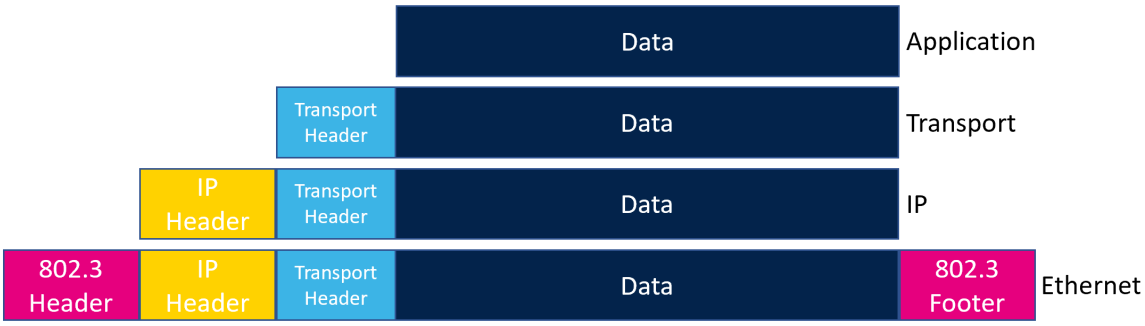


Figure 1.9 – Encapsulation de données.

Parmi les protocoles de transport les plus courants, nous détaillons UDP et TCP et abordons de façon plus succincte le protocole SCTP.

UDP Le protocole UDP fournit un mécanisme simple aux applications qui souhaitent échanger des données sans exiger de fiabilité. Il est orienté message et ne nécessite pas une phase de connexion à l'hôte distant. Ce standard est décrit dans la RFC 768 (Postel, 1980). La Figure 1.10 montre l'en-tête UDP.

	Bit																														
Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source Port																Destination Port															
4	Length																Checksum															

Figure 1.10 – Format de l'en-tête UDP.

Grâce aux numéros de port, un multiplexage/démultiplexage entre les multiples applications et la couche réseau est possible. Le calcul de la somme de contrôle, couvrant l'en-tête et les données, est optionnel et UDP ne possède aucun autre mécanisme de fiabilisation.

TCP Le protocole TCP est plus complexe que UDP et apporte des services supplémentaires. Il fournit les fonctions suivantes :

— le multiplexage/démultiplexage pour les différentes applications accédant au réseau ;
— l'établissement, la gestion et la fermeture de la connexion ;
— le transfert de données avec la couche réseau ;
— la fiabilisation de la connexion ;
— le contrôle de flux et la gestion nécessaire afin d'éviter des congestions.

Compte-tenu de l'ensemble des services apporté par TCP, la complexité de l'en-tête est notable sur la FIGURE 1.11.

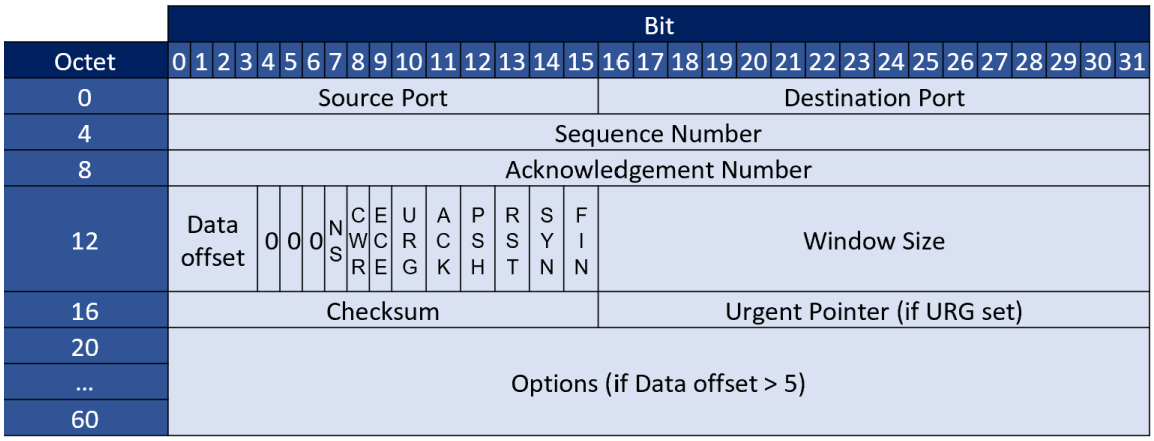


FIGURE 1.11 – Format de l'en-tête TCP.

Le respect des séquences de messages est possible grâce aux numéros de séquences. Les accusés de réception permettent d'identifier les données perdues pour les retransmettre. Ce mécanisme suppose alors une mémorisation des données entre l'envoi et la réception de l'acquittement.

TCP fonctionne en mode connecté. L'établissement d'une session est nécessaire avant l'échange de données. Lorsqu'elle n'est plus utile, la session est fermée. Ces phases sont décrites dans le diagramme d'état de la FIGURE 1.12.

En comparaison à UDP, TCP offre un débit plus réduit. Plusieurs facteurs expliquent cette diminution. Tout d'abord, la taille de l'en-tête réduit la quantité de données utiles pouvant être transportées dans chaque paquet IP. De plus, les gestions d'acquittement, de contrôle de flux, de congestion et de retransmission empêchent une utilisation maximale de la capacité de la couche réseau.

Le protocole TCP est détaillé par la RFC 793 (POSTEL, 1981c), puis les RFC 1122, 316, 6093 et 6528 qui ont apporté de manière incrémentale des améliorations (BRADEN, 1989 ; GONT & BELLOVIN, 2012 ; GONT & YOURTCHENKO, 2011 ; RAMAKRISHNAN et al., 2001).

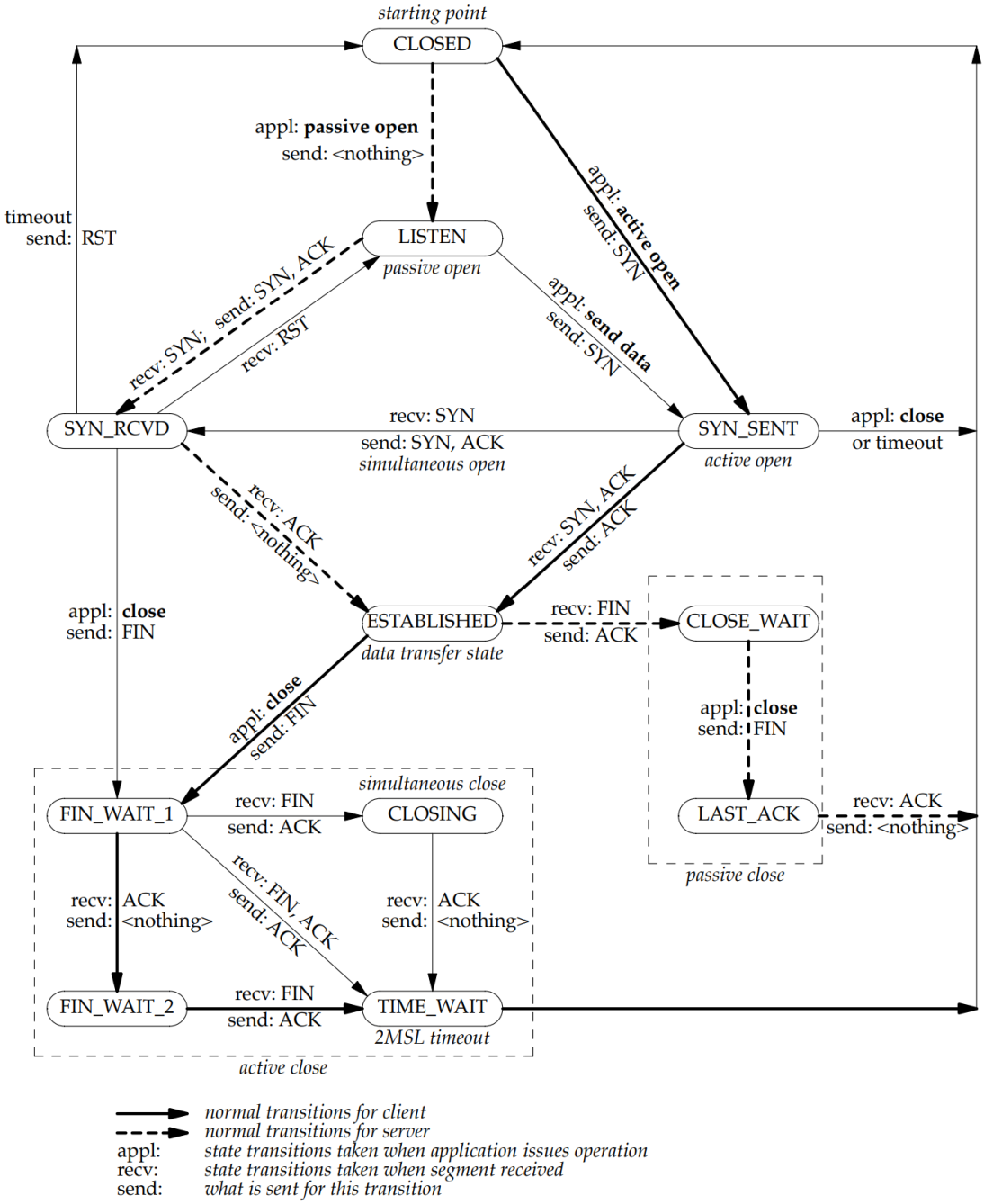


FIGURE 1.12 – Diagramme d'état TCP (WRIGHT & STEVENS, 1995, p. 827).

SCTP Le protocole de transmission de contrôle de flux (SCTP) est un protocole pensé à l'origine pour porter la pile de protocole SS7 utilisée pour la signalisation téléphonique sur IP. Ce protocole reprend des éléments des protocoles UDP et TCP. Comme UDP, il est orienté message. Toutefois, il autorise un transport fiable avec remise en ordre des données et une adaptation de débit afin d'éviter les congestions comme le permet TCP.

SCTP supporte le transfert de données sur de multiples flux. Un message perdu sur un flux n'influencera pas les autres flux. Dans la signalisation téléphonique, la préservation stricte des séquences n'est pas totalement nécessaire. Il suffit que la séquence pour un flux donné soit préservée pour que ce flux continue de fonctionner.

Enfin, un point d'accès SCTP peut utiliser plusieurs adresses IP. Si ces adresses sont routées sur des chemins différents, cette fonctionnalité contribue à améliorer la résilience à une défaillance sur le réseau.

Le protocole est décrit dans la RFC 4960 (Stewart, 2007).

1.2.2.4 Couche application

De nombreux protocoles existent dans la couche application et notre objectif se limite à décrire uniquement ceux qui présentent un intérêt pour les chapitres suivants. Nous traitons donc les protocoles HTTP, FTP et SSH.

HTTP Le protocole de transfert hypertexte HTTP est un protocole client-serveur fondamental pour le *World Wide Web*. Le client est généralement un navigateur web et envoie des requêtes vers un serveur dont les réponses permettent de récupérer des ressources comme des fichiers HTML.

HTTP a évolué avec trois révisions majeures et sont spécifiées dans les RFC 1945, 7540 et 9114 (Belshe et al., 2015 ; Berners-Lee et al., 1996 ; Bishop, 2022).

FTP Le protocole de transfert de fichier FTP est également un protocole client-serveur dont le rôle est, comme son nom l'indique, de transférer les fichiers d'un hôte à un autre. Dans ce contexte, le terme utilisateur FTP est privilégié au terme client FTP.

Le protocole établit deux connexions sur la couche transport TCP. La première, sur le port vingt-et-un, permet le transfert d'informations de contrôle et la deuxième, sur le port vingt, est utilisé pour le transfert de données. FTP nécessite une phase d'authentification de l'utilisateur sur le port de contrôle. Les droits de l'utilisateur servent au contrôle d'accès aux fichiers.

Les détails du protocole FTP sont décrits dans la RFC 969 (Postel & Reynolds, 1985).

SSH Le protocole de shell sécurisé SSH a pour objet d'établir une connexion sécurisée au niveau applicatif sur un canal non sécurisé. Il est basé sur une architecture client-serveur. Le client se connecte au serveur en s'authentifiant grâce à des services de cryptographie asymétrique et de hachage. Une fois la connexion établie, le transfert de données met

en œuvre des algorithmes de cryptographie symétrique afin de garantir le chiffrement et l'intégrité. Les choix des algorithmes et de leurs paramètres sont négociés par les deux parties à l'établissement de la connexion. SSH utilise la couche de transport TCP sur le port vingt-trois.

Les détails du fonctionnement de SSH sont décrits dans les RFC 4250, 4251, 4252, 4253, 4254 (Lehtinen & Lonvick, 2006 ; Ylonen & Lonvick, 2006a, 2006b, 2006c, 2006d).

1.2.3 Sécurisation des communications

Différentes couches du modèle OSI offrent des moyens pour sécuriser les communications. À titre d'exemple, la Figure 1.13 illustre quelques possibilités. SSH ayant déjà été abordé, nous décrivons succinctement les trois autres protocoles mentionnés.

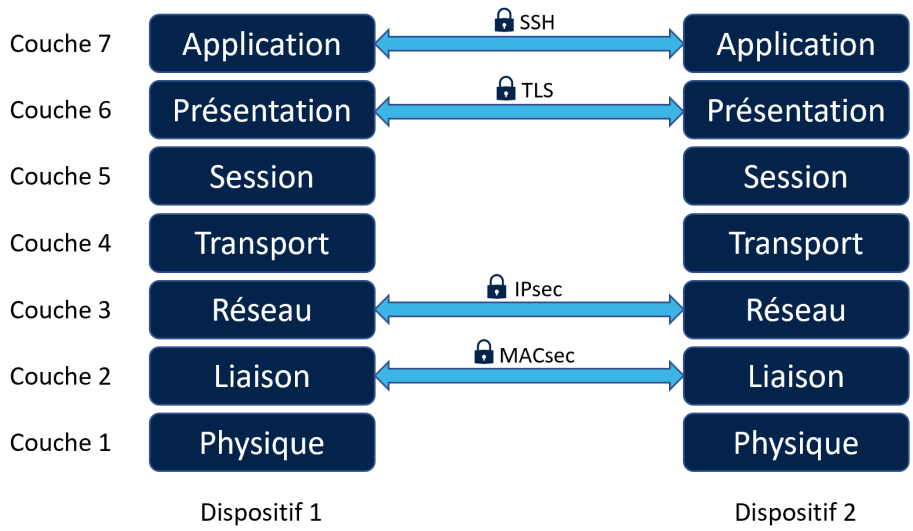


Figure 1.13 – Exemples de communications sécurisées sur le modèle OSI.

Le standard IEEE 802.1ae (« IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security », 2018), appelé MACsec, permet de chiffrer et/ou authentifier les échanges au niveau de la couche liaison de données Ethernet.

Le protocole IPsec sur la couche réseau permet de remplacer le protocole IP en apportant une sécurisation. Le point d'entrée dans les spécifications de ce protocole est la RFC 4301 (Kent & Seo, 2005).

Le protocole de sécurité de la couche de transport TLS sur la couche présentation permet l'établissement d'une connexion sécurisé aux applications, notamment pour le protocole HTTPS (Rescorla, 2000). TLS a évolué pour améliorer le niveau de protection grâce à de meilleurs algorithmes cryptographiques. La version 1.3, la plus récente, est définie dans la RFC 8446 (Rescorla, 2018).

La sécurisation des communications est un élément primordial et doit être utilisée dès que nécessaire. De nombreux objets connectés offrent une interface pour les configurer, au minimum par un système d'identifiant composé d'un nom d'utilisateur et d'un mot de passe. Par négligence ou méconnaissance, des utilisateurs laissent les identifiants par défaut. Ces comportements reviennent à mettre un verrou sur une porte et à laisser la clef sous un pot de fleurs, à portée de main du premier malandrin.

La mise en place d'un pare-feu est une technique largement utilisée, en particulier dans les entreprises. Dans les usages domestiques, le risque est important de retomber dans une configuration incomplète laissant des portes ouvertes.

Dans notre contexte de recherche, un pare-feu n'est pas vraiment une solution envisageable. En effet, ce serait trop lourd d'adapter la configuration à chaque fois qu'un passager dans un véhicule souhaite se connecter à des services extérieurs par le modem cellulaire de la voiture. Avec un dispositif éducatif, le même type de contraintes s'applique. Il est nécessaire de laisser des possibilités de connexions ouvertes. Toutefois, une sécurisation par détection d'intrusions réseau devient nécessaire.

1.3 Synthèse

Dans une première section, les objets connectés ont d'abord été décrits en tant que systèmes embarqués communicants.

Nous avons évoqué des contraintes s'appliquant aux objets connectés, telles que la fiabilité, la réactivité, la sécurité et la sûreté de fonctionnement. Selon le type d'objet connecté, seules une partie d'entre elles peuvent s'appliquer.

Ensuite, les aspects matériels et logiciels ont été couverts en commençant avec des éléments de base, programmables ou non, pouvant être intégrés dans des systèmes d'une complexité variable, allant d'un simple microcontrôleur à un système sur puce.

Enfin, nous avons traité les propriétés de cybersécurité, les menaces, les vulnérabilités et les attaques ainsi que les contre-mesures envisageables tant au niveau matériel que logiciel.

Dans une deuxième section, notre point d'attention s'est porté sur les communications. La description des modèles OSI et TCP/IP ont permis de poser un cadre sur le rôle des différentes couches des piles protocolaires.

Nous avons ensuite décrit un ensemble de protocoles. Les attaques peuvent s'appuyer sur certaines de leurs particularités. De plus, les corpus numériques de détection d'intrusions contiennent des échanges utilisant ces protocoles. Leur connaissance est importante tant pour comprendre les attaques que pour amener des propositions dans les mécanismes de détection d'intrusions réseau.

Pour finir, la sécurisation des communications a été évoquée, montrant qu'il est possible de mettre en place des mécanismes d'authentification, de chiffrement et d'intégrité à différents niveaux du modèle OSI. Cependant, ce n'est pas toujours suffisant pour empêcher les attaques, en particulier sur les couches 6 et 7.

Le chapitre suivant aborde la détection d'intrusions sur des dispositifs communicants. En particulier, il présente différents types d'attaques sur plusieurs protocoles de communication dont certains possèdent un mécanisme de sécurisation, les techniques de détection ainsi que les corpus numériques de ce domaine.

Références

Akram, A. & Sawalha, L., (2019), A Study of Performance and Power Consumption Differences Among Different ISAs, *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 628-632, https://doi.org/10.1109/DSD.2019.00098

Arkko, J. & Pignataro, C., (2009), *IANA Allocation Guidelines for the Address Resolution Protocol (ARP)* (RFC N° 5494), RFC Editor, http://www.rfc-editor.org/rfc/rfc5494.txt

Barkalov, A., Titarenko, L. & Mazurkiewicz, M., (2019), *Foundations of embedded systems*, Springer International Publishing.

Belshe, M., Peon, R. & Thomson, M., (2015), *Hypertext Transfer Protocol Version 2 (HTTP/2)* (RFC N° 7540), RFC Editor, http://www.rfc-editor.org/rfc/rfc7540.txt

Berners-Lee, T., Fielding, R. T. & Nielsen, H. F., (1996), *Hypertext Transfer Protocol – HTTP/1.0* (RFC N° 1945), RFC Editor, http://www.rfc-editor.org/rfc/rfc1945.txt

Bishop, M., (2022), *HTTP/3* (RFC N° 9114), RFC Editor, http://www.rfc-editor.org/rfc/rfc9114.txt

Braden, R., (1989), *Requirements for Internet Hosts - Communication Layers* (STD N° 3), RFC Editor, http://www.rfc-editor.org/rfc/rfc1122.txt

Cain, B., Deering, S., Kouvelas, I., Fenner, B. & Thyagarajan, A., (2002), *Internet Group Management Protocol, Version 3* (RFC N° 3376), RFC Editor, http://www.rfc-editor.org/rfc/rfc3376.txt

Cheshire, S., (2008), *IPv4 Address Conflict Detection* (RFC N° 5227), RFC Editor, http://www.rfc-editor.org/rfc/rfc5227.txt

Comer, D. E., (2000), *Internetworking with TCP/IP Vol.1 : Principles, Protocols, and Architecture* (4e éd.), Pearson.

Conte, G., Tommesani, S. & Zanichelli, F., (2000), The long and winding road to high-performance image processing with MMX/SSE, *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, 302-310, https://doi.org/10.1109/CAMP.2000.875989

Coppola, M., Kornaros, G., Andrade, L. & Rousseau, F., (2021), Automation for industry 4.0 by using secure lorawan edge gateways, *Multi-Processor System-on-Chip*, *2*, 49-66.

Cotton, M., Eggert, L., Touch, J., Westerlund, M. & Cheshire, S., (2011), *Internet Assigned Numbers Authority (IANA) Procedures for the Management of*

the Service Name and Transport Protocol Port Number Registry (BCP N° 165), RFC Editor, http://www.rfc-editor.org/rfc/rfc6335.txt

Dang, L. M., Piran, M. J., Han, D., Min, K. & Moon, H., (2019), A Survey on Internet of Things and Cloud Computing for Healthcare, *Electronics*, *8 7*, https://doi.org/10.3390/electronics8070768

Debouk, R. et al., (2018), Overview of the 2nd Edition of ISO 26262 : Functional safety-road vehicles, *General Motors Company, Warren, MI, USA*.

Deering, S. & Hinden, R., (2017), *Internet Protocol, Version 6 (IPv6) Specification* (STD N° 86), RFC Editor, http://www.rfc-editor.org/rfc/rfc8200.txt

Deering, S., (1989), *Host extensions for IP multicasting* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc1112.txt

Eastlake, D. & Abley, J., (2013), *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters* (BCP N° 141), RFC Editor, http://www.rfc-editor.org/rfc/rfc7042.txt

Ejaz, W. & Anpalagan, A., (2019), Internet of Things for Smart Cities : Overview and Key Challenges. *Internet of Things for Smart Cities : Technologies, Big Data and Security* (p. 1-15), Springer International Publishing, https://doi.org/10.1007/978-3-319-95037-2_1

Elnawawy, M., Farhan, A., Nabulsi, A. A., Al-Ali, A. & Sagahyroon, A., (2019), Role of FPGA in Internet of Things Applications, *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 1-6, https://doi.org/10.1109/ISSPIT47144.2019.9001747

Fenner, W. C., (1997), *Internet Group Management Protocol, Version 2* (RFC N° 2236), RFC Editor, http://www.rfc-editor.org/rfc/rfc2236.txt

Finlayson, R., Mann, T., Mogul, J. & Theimer, M., (1984), *A Reverse Address Resolution Protocol* (STD N° 38), RFC Editor, http://www.rfc-editor.org/rfc/rfc903.txt

Friha, O., Ferrag, M. A., Shu, L., Maglaras, L. & Wang, X., (2021), Internet of Things for the Future of Smart Agriculture : A Comprehensive Survey of Emerging Technologies, *IEEE/CAA Journal of Automatica Sinica*, *8*, 718-752.

Gont, F. & Bellovin, S., (2012), *Defending against Sequence Number Attacks* (RFC N° 6528), RFC Editor, http://www.rfc-editor.org/rfc/rfc6528.txt

Gont, F. & Yourtchenko, A., (2011), *On the Implementation of the TCP Urgent Mechanism* (RFC N° 6093), RFC Editor, http://www.rfc-editor.org/rfc/rfc6093.txt

Henzinger, T. A. & Sifakis, J., (2007), The Discipline of Embedded Systems Design, *Computer*, *40 10*, 32-40, https://doi.org/10.1109/MC.2007.364

Herculano-Houzel, S., (2020), Chapter 33 - Remarkable, But Not Special : What Human Brains Are Made of, In J. H. Kaas (Éd.), *Evolutionary Neuroscience (Second Edition)* (Second Edition, p. 803-813), Academic Press, https://doi.org/10.1016/B978-0-12-820584-6.00033-7

IEEE Standard for Ethernet, (2018), *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, 1-5600, https://doi.org/10.1109/IEEESTD.2018.8457469

IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Redline, (2021), *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline*, 1-7524.

IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery, (2016), *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, 1-146, https://doi.org/10.1109/IEEESTD.2016.7433915

IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security, (2018), *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, 1-239, https://doi.org/10.1109/IEEESTD.2018.8585421

ISO, (2009), ISO/IEC 15408 : Information technology — Security techniques — Evaluation criteria for IT security, Récupérée 3 août 2022, à partir de https://www.iso.org/standard/50341.html

ISO, (2021), ISO/SAE 21434 : Road vehicles — Cybersecurity engineering, Récupérée 3 août 2022, à partir de https://www.iso.org/standard/70918.html

ITU, (2021), Measuring digital development, Facts and figures, Récupérée 26 juillet 2022, à partir de https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2021.pdf

Jabbar, W. A., Wei, C. W., Azmi, N. A. A. M. & Haironnazli, N. A., (2021), An IoT Raspberry Pi-based parking management system for smart campus, *Internet of Things*, *14*, 100387, https://doi.org/10.1016/j.iot.2021.100387

Kent, S., (2005a), *IP Authentication Header* (RFC N° 4302) [http://www.rfc-editor.org/rfc/rfc4302.txt], RFC Editor, RFC Editor, http://www.rfc-editor.org/rfc/rfc4302.txt

Kent, S., (2005b), *IP Encapsulating Security Payload (ESP)* (RFC N° 4303) [http://www.rfc-editor.org/rfc/rfc4303.txt], RFC Editor, RFC Editor, http://www.rfc-editor.org/rfc/rfc4303.txt

Kent, S. & Seo, K., (2005), *Security Architecture for the Internet Protocol* (RFC N° 4301), RFC Editor, http://www.rfc-editor.org/rfc/rfc4301.txt

Kong, X., Wang, C., Sun, S. & Gao, F., (2012), A Method of Digital to Shaft-Angle Converting Using ARM Series MCU & CPLD, *2012 Fifth International Symposium on Computational Intelligence and Design*, *2*, 262-265, https://doi.org/10.1109/ISCID.2012.217

Krishna, R. R., Priyadarshini, A., Jha, A. V., Appasani, B., Srinivasulu, A. & Bizon, N., (2021), State-of-the-Art Review on IoT Threats and Attacks : Taxonomy, Challenges and Solutions, *Sustainability*, *1316*, https://doi.org/10.3390/su13169463

Kumar, S., Dalal, S. & Dixit, V., (2014), The OSI model : Overview on the seven layers of computer networks, *International Journal of Computer Science and Information Technology Research*, *2 3*, 461-466.

Kurachi, R., Takada, H., Tanabe, M., Anzai, J., Takei, K., Iinuma, T., Maeda, M. & Matsushima, H., (2018), Improving secure coding rules for automotive software by using a vulnerability database, *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 1-8, https://doi.org/10.1109/ICVES.2018.8519496

Lee, E. A., (2007), *Computing Foundations and Practice for Cyber-Physical Systems : A Preliminary Report* (rapp. tech. UCB/EECS-2007-72), EECS Department, University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-72.html

Lehtinen, S. & Lonvick, C., (2006), *The Secure Shell (SSH) Protocol Assigned Numbers* (RFC N° 4250), RFC Editor, http://www.rfc-editor.org/rfc/rfc4250.txt

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H. & Zhao, W., (2017), A Survey on Internet of Things : Architecture, Enabling Technologies, Security and Privacy, and Applications, *IEEE Internet of Things Journal*, *4 5*, 1125-1142, https://doi.org/10.1109/JIOT.2017.2683200

Madakam, S., Lake, V., Lake, V., Lake, V. et al., (2015), Internet of Things (IoT) : A literature review, *Journal of Computer and Communications*, *3 05*, 164.

Marikyan, D., Papagiannidis, S. & Alamanos, E., (2019), A systematic review of the smart home literature : A user perspective, *Technological Forecasting and Social Change*, *138*, 139-154, https://doi.org/10.1016/j.techfore.2018.08.015

Marwedel, P., (2021), *Embedded System Design : Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things* (4th), Springer Publishing Company, Incorporated, https://doi.org/10.1007/978-3-030-60910-8

McGregor, G., (1992), *The PPP Internet Protocol Control Protocol (IPCP)* (RFC N° 1332), RFC Editor, http://www.rfc-editor.org/rfc/rfc1332.txt

Miller, C., (2019), Lessons learned from hacking a car, *IEEE Design & Test*, *36 6*, 7-9, https://doi.org/10.1109/MDAT.2018.2863106

NXP, (2021), i.MX RT1170 CROSSOVER MCUs Ushering in the GHZ ERA, Récupérée 8 janvier 2021, à partir de https://www.nxp.com/docs/en/fact-sheet/i.MX-RT1170-FS.pdf

Plummer, D. C., (1982), *Ethernet Address Resolution Protocol : Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware* (STD N° 37), RFC Editor, http://www.rfc-editor.org/rfc/rfc826.txt

Postel, J., (1980), *User Datagram Protocol* (STD N° 6), RFC Editor, http://www.rfc-editor.org/rfc/rfc768.txt

Postel, J., (1981a), *Internet Control Message Protocol* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc792.txt

Postel, J. & Reynolds, J., (1985), *File Transfer Protocol* (STD N° 9), RFC Editor, http://www.rfc-editor.org/rfc/rfc959.txt

Postel, J., (1981b), *Internet Protocol* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc791.txt

Postel, J., (1981c), *Transmission Control Protocol* (STD N° 7), RFC Editor, http://www.rfc-editor.org/rfc/rfc793.txt

Ramakrishnan, K., Floyd, S. & Black, D., (2001), *The Addition of Explicit Congestion Notification (ECN) to IP* (RFC N° 3168), RFC Editor, http://www.rfc-editor.org/rfc/rfc3168.txt

Rayes, A. & Salam, S., (2017), The Internet in IoT—OSI, TCP/IP, IPv4, IPv6 and Internet Routing. *Internet of Things From Hype to Reality : The Road to Digitization* (p. 35-56), Springer International Publishing, https://doi.org/10.1007/978-3-319-44860-2_2

Rescorla, E., (2000), *HTTP Over TLS* (RFC N° 2818), RFC Editor, http://www.rfc-editor.org/rfc/rfc2818.txt

Rescorla, E., (2018), *The Transport Layer Security (TLS) Protocol Version 1.3* (RFC N° 8446), RFC Editor, http://www.rfc-editor.org/rfc/rfc8446.txt

Rizvi, S., Kurtz, A., Pfeffer, J. & Rizvi, M., (2018), Securing the Internet of Things (IoT) : A Security Taxonomy for IoT, *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering*, 163-168, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00034

Romkey, J., (2017), Toast of the IoT : The 1990 Interop Internet Toaster, *IEEE Consumer Electronics Magazine*, *6 1*, 116-119, https://doi.org/10.1109/MCE.2016.2614740

Segars, S., (1997), ARM7TDMI power consumption, *IEEE Micro*, *174*, 12-19, https://doi.org/10.1109/40.612178

Serpanos, D., (2019), There Is No Safety Without Security and Dependability, *Computer*, *52 6*, 78-81, https://doi.org/10.1109/MC.2019.2903360

Shah, D. K., B., C. R., Sen, D. & Goyal, S., (2015), Vehicle parking system implementation using CPLD, *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, 1-5, https://doi.org/10.1109/ICCICT.2015.7045668

Sharma, N., Shamkuwar, M. & Singh, I., (2019), The History, Present and Future with IoT, In V. E. Balas, V. K. Solanki, R. Kumar & M. Khari (Éd.), *Internet of Things and Big Data Analytics for Smart Generation* (p. 27-51), Springer International Publishing, https://doi.org/10.1007/978-3-030-04203-5_3

Smith, S., (2020), Neon Coprocessor. *Programming with 64-Bit ARM Assembly Language : Single Board Computer Development for Raspberry Pi and Mobile Devices* (p. 291-306), Apress, https://doi.org/10.1007/978-1-4842-5881-1_13

Stewart, R., (2007), *Stream Control Transmission Protocol* (RFC N° 4960), RFC Editor, http://www.rfc-editor.org/rfc/rfc4960.txt

STMicroelectronics, (2020), STM32MP1 Series Microprocessors, Récupérée 3 août 2022, à partir de https://www.st.com/content/ccc/resource/sales_and_marketi

ng/promotional_material/flyer/group0/1f/4c/20/0a/43/94/4f/70/flstm32mp/files/flstm32mp.pdf/jcr:content/translations/en.flstm32mp.pdf

STMicroelectronics, (2022), Data Brief - Stellar SR6 G7 line, Récupérée 26 juillet 2022, à partir de https://www.st.com/resource/en/data_brief/sr6g7c4.pdf

Suresh, P., Daniel, J. V., Parthasarathy, V. & Aswathy, R. H., (2014), A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment, *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, 1-8, https://doi.org/10.1109/ICSEMR.2014.7043637

Van, H. P., (2021), *State of IoT - summer report*. Récupérée 26 août 2022, à partir de https://iot-analytics.com/product/state-of-iot-summer-2021/

Wright, G. R. & Stevens, W. R., (1995), *TCP/IP Illustrated, Volume 2 (paperback) : The Implementation*, Addison-Wesley Professional.

Xu, H., Yu, W., Griffith, D. & Golmie, N., (2018), A Survey on Industrial Internet of Things : A Cyber-Physical Systems Perspective, *IEEE Access*, *6*, 78238-78259, https://doi.org/10.1109/ACCESS.2018.2884906

Xu, J., Wang, B., Li, J., Hu, C. & Pan, J., (2017), Deep learning application based on embedded GPU, *2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS)*, 1-4, https://doi.org/10.1109/EIIS.2017.8298723

Ylonen, T. & Lonvick, C., (2006a), *The Secure Shell (SSH) Authentication Protocol* (RFC N° 4252), RFC Editor, http://www.rfc-editor.org/rfc/rfc4252.txt

Ylonen, T. & Lonvick, C., (2006b), *The Secure Shell (SSH) Connection Protocol* (RFC N° 4254), RFC Editor, http://www.rfc-editor.org/rfc/rfc4254.txt

Ylonen, T. & Lonvick, C., (2006c), *The Secure Shell (SSH) Protocol Architecture* (RFC N° 4251), RFC Editor, http://www.rfc-editor.org/rfc/rfc4251.txt

Ylonen, T. & Lonvick, C., (2006d), *The Secure Shell (SSH) Transport Layer Protocol* (RFC N° 4253), RFC Editor, http://www.rfc-editor.org/rfc/rfc4253.txt

Chapitre 2

Détection d'intrusions réseau

« Ce que font les hackeurs, c'est comprendre la technologie et l'expérimenter d'une manière que beaucoup de gens n'auraient jamais imaginée. »

Eric "Emmanuel Goldstein" Corley
Journaliste, éditeur et figure du Hacking (1959 –)

Ce travail de recherche a pour but d'amener des contributions novatrices sur la détection d'intrusions et comme doivent le faire les hackeurs, nous devons d'abord comprendre la technologie pour lancer ensuite les expériences qui pourront confirmer ou infirmer nos hypothèses.

Dans ce chapitre, nous commençons par une description des familles de cyberattaques en rapport avec les intrusions sur les réseaux. Dans un deuxième temps, les techniques de détection d'intrusions seront décrites. Pour finir, nous abordons les corpus numériques qui permettent l'étude des systèmes de détection d'intrusions.

2.1 Cyberattaques d'intrusions réseau

Nous décrivons dans cette section les différentes façons de s'introduire dans un réseau informatique ainsi que quelques exemples d'outils pouvant être facilement trouvés sur l'internet. De tels programmes sont disponibles dans la distribution Kali Linux (Kali, 2017), une suite d'outils de tests de pénétration destinée à aider les professionnels de la sécurité informatique lors de tests d'infrastructure informatique ou lors d'audits de sécurité. Les classifications des attaques de cybersécurité ont été étudiées (Derbyshire et al., 2018 ; Simmons et al., 2009) et aboutissent à différentes taxonomies, souvent comprenant différentes dimensions. L'objectif ici n'est pas de faire une revue complète des taxonomies, mais de faciliter la compréhension des corpus numériques étudiés en section 2.3. Pour cela, nous utilisons une classification selon la méthode d'opération telle que décrite dans (Kjaerland, 2006). Cette classification se réfère à différentes méthodes utilisées par les hackeurs pour porter des attaques. Chacun des paragraphes suivants reprennent une de ces méthodes dans le but d'expliquer les attaques et certains outils permettant de les mener.

Déni de Service (DoS) ou Déni de Service Distribué (DDoS) Ces attaques ont pour but de perturber les services. Pour ce faire, elles inondent généralement les ressources

du système par différentes techniques. Des exemples de tels outils pour générer les ensembles de données sont Low and High Orbic Ion Cannon, GoldenEye, Hulk, Slowloris, SlowHttpTest.

Goldeneye est un outil développé en Python pour effectuer une attaque DoS au niveau HTTP, en utilisant les options *KeepAlive* et *Cache-Control* jusqu'à ce que tous les sockets du serveur HTTP soient consommés. Slowloris est une attaque à faible bande passante sur HTTP. En envoyant des requêtes HTTP partielles, elle maintient la connexion ouverte et surcharge les ressources de la cible. De plus amples informations sur Slowloris et Goldeneye sont disponibles dans (Shorey et al., 2018). D'autres outils comme Hulk, SlowHTTPtest, Low and High Orbic Ion Cannon (LOIC et HOIC) permettent de mener des attaques DoS/DDoS.

Compromission d'un utilisateur Cette catégorie d'attaques vise à découvrir un mot de passe afin d'obtenir un accès à un compte informatique (Bošnjak et al., 2018) pour accéder aux données de l'utilisateur et potentiellement chercher le mot de passe administrateur par la suite. Ces attaques peuvent être exécutées avec des outils comme Hydra, Ncrack et Patator pour SSH ou FTP.

Compromission de l'administrateur Ce type d'attaque permet d'obtenir la totalité des droits d'administration. Avec des droits supérieurs à ceux de l'utilisateur normal, l'attaquant a la liberté de faire tout ce qu'il souhaite sur le système d'information, comme modifier les paramètres, installer ou supprimer des logiciels, mettre à niveau ou rétrograder le micrologiciel ou le système d'exploitation, lire, écrire ou supprimer des fichiers. Ces attaques peuvent consister à découvrir un mot de passe de manière similaire à un compte utilisateur. Une alternative vise à obtenir en premier lieu les droits utilisateurs ou à installer un logiciel malveillant de type rootkit puis, par une élévation de privilège, obtenir les droits administrateur.

Compromission d'un site Web Les serveurs Web peuvent contenir des vulnérabilités exploitées pour créer une attaque. Les serveurs d'applications peuvent être attaqués par force brute, par exemple en essayant de nombreuses paires d'identifiant et de mot de passe à partir d'un dictionnaire ou en injectant des chaînes spécifiques dans les champs de saisie. Une telle chaîne peut être un script qui sera exécuté en cas de cross-site scripting (XSS) ou une requête SQL malformée. Les vulnérabilités et les attaques par injection SQL ont été décrites dans (Fonseca et al., 2009). Les attaques contre les serveurs d'applications ne se limitent pas aux injections XSS et SQL. Une taxonomie des attaques Web est fournie dans (Lai et al., 2008). Selenium est un outil de test généralement utilisé pour interagir avec les pages Web de manière automatique. Son usage peut être détourné pour automatiser des attaques.

L'attaque Heartbleed n'est pas une compromission de site Web, mais comme il est basé sur une connexion sécurisée, par exemple une communication basée sur SSH, nous prenons le parti d'étendre cette catégorie pour couvrir Heartbleed. Cette célèbre attaque

utilise une vulnérabilité dans certaines versions d'une boîte à outils cryptographiques appelée OpenSSL, notamment utilisée à l'établissement de la connexion sécurisée. Cette vulnérabilité peut être utilisée pour voler des informations sensibles. (Durumeric et al., 2014) détaille la vulnérabilité et l'attaque.

Reconnaissance Cette catégorie ne correspond pas tout à fait à une attaque dans le sens où le but est de collecter de l'information avant de lancer une attaque (Yadav & Rao, 2015). Ces reconnaissances consistent à scanner ou à sonder les réseaux pour découvrir les services disponibles. Il est possible de scanner des réseaux depuis l'internet, mais aussi depuis un réseau local après l'installation d'un logiciel malveillant.

Nmap, SCTPscan ou unicornscan sont des outils bien connus pour scanner les réseaux à des fins légitimes mais sont également utilisés pour lancer des attaques de scan de ports.

Installation de logiciel malveillant Les logiciels malveillants aussi dénommés *malwares* (version contractée de l'anglais *malicious software*) correspondent à tout type de programme ou de fichier nuisible pour un ordinateur. Les objectifs de ces *malwares* varient (Vignau et al., 2021) et recouvrent du déni de service, de l'espionnage, des intérêts financiers ou encore de la distribution d'attaques.

Les *malwares* prennent plusieurs formes et varient en termes de dangerosité. Les virus informatiques sont des programmes ou des morceaux de programmes qui s'insèrent dans un autre logiciel lors de son exécution. Ils se diffusent par partage de fichiers entre ordinateurs. Les vers se répliquent également et détruisent des fichiers présents sur la machine infectée. Certains *malwares* se cachent sous l'apparence d'un programme normal, ce sont des chevaux de Troie. Un botnet est un réseau de bots, des programmes permettant d'automatiser des tâches plus rapidement qu'un humain. Les bots sont des programmes exécutant de manière automatique des tâches sur des ordinateurs. La particularité d'un botnet consiste à utiliser des bots malveillants sur des ordinateurs infectés et connectés à l'internet. Les réseaux de bots se composent de trois éléments : les zombies, les serveurs de commande et de contrôle (C&C) et le botmaster. Les zombies se connectent au C&C et le botmaster peut interagir avec les bots pour lancer la mise à jour du code du bot et lancer des actions malveillantes à partir des bots. Certaines de leurs tâches typiques consistent à envoyer des spams, à exécuter des dénis de service distribués et à voler des informations sensibles. Le cycle de vie et les activités des botnets sont décrits dans (Eslahi et al., 2012). Les rançongiciels aussi appelés *ransomwares* (de l'anglais *ransom software*) bloquent l'utilisation de l'ordinateur infecté et demande le paiement d'une rançon pour déverrouiller le système informatique. Parmi les moins néfastes des *malwares*, les logiciels espions sont des logiciels qui suivent les activités de l'utilisateur sur l'internet afin d'envoyer des publicités ciblées. Ces logiciels sont souvent connus sous leurs noms anglais de *spyware* et *adware*.

Les attaques de cette catégorie ont la caractéristique commune d'être exécutées en deux étapes. Tout d'abord, un logiciel malveillant ou un fichier déclenchant une faille

de sécurité dans une application est chargé, par exemple à partir d'un site Web, d'une pièce jointe à un courriel ou encore dans un fichier sur un support de mémoire non volatile amovible. Ensuite, le fichier malveillant s'exécute ou la vulnérabilité de l'application ouvre une porte dérobée qui peut être utilisée par un pirate distant, par exemple pour analyser le réseau interne à la recherche d'autres vulnérabilités.

La distribution Kali Linux contient un outil appelé Metasploit-framework fournissant une liste d'exploits, des modules permettant de lancer des reconnaissances, mais aussi de générer des *shellcodes* et d'insérer des exploits dans des fichiers.

2.2 Techniques de détection d'intrusions réseau

Afin de détecter les cyberattaques, des systèmes adaptés, appelés systèmes de détection d'intrusions (IDS selon l'acronyme anglais pour *Intrusion Detection System*) ont été développés.

2.2.1 Aperçu général des IDS

Un système de détection d'intrusions surveille dynamiquement les événements qui ont lieu dans un environnement et décide s'ils sont symptomatiques d'une attaque ou bien constituent une utilisation légitime de cet environnement (Debar et al., 1999). Plusieurs critères des IDS permettent de les classer (Milenkoski et al., 2015).

Plateforme de surveillance Un IDS qui s'exécute directement sur un hôte surveille le système sur lequel il est déployé pour détecter les attaques locales. Dans un tel IDS, la surveillance se fait par l'analyse de fichiers de log en particulier via les enchainements d'appels à des fonctions systèmes. *A contrario*, un IDS surveillant le trafic depuis un nœud de connexion réseau détecte les attaques à distance. Une approche hybride consiste à combiner un IDS sur un hôte et un IDS surveillant le réseau.

Méthode de détection d'attaques Un IDS qui détecte les attaques par une mauvaise utilisation du système ou du réseau évalue les activités par rapport à un ensemble de signatures d'attaques connues. Une alerte est générée lorsqu'un motif particulier est reconnu. Alternativement, un IDS qui analyse le profil d'activité détecte les attaques en le comparant à un profil de référence, une sorte de modèle de normalité. Cette deuxième méthode détecte les anomalies en se basant sur une représentation latente construite à partir des caractéristiques du trafic surveillé. Lorsque le profil mesuré s'éloigne suffisamment du modèle de référence, une anomalie que l'on va attribuer à une attaque peut être considérée. Là encore, un modèle hybride est possible en combinant les deux méthodes de détection précédemment citées.

Architecture de déploiement Lorsque son déploiement est localisé sur une seule machine, l'IDS est dit centralisé. Il est possible de déployer un ensemble de détection d'intrusions sur plusieurs machines qui collaborent pour détecter les attaques et ainsi obtenir un IDS distribué.

Les objets connectés partagent comme caractéristique d'être regroupés en réseaux. Comme notre activité de recherche se concentre sur ce type d'objets intelligents, nous avons décidé de focaliser notre étude sur les IDS surveillants les activités réseau, appelés NIDS pour Network-based Intrusion Detection System.

2.2.2 IDS basés sur la surveillance réseau

Comme tous les IDS, ceux basés sur une surveillance réseau peuvent utiliser deux méthodes de détection des attaques précitées et que nous allons maintenant détailler. Par souci de simplification, lorsque nous parlons d'IDS, sans autre précision, dans la suite de ce document, il s'agit d'un système de détection d'intrusions réseau selon la définition 3.

Définition 3: IDS

Un IDS est un système de détection d'intrusions, un processus d'identification d'activités malicieuses visant un ordinateur ou des ressources en réseau (Kruegel et al., 2004), dans le but de discriminer les tentatives d'intrusions d'un usage normal. Parmi les différents types d'IDS existant, notre définition se limite aux systèmes basés sur la surveillance réseau.

2.2.2.1 IDS basé sur les paquets

Ces IDS s'appuient sur une analyse de chaque paquet reçu sur le lien de communication, portant sur l'en-tête et la charge utile. Cette technique est connue sous l'acronyme anglais DPI (Deep Packet Inspection). Avec l'augmentation des débits sur les connexions Ethernet, les techniques basées sur l'analyse des paquets, avec des outils tels que Snort (Roesch, 1999) et Bro (Paxson, 1999) - maintenant appelé Zeek - atteignent des limites (Bul'ajoul et al., 2015 ; Ming Gao et al., 2006). En effet, plus le nombre de paquets à analyser sur un intervalle de temps est élevé, plus la quantité de ressources pour les traiter sans goulet d'étranglement augmente.

Pour illustrer le propos, nous considérons l'utilisation d'un lien Ethernet à 100 Mbps. Nous prenons une taille maximum de la charge utile de 1 500 octets, soit une taille de paquet de 1 522 octets et la taille minimale de 64 octets. Cette taille minimale est typique d'une attaque testant les ports ouverts sur un réseau. La Table 2.1 indique la périodicité minimale à laquelle chaque paquet Ethernet doit être analysé pour ces deux tailles de paquets. D'un côté, les paquets peuvent arriver toutes les 5.12 µs et l'analyse sera rapide, car limitée à 64 octets. De l'autre côté, la périodicité des paquets est moins critique, mais l'analyse portera sur 1 522 octets.

Table 2.1 – Périodicité minimale des paquets Ethernet à 100 Mbps.

Taille (octect)	**Périodicité minimale** (µs)
64	5,12
1 522	121,76

Lorsque le débit augmente à 1 Gbps, la périodicité est divisée par dix. En effet, même si certaines implémentations supportent une charge utile jusqu'à 64 kilo-octets, cet usage n'est pas répandu. C'est notamment le cas dans le domaine automobile avec certains constructeurs qui impose une charge utile maximale de 1 500 octets. Dans le pire cas, il faut être capable d'analyser un paquet de 64 octets toutes les 512 ns, un temps correspondant approximativement à deux cents cycles horloge d'un processeur Cortex-R52 à 400 MHz comme celui présent dans le microcontrôleur Stellar (STMicroelectronics, 2022). C'est une contrainte extrêmement forte pour un système embarqué.

2.2.2.2 IDS basé sur les conversations

Face au problème identifié lorsque les analyses portent sur les paquets, une approche consistant à analyser des conversations semble intéressante. Avant d'aller plus loin, il est utile de définir ce qu'est une conversation en nous basant sur celle donnée par le groupe de travail IPFIX au sein de l'IETF (Claise et al., 2013).

Définition 4: Une conversation

Une conversation est définie comme un ensemble de paquets IP passant par un point d'observation du réseau pendant un intervalle donné. Tous les paquets appartenant à une conversation particulière possèdent un ensemble de propriétés communes. Chaque propriété est définie comme le résultat de l'application d'une fonction sur les valeurs de :

- au moins un champ de l'en-tête du paquet ;
- au moins une caractéristique du paquet ;
- au moins un champ dérivé du traitement du paquet.

Les propriétés communes à chaque conversation permettent de définir une clef identifiant la conversation. Un exemple commun consiste à utiliser une combinaison des adresses IP source et destination, des ports de transport source et destination et du protocole de transport pour identifier un groupe de paquet correspondant à une communication dans une direction donnée.

Avec cette définition, une connexion TCP génèrera systématiquement deux conversations, une dans chaque direction, alors qu'une diffusion en continu (streaming) utilisant un protocole de transport comme l'UDP peut créer une seule conversation. L'IETF a étendu la définition à des conversations bidirectionnelles (Trammell & Boschi, 2008),

ce qui permet d'avoir une seule conversation par connexion TCP.

Un système de détection d'intrusions basé sur les conversations réseau nécessite un système d'extraction des caractéristiques décrivant ces conversations qui seront ensuite utilisées afin de détecter les éventuelles attaques. À titre d'exemple, ces méta-données peuvent indiquer, entre autres, le nombre de paquets échangés, le débit moyen, la durée des conversations, les statistiques sur les temps d'arrivée inter-paquet comme les temps moyen, minimum, maximum et leur écart-type.

Ce type d'IDS possède donc deux activités. La première extrait les informations des en-têtes de chaque paquet pour mettre à jour les caractéristiques de chaque conversation. tandis que la seconde va chercher à identifier la nature de la conversation pour détecter les attaques. Dans un système contraint comme un objet connecté, cette méthode est moins consommatrice de ressources et donc plus adaptée.

2.3 Corpus numériques de détection d'intrusions réseau

L'étude des cyberattaques sur des réseaux nécessite de disposer d'enregistrements contenant du trafic normal et du trafic correspondant à des attaques.

Dans cette section, nous abordons d'abord ce sujet de façon générale. Ensuite, un aperçu chronologique de quelques corpus numériques susceptibles de correspondre aux besoins de notre étude sont évoqués. Enfin, les plus pertinents sont ensuite étudiés de façon plus détaillée.

2.3.1 Généralités

Les premiers corpus d'intrusions réseau sont apparus avec le développement des réseaux d'ordinateurs puis se sont diversifiés avec l'interconnexion de différents appareils électroniques. Comme illustré par la Figure 1 du chapitre Introduction, les objets connectés utilisent tout un panel d'interfaces réseau. En conséquence, il existe différents corpus d'intrusions réseau pour des attaques spécifiques à certaines interfaces de communication, ou encore se limitant à certains types d'attaques spécifiques.

Un exemple typique concerne les objets connectés sur des réseaux LPWAN qui utilisent un protocole de routage adapté aux réseaux basse consommation et avec pertes, appelé RPL (pour Routing Protocol for Low power and lossy network) et défini par l'IETF (Winter et al., 2012). Un corpus spécifique, RPL-NIDDS17, a été proposé (Verma & Ranga, 2019b) et étudié (Verma & Ranga, 2019a) pour ce type d'IoT.

De la même manière, la détection d'intrusions sur un réseau de type Bluetooth PAN (pour Personal Area Network) est particulière et peu comparable avec des attaques sur un réseau WLAN ou Ethernet. En conséquence, ce cas spécifique nécessite un corpus approprié (Satam et al., 2018).

Plusieurs corpus récents ont été définis spécifiquement pour certains types d'attaques sur les objets connectés. C'est notamment le cas pour CIC-DDoS2019 (Sharafaldin et al., 2019) qui ne contient que des attaques de type déni de service alors que Bot-IoT (Koroniotis et al., 2019) contient uniquement des attaques de type Bot. IoTID20 (Ullah & Mahmoud, 2020), plus récent, résout quelques faiblesses observées dans un précédent corpus et est, lui aussi, dédié aux attaques Bot.

Pour former le corpus IoT-23 (Garcia et al., 2020), plusieurs attaques de logiciels malveillants ont été enregistrées sur un réseau composé de trois objets connectés spécifiques - une lampe intelligente Philips Hue, un assistant vocal Amazon Echo et une serrure intelligente Somfy.

Des corpus comme CICAndMal2017, CICMalDroid2020 sont, quant à eux, dédiés aux logiciels malveillants sur Android. Ces corpus ont en commun de couvrir un nombre restreint d'attaques et ne sont pas forcément représentatifs de la diversité des objets connectés.

Le manque d'hétérogénéité dans les corpus peut être vu comme un problème (Booij et al., 2022). Aussi, nous allons focaliser la suite de l'analyse sur des corpus présentant une diversité plus importante et étant plus représentatifs des deux cas de notre étude.

2.3.2 Revue de corpus numériques de détection d'intrusions

Cette section aborde de manière chronologique, selon la frise de la Figure 2.1, plusieurs corpus numériques qui sont très largement utilisés dans les publications sur les systèmes de détection d'intrusions réseau basés sur les conversations. Les éléments importants ainsi qu'une analyse des problèmes déjà connus sont donnés afin de permettre ultérieurement de faire un choix de corpus pour notre étude.

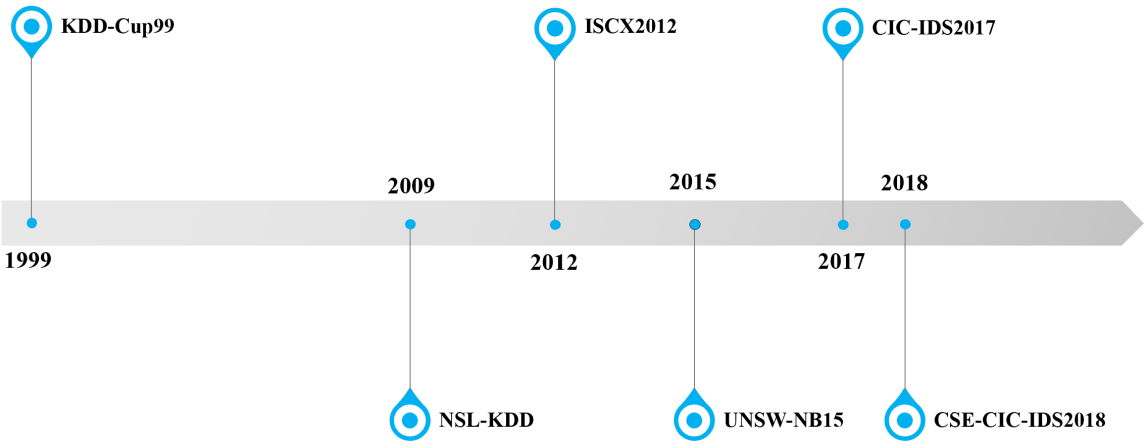


Figure 2.1 – Frise chronologique de quelques corpus numériques.

La DARPA, agence du département de la Défense des États-Unis pour les projets de recherche avancée, a fourni l'un des tout premiers corpus de détection d'intrusions réseau

en 1998. Il a été conçu pour l'analyse de sécurité des réseaux en exposant les problèmes associés à l'injection artificielle d'attaques et de trafic bénin à partir d'une simulation. Ce corpus a été modifié pour être utilisé lors de la troisième compétition internationale sur les outils de découverte de connaissances et d'extraction de données. Il est connu sous le nom de KDD-Cup99 et est constitué d'environ 800 Mo de données sous forme de fichiers CSV. Comme ce jeu de données est public, il a largement été utilisé et étudié. L'une des principales critiques sur ce corpus est que le trafic normal simulé n'est pas réaliste et conduit à des taux de faux positifs artificiellement bas (McHugh, 2000). Plusieurs autres incohérences avec les captures d'un trafic réel font que les attaques de ce corpus sont beaucoup plus faciles à détecter que dans un trafic ordinaire (Brown et al., 2009).

Le corpus NSL-KDD a été conçu en 2009 par l'université canadienne du Nouveau-Brunswick, à partir de KDD-Cup99 en résolvant une partie des problèmes identifiés (Tavallaee et al., 2009). Les données redondantes de KDD-Cup99 étant supprimées, NSL-KDD est significativement plus petit et contient 71 Mo de données sous forme de fichiers CSV. Comme pour son prédécesseur, les conversations sont caractérisées par quarante-et-une données et sont labellisées pour indiquer quatre types d'attaques différentes au lieu des vingt-deux de son prédécesseur.

Depuis 2009, NSL-KDD a été largement utilisé pour étudier les systèmes de détection d'intrusions et, bien que les attaques aient beaucoup évolué depuis 1998, KDD-Cup99 et NSL-KDD restent des sujets d'étude plus de vingt ans après la création des données (Devarakonda et al., 2022 ; Hussain & Hnamte, 2021 ; Sharma & Yadav, 2021).

En 2012, la même université a produit un corpus appelé ISCX-2012 basé sur le trafic généré selon la définition de deux profils (Shiravi et al., 2012). Le premier correspond aux scénarios de plusieurs attaques tandis que le second correspond à un trafic normal utilisant les protocoles HTTP, SMTP, SSH, IMAP, POP3 et FTP.

Les échanges réseau ont été enregistrés durant sept jours et sont disponibles sous une forme brute au format Packet CAPture (PCAP) ainsi que sous forme de fichiers XML contenant les caractéristiques des conversations et le label. La disponibilité des 84 Go de fichiers PCAP présente l'avantage de pouvoir recalculer les caractéristiques des conversations. Bien que plusieurs types d'attaque soient utilisés, toutes les conversations correspondantes n'ont qu'un seul label.

On peut toutefois noter que ce corpus n'utilise pas le protocole HTTPS. À ce titre, il n'est pas représentatif des échanges réseau actuels puisqu'en 2017, le ratio des chargements de page Web en HTTPS était déjà estimé entre 50% et 70% (Felt et al., 2017) et a probablement augmenté depuis.

En 2015, l'Université de Nouvelle-Galles du Sud (Australie) a proposé un nouvel ensemble de données appelé UNSW-NB15 (Moustafa & Slay, 2015), généré par simulation et représentant trente-et-une heures de trafic avec un total de 2 540 044 instances. Il contient des attaques modernes et à faible empreinte qui sont regroupées en neuf familles d'attaques différentes. UNSW-NB15 distingue les conversations avec quarante-neuf caractéristiques. Les fichiers de capture de paquets (PCAP) contenant des données brutes sont

disponibles.

L'Institut canadien pour la cybersécurité (CIC) de l'Université du Nouveau-Brunswick a publié l'ensemble de données CIC-IDS2017 selon le cadre défini dans (Sharafaldin et al., 2018) dans l'intention de résoudre les lacunes des ensembles de données précédents. L'infrastructure réseau est divisée en deux parties. La première contient 4 machines lançant les attaques vers dix machines victimes. Cinquante gigaoctets de données brutes sous forme de fichiers PCAP sont fournis ainsi qu'un ensemble de quatre-vingt-quatre caractéristiques dans des fichiers CSV. Le trafic réseau a été enregistré pendant cinq jours, ce qui a donné un total de 2 830 743 instances. Dans le cadre d'un projet de collaboration entre l'Établissement de Sécurité des Communications (CSE) et le CIC, un ensemble de données encore plus récent et plus important (400 Go) appelé CSE-CIC-IDS2018 (Canadian Institute for Cybersecurity, 2018) a été publié. L'infrastructure est plus complexe que celle de CIC-IDS2017 avec cinquante machines chargées des attaques et quatre cent cinquante machines victimes (dont trente serveurs) dans cinq sous-réseaux différents. L'ensemble de données contient l'enregistrement de dix jours de trafic, correspondant à 16 233 002 instances de conversations, chacune caractérisée par quatre-vingt caractéristiques communes à CIC-IDS2017.

Un comparatif de ces corpus numérique de détection d'intrusions réseau est donné dans la Table 2.2, montrant la date de création et caractérisant la taille du corpus au travers des nombres d'instances, de caractéristiques des conversations et de classes.

2.3.3 Étude détaillée des corpus numériques récents

Les trois corpus les plus récents semblent les plus appropriés pour notre étude, aussi une étude détaillée de chacun d'entre eux est nécessaire.

2.3.3.1 UNSW-NB15

Le corpus UNSW-NB15 contient un mélange de trafic normal et de différentes attaques, généré avec l'outil PerfectStorm (IXIA, 2016), un générateur de trafic de la société IXIA, rachetée en 2017 par Keysight. L'utilisation de cet outil a permis de simuler neuf familles d'attaques à partir d'informations disponibles CVE qui identifient, définissent et cataloguent des vulnérabilités rendues publiques sur un site internet officiel [1].

Le trafic réseau a été capturé sous forme de paquets avec l'outil TCPDUMP et enregistré en fichiers PCAP pendant deux sessions, l'une de seize heures et l'autre de quinze heures. Lors de la première session, le générateur de trafic est configuré pour produire une attaque par seconde. Pour la seconde, les attaques sont générées à raison de dix par seconde. La Table 2.3 fournit les statistiques du corpus pour chacune des sessions. Le nombre total d'attaques est de 321 283, ce qui représente 14.48 % des conversations du corpus complet. Malgré un nombre total de labels supérieur au nombre de conversations

1. https://cve.mitre.org/

TABLE 2.2 – Comparaison de corpus numériques.

Corpus	Création	Instances	Caractéristiques	Classes
KDD-Cup99	1998	3 883 370	41	23
NSL-KDD	2009	125 973	41	5
ISCX-2012	2012	2 028 005	18	2
UNSW-NB15	2015	2 540 044	49	9
CIC-IDS2017	2017	2 830 743	84	15
CSE-CIC-IDS2018	2018	16 232 943	80	15

(a) Caractéristiques numériques.

Corpus	Fichiers	Remarques
KDD-Cup99	CSV	Problèmes connus, pas représentatif
NSL-KDD	CSV	Basé sur KDD-Cup99, pas représentatif
ISCX-2012	PCAP, XML	Manque de représentativité
UNSW-NB15	PCAP, CSV	Trafic **simulé**, peu/pas de problème connu
CIC-IDS2017	PCAP, CSV	Trafic **réel**, peu/pas de problème connu
CSE-CIC-IDS2018	PCAP, CSV	Trafic **réel**, peu/pas de problème connu

(b) Caractéristiques catégorielles.

laissant supposer des erreurs, nous prenons le parti de conserver les valeurs données par les créateurs du corpus numérique (MOUSTAFA & SLAY, 2015).

TABLE 2.3 – Statistiques du corpus UNSW-NB15.

Caractéristiques	Première session	Deuxième session
Durée	16h	15h
Nombre de conversations	987 627	976 882
Instances d'attaques	22 215	299 068
Instances de trafic normal	1 064 987	1 153 774

Ces données brutes ont été converties en quarante-neuf caractéristiques grâce à des outils d'analyse de réseau ainsi que douze algorithmes spécifiques développés par l'équipe qui a généré le corpus. Les labels ont été apposés à partir des rapports générés par l'outil PerfectStorm. Ces rapports fournissent onze caractéristiques utilisées pour associer un label : les heures de début et de fin, la catégorie et sous-catégorie de l'attaque, le protocole, l'adresse et le port source, l'adresse et le port destination, le nom et la référence de l'attaque. La TABLE 2.4 liste toutes les caractéristiques. Les trente-cinq premières sont calculées par les outils d'analyse de réseau, les douze suivantes par les algorithmes spécifiques. Enfin, la famille d'attaques ainsi que le label complètent la liste.

TABLE 2.4 – Caractéristiques du corpus UNSW-NB15.

N○	Nom	Description
1	srcip	adresse IP source
2	sport	port source
3	dstip	adresse IP destination
4	dsport	port destination
5	proto	protocole
6	state	état (dépendant du protocole)
7	dur	durée total
8	sbytes	nombre d'octets envoyés par la source
9	dbytes	nombre d'octets envoyés par la destination
10	sttl	durée de vie source vers destination
11	dttl	durée de vie destination vers source
12	sloss	nombre de paquets source retransmis ou éliminés
13	dloss	nombre de paquets destination retransmis ou éliminés
14	service	http, ftp, ssh, dns, ..., autre (-)
15	sload	bits par seconde source
16	dload	bits par seconde destination
17	spkts	nombre de paquets source vers destination
18	dpkts	nombre de paquets destination vers source
19	swin	fenêtre TCP source
20	dwin	fenêtre TCP destination
21	stcpb	numéro de séquence source
22	dtcpb	numéro de séquence destination
23	smeansz	moyenne de la taille des paquets source
24	dmeansz	moyenne de la taille des paquets destination
25	trans_depth	profondeur de la connexion des transactions requête/réponse HTTP
26	res_bdy_len	taille des données utiles transférées vers le HTTP du serveur
27	sjit	gigue source
28	djit	gigue destination
29	stime	horodatage de début
30	ltime	horodatage de fin
31	sintpkt	temps d'arrivée inter-paquet source
32	dintpkt	temps d'arrivée inter-paquet destination
33	tcprtt	somme des caractéristiques 'synack' et 'ackdat'
34	synack	temps entre les drapeaux TCP SYN et SYN_ACK
35	ackdat	temps entre les drapeaux TCP SYN_ACK et ACK
36	is_sm_ips_ports	vaut 1 si adresse et port sont identiques pour la source et la destination, 0 sinon
37	ct_state_ttl	nombre pour chaque état, en fonction des plages ttl
38	ct_flw_http_mthd	nombre de conversations qui contient des méthodes comme get/post en HTTP
39	is_ftp_login	vaut 1 si la session FTP est accédé avec nom/mot de passe, 0 sinon
40	ct_ftp_cmd	nombre de conversations qui contient une commande FTP
41	ct_srv_src	nombre de connections qui contient les mêmes service et adresse source parmi 100 connexions
42	ct_srv_dst	nombre de connections qui contient les mêmes service et adresse destination parmi 100 connexions
43	ct_dst_ltm	nombre de connections qui contient la même adresse source parmi 100 connexions
44	ct_src_ltm	nombre de connections qui contient la même adresse destination parmi 100 connexions
45	ct_src_dport_ltm	nombre de connections qui contient les mêmes port et adresse source parmi 100 connexions
46	ct_dst_sport_ltm	nombre de connections qui contient les mêmes port et adresse destination parmi 100 connexions
47	ct_dst_src_ltm	nombre de connections qui contient les mêmes adresse source et adresse destination parmi 100 connexions
48	attack_cat	nom de la catégorie d'attaques
49	Label	vaut 0 si trafic 0, 1 si attaque

Comme les fichiers de configuration des outils ainsi que les algorithmes ne sont pas mis à disposition, il est impossible de recalculer les caractéristiques à partir des fichiers PCAP.

Les auteurs de ce corpus ont réalisé une analyse statistique et une comparaison avec le corpus ancien KDD-Cup99 (Moustafa & Slay, 2016). Les résultats expérimentaux montrent que ce corpus est plus complexe que KDD-Cup99 et les auteurs proposent de l'utiliser pour évaluer les systèmes de détection d'intrusions réseau. Cette complexité accrue est confirmée par une plus grande difficulté à détecter les attaques avec un taux de faux positif plus élevé et une différence de l'ordre de 10% sur la justesse (Khan et al., 2019).

2.3.3.2 CIC-IDS2017

Contrairement au corpus UNSW-NB15, celui-ci n'est pas issu d'une simulation, mais d'un réel enregistrement de trafic. Au lieu d'utiliser un générateur de trafic, les créateurs de ce corpus ont lancé des attaques en utilisant des outils de test de pénétration de réseaux. L'installation se compose de deux réseaux : le premier lance des attaques depuis quatre machines différentes ; le second contient dix machines, des PC et des serveurs utilisant différents systèmes d'exploitation (Windows, Ubuntu, MacOS) et qui subissent les attaques. La topologie du réseau correspondante est représentée sur la Figure 2.2 (Sharafaldin et al., 2018).

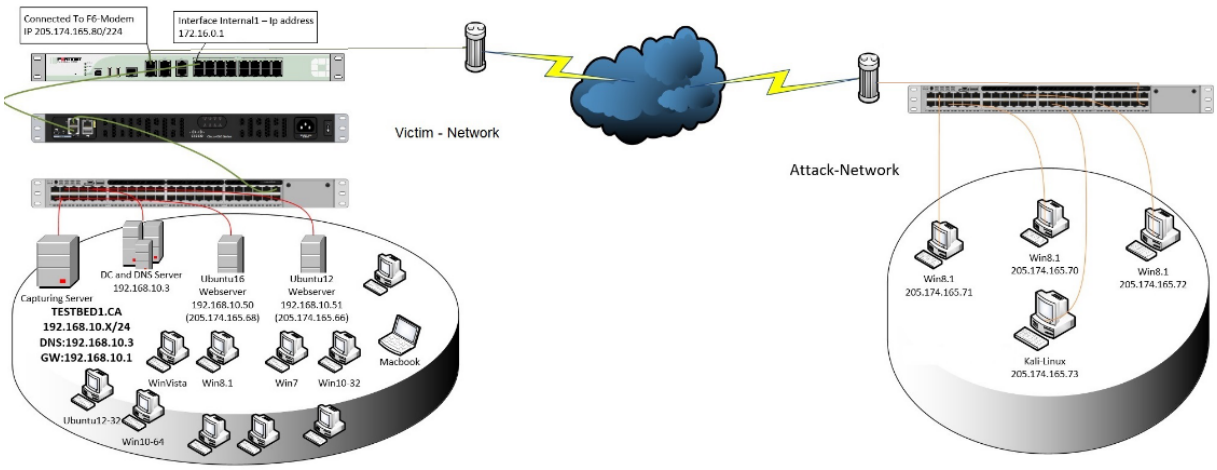


Figure 2.2 – Topologie réseau utilisée pour enregistrer le corpus CIC-IDS2017.

Les données ont été enregistrées durant cinq jours consécutifs en 2017. Le premier jour ne contient que du trafic normal. Durant la journée du mardi, les attaques en force brute ont été lancées sur des sessions FTP et SSH. Différentes attaques en déni de service et l'attaque Heartbleed ont eu lieu le mercredi. Le jeudi, des attaques Web (cross-site scripting, injection SQL, force brute) et d'infiltration ont été menées. Enfin, l'enregistrement du vendredi contient des attaques en déni de service distribué, des attaques de bot et

de reconnaissance par du scan de ports (Canadian Institute for Cybersecurity, 2017b).

Les données brutes sont disponibles au format PCAP. Des caractéristiques en ont été extraites grâce à un outil appelé CICFlowMeter, développé par le même laboratoire, le CIC (Canadian Institute for Cybersecurity, 2017a). La liste des caractéristiques est donnée dans la Table 2.5.

Le code source de l'outil est disponible (Canadian Institute for Cybersecurity, 2020) et permet de reproduire les caractéristiques, à l'exception des labels. En effet, l'outil de labellisation est un outil propriétaire que le CIC ne souhaite pas partager. Les labels correspondent aux noms des attaques et le trafic normal est labellisé comme Bénin.

Ce corpus a été analysé par leurs créateurs en utilisant différents algorithmes d'apprentissage automatique (Sharafaldin et al., 2018). Leurs résultats montrent que les algorithmes classiques donnent de meilleurs résultats que les réseaux de neurones. Ce corpus est également largement utilisé pour l'évaluation des systèmes de détection d'intrusions (Gamage & Samarabandu, 2020 ; Jiang et al., 2018 ; Ullah & Mahmoud, 2019 ; Ustebay et al., 2018).

2.3.3.3 CSE-CIC-IDS2018

Ce corpus est très similaire à CIC-IDS2017. En effet, il a été généré par le CIC en collaboration avec l'Établissement de Sécurité des Communications (CSE) canadien en utilisant les mêmes outils. Les données brutes ont été enregistrées au format PCAP et les caractéristiques des conversations ont été extraites à l'aide de CICFlowMeter. Les caractéristiques sont donc identiques. On notera toutefois que leurs noms diffèrent et que quatre caractéristiques ne sont pas incluses : l'identifiant de conversation, les adresses IP et ports source et destination.

Une différence majeure entre CSE-CIC-IDS2018 et CIC-IDS2017 porte sur une mise à plus grande échelle avec une topologie réseau beaucoup plus complexe telle que représentée sur la Figure 2.3 et une quantité plus élevée de données collectées (Canadian Institute for Cybersecurity, 2018).

Le réseau attaqué est représentatif d'un cas réel en se composant de sous-réseaux pouvant correspondre à cinq départements d'une entreprise avec une salle de serveurs. Les machines utilisent les deux systèmes d'exploitation : Windows et Linux. Les attaques sont lancées depuis cinquante machines différentes, utilisant également Windows et Linux.

Comme pour CIC-IDS207, il est possible de régénérer les caractéristiques à partir de l'outil CICFlowMeter mais il n'est toujours pas possible de labelliser les conversations. Toutefois, le CIC fournit en plus des fichiers PCAP et de CSV des fichiers répertoriant les événements détectés par Windows et Linux sur les machines attaquées. Ces informations peuvent être utiles pour déterminer le label de chaque conversation et les couplant avec les informations disponibles (Canadian Institute for Cybersecurity, 2018) comme les adresses ciblées par les attaques ainsi que les horodatages des attaques.

TABLE 2.5 – Caractéristiques du corpus CIC-IDS2017.

Feature name	Description	Feature name	Description
Flow ID	Flow identifier	Fwd Packets/s	Number of forward packets per second
Source IP	IP address of the initiator	Bwd Packets/s	Number of backward packets per second
Source Port	Port of the initiator	Min Packet Length	Minimum length of a packet
Destination IP	IP address of the receiver	Max Packet Length	Maximum length of a packet
Destination Port	Port of the receiver of the flow	Packet Length Mean	Mean length of a packet
Protocol	Protocol number	Packet Length Std	Standard deviation length of a packet
Timestamp	Timestamp of the first packet of the flow	Packet Length Variance	Variance length of a packet
Flow Duration	Duration of the flow in microsecond	FIN Flag Count	Number of packets with FIN
Total Fwd Packets	Number of packets in forward direction	SYN Flag Count	Number of packets with SYN
Total Bwd Packets	Number of packets in backward direction	RST Flag Count	Number of packets with RST
Total Length of Fwd Packets	Total size of packets in forward direction	PSH Flag Count	Number of packets with PUSH
Total Length of Bwd Packets	Total size of packets in backward direction	ACK Flag Count	Number of packets with ACK
Fwd Packet Length Max	Maximum size of packet in forward direction	URG Flag Count	Number of packets with URG
Fwd Packet Length Min	Minimum size of packet in forward direction	CWE Flag Count	Number of packets with CWE
Fwd Packet Length Mean	Mean size of packet in forward direction	ECE Flag Count	Number of packets with ECE
Fwd Packet Length Std	Standard deviation size of packet in forward direction	Down/Up Ratio	Download and upload ratio
Bwd Packet Length Max	Maximum size of packet in backward direction	Average Packet Size	Average size of packet
Bwd Packet Length Min	Minimum size of packet in backward direction	Avg Fwd Segment Size	Average size observed in forward direction
Bwd Packet Length Mean	Mean size of packet in backward direction	Avg Bwd Segment Size	Average size observed in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction	Fwd Header Length.1	Redundant field with Fwd Header Length
Flow Bytes/s	Number of flow bytes per second	Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in forward direction
Flow Packets/s	Number of flow packets per second	Fwd Avg Packets/Bulk	Average number of packets bulk rate in forward direction
Flow IAT Mean	Mean time between two packets sent in flow	Fwd Avg Bulk Rate	Average number of bulk rate in forward direction
Flow IAT Std	Standard deviation time between two packets sent in flow	Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in backward direction
Flow IAT Max	Maximum time between two packets sent in flow	Bwd Avg Packets/Bulk	Average number of packets bulk rate in backward direction
Flow IAT Min	Minimum time between two packets sent in flow	Bwd Avg Bulk Rate	Average number of bulk rate in backward direction
Fwd IAT Total	Total time between two packets sent in forward direction	Subflow Fwd Packets	Average number of packets in a sub flow in forward direction
Fwd IAT Mean	Mean time between two packets sent in forward direction	Subflow Fwd Bytes	Average number of bytes in a sub flow in forward direction
Fwd IAT Std	Standard deviation time between two packets sent in forward direction	Subflow Bwd Packets	Average number of packets in a sub flow in backward direction
Fwd IAT Max	Maximum time between two packets sent in forward direction	Subflow Bwd Bytes	Average number of bytes in a sub flow in backward direction
Fwd IAT Min	Minimum time between two packets sent in forward direction	Init_Win_bytes_forward	Total number of bytes sent in initial window in forward direction
Bwd IAT Total	Total time between two packets sent in backward direction	Init_Win_bytes_backward	Total number of bytes sent in initial window in backward direction
Bwd IAT Mean	Mean time between two packets sent in backward direction	act_data_pkt_fwd	Count of packets with ≥ 1 byte of TCP payload in forward direction
Bwd IAT Std	Standard deviation time between two packets sent in backward direction	min_seg_size_forward	Minimum segment size observed in forward direction
Bwd IAT Max	Maximum time between two packets sent in backward direction	Active Mean	Mean time a flow was active before becoming idle
Bwd IAT Min	Minimum time between two packets sent in backward direction	Active Std	Standard deviation time a flow was active before becoming idle
Fwd PSH Flags	Number of times the PSH flag was set in packets in forward direction	Active Max	Maximum time a flow was active before becoming idle
Bwd PSH Flags	Number of times the PSH flag was set in packets in backward direction	Active Min	Minimum time a flow was active before becoming idle
Fwd URG Flags	Number of times the URG flag was set in packets in forward direction	Idle Mean	Mean time a flow was idle before becoming active
Bwd URG Flags	Number of times the URG flag was set in packets in backward direction	Idle Std	Standard deviation time a flow was idle before becoming active
Fwd Header Length	Total bytes used for headers in forward direction	Idle Max	Maximum time a flow was idle before becoming active
Bwd Header Length	Total bytes used for headers in backward direction	Idle Min	Minimum time a flow was idle before becoming active

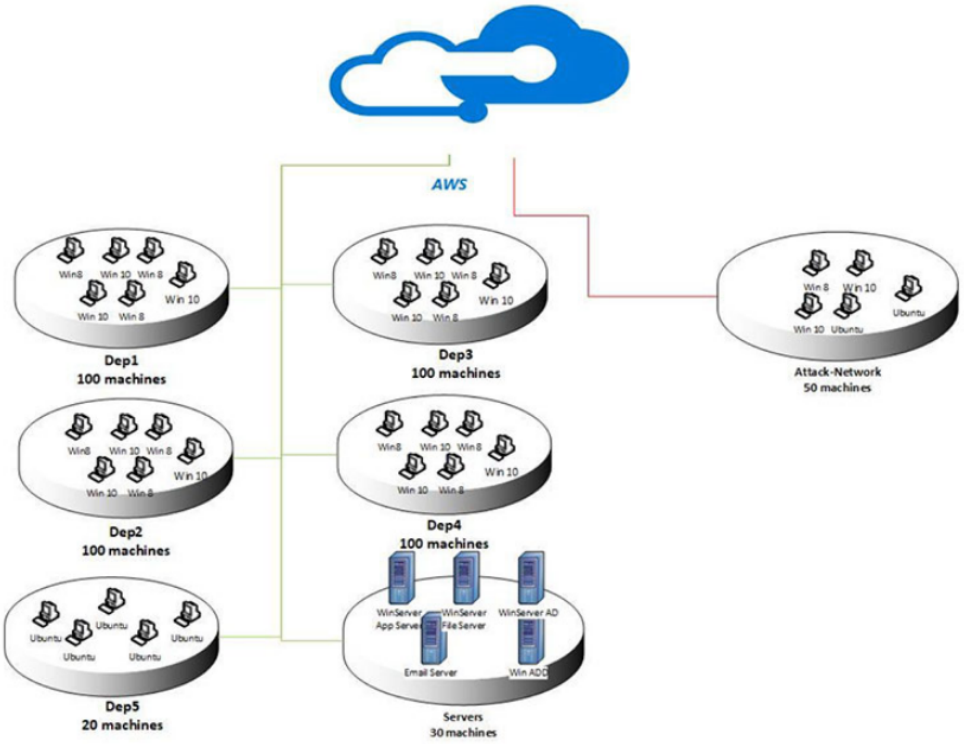


FIGURE 2.3 – Topologie réseau utilisée pour enregistrer le corpus CSE-CIC-IDS2018.

Ce corpus est devenu une référence pour l'étude des systèmes de détection d'intrusions aussi bien avec des algorithmes classiques d'apprentissage automatique qu'avec des réseaux de neurones (HUA, 2020 ; KARATAS et al., 2020 ; PRASAD et al., 2019). Comme nous l'avons déjà remarqué pour le corpus CIC-IDS2017, les réseaux de neurones ont tendance à donner des résultats moins bons que les algorithmes classiques.

2.4 Synthèse

Au cours de ce chapitre, nous avons dans un premier temps abordé différentes catégories d'attaques d'intrusions réseau, en suivant une taxonomie existante. Pour chacune d'elles, des exemples d'attaques ont été détaillés dans leur fonctionnement ainsi que les outils permettant de les lancer.

Dans un deuxième temps, différents types de systèmes de détection d'intrusions ont été présentés. Après avoir défini la notion de conversation, les techniques basées sur les paquets et sur les conversations ont été expliquées. Pour des raisons de performances, celle basée sur les conversations est plus adaptée aux systèmes embarqués, et convient mieux à notre contexte de recherche.

Pour finir, une revue de différents corpus numériques de détection d'intrusions réseau créés entre 1998 et 2018 a été menée. De cette première analyse, nous avons étudié en

détail les corpus numériques UNSW-NB15, CIC-IDS2017 et CSE-CIC-IDS2018, ceux-ci étant les trois plus récents.

L'ensemble de ces éléments est essentiel pour choisir parmi les différents corpus numériques de détection d'intrusions ceux que nous utilisons pour ce travail de recherche. D'autre part, ces éléments nous permettent de réaliser une étude critique des corpus numériques et d'amener des contributions dans la Partie II de ce document.

Nous abordons dans le chapitre suivant l'apprentissage automatique en décrivant les différentes méthodes d'apprentissage ainsi que les types de tâches qu'elles permettent d'adresser. Nous y présentons également quelques algorithmes souvent utilisés sur les corpus numériques de détection d'intrusions et en particulier pour les phases d'expérimentation de nos travaux de recherche.

Références

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N. & Hartog, F. T. H. d., (2022), ToN_IoT : The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets, *IEEE Internet of Things Journal*, *9 1*, 485-496, https://doi.org/10.1109/JIOT.2021.3085194

Bošnjak, L., Sreš, J. & Brumen, B., (2018), Brute-force and dictionary attack on hashed real-world passwords, *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1161-1166, https://doi.org/10.23919/MIPRO.2018.8400211

Brown, C., Cowperthwaite, A., Hijazi, A. & Somayaji, A., (2009), Analysis of the 1999 DARPA/Lincoln Laboratory IDS evaluation data with NetADHICT, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-7, https://doi.org/10.1109/CISDA.2009.5356522

Bul'ajoul, W., James, A. & Pannu, M., (2015), Improving network intrusion detection system performance through quality of service configuration and parallel technology [Special Issue on Optimisation, Security, Privacy and Trust in E-business Systems], *Journal of Computer and System Sciences*, *81 6*, 981-999, https://doi.org/10.1016/j.jcss.2014.12.012

Canadian Institute for Cybersecurity, (2017a), Applications - CICFlowMeter (formerly ISCXFlowMeter), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/research/applications.html

Canadian Institute for Cybersecurity, (2017b), Intrusion Detection Evaluation Dataset (CICIDS2017), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2017.html

Canadian Institute for Cybersecurity, (2018), CSE-CIC-IDS2018 on AWS, A collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2018.html

Canadian Institute for Cybersecurity, (2020), CicFlowMeter source code, Récupérée 24 juillet 2022, à partir de https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter

Claise, B., Trammell, B. & Aitken, P., (2013), *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* (STD N° 77), RFC Editor, http://www.rfc-editor.org/rfc/rfc7011.txt

Debar, H., Dacier, M. & Wespi, A., (1999), Towards a taxonomy of intrusion-detection systems, *Computer Networks*, *31 8*, 805-822, https://doi.org/10.1016/S1389-1286(98)00017-6

Derbyshire, R., Green, B., Prince, D., Mauthe, A. & Hutchison, D., (2018), An Analysis of Cyber Security Attack Taxonomies, *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 153-161, https://doi.org/10.1109/EuroSPW.2018.00028

Devarakonda, A., Sharma, N., Saha, P. & Ramya, S., (2022), Network intrusion detection : a comparative study of four classifiers using the NSL-KDD and KDD'99 datasets, *Journal of Physics : Conference Series*, *2161 1*, 012043, https://doi.org/10.1088/1742-6596/2161/1/012043

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M. & Halderman, J. A., (2014), The Matter of Heartbleed, *Proceedings of the 2014 Conference on Internet Measurement Conference*, 475-488, https://doi.org/10.1145/2663716.2663755

Eslahi, M., Salleh, R. & Anuar, N. B., (2012), Bots and botnets : An overview of characteristics, detection and challenges, *IEEE International Conference on Control System, Computing and Engineering*, 349-354, https://doi.org/10.1109/ICCSCE.2012.6487169

Felt, A. P., Barnes, R., King, A., Palmer, C., Bentzel, C. & Tabriz, P., (2017), Measuring HTTPS Adoption on the Web, *Proceedings of the 26th USENIX Conference on Security Symposium*, 1323-1338.

Fonseca, J., Vieira, M. & Madeira, H., (2009), Vulnerability attack injection for web applications, *IEEE/IFIP International Conference on Dependable Systems Networks*, 93-102, https://doi.org/10.1109/DSN.2009.5270349

Gamage, S. & Samarabandu, J., (2020), Deep learning methods in network intrusion detection : A survey and an objective comparison, *Journal of Network and Computer Applications*, *169*, 102767, https://doi.org/10.1016/j.jnca.2020.102767

Garcia, S., Parmisano, A. & Erquiaga, M. J., (2020), IoT-23 : A labeled dataset with malicious and benign IoT network traffic, https://doi.org/10.5281/zenodo.4743746

Hua, Y., (2020), An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM, *Information Communication Technologies Conference (ICTC)*, 125-130, https://doi.org/10.1109/ICTC49638.2020.9123302

Hussain, J. & Hnamte, V., (2021), Deep Learning Based Intrusion Detection System : Software Defined Network, *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 1-6, https://doi.org/10.1109/ASIANCON51346.2021.9544913

IXIA, (2016), PerfectStorm, Industry's highest performing application and security test platform, Récupérée 24 juillet 2022, à partir de https://www.keysight.com/fr/en/products/network-test/network-test-hardware/perfectstorm.html

Jiang, J., Yu, Q., Yu, M., Li, G., Chen, J., Liu, K., Liu, C. & Huang, W., (2018), ALDD : A Hybrid Traffic-User Behavior Detection Method for Application Layer DDoS, *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 1565-1569, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00225

Kali, (2017), Kali Linux, The Most Advanced Penetration Testing Distribution, Récupérée 24 juillet 2022, à partir de https://www.kali.org/

Karatas, G., Demir, O. & Sahingoz, O. K., (2020), Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset, *IEEE Access*, *8*, 32150-32162, https://doi.org/10.1109/ACCESS.2020.2973219

Khan, F. A., Gumaei, A., Derhab, A. & Hussain, A., (2019), TSDL : A TwoStage Deep Learning Model for Efficient Network Intrusion Detection, *IEEE Access*, 1-1, https://doi.org/10.1109/ACCESS.2019.2899721

Kjaerland, M., (2006), A taxonomy and comparison of computer security incidents from the commercial and government sectors, *Computers & Security*, *25 7*, 522-538, https://doi.org/10.1016/j.cose.2006.08.004

Koroniotis, N., Moustafa, N., Sitnikova, E. & Turnbull, B., (2019), Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics : Bot-IoT dataset, *Future Generation Computer Systems*, *100*, 779-796, https://doi.org/10.1016/j.future.2019.05.041

Kruegel, C., Valeur, F. & Vigna, G., (2004), *Intrusion detection and correlation : challenges and solutions* (T. 14), Springer Science & Business Media, https://doi.org/https://doi.org/10.1007/b101493

Lai, J., Wu, J., Chen, S., Wu, C. & Yang, C., (2008), Designing a Taxonomy of Web Attacks, *International Conference on Convergence and Hybrid Information Technology (ICHIT)*, 278-282, https://doi.org/10.1109/ICHIT.2008.280

McHugh, J., (2000), Testing Intrusion Detection Systems : A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory, *ACM Trans. Inf. Syst. Secur.*, *3 4*, 262-294, https://doi.org/10.1145/382912.382923

Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A. & Payne, B. D., (2015), Evaluating Computer Intrusion Detection Systems : A Survey of Common Practices, *ACM Comput. Surv.*, *48 1*, https://doi.org/10.1145/2808691

Ming Gao, Kenong Zhang & Jiahua Lu, (2006), Efficient packet matching for gigabit network intrusion detection using TCAMs, *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, *2*, 6 pp.-254, https://doi.org/10.1109/AINA.2006.164

Moustafa, N. & Slay, J., (2015), UNSW-NB15 : A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set), *Military Communications and Information Systems Conference (MilCIS)*, 1-6, https://doi.org/10.1109/MilCIS.2015.7348942

Moustafa, N. & Slay, J., (2016), The evaluation of Network Anomaly Detection Systems : Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, *Information Security Journal : A Global Perspective*, *25 1-3*, 18-31, https://doi.org/10.1080/19393555.2015.1125974

Paxson, V., (1999), Bro : a system for detecting network intruders in real-time, *Computer Networks*, *31 23*, 2435-2463, https://doi.org/10.1016/S1389-1286(99)00112-7

Prasad, M. D., V, P. B. & Amarnath, C., (2019), Machine Learning DDoS Detection Using Stochastic Gradient Boosting, *International Journal of Computer Sciences and Engineering*, *7*, 157-166, https://doi.org/10.26438/ijcse/v7i4.157166

Roesch, M., (1999), Snort - Lightweight Intrusion Detection for Networks, *Proceedings of the 13th USENIX Conference on System Administration*, 229-238, https://doi.org/10.5555/1039834.1039864

Satam, P., Satam, S. & Hariri, S., (2018), Bluetooth Intrusion Detection System (BIDS), *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, 1-7, https://doi.org/10.1109/AICCSA.2018.8612809

Sharafaldin, I., Lashkari, A. H., Hakak, S. & Ghorbani, A. A., (2019), Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy, *International Carnahan Conference on Security Technology (ICCST)*, 1-8, https://doi.org/10.1109/CCST.2019.8888419

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., (2018), Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, 108-116, https://doi.org/10.5220/0006639801080116

Sharma, N. V. & Yadav, N. S., (2021), An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers, *Microprocessors and Microsystems*, *85*, 104293, https://doi.org/10.1016/j.micpro.2021.104293

Shiravi, A., Shiravi, H., Tavallaee, M. & Ghorbani, A. A., (2012), Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection, *Comput. Secur.*, *31 3*, 357-374, https://doi.org/10.1016/j.cose.2011.12.012

Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A. & Mishra, A. K., (2018), Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools, *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 318-322, https://doi.org/10.1109/ICACCI.2018.8554590

Simmons, C. B., Ellis, C., Shiva, S., Dasgupta, D. & Wu, Q., (2009), AVOIDIT : A Cyber Attack Taxonomy, *CTIT technical reports series*.

STMicroelectronics, (2022), Data Brief - Stellar SR6 G7 line, Récupérée 26 juillet 2022, à partir de https://www.st.com/resource/en/data_brief/sr6g7c4.pdf

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A., (2009), A Detailed Analysis of the KDD CUP 99 Data Set, *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6, https://doi.org/10.1109/CISDA.2009.5356528

Trammell, B. & Boschi, E., (2008), *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)* (STD N° 5103), RFC Editor, http://www.rfc-editor.org/rfc/rfc5103.txt

Ullah, I. & Mahmoud, Q. H., (2019), A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks, *16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 1-6, https://doi.org/10.1109/CCNC.2019.8651782

Ullah, I. & Mahmoud, Q. H., (2020), A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks, In C. Goutte & X. Zhu (Éd.), *Advances in Artificial Intelligence* (p. 508-520), Springer International Publishing, https://doi.org/10.1007/978-3-030-47358-7_52

Ustebay, S., Turgut, Z. & Aydin, M. A., (2018), Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier, *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 71-76, https://doi.org/10.1109/IBIGDELFT.2018.8625318

Verma, A. & Ranga, V., (2019a), ELNIDS : Ensemble Learning based Network Intrusion Detection System for RPL based Internet of Things, *2019 4th International Conference on Internet of Things : Smart Innovation and Usages (IoT-SIU)*, 1-6, https://doi.org/10.1109/IoT-SIU.2019.8777504

Verma, A. & Ranga, V., (2019b), Evaluation of Network Intrusion Detection Systems for RPL Based 6LoWPAN Networks in IoT, *Wireless Personal Communications*, *108 3*, 1571-1594, https://doi.org/10.1007/s11277-019-06485-w

Vignau, B., Khoury, R., Hallé, S. & Hamou-Lhadj, A., (2021), The evolution of IoT Malwares, from 2008 to 2019 : Survey, taxonomy, process simulator and perspectives, *Journal of Systems Architecture*, *116*, 102143, https://doi.org/10.1016/j.sysarc.2021.102143

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J. & Alexander, R., (2012), *RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks* (RFC N° 6550), RFC Editor, http://www.rfc-editor.org/rfc/rfc6550.txt

Yadav, T. & Rao, A. M., (2015), Technical Aspects of Cyber Kill Chain, In J. H. Abawajy, S. Mukherjea, S. M. Thampi & A. Ruiz-Martínez (Éd.), *Security in Computing and Communications* (p. 438-452), Springer International Publishing.

Chapitre 3

Apprentissage automatique

« *La tristesse de l'intelligence artificielle est qu'elle est sans artifice, donc sans intelligence.* »

Jean BAUDRILLARD
Sociologue et philosophe (1929 – 2007)

Depuis le début de son apparition, l'intelligence artificielle (IA) fait partie des sujets capables de déclencher des passions, que ce soit de l'espoir, des rêves avec des machines évitant aux humains de réaliser des tâches fastidieuses ou des peurs.

La littérature comme le cinéma s'est emparé du sujet avec des œuvres, souvent dystopiques, comme les romans d'Isaac Asimov, des films telles que *I, Robot*, *Matrix* ou *Ex Machina*, ou encore des bandes dessinées comme *Carbone & Silicium*. L'imaginaire collectif a associé l'IA à des humanoïdes capables de prendre le contrôle sur les humains.

Les détails fournis dans ce chapitre montre que l'apprentissage automatique ou machine learning (ML), un sous-domaine de l'IA, est essentiellement basé sur l'optimisation d'un problème exposé sous forme mathématique, afin de spécialiser un programme à exécuter une tâche prévue par l'informaticien.

Les moyens ingénieux de résoudre les problèmes sont bien dans la tête des humains et non dans une machine. Nous sommes loin d'un ordinateur doté d'une quelconque forme d'intelligence. Pour autant, la résolution de problèmes complexes grâce à ces techniques d'IA provoque une joie, une extase rendant moins triste l'intelligence artificielle qui, finalement, porte bien mal son nom.

Pour lever le voile sur l'apprentissage automatique, la section 3.1 précise ce qu'est le machine learning et transpose la définition à la détection d'intrusions réseau. Ensuite, les différentes formes ainsi que les types de tâche d'apprentissage automatique sont présentés en section 3.2. Pour finir, le chapitre expose les algorithmes de machine learning généralement utilisés pour détecter des intrusions réseau ainsi que la procédure pour les entraîner.

3.1 Définitions

L'apprentissage automatique recouvre un ensemble de techniques très larges et sert des objectifs très différents. Avant de plonger dans les détails, une clarification du ML est nécessaire.

Plusieurs définitions expliquent ce qu'est l'apprentissage automatique. Commençons par une première définition de 1959 attribuée à Arthur Samuel : « *Champ d'études donnant aux ordinateurs la capacité d'apprendre sans être explicitement programmé.* » Une telle définition, simple en apparence, reste difficile à comprendre sortie de son contexte, *a priori*, le jeu d'échec (Samuel, 1959).

Une autre définition, plus technique, est proposé par Tom Mitchell (Mitchell, 1997) : « *On dit d'un programme informatique qu'il apprend de l'expérience E concernant une certaine classe de tâches T et d'une mesure de performance P, si sa performance aux tâches dans T, telle que mesurée par P, s'améliore avec l'expérience E.* »

La définition 5 est une transposition de la définition de Tom Mitchell à la détection d'intrusions.

Définition 5: Apprentissage automatique pour la détection d'intrusion

Posons les éléments suivants :
- soit E, une expérience d'utilisation d'un réseau informatique contenant des données de trafic normal et des cyberattaques, enregistrées et labellisées ;
- soit T, une tâche de classification prédisant le type de trafic ;
- soit P, une métrique telle que la justesse.

Un système de détection d'intrusions réseau utilisant une technique d'apprentissage automatique est un programme informatique qui apprend, à partir de l'ensemble E des données enregistrées sur un réseau, à réaliser la tâche T de classification en trafic normal ou en attaques, si la performance P de justesse de la classification s'améliore avec l'expérience E.

3.2 Méthodes d'apprentissage et types de tâches

La catégorisation des méthodes de ML est réalisable selon différents critères. Par exemple, ces classements peuvent se baser sur la façon dont l'humain intervient lors de l'apprentissage. Un autre classement est possible en fonction de l'entraînement à la volée, à chaque fois qu'une nouvelle donnée est disponible, ou bien à partir d'un ensemble de données disponibles dès le début de l'entraînement. Une autre possibilité serait l'absence ou non d'entraînement, par exemple en comparant systématiquement une nouvelle donnée à celles déjà connues. Ces critères ne sont pas exhaustifs et peuvent être combinés.

La Figure 3.1 donne une représentation partielle de différentes méthodes. Elle est basée sur le type d'apprentissage et les types de tâches associées.

Son contenu illustre un niveau suffisant pour la compréhension de nos travaux de recherches.

Parmi les algorithmes de ML, certaines méthodes sont basées sur le niveau de supervision apporté à la création du modèle, ou encore sur l'interaction entre le modèle et son environnement.

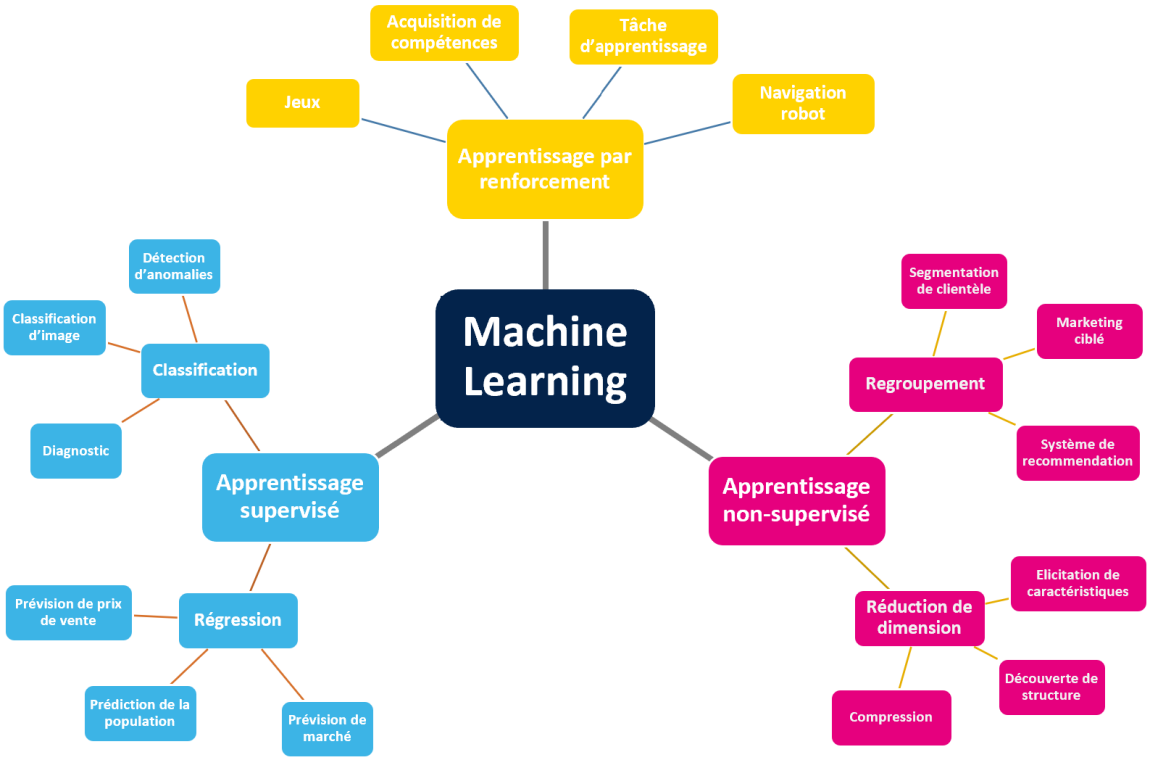


FIGURE 3.1 – Catégories d'apprentissage automatique avec exemples.

3.2.1 Apprentissage supervisé

Dans le cas de l'apprentissage supervisé, l'objectif est d'apprendre une estimation d'une fonction inconnue f, de correspondance entre les données d'entrées $\boldsymbol{x}$ et des données de sorties y telle que $y = f(\boldsymbol{x})$.

Un jeu de données $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$ est un ensemble composé de N exemples de paires d'entrée et sortie $(\boldsymbol{x_i}, y_i)$ (Murphy, 2012). Dans sa forme la plus simple, $\boldsymbol{x}$ est un vecteur contenant les caractéristiques d'entrée et y est la valeur de sortie, qu'elle soit catégorielle ou continue.

Le principe de ce type d'apprentissage consiste à utiliser les données de $\mathcal{D}$ (ou un sous-ensemble) afin que le modèle apprenne via la fonction d'estimation $\hat{f}$ à prédire la valeur sortie en minimisant une fonction de coût.

Lorsque l'apprentissage est terminé, le modèle est alors capable pour toute nouvelle valeur de $\boldsymbol{x}$ de prédire une valeur estimée $\hat{y}$ avec une erreur minimale.

$$\forall \boldsymbol{x} \notin \mathcal{D}, \hat{y} = \hat{f}(\boldsymbol{x})$$

Le point fort de ce type d'apprentissage est sa capacité à extrapoler ou à généraliser ses sorties pour que le système se comporte correctement dans des situations non présentes dans les données d'entraînement.

Une variante d'apprentissage supervisé consiste à utiliser une fonction $\hat{f}$ produisant une distribution de probabilité sur les classes. Dans ce cas, la classe prédite est celle ayant la probabilité la plus élevée.

L'apprentissage supervisé est très utilisé pour deux types de tâches : la classification et la régression.

Classification Pour ce type de tâches, les valeurs de sortie y_i appartiennent à un ensemble fini. La dimension de cet ensemble correspond au nombre de classes contenu dans $\mathcal{D}$. Ainsi, pour classer des images labellisées chien ou chat, y_i ne peut prendre que deux valeurs, l'une correspondant au chien, l'autre au chat.

Les outils de classification sont utilisables pour établir notamment des diagnostics, par exemple dans le domaine médical. Dans certains cas, en fonction des données disponibles, ils peuvent également être appliqués à la détection d'anomalies.

Dans un corpus numérique de détection d'intrusions réseau, chaque exemple ou instance d'attaques est associée à l'un des m labels d'attaques. La sortie y_i peut prendre $m + 1$ valeurs différentes, le trafic normal s'ajoutant aux m attaques.

Régression La régression est une tâche ressemblant à la classification. Toutefois, la valeur de sortie est continue contrairement à la classification utilisant un ensemble fini de valeurs. Un exemple d'utilisation classique est l'estimation d'un prix de vente dans l'immobilier. L'ensemble $\mathcal{D}$ contient des ventes déjà réalisées pour lesquelles $\boldsymbol{x}$ contient les caractéristiques, telles que la superficie de l'habitation, le nombre de pièces, la proximité

de commerce ou d'école, et y le prix de vente effectif. Le prix vente d'un nouveau bien est estimé et prend une valeur continue.

De la même manière, les estimations de progression de la population ou d'évolution d'un marché donné se basent sur des techniques de régression.

3.2.2 Apprentissage non-supervisé

En apprentissage non-supervisé, une différence majeure réside dans le jeu de données $\mathcal{D} = \{\boldsymbol{x_i}\}_{i=1}^{N}$. Il n'est composé que de N exemples d'entrée $\boldsymbol{x}$ sans valeur de sortie y_i. Le système essaie d'apprendre de lui-même à partir des entrées, sans avoir de sorties. L'absence de valeur y_i empêche l'usage d'une métrique d'erreur puisqu'il n'est plus possible d'estimer la sortie observée à une valeur de référence.

L'objectif consiste alors à trouver des motifs utiles sur la structure au sein de $\mathcal{D}$. En conséquence, il adresse des tâches très différentes, explicitées ci-après.

Regroupement Un algorithme de regroupement (*clustering* en anglais) sépare l'ensemble $\mathcal{D}$ en k sous-groupes. Ces regroupements imposent de mesurer une certaine similarité ou dissimilarité entre les instances présentes dans $\mathcal{D}$ (Omran et al., 2007). La division des données est réalisable en simples partitions ou de manière hiérarchique (Murtagh & Contreras, 2017), faisant ainsi apparaître des sous-groupes.

Ce type de tâches présente un intérêt pour la segmentation de clientèle et la promotion ciblée. Les systèmes de recommandations sur les sites Web marchand ou encore sur les sites de vidéo à la demande essaient de proposer des articles ou des vidéos similaires à ceux déjà achetés. En faisant des propositions pertinentes, les entreprises utilisant ces systèmes de recommandations maximisent leurs chances de garder le client captif.

Réduction de dimension Lors de la création d'un jeu de données, les créateurs peuvent être tentés d'enregistrer un grand nombre de caractéristiques d'entrées. Pour autant, certaines d'entre elles ne sont pas forcément nécessaires ou portent très peu d'information.

La réduction de dimension se définit d'un point de vue mathématique de la façon suivante (Kumar, 2009). Étant donné un vecteur d'entrée $\boldsymbol{x}$ de dimension r, il s'agit de trouver un vecteur de sortie $\boldsymbol{x'}$ de dimension $k < r$ préservant autant que possible, selon un critère prédéfini, l'information principale des données d'origine.

La réduction de dimension avant l'utilisation d'un autre algorithme conduit, par exemple, à un classifieur de moindre complexité, sans impacter fortement les performances de classification.

3.2.3 Apprentissage semi-supervisé

Bien que l'apprentissage semi-supervisé n'apparaissent pas sur la Figure 3.1, il mérite quelques mots. Il se situe à mi-chemin entre l'apprentissage supervisé et non-supervisé

dans le sens où une partie seulement des entrées possèdent une sortie. Un jeu de données pour l'apprentissage supervisé peut aussi être utilisé en ignorant une partie des sorties. L'intérêt principal réside toutefois dans l'utilisation de données partiellement annotées. C'est une solution à la difficulté d'obtenir des jeux de données complètement et correctement labellisés.

L'apprentissage semi-supervisé sert aussi bien à de la classification que du regroupement contraint (X. Zhu & Goldberg, 2009).

3.2.4 Apprentissage par renforcement

L'apprentissage par renforcement diffère totalement des types d'apprentissage évoqués jusqu'ici. Un agent interagit avec son environnement comme illustré sur la Figure 3.2. Basé sur ses observations, il décide de prendre des actions. L'apprentissage progresse en réalisant des essais et des erreurs. Un système de récompenses, positives ou négatives, permet à l'agent d'apprendre par lui-même une politique définissant quelles actions il doit prendre, dans une situation donnée, pour maximiser ses récompenses.

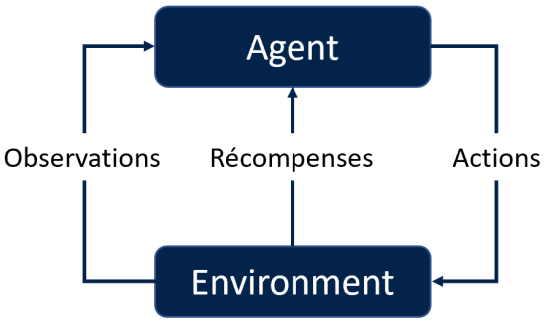


Figure 3.2 – Apprentissage par renforcement.

Une des difficultés dans ce type d'apprentissage consiste à trouver le bon compromis entre l'exploration de nouvelles stratégies et l'exploitation de stratégies déjà identifiées (Sutton & Barto, 2018).

L'apprentissage par renforcement présente un intérêt notamment pour l'acquisition de connaissances ou de compétences (Li et al., 2019), pour les jeux (Silver et al., 2017) ou encore pour la mobilité des robots (K. Zhu & Zhang, 2021).

3.3 Algorithmes de ML

Après la revue de différents types d'apprentissage et de leurs tâches associées, vient naturellement le choix des algorithmes. Cette décision n'est pas aisée, certains algorithmes pouvant être utilisés dans différents types de tâches. Pour compliquer davantage le choix, la taille du corpus numérique est importante. En effet, Banko et Brill ont montré qu'avec une quantité de données suffisamment grande, le choix de l'algorithme présente assez peu

d'importance. Ce point est illustré par la FIGURE 3.3, extraite de leur publication (BANKO & BRILL, 2001).

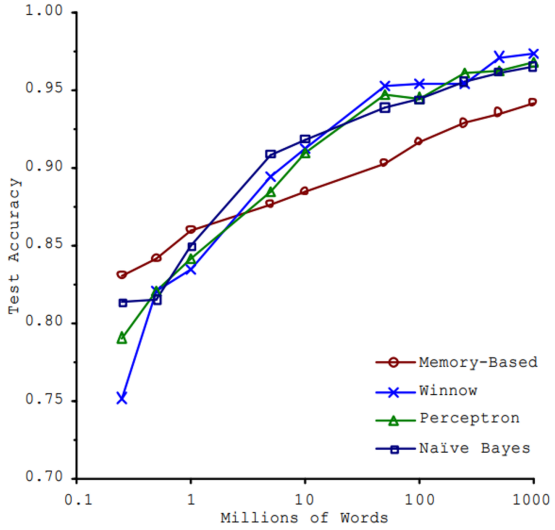


FIGURE 3.3 – Courbes d'apprentissage de désambiguïsation pour un ensemble de confusion montrant l'importance de la quantité de données.

Selon les auteurs, « ces résultats suggèrent que nous devrions peut-être reconsidérer le compromis entre le temps et l'argent consacrés au développement d'algorithmes et ceux consacrés au développement de corpus ». Néanmoins, leurs conclusions ont été obtenues avec des corpus allant jusqu'à un milliard de mots. En pratique, les jeux de données sont souvent de taille réduite et dans ce cas, le choix de l'algorithme reste un élément primordial.

La boîte à outils du machine learning est très fournie, les algorithmes sont nombreux et nous étudions quelques algorithmes régulièrement utilisés dans le contexte de la détection d'intrusions. Avec l'augmentation de performance des ordinateurs, des solutions d'apprentissage automatique à base de réseaux de neurones ont vu le jour et se sont développées. Pour distinguer les algorithmes détaillés ici, nous aborderons ceux généralement plus anciens, dit « classiques » dans la section 3.3.1. Les solutions utilisant des réseaux de neurones sont traitées dans la section 3.3.2. Enfin, la section 3.3.3 explicite le processus d'entraînement et ses challenges.

3.3.1 Algorithmes classiques

3.3.1.1 Analyse discriminante linéaire et quadratique

L'analyse discriminante linéaire (LDA) est une généralisation de l'analyse discriminante de Fisher (FISHER, 1936), une technique statistique pouvant être utilisée pour une réduction de dimensionnalité ou pour faire de la classification.

L'idée est de trouver un vecteur sur lequel les données sont projetées. Cette projection correspond bien à une réduction de dimensionnalité. En choisissant judicieusement ce vecteur, les données projetées peuvent être facilement séparées et c'est ainsi que nous obtenons un classifieur.

La FIGURE 3.4 (BISHOP, 2006, p. 188) illustre un exemple simple permettant de visualiser le fonctionnement de l'analyse discriminante de Fisher sur deux classes. La partie gauche montre un mauvais choix conduisant à une superposition des deux distributions. La partie droite illustre un bon choix de projection séparant parfaitement les deux classes, sans superposition de celles-ci.

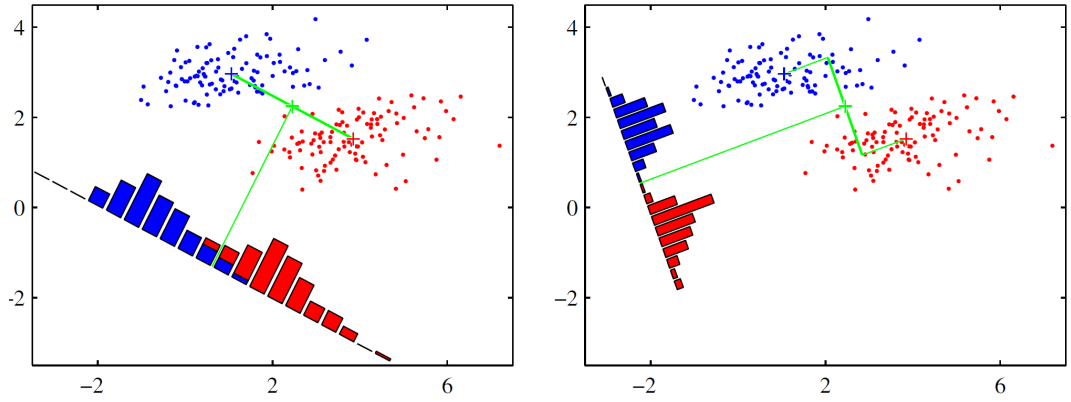


FIGURE 3.4 – Projection et classification binaire par l'analyse discriminante de Fisher

Dans le cas d'une classification binaire, LDA projette un vecteur de caractéristiques $\boldsymbol{x}$ à p dimensions sur un hyperplan divisant l'espace en deux demi-espaces (DUDA et al., 2001). Chaque demi-espace correspond à une classe. La limite de décision, donnée par l'équation 3.1 est caractérisée par le vecteur $\boldsymbol{w}$ normal à l'hyperplan et le seuil w_0.

$$[w_1, \ldots, w_p]^T . [x_1, \ldots, x_p] = \boldsymbol{w}^T . \boldsymbol{x} + w_0 = 0 \tag{3.1}$$

Cette méthode fonctionne sur la base de deux hypothèses que doivent respecter les données. La première concerne les distributions conditionnelles de classe $P(\boldsymbol{x}|classe = 0)$ et $P(\boldsymbol{x}|classe = 1)$ devant suivre des distributions normales de moyennes $\boldsymbol{\mu_0}$ et $\boldsymbol{\mu_1}$ et de matrice de covariances $\boldsymbol{\Sigma_0}$ et $\boldsymbol{\Sigma_1}$. Dans ces conditions, pour calculer le vecteur $\boldsymbol{w}$, la stratégie de classification optimale est d'affecter les entrées à la première classe si le rapport de log-vraisemblance $\ln\left(\frac{P(x|classe=0)}{P(x|classe=1)}\right)$ est supérieur à un seuil S et de les affecter à la seconde classe s'il est inférieur. Puisque les deux distributions sont gaussiennes, ceci se réduit à l'équation 3.2.

$$(\boldsymbol{x} - \boldsymbol{\mu_0})^T \boldsymbol{\Sigma_0^{-1}} (\boldsymbol{x} - \boldsymbol{\mu_0}) - (\boldsymbol{x} - \boldsymbol{\mu_1})^T \boldsymbol{\Sigma_1^{-1}} (\boldsymbol{x} - \boldsymbol{\mu_1}) > S \tag{3.2}$$

En l'état, l'équation est de forme quadratique et son utilisation correspond à l'analyse discriminante quadratique (QDA). Pour obtenir la forme linéaire, LDA ajoute une condition d'homoscédasticité. Dans ce cas, $\boldsymbol{\Sigma_0} = \boldsymbol{\Sigma_1} = \boldsymbol{\Sigma}$ et l'équation 3.2 se simplifie pour

donner l'équation 3.3.

$$\boldsymbol{w}^T \boldsymbol{x} > S \text{ avec } \boldsymbol{w} = \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu_0} - \boldsymbol{\mu_1}) \tag{3.3}$$

LDA comme QDA se généralise à des classifications multi-classes.

3.3.1.2 Machine à vecteurs de support

Le concept de machine à vecteurs de support (SVM) est simple à comprendre, sans même utiliser de formules mathématiques. Un exemple de classification binaire en deux dimensions est illustré par la FIGURE 3.5.

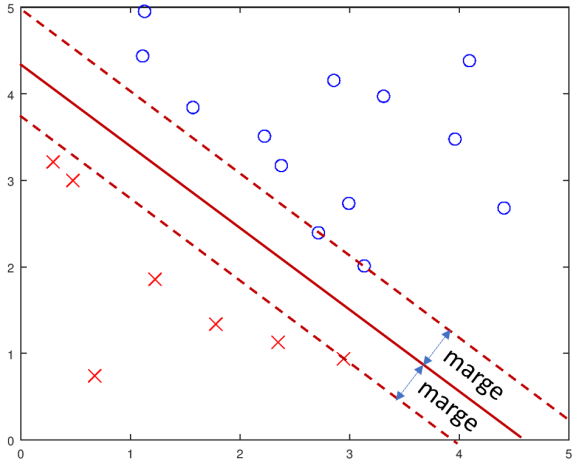


FIGURE 3.5 – Classification binaire avec SVM.

L'idée du SVM est de définir un plan, ou plus généralement un hyperplan, s'appuyant sur des points du jeu de données et maximisant la marge séparant les classes. La sélection d'un tel hyperplan augmente la capacité de généralisation du SVM en classifiant correctement des données non rencontrées durant l'entraînement.

Lorsqu'un corpus numérique contient des classes non séparables linéairement, SVM reste une option valable. Avant de rechercher l'hyperplan optimal, les vecteurs d'entrées du corpus vont être projetés dans un espace de plus grande dimension, avec l'espoir que les vecteurs dans ce nouvel espace puissent être séparables avec des frontières linéaires (CORTES & VAPNIK, 1995). Dans ces cas, les transformations se font par l'utilisation de noyaux, par exemple basée sur une fonction polynomiale ou encore une fonction à base radiale gaussienne.

L'implémentation de la fonction d'apprentissage pour les méthodes à noyaux a été optimisé dans le but d'accélérer les calculs et réduire l'empreinte mémoire (PLATT, 1998). Malgré cela, la durée d'entraînement varie entre $\mathcal{O}(m^2 \times n)$ à $\mathcal{O}(m^3 \times n)$ où m est le nombre d'instances et n le nombre de caractéristiques (GERON, 2019, p. 162). En comparaison, la durée d'entraînement pour un SVM linéaire est en $\mathcal{O}(m \times n)$. En conséquence, les

techniques de SVM à noyaux sont difficilement utilisables pour des corpus numériques de grandes dimensions.

En plus de son intérêt pour les tâches de classification, SVM supporte également les tâches de régression (Drucker et al., 1996). Dans ce cas, l'objectif consiste à déterminer un hyperplan avec une marge contenant le maximum de données du corpus.

3.3.1.3 *K* plus proches voisins

L'algorithme des k plus proches voisins (k-NN) est un classifieur non paramétrique, basé sur les distributions de probabilité (Bishop, 2006, p. 120). Il appartient à la famille d'apprentissage basé sur les instances, souvent appelé apprentissage paresseux. Contrairement à ceux basés sur des modèles, les algorithmes d'apprentissage basés sur les instances ne nécessitent pas de phase d'entraînement. Les traitements sont retardés jusqu'à l'arrivée d'une nouvelle instance à classer, moment auquel elle est comparée à l'ensemble des données d'entraînement.

Le principe de fonctionnement de cet algorithme est extrêmement simple à comprendre et est illustré par la Figure 3.6.

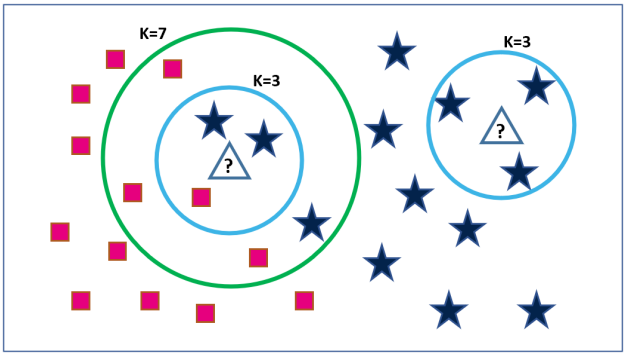


Figure 3.6 – Exemples de classification binaire avec les k plus proches voisins.

Pour les deux nouvelles instances à classer, identifiées par des triangles contenant un point d'interrogation, les distances entre ces instances et toutes les instances déjà connues sont mesurées. Seuls les k plus proches voisins sont gardés et la classe majoritaire de cet ensemble est attribuée à l'instance à classer. Le choix de k est important puisqu'il peut influencer la classification. Les deux cercles concentriques, vert pour $k = 3$ et bleu pour $k = 7$, montre que la classe allouée varie, dans cet exemple, en fonction de la valeur de k.

k-NN étant un algorithme basé sur les distances, il est nécessaire de sélectionner une métrique de distance. Parmi celles régulièrement utilisées, les quatre fonctions de distance d_1, d_2, d_3 et d_4 respectivement Manhatan, euclidienne, Minkowski et Mahalanobis entre deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ de dimension n sont donnée par les équations 3.4 à 3.7. La distance Mahalanobis fait intervenir un vecteur $\boldsymbol{\mu}$, de dimension n, des valeurs moyennes des caractéristiques et une matrice de covariance $\boldsymbol{\Sigma}$.

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.4}$$

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.5}$$

$$d_3(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{3.6}$$

$$d_4(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \tag{3.7}$$

L'algorithme est généralement décrit en utilisant la distance euclidienne (Mitchell, 1997).

Notons que l'utilisation d'une métrique de distance nécessite une préparation des données avant classification, afin de placer toutes les caractéristiques sur une même échelle de valeurs.

Comme pour SVM, k-NN supporte également les tâches de régression (Altman, 1992). Dans ce cas, la valeur de sortie correspond à la valeur moyenne des k plus proches voisins.

3.3.1.4 Arbre de décision et forêt aléatoire

Arbre de décision L'apprentissage par arbre de décision (DT) est une méthode parmi les plus populaires en inférence inductive. Un arbre de décision est descendant et classe les instances en les triant depuis la racine juqu'aux feuilles, chacune d'elles représentant une classe. Très facile à comprendre et à interpréter, il s'implémente simplement avec une série de règles *si-alors-sinon*. Chaque nœud est un point de décision, comportant un test sur une caractéristique unique, permettant de séparer au mieux les données selon leur classe. Un nœud conduit soit à un nœud suivant, soit une feuille terminale.

La construction de l'arbre nécessite, pour chaque nœud, de sélectionner la caractéristique séparant au mieux l'ensemble d'entrées en sous-ensembles de sorties. Les critères de sélection se basent généralement sur l'entropie, le gain d'information ou encore sur l'impureté de Gini.

L'entropie H, utilisée en théorie de l'information, est donnée par l'équation 3.8 et mesure l'impureté dans un ensemble d'instances d'entraînement.

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \tag{3.8}$$

Le gain d'information est un critère dérivé de l'entropie, connaissant la valeur d'un attribut (Quinlan, 1993). il permet de mesurer l'efficacité d'un attribut à classifier les données en sous-ensembles.

L'impureté de Gini est une mesure de la probabilité de classification incorrecte d'une nouvelle instance possédant une variable aléatoire, dans l'hypothèse où cette instance

serait classée au hasard selon la distribution des étiquettes de classe dans le sous-ensemble de données. L'impureté de Gini est donnée par l'équation 3.9 où n représente le nombre de classes et P_i le ratio d'éléments labellisés avec la classe i dans l'ensemble.

$$Gini = 1 - \sum_{i=1}^{n} {P_i}^2 \tag{3.9}$$

La structure de l'arbre et ses performances varient très peu selon le critère choisi (Raileanu & Stoffel, 2004). L'impureté de Gini est légèrement plus rapide à calculer que les autres critères grâce à l'absence de logarithme, mais peut néanmoins conduire à un arbre moins équilibré (Geron, 2019, p. 181).

Les arbres de décision les plus connus sont ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) et CART (Breiman, 2017). Avec ID3, la séparation à chaque nœud a lieu en trouvant la caractéristique, devant impérativement être catégorielle, selon un critère d'impureté tel que l'entropie ou l'index Gini.

C4.5 supprime la restriction des caractéristiques catégorielles et supporte des valeurs continues. Les arbres sont développés jusqu'à leur taille maximale, puis une étape d'élagage est généralement appliquée dans le but d'améliorer la capacité de l'arbre à se généraliser aux données jamais rencontrées. L'algorithme C4.5 donne de meilleures performances que ID3, tant en termes de justesse qu'en temps d'exécution (Hssina et al., 2014). Ce résultat n'est pas surprenant puisque C4.5 est plus récent et conçu par le même chercheur (Quinlan).

CART est très proche de l'algorithme C4.5. Il diffère en étant un arbre binaire et en utilisant l'impureté de Gini par défaut. La séparation à chaque nœud en deux sous-ensembles utilise une caractéristique c et un seuil s. L'algorithme recherche parmi toutes les paires (c, s) celle qui produira le sous-ensemble le plus pur. Comme pour C4.5, un élagage est généralement appliqué pour une meilleure généralisation. De plus, cet algorithme prend en charge les variables cibles numériques permettant son usage sur des tâches de régression.

Forêt aléatoire Une forêt aléatoire (RF) est une extension mettant en parallèle un ensemble d'arbres de décision. Un arbre de décision seul peut avoir un biais élevé. La technique dite de *bagging*, mettant en parallèle un nombre suffisant d'arbres, obtient de meilleurs résultats quand les erreurs non corrélées de chaque arbre s'annulent en moyenne.

Chaque arbre est entraîné sur un sous-ensemble des données d'entraînement différent, réalisé par un tirage aléatoire avec remise afin de limiter les biais si les classes ne sont pas équilibrées. La Figure 3.7 présente le schéma d'une forêt composée de quatre arbres. Chaque arbre de décision va fournir sa prédiction à un classifieur final, par exemple basé sur un vote à la majorité, dans le but d'estimer la classe de l'instance testée.

L'utilisation d'une sélection aléatoire des caractéristiques pour séparer chaque nœud (Breiman, 2001) améliore la robustesse du classifieur en augmentant la diversité des arbres générés et en décorrélant davantage les erreurs produites par chaque arbre.

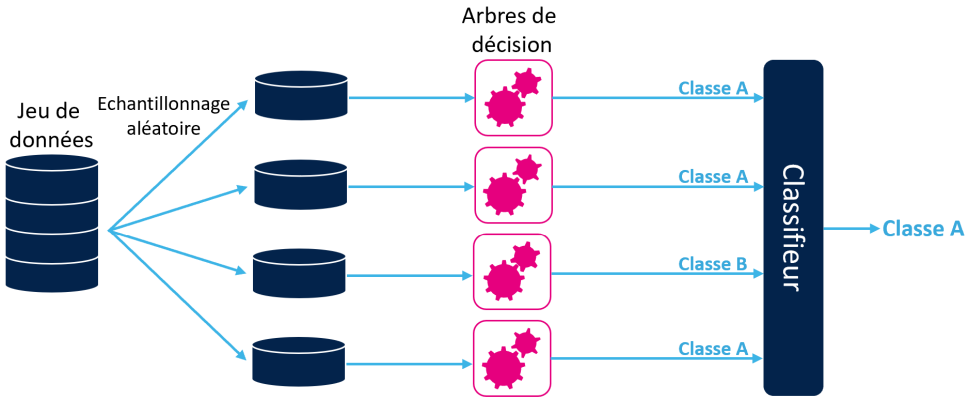


FIGURE 3.7 – Schéma d'une forêt aléatoire.

Le caractère aléatoire peut être renforcé en utilisant des seuils aléatoires sur les caractéristiques pour créer des forêts appelées *Extremely Randomized Tree* (GEURTS et al., 2006).

3.3.2 Réseaux de neurones

Le développement de réseaux de neurones n'est pas nouveau. Il a commencé par une approche en neuroscience dans les années 1940. En 1958, Franck ROSENBLATT, un psychologue américain, a créé le plus simple d'entre-eux : le perceptron (ROSENBLATT, 1958). Comme décrit sur la FIGURE 3.8, ce réseau ne posséde qu'un seul nœud. dans laquelle x_i représente l'entrée venant de la synapse i, w_i le poids de la synapse i, f une fonction non-linéaire et y la sortie du neurone. Le cas de x_0 de valeur unitaire est particulier et représente le biais, appris par le poids w_0.

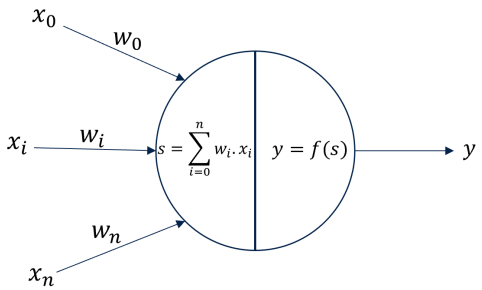


FIGURE 3.8 – Représentation mathématique du perceptron de Rosenblatt.

Ce modèle basique se limitait à séparer des classes de façon linéaire, mais il est néanmoins à l'origine d'une grande variété de réseaux de neurones. De nombreux modèles, très différents et atteignant une très grande complexité, ont été conçus et sont en partie mentionnés dans la FIGURE 3.9 (van VEEN, 2016).

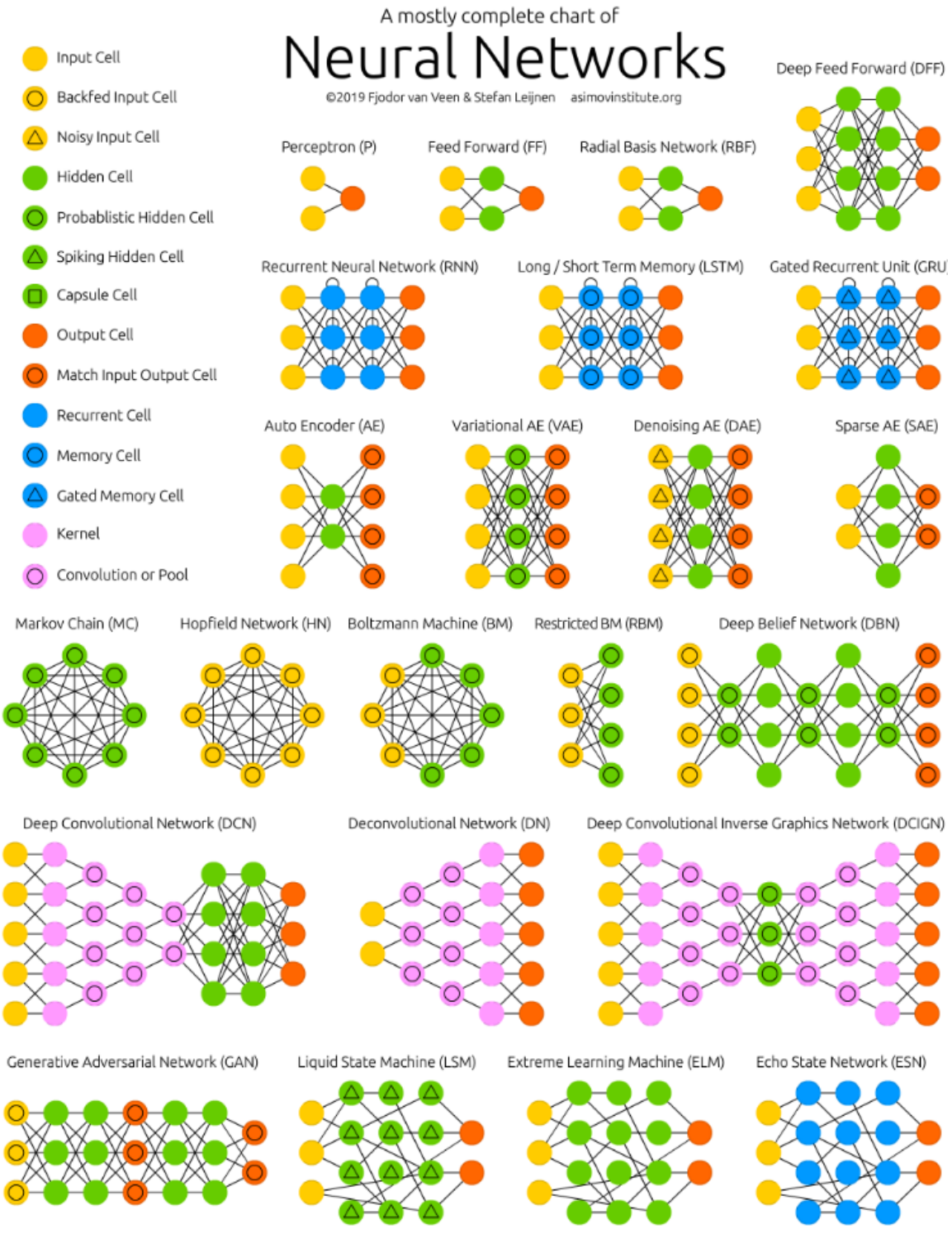


FIGURE 3.9 – Panel de réseaux de neurones.

Nous étudions plus en détail le perceptron multi-couches (MLP), les réseaux récurrents (RNN), les réseaux convolutifs (CNN) et les auto-encodeurs (AE).

3.3.2.1 Perceptron multi-couches

Le MLP, aussi appelé réseau de neurones à propagation avant (*feed-forward neural network* en anglais), est une extension du perceptron de Rosenblatt contenant davantage de nœuds, sur plusieurs couches, comme illustrée sur la Figure 3.10.

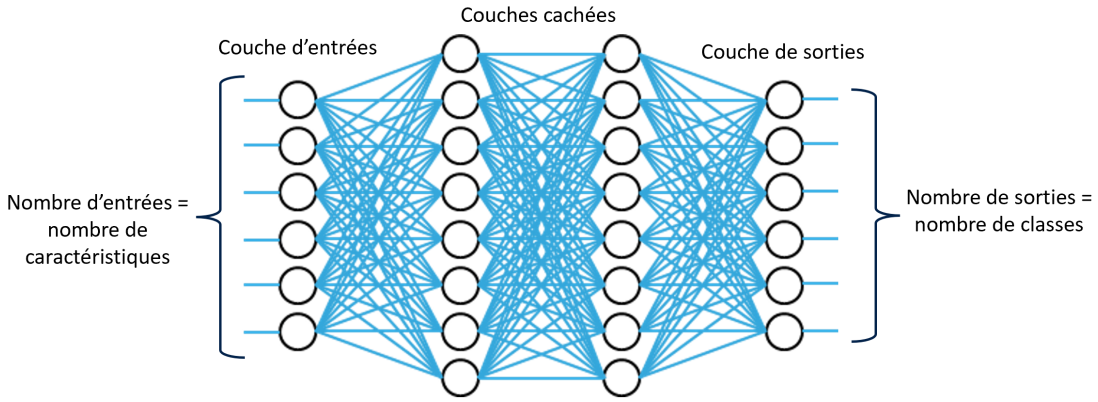


Figure 3.10 – Structure d'un perceptron multi-couches.

Il est composé d'une couche d'entrée, suivie d'un nombre variable de couches cachées, puis d'une couche de sortie. Chaque nœud est connecté à tous ceux de la couche suivante.

Le MLP est capable de faire des séparations de classes non-linéaires. Il a été démontré qu'un modèle à une seule couche cachée est un approximateur universel (Hornik, 1991). Toutefois, il est généralement plus efficace d'augmenter le nombre de couches plutôt que d'augmenter le nombre de nœuds d'une couche unique.

Les fonctions non-linéaires des nœuds peuvent être choisies dans une liste de fonctions couramment utilisées. Les fonctions sigmoïde σ et tangente hyperbolique $tanh$ sont données respectivement par les équations 3.10 et 3.11.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.11}$$

Ces deux équations présentent deux problèmes importants. D'une part, elles comportent des plateaux à leurs extrémités posant un problème lors de la phase d'entraînement avec des gradients extrêmement faibles dans ces zones. D'autre part, le calcul des exponentielles peut être coûteux sur un système contraint en ressources.

Des alternatives sont possibles avec les fonctions dérivant de la fonction unité linéaire rectifié ReLU, LeakyReLU, et SELU données respectivement par les équations 3.12 à 3.14.

$$relu(x) = \begin{cases} 0 & \text{pour } x < 0 \\ x & \text{pour } x \geq 0 \end{cases} \tag{3.12}$$

$$leakyrelu(x) = \begin{cases} \alpha.x & \text{pour } x < 0 \\ x & \text{pour } x \geq 0 \end{cases} \tag{3.13}$$

$$selu(x) = \lambda. \begin{cases} \alpha.(e^x - 1) & \text{pour } x < 0 \\ x & \text{pour } x \geq 0 \end{cases} \tag{3.14}$$

Les fonctions ReLU et LeakyReLU ont l'avantage d'être extrêmement simples à calculer. Quant à la fonction SELU, plus récente, elle présente un inconvénient et un avantage. D'un côté, elle nécessite une exponentielle. De l'autre, elle facilite l'entraînement des réseaux de neurones en auto-normalisant les sorties quand $\lambda = 1.0507$ et $\alpha = 1.6733$ (Klambauer et al., 2017) et donne généralement de meilleurs résultats.

Pour les problèmes de classification, la couche de sortie utilise généralement la fonction softmax pour chaque sortie i, donnée par l'équation 3.15.

$$softmax\,(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{N_c} e^{x_k}} \text{ pour } i \in [1; N_c] \tag{3.15}$$

La fonction softmax, aussi appelée exponentielle normalisée, fournit une distribution de probabilité discrète pour un nombre de classes N_c (Bishop, 2006, p. 198). Cette fonction généralise la fonction sigmoïde, utilisée pour représenter une distribution de probabilité lors d'une classification binaire.

3.3.2.2 Réseaux récurrents

Contrairement aux réseaux de neurones à propagation avant, les réseaux récurrents (RNN) possèdent une boucle de rétroaction. Chaque nœud à l'instant t prend l'entrée x_t et calcule, d'une part, sa sortie y_t, et d'autre part, un état caché h_t, tout en prenant en compte l'état caché de l'instant précédent h_{t-1}. Pour faciliter la compréhension, le nœud du RNN peut être dépliée dans le temps comme sur la Figure 3.11.

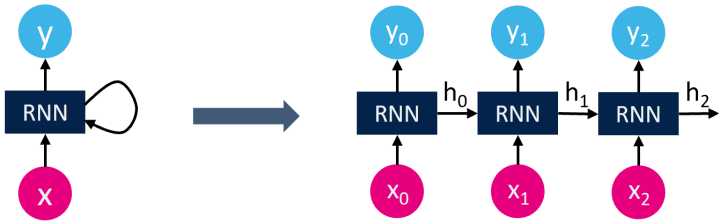


Figure 3.11 – Structure d'un nœud RNN déplié dans le temps.

Le RNN, grâce à sa boucle de rétroaction, agit comme une cellule mémoire. Il est capable de reconnaître des motifs dans des séries temporelles et de prédire une évolution future. Dans le domaine automobile, les RNN peuvent être utiles dans l'analyse de trajectoire et éviter des accidents.

De manière générale, les RNN prennent en entrées des séries de tailles variables et peuvent également générer des séries de tailles également variables. Les RNN sont typiquement utilisés pour modéliser des séquences :

— prédire une valeur à partir d'une série de mesures passées ;
— prédire une série de mots à partir d'une image pour du sous-titrage ;
— prédire une série de mots à partir d'une série de mots pour de la traduction automatique, sans que les séries soient de même taille.

Le RNN de base présentent des inconvénients pour leur entraînement et sa capacité de mémorisation qui est très courte. Ces problèmes ont été adressés par des versions améliorées comme celle à mémoire court et long terme (LSTM) (Gers et al., 2000 ; Hochreiter & Schmidhuber, 1997) et l'unité récurrente à porte (GRU) (Cho et al., 2014). Le LSTM a une structure complexe et possède deux états cachés, l'un pour le court terme et l'autre pour le long terme. Le concept consiste à apprendre une entrée importante grâce à une porte d'entrée, la stocker et la conserver plus ou moins longtemps grâce à la porte d'oubli et l'extraire lorsqu'elle est requise grâce à la porte de sortie. De nombreuses variantes de LSTM existent. Le GRU est une variante de LSTM simplifié donnant d'aussi bons résultats (Greff et al., 2017) et justifiant ainsi une utilisation croissante du GRU.

3.3.2.3 Réseaux convolutifs

Les réseaux de neurones convolutifs (CNN) sont un des meilleurs exemples de l'impact de la neurobiologie sur l'apprentissage automatique. Le néocognitron (Fukushima & Miyake, 1982), inspiré d'études sur le cortex visuel, a évolué pour aboutir au CNN capable de reconnaitre les nombres manuscrits sur les chèques avec l'architecture LeNet5 (Lecun et al., 1998). Les CNN sont largement utilisés dans les tâches visuelles telles que la classification d'objets ou la segmentation d'images.

Ces réseaux de neurones ont la particularité d'utiliser des couches de convolution, composée de trois étapes comme indiquées dans la Figure 3.12 (Goodfellow et al., 2016).

L'étape de convolution fait glisser une fenêtre sur les données d'entrées (Albawi et al., 2017). La transformation affine réalisée n'est autre qu'un filtre produisant des cartes de caractéristiques, appelée *feature maps* en anglais. La couverture spatiale d'un filtre par rapport à l'image source est appelée son champ réceptif. L'utilisation d'un champ réceptif réduit considérablement le nombre de poids qui seraient nécessaires si chaque point d'entrée était connecté à un nœud du réseau de neurones. La couche de détection de la deuxième étape est généralement une fonction d'activation non-linéaire de type ReLU. Enfin, la dernière étape de *pooling* permet de réduire la dimension des données à traiter par la suite grâce à une fonction, souvent MaxPooling (Zhou & Chellappa, 1988). Cette

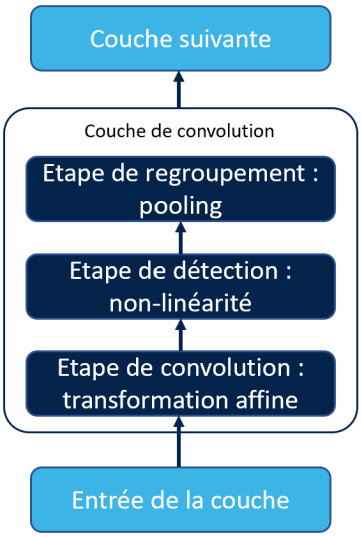


FIGURE 3.12 – Étapes d'une couche de convolution.

fonction retourne la valeur maximale à l'intérieur d'un voisinage rectangulaire, aidant ainsi à rendre la représentation un peu plus invariante à des petites translations de l'entrée.

Dans une structure typique, les couches de convolution sont empilées et les données de la dernière de ces couches sont mises à plat pour être injectées dans un MLP comme le montre la FIGURE 3.13.

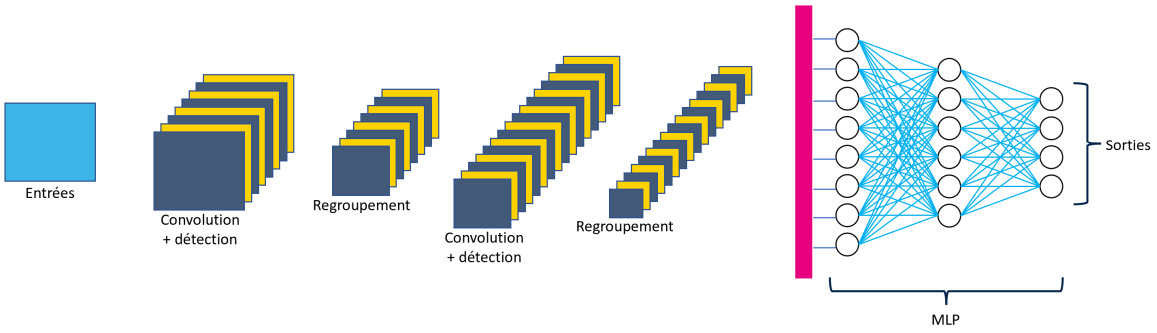


FIGURE 3.13 – Structure simplifiée d'un CNN

Plus les tâches à réaliser sont complexes et plus les nombres de couches et de *feature maps* sont grandes. Ces CNN conduisent à un grand nombre de paramètres à apprendre. En conséquence, la phase d'entraînement est extrêmement longue et nécessite des machines puissantes. L'utilisation de la fonction d'activation non-linéaire ReLU simplifie les calculs.

Pour compenser l'augmentation toujours croissante des modèles à base de CNN, les poids sont souvent quantifiés sur des entiers codés sur 8 bits. Des quantifications plus agressives montrent des résultats faiblement impactés avec une diminution significative

de la taille mémoire et des temps de calculs nécessaires (Courbariaux et al., 2015 ; Dockhorn et al., 2021).

L'utilisation de CNN va bien au-delà du simple traitement d'images. En utilisant des convolutions sur une dimension, il devient possible de traiter par exemple des signaux audio. Les couches de convolution peuvent être utilisées avec des RNN et obtenir des effets mémoires plus longs.

Dans le domaine de l'imagerie médicale, les convolutions en trois dimensions sont utilisables, par exemple, pour traiter les images de tomodensitométrie.

3.3.2.4 Réseaux auto-encodeur

L'objectif des réseaux auto-encodeur (AE) est d'apprendre à reproduire ses entrées à la sortie en passant par une représentation minimale. L'AE le plus simple est un cas particulier de MLP. Sa structure est symétrique avec un même nombre d'entrées et de sorties, une diminution du nombre de nœuds par couche, suivie d'une augmentation comme le montre la Figure 3.14.

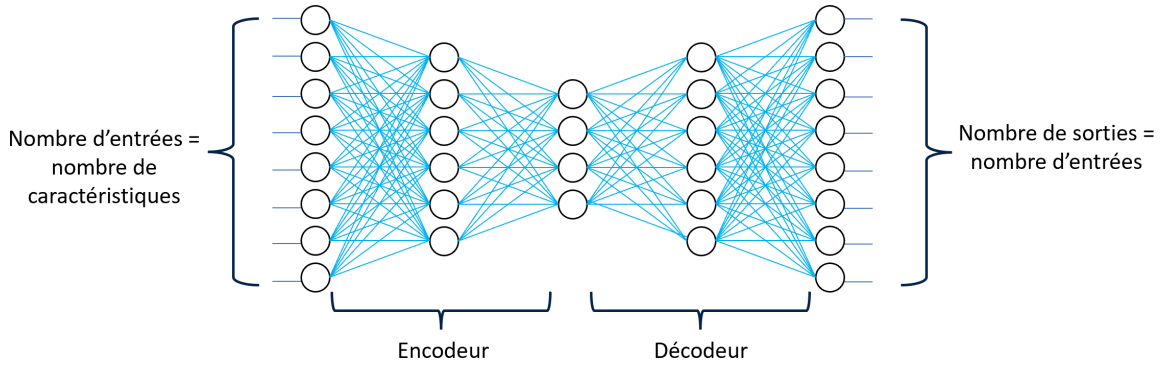


Figure 3.14 – Structure de l'auto-encodeur.

La première partie réalise un encodeur en cherchant une représentation des données d'entrées sur un ensemble de dimension réduite. Quant à la seconde, elle réalise l'opération inverse en transformant les données encodées vers l'ensemble de dimension d'origine. Ce schéma peut être adapté pour utiliser des couches convolutives ou pour gérer des séquences comme les RNN.

Les AE sont utilisés par exemple en apprentissage non-supervisé, pour réduire la dimension de l'ensemble d'entrées, et donc le nombre de caractéristiques. Lorsque l'erreur de reconstruction entre les entrées et les sorties est faible, la couche centrale est de dimension suffisante pour contenir l'information utile des caractéristiques d'entrées. Toutes les données d'entrées peuvent alors être converties par l'encodeur avant leur utilisation à l'entrée d'un système de classification et ainsi réduire sa complexité.

En apprentissage semi-supervisé, l'AE peut être entrainé à reproduire une classe particulière, correspondant par exemple au fonctionnement nominal d'un système. Une aug-

mentation de l'erreur de construction indique un fonctionnement anormal, c'est donc un détecteur d'anomalie.

En changeant légèrement la symétrie, l'AE peut servir de débruiteur. Il suffit pour cela d'ajouter juste après la couche d'entrée un bruit. L'entraînement de l'AE cherche à reproduire les entrées donc à supprimer le bruit.

3.3.3 Processus d'entraînement

L'entraînement d'un réseau de neurones est similaire à celui d'autres algorithmes d'apprentissage automatique, tel que la régression logistique, avec quelques subtilités pour adresser ses particularités. L'algorithme d'apprentissage utilise la descente du gradient (Cauchy et al., 1847).

3.3.3.1 Descente du gradient

Le processus d'entraînement suit le schéma décrit par la Figure 3.15.

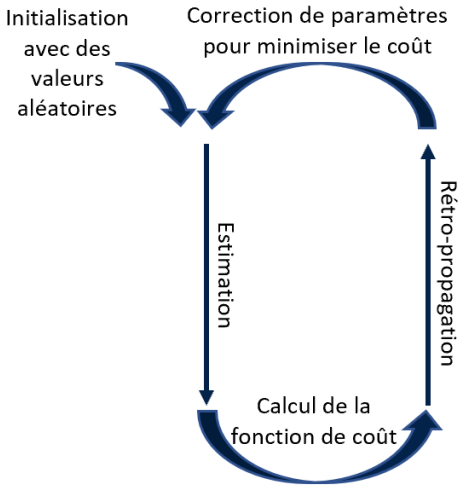


Figure 3.15 – Schéma d'entraînement d'un réseau de neurones.

Les poids du réseau sont initialisés avec des valeurs aléatoires. L'algorithme d'entraînement utilise les entrées pour calculer les sorties du réseau de neurones. Il mesure ensuite l'erreur de prédiction grâce à une fonction de coût comparant les sorties souhaitées et les prédictions. Les contributions de chaque poids à l'erreur sont mesurés grâce au gradient d'erreur et sont rétro-propagés jusqu'à la couche d'entrée. Enfin, les poids sont mis à jour pour diminuer l'erreur.

Ce cycle, appliquée à toutes les instances d'entraînement, est appelé une époque, et est exécuté à de multiples reprises, jusqu'à l'obtention d'une erreur acceptable.

Reprenons l'exemple du perceptron de la FIGURE 3.8 dans lequel la sortie y est donnée par l'équation 3.16.

$$y = f\left(\sum_{i=0}^{n} w_i.x_i\right) \tag{3.16}$$

La mise à jour des poids w_i se fait grâce à l'équation 3.17 où α est le taux d'apprentissage.

$$w_i = w_i - \alpha \cdot \frac{\partial y}{\partial w_i} \tag{3.17}$$

Lors de la rétro-propagation, la dérivée partielle donne la pente de la fonction de coût et α la taille du pas fait dans la direction de la pente.

3.3.3.2 Challenges de l'entraînement

Un des challenges de l'entraînement est lié à l'utilisation de fonction non-linéaire. Les fonctions de coût avec les fonctions d'activation les plus pertinentes ne sont pas convexes et il n'y a aucune garantie de déterminer un minimum global. Des paramètres initiaux différents risquent de conduire la descente de gradient à un minimum local. Par ailleurs, il est recommandé d'initialiser les paramètres de manière aléatoire, selon des distributions centrées en zéro mais avec des variances spécifiques, dépendantes des fonctions d'activations choisies (GLOROT & BENGIO, 2010 ; HE et al., 2015).

Les fonctions de coûts peuvent présenter des points où le gradient est extrêmement faible, typiquement des points-selles. Un entraînement sur un grand nombre d'époques est nécessaire, mais peut ne pas être suffisant. Pour contourner ce problème, plusieurs algorithmes ont la capacité d'améliorer la descente de gradient. Nous en citons quelques-uns, sans les expliquer :

— la méthode d'accélération, aussi appelée d'inertie (POLYAK, 1964) ;
— le gradient accéléré de Nesterov (NESTEROV, 1983) ;
— l'algorithme Adagrad (DUCHI et al., 2011) ;
— l'algorithme Adam (KINGMA & BA, 2015) et sa variante, Nadam, appliquant l'accélération de Nesterov (DOZAT, 2016).

Toutes les versions d'algorithmes de descente du gradient nécessitent des réglages de variables appelées hyper-paramètres. Le choix de ces derniers est important pour obtenir rapidement de bons résultats et éviter une divergence lors de l'entraînement.

Aux variables de ces algorithmes s'ajoutent d'autres hyper-paramètres tels que le *dropout* permettant de contrôler le compromis biais/variance et éviter le sous-apprentissage et le sur-apprentissage (SRIVASTAVA et al., 2014).

Toutefois, aucune règle magique ne permet d'identifier facilement les valeurs à utiliser. Leur recherche est coûteuse en temps. Compte-tenu du nombre souvent élevé d'hyper-paramètres, le nombre de combinaisons à tester serait trop long et trop fastidieux. Dans pareil cas, il est préférable d'utiliser une recherche aléatoire (BERGSTRA & BENGIO, 2012).

3.4 Synthèse

Dans ce chapitre, nous avons dans un premier temps expliqué ce qu'est l'apprentissage automatique. Deux définitions permettent de mieux comprendre le concept de machine learning.

Nous avons ensuite présenté différentes méthodes d'apprentissage et les types de tâches qui leur sont associés. Nous avons couvert les apprentissages supervisé, non-supervisé, semi-supervisé et l'apprentissage par renforcement. Cette partie permet de se situer dans le contexte de la problématique énoncée page 19 « *Les objets connectés utilisant des communications par protocole IP peuvent-ils être protégés des tentatives d'intrusions réseau grâce à des techniques d'apprentissage automatique supervisé ?* ».

Dans un deuxième temps, nous avons abordé quelques algorithmes du machine learning. Les algorithmes d'analyse discriminante LDA, QDA, les machines à vecteurs de support SVM, les k plus proches voisins k-NN, les arbres de décision et les forêts aléatoires appartiennent à la famille des algorithmes classiques du domaine.

Nous avons ensuite couvert les réseaux de neurones. Plusieurs familles ont été décrites telles que les MLP, les RNN, les CNN et les auto-encodeurs.

Enfin, le processus d'entraînement des réseaux de neurones a été présenté. Plusieurs algorithmes optimisant la descente de gradient ont été évoqués ainsi que quelques challenges liés à l'entrainement des réseaux de neurones.

La bonne compréhension de ces différentes familles d'algorithmes de l'apprentissage automatique est essentielle. En effet, ces algorithmes sont utilisés dans les chapitres 4 et 5. L'interprétation des résultats obtenus nécessite de comprendre leur fonctionnement et leur limite.

Après avoir un état de l'art sur les objets connectés, la détection d'intrusions réseau et le machine learning, nous avons maintenant les éléments pour aborder les trois propositions adressant les hypothèses énoncées en introduction.

Références

Albawi, S., Mohammed, T. A. & Al-Zawi, S., (2017), Understanding of a convolutional neural network, *2017 International Conference on Engineering and Technology (ICET)*, 1-6, https://doi.org/10.1109/ICEngTechnol.2017.8308186

Altman, N. S., (1992), An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression, *The American Statistician*, *46 3*, 175-185, https://doi.org/10.1080/00031305.1992.10475879

Banko, M. & Brill, E., (2001), Scaling to very very large corpora for natural language disambiguation, *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, 26-33.

Bergstra, J. & Bengio, Y., (2012), Random Search for Hyper-parameter Optimization, *The Journal of Machine Learning Research*, *13*, 281-305.

Bishop, C. M., (2006), *Pattern Recognition and Machine Learning* (M. Jordan, J. Kleinberg & B. Schölkopf, Éd. ; T. 4), Springer, https://doi.org/10.1117/1.2819119

Breiman, L., (2001), Random forests, *Machine learning*, *45 1*, 5-32.

Breiman, L., (2017), *Classification and Regression Trees*, CRC Press.

Cauchy, A. et al., (1847), Méthode générale pour la résolution des systemes d'équations simultanées, *Comp. Rend. Sci. Paris*, *25 1847*, 536-538.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y., (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv :1406.1078*.

Cortes, C. & Vapnik, V., (1995), Support-vector networks, *Machine learning*, *20 3*, 273-297.

Courbariaux, M., Bengio, Y. & David, J.-P., (2015), BinaryConnect : Training Deep Neural Networks with binary weights during propagations, In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2015/file/3e15cc11f979ed25912dff5b0669f2cd-Paper.pdf

Dockhorn, T., Yu, Y., Sari, E., Zolnouri, M. & Partovi Nia, V., (2021), Demystifying and Generalizing BinaryConnect, In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang & J. W. Vaughan (Éd.), *Advances in Neural Information Processing Systems* (p. 13202-13216), Curran Associates, Inc., https://proceedings.neurips.cc/paper/2021/file/6e0cf80a83327822a972bcde3c1d9740-Paper.pdf

Dozat, T., (2016), Incorporating nesterov momentum into adam.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. & Vapnik, V., (1996), Support Vector Regression Machines, In M. Mozer, M. Jordan & T. Petsche (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/1996/file/d38901788c533e8286cb6400b40b386d-Paper.pdf

Duchi, J., Hazan, E. & Singer, Y., (2011), Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, *12 61*, 2121-2159, http://jmlr.org/papers/v12/duchi11a.html

Duda, R. O., Hart, P. E. & Stork, D. G., (2001), *Pattern Classification* (2e éd.), Wiley.

Fisher, R. A., (1936), The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, *7 2*, 179-188, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Fukushima, K. & Miyake, S., (1982), Neocognitron : A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition, In S.-i. Amari & M. A. Arbib (Éd.), *Competition and Cooperation in Neural Nets* (p. 267-285), Springer Berlin Heidelberg.

Geron, A., (2019), *Hands-on machine learning with scikit-learn, keras, and TensorFlow : Concepts, tools, and techniques to build intelligent systems* (2e éd.), O'Reilly Media.

Gers, F. A., Schmidhuber, J. & Cummins, F., (2000), Learning to Forget : Continual Prediction with LSTM, *Neural Computation*, *12 10*, 2451-2471, https://doi.org/10.1162/089976600300015015

Geurts, P., Ernst, D. & Wehenkel, L., (2006), Extremely randomized trees, *Machine learning*, *63 1*, 3-42.

Glorot, X. & Bengio, Y., (2010), Understanding the difficulty of training deep feedforward neural networks, In Y. W. Teh & M. Titterington (Éd.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (p. 249-256), PMLR, https://proceedings.mlr.press/v9/glorot10a.html

Goodfellow, I., Bengio, Y. & Courville, A., (2016), *Deep Learning*, MIT Press.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J., (2017), LSTM : A Search Space Odyssey, *IEEE Transactions on Neural Networks and Learning Systems*, *28 10*, 2222-2232, https://doi.org/10.1109/TNNLS.2016.2582924

He, K., Zhang, X., Ren, S. & Sun, J., (2015), Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Hochreiter, S. & Schmidhuber, J., (1997), Long Short-Term Memory, *Neural Computation*, *9 8*, 1735-1780, https://doi.org/10.1162/neco.1997.9.8.1735

Hssina, B., Merbouha, A., Ezzikouri, H. & Erritali, M., (2014), A comparative study of decision tree ID3 and C4.5, *International Journal of Advanced Computer Science and Applications*, *4 2*, 13-19.

Kingma, D. P. & Ba, J., (2015), Adam : A Method for Stochastic Optimization, *3rd International Conference for Learning Representations*.

Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S., (2017), Self-Normalizing Neural Networks, *Advances in Neural Information Processing Systems*, 971-980.

Kumar, A. C., (2009), Analysis of unsupervised dimensionality reduction techniques, *Computer science and information systems*, *6 2*, 217-227.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P., (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, *86 11*, 2278-2324, https://doi.org/10.1109/5.726791

Li, F., Jiang, Q., Zhang, S., Wei, M. & Song, R., (2019), Robot skill acquisition in assembly process using deep reinforcement learning [Deep Learning for Intelligent Sensing, Decision-Making and Control], *Neurocomputing*, *345*, 92-102, https://doi.org/10.1016/j.neucom.2019.01.087

Mitchell, T., (1997), *Machine Learning*, McGraw-Hill Professional.

Murphy, K. P., (2012), *Machine Learning : A Probabilistic Perspective*, MIT Press.

Murtagh, F. & Contreras, P., (2017), Algorithms for hierarchical clustering : an overview, II, *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, *7 6*, e1219.

Nesterov, Y., (1983), A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2), *Doklady an ussr*, *269*, 543-547.

Omran, M. G., Engelbrecht, A. P. & Salman, A., (2007), An overview of clustering methods, *Intelligent Data Analysis*, *11 6*, 583-605.

Platt, J., (1998), *Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines* (rapp. tech. MSR-TR-98-14), Microsoft, https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/

Polyak, B. T., (1964), Some methods of speeding up the convergence of iteration methods, *Ussr computational mathematics and mathematical physics*, *4 5*, 1-17.

Quinlan, J. R., (1986), Induction of decision trees, *Machine learning*, *1 1*, 81-106.

Quinlan, J. R., (1993), *C4.5 : Programs for Machine Learning*, Morgan Kaufmann.

Raileanu, L. E. & Stoffel, K., (2004), Theoretical comparison between the gini index and information gain criteria, *Annals of Mathematics and Artificial Intelligence*, *41 1*, 77-93, https://doi.org/10.1023/B:AMAI.0000018580.96245.c6

Rosenblatt, F., (1958), The Perceptron : A Probabilistic Model for Information Storage and Organization in The Brain, *Psychological Review*, 65-386.

Samuel, A. L., (1959), Some Studies in Machine Learning Using the Game of Checkers, *IBM Journal of Research and Development*, *3 3*, 210-229, https://doi.org/10.1147/rd.33.0210

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al., (2017), Mastering the game of go without human knowledge, *nature*, *550 7676*, 354-359.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R., (2014), Dropout : a simple way to prevent neural networks from overfitting, *The journal of machine learning research*, *15 1*, 1929-1958.

Sutton, R. S. & Barto, A. G., (2018), *Reinforcement Learning : An Introduction* (2e éd.), Bradford Books.

van Veen, F., (2016), The neural network zoo, https://www.asimovinstitute.org/neural-network-zoo/

Zhou & Chellappa, (1988), Computation of optical flow using a neural network, *IEEE 1988 International Conference on Neural Networks*, 71-78 vol.2, https://doi.org/10.1109/ICNN.1988.23914

Zhu, K. & Zhang, T., (2021), Deep reinforcement learning based mobile robot navigation : A review, *Tsinghua Science and Technology*, *26 5*, 674-691, https://doi.org/10.26599/TST.2021.9010012

Zhu, X. & Goldberg, A., (2009), *Introduction to semi-supervised learning*, Springer.

Deuxième partie

Propositions

Chapitre 4

Réseau de neurones : une approche viable

> « *On ne peut se passer d'une méthode pour se mettre en quête de la vérité des choses.* »
>
> René Descartes
> *Mathématicien, physicien et philosophe (1596 – 1650)*

Après une description de l'état de l'art dans la première partie de ce document, l'objet de ce chapitre est d'amener une première contribution à la problématique en répondant à la première hypothèse énoncée page 20 et reformulée ici :

Un réseau de neurones potentiellement exécutable sur un accélérateur matériel modeste peut atteindre d'aussi bonnes performances de détection d'intrusions réseau que les algorithmes classiques d'apprentissage automatique.

Nous avons vu dans la section 2.3.3 que plusieurs publications sur la détection d'intrusions utilisant les corpus numériques récents tendent à montrer que les réseaux de neurones conduisent à de moins bonnes performances de détection que les algorithmes de machine learning classiques. Notre hypothèse est donc en contradiction avec ces résultats. Deux choix s'offrent à nous : ou notre hypothèse est fausse, ou les résultats de publications existantes doivent être remis en cause. Comme le suggère Descartes, nous commençons ce chapitre par la description de la méthodologie que nous employons dans la section 4.1. Dans un deuxième temps, nous décrivons les expérimentations menées et les résultats obtenus en section 4.2, avec l'objectif de lever l'incertitude sur la possibilité de détecter convenablement les intrusions réseau avec un réseau de neurones. Enfin, pour clore ce chapitre, la section 4.3 propose une synthèse sur cette première contribution.

4.1 Méthodologie

La vérification de notre hypothèse nécessite de comparer les performances de différents algorithmes d'apprentissage automatique avec, d'une part, les algorithmes classiques, et d'autre part, au moins un réseau de neurones. Pour que les comparaisons aient un sens, nous définissons tout d'abord un cadre d'évaluation tel que décrit en Figure 4.1. La première étape, décrite en section 4.1.1, consiste à choisir deux corpus numériques qui seront identiques pour toutes les mesures de performances. Le premier sert à vérifier notre hypothèse tandis que le second permet de tester la généralisation de notre méthode.

Ensuite, les données contenues dans les corpus numériques choisis sont préparées pour servir à l'entraînement et au test de chaque algorithme selon la description donnée en section 4.1.2. La section 4.1.3 décrit les modèles testés. Enfin, la section 4.1.4 explique les différentes métriques qui vont servir aux comparaisons.

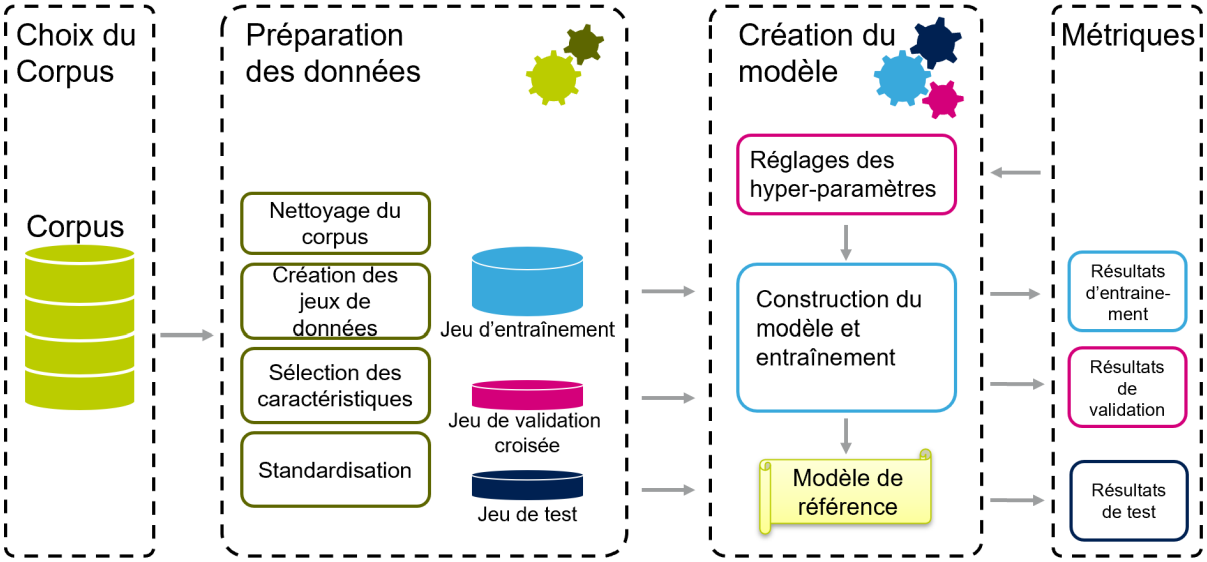


FIGURE 4.1 – Vue générale du cadre d'évaluation des algorithmes.

4.1.1 Choix des corpus numériques

La sélection d'un corpus numérique est une étape importante pour toutes les études basées sur le machine learning. Afin d'être en mesure d'arrêter un choix, nous définissons cinq critères de sélection :

- C1 : l'interface de communication doit correspondre à nos deux cas d'étude basés sur un protocole IP ;
- C2 : le corpus doit contenir des données en grande quantité ;
- C3 : les données doivent être représentatives d'un trafic réseau réel ;
- C4 : les enregistrements réseau doivent contenir des attaques récentes et variées ;
- C5 : le corpus doit contenir les données brutes.

Les critères C1 à C4 sont classiques dans les problèmes adressés par l'apprentissage automatique. Le critère C5 est important afin de pouvoir recalculer les caractéristiques des conversations ou d'en définir de nouvelles si nécessaire. Il permet également de rejouer les échanges sur un système embarqué similaire à nos deux domaines d'étude.

Reprenons quelques-uns des corpus mentionnés dans la section 2.3, dans l'ordre chronologique de leur création, pour les analyser selon nos cinq critères et dont la synthèse est donnée par la TABLE 4.1.

TABLE 4.1 – Comparaison des corpus numériques selon les critères définis.

Corpus	**C1**	**C2**	**C3**	**C4**	**C5**
KDD-Cup99	✓	✓	✗	✗	✗
NSL-KDD	✓	✗	✗	✗	✗
ISCX2012	✓	✓	✗	✗	✓
UNSW-NB15	✓	✓	✗	✓	✓
CIC-IDS2017	✓	✓	✓	✓	✓
RPL-NIDDS17	✗	✗	✗	✗	✓
CSE-CIC-IDS2018	✓	✓	✓	✓	✓
BIDS	✗	✗	✓	✗	✗
CIC-DDoS2019	✓	✓	✗	✗	✓
IoTID20	✓	✗	✓	✗	✗
IoT-23	✓	✓	✗	✗	✗

Le corpus KDD-Cup99 correspond bien à des communications IP, mais il est ancien. De plus, les données, en grande quantité, ne sont pas représentatives d'un trafic réseau actuel et des problèmes de qualités sont connus (McHUGH, 2000). Enfin, les données brutes ne sont pas disponibles.

Basé sur KDD-Cup99, le corpus NSL-KDD en partage la plupart des caractéristiques. Même si une partie des problèmes a été corrigée, la qualité n'est toujours pas au rendez-vous et la quantité de données est largement réduite à cause des redondances qui ont été supprimées (TAVALLAEE et al., 2009).

Bien qu'il respecte la plupart des critères, ISCX2012 contient un trafic ancien, sans trafic HTTPS (SHARAFALDIN et al., 2018). De plus, la distribution des attaques simulées n'est pas basée sur des statistiques réelles.

UNSW-NB15 est un corpus récent, avec des attaques récentes et variées. La quantité de données est suffisante et les données brutes sont disponibles. Le critère C3 n'est pas respecté puisque le trafic est simulé et non un enregistrement d'un trafic réel (MOUSTAFA & SLAY, 2015).

Quant à CIC-IDS2017, il respecte l'ensemble des critères.

RPL-NIDDS17 et BIDS ne correspondent pas au type de trafic que nous étudions dans les voitures connectées et le mur d'écrans utilisé dans un contexte d'éducation. D'autre part, la quantité de données, la variété des attaques ne sont pas suffisantes.

Comme CIC-IDS2017, le corpus CSE-CIC-IDS2018 respecte tous les critères et contient une plus grande quantité de données.

CIC-DDoS2019 (SHARAFALDIN et al., 2019) et IoTID20 (ULLAH & MAHMOUD, 2020) ne contiennent pas d'attaques variées mais se limitent à des attaques de type déni de service pour le premier et de type Bot pour le second. Enfin, pour IoT-23 (GARCIA et al., 2020), le manque de représentativité par rapport à nos domaines d'étude et de diversité

dans les attaques ainsi que l'absence de données brutes font que ce corpus échoue sur trois de nos critères.

Une approche complémentaire consiste à considérer le nombre de citations des corpus. Cette méthode a ses limites dans le sens où un corpus peut être cité pour mettre en avant ses défauts comme pour être utilisé dans l'étude des systèmes de détection d'intrusions réseau. La FIGURE 4.2 illustre le nombre de citations, par an, des publications correspondant aux corpus couramment utilisés en détection d'intrusions réseau, tel qu'indiqué par Google Scholar.

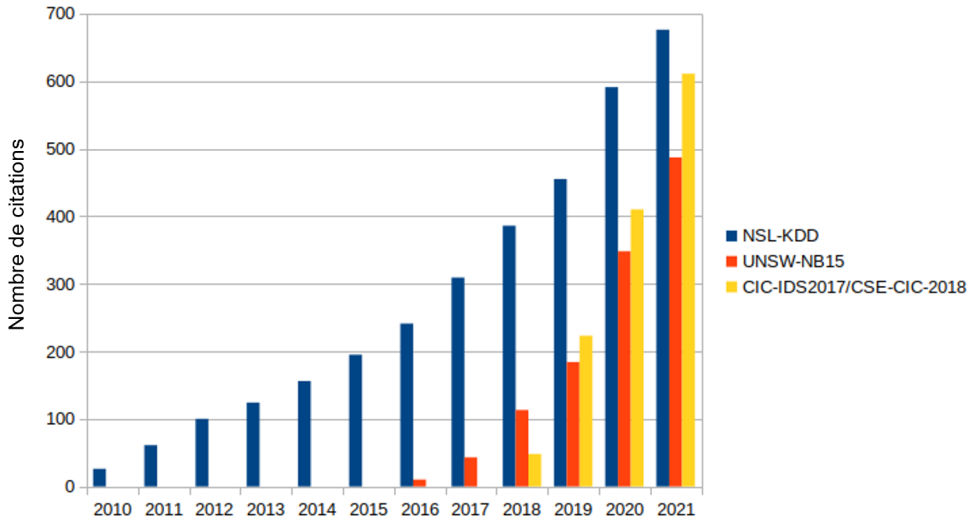


FIGURE 4.2 – Nombre de citations des corpus par an dans Google Scholar.

Tout d'abord, on remarque que NSL-KDD est cité à de nombreuses reprises. Ceci s'explique par deux raisons :

— beaucoup d'études ont été et sont encore menées sur ce corpus ;
— les études sur les corpus plus récents continuent de le citer pour indiquer ses limitations.

Les corpus CIC-IDS2017 et CSE-CIC-IDS2018 sont agrégés, car leurs auteurs demandent de citer la même publication. Ces corpus deviennent rapidement des références et UNSW-NB15, plus vieux de deux ans, est moins souvent cité.

En conclusion, la TABLE 4.1 montre que trois corpus sortent du lot : UNSW-NB15, CIC-IDS2017 et CSE-CIC-IDS2018. Sachant que UNSW-NB15 est le seul d'entre eux à ne pas respecter le critère C3 - le trafic est simulé - et considérant un nombre de citations plus important pour les corpus du CIC, nous arrêtons notre choix sur l'utilisation de CIC-IDS2017 pour vérifier notre hypothèse et de CSE-CIC-IDS2018 pour vérifier la possibilité de généraliser nos conclusions.

4.1.2 Préparation des données

4.1.2.1 Nettoyage du corpus numérique

Une série d'opérations a été menée pour détecter la présence de lignes vides, de caractéristiques redondantes et de valeurs non numériques dans des champs numériques.

Ce contrôle systématique a permis de supprimer plus de 280 000 cas où toutes les caractéristiques étaient vides sur un total de 2 830 743.

La longueur de l'en-tête des messages émis apparaissant deux fois, une instance a été supprimée.

La plupart des caractéristiques sont numériques, mais certains de ces champs peuvent contenir des chaînes de caractères comme "NaN" ou "Infini". Ces cas apparaissent dans 6 types d'attaques : BENIGN, FTP-PATATOR, DoS Hulk, Bot, PortScan et DDoS. Comme l'indique la Table 4.2, la quantité de ces cas est négligeable dans chaque type de trafic et ces instances ont simplement été supprimées.

Table 4.2 – Quantité de conversations dans CIC-IDS2017.

Traffic	**Instances d'origine**	**NaN/Infini**	**Après nettoyage**
BENIGN	2 273 097	1 777	2 271 320
Bot	1 966	10	1 956
DDoS	128 027	2	128 025
DoS GoldenEye	10 293	0	10 293
DoS Hulk	231 073	949	230 124
DoS Slowhttptest	5 499	0	5 499
DoS Slowloris	5 796	0	5 796
FTP-PATATOR	7 938	3	7 935
Heartbleed	11	0	11
Infiltration	36	0	36
PortScan	158 930	126	158 804
SSH-PATATOR	2 897	0	2 897
WebAttack BruteForce	1 507	0	1 507
WebAttack SQL Injection	21	0	21
WebAttack XSS	652	0	652

4.1.2.2 Création des jeux de données d'entraînement, de validation et de test

CIC-IDS2017 est un corpus très déséquilibré à double titre. D'une part, le trafic normal (BENIGN) représente environ 80% du trafic total, et d'autre part, certaines attaques comme Heartbleed, Infiltration, WebAttack SQL injection sont très largement sous-représentées comme on peut le noter dans la Table 4.2.

Sachant qu'un corpus déséquilibré est un défi pour l'apprentissage supervisé, nous avons préparé des jeux d'entraînement, de validation croisée et de test en prenant de manière aléatoire respectivement 50%, 25% et 25% de chaque type d'attaques, tout en

garantissant que chaque instance n'est utilisée qu'une seule fois. Ensuite, chaque jeu de données a été complété par des instances de trafic normal également choisies aléatoirement sans remise. Ce procédé permet d'équilibrer la quantité de trafic normal et d'attaques, mais reste déséquilibré en termes de type d'attaques.

Les jeux d'entraînement et de validation croisée servent uniquement à l'apprentissage et au réglage des hyper-paramètres. Les mesures de performances sont effectuées avec les données du jeu de test qui n'ont jamais été vues par les algorithmes de machine learning durant la phase d'apprentissage. La quantité de chaque type de trafic pour chaque jeu de données est décrite dans la Table 4.3.

Table 4.3 – Contenu des jeux d'entraînement, de validation croisée et de test de CIC-IDS2017.

Classes	**Jeu d'entraînement**	**Jeu de validation croisée**	**Jeu de test**
BENIGN	278 274	139 135	139 135
Bot	978	489	489
DDoS	64 012	32 006	32 006
DoS GoldenEye	5 146	2 573	2 573
DoS Hulk	115 062	57 531	57 531
DoS Slowhttptest	2 749	1 374	1 374
DoS Slowloris	2 898	1 449	1 449
FTP-PATATOR	3 967	1 983	1 983
Heartbleed	5	2	2
Infiltration	18	9	9
PortScan	79 402	39 701	39 701
SSH-PATATOR	2 948	1 474	1 474
WebAttack BruteForce	753	376	376
WebAttack SQL Injection	10	5	5
WebAttack XSS	326	163	163
Total	**556 548**	**278 270**	**278 270**

4.1.2.3 Sélection des caractéristiques

La sélection des caractéristiques parmi les quatre-vingt-quatre proposées est l'une des étapes fondamentales de l'apprentissage automatique influençant grandement les performances du modèle. Des caractéristiques non pertinentes peuvent avoir un impact négatif et entraîner une diminution de la précision des prédictions.

L'analyse du corpus numérique a révélé huit caractéristiques qui ne sont pas informatives. Leur valeur est constante quel que soit le type de trafic. Par conséquent, les caractéristiques "Bwd PSH Flags", "Bwd URG Flags", "Fwd Avg Bytes/Bulk", "Fwd Avg Packets/Bulk", "Fwd Avg Bulk Rate", "Bwd Avg Bytes/Bulk", "Bwd Avg Packets/Bulk", "Bwd Avg Bulk Rate" ont été supprimées.

Afin que le modèle n'apprenne pas quand les attaques se produisent, la caractéristique "Timestamp" n'est pas considérée comme informative pour notre étude. Chaque instance est caractérisée par son port et son adresse IP d'origine et de destination, son protocole et un identifiant de conversations contenant les mêmes informations.

Comme la caractéristique "FlowID" est redondante avec d'autres champs, elle a été retirée du jeu de données. Nous avons également décidé de retirer l'adresse IP source/destination et le port source, car ces caractéristiques ne sont pas pertinentes dans un système générique de détection des intrusions. Il en résulte alors un corpus numérique contenant soixante-dix caractéristiques.

4.1.2.4 Standardisation

Le pré-traitement des données est essentiel pour préparer l'ensemble des données en vue d'un apprentissage efficace. En particulier, un réseau neuronal apprend mieux lorsque toutes les caractéristiques sont mises à l'échelle dans la même plage de valeurs. Ceci est utile lorsque les entrées sont à des échelles très différentes comme c'est le cas avec le corpus CIC-IDS2017. Cette mise à échelle est également nécessaire pour certains algorithmes de machine learning qui procèdent à des calculs de distance. Dans notre implémentation, les données sont standardisées. Pour chaque caractéristique $\mathcal{F}_j$, chaque valeur x_i est transformée en valeur standardisée $x_{n,i}$ selon l'équation (4.1) où $\mu^{(\mathcal{F}_j)}$ et $\sigma^{(\mathcal{F}_j)}$ sont respectivement la moyenne et l'écart-type de la caractéristique $\mathcal{F}_j$.

$$x_{n,i}^{(\mathcal{F}_j)} = \frac{x_i^{(\mathcal{F}_j)} - \mu^{(\mathcal{F}_j)}}{\sigma^{(\mathcal{F}_j)}} \tag{4.1}$$

Le choix de la standardisation nous garantit que la distribution de chaque caractéristique aura une moyenne centrée en zéro et un écart-type égal à un. L'utilisation d'une mise à l'échelle dans l'intervalle $[0, 1]$ ou $[-1, 1]$ est inappropriée. En effet, la présence de valeurs extrêmes dans le corpus, dont on ne peut pas présupposer qu'elles soient aberrantes et donc à supprimer, compresse excessivement les plages utiles des caractéristiques et rend plus difficile l'apprentissage.

4.1.3 Création des modèles

La création des modèles comprend deux étapes majeures. La première consiste à choisir des modèles d'apprentissage automatique alors que la deuxième étape adresse l'entraînement des modèles retenus.

4.1.3.1 Choix des modèles

La problématique porte sur l'étude de la protection des objets connectés en utilisant des techniques d'apprentissage automatique supervisé. Elle limite les modèles qu'on peut utiliser aux algorithmes de classifications, en supprimant les algorithmes non-supervisés ou

semi-supervisés. La complexité de ces modèles doit être adaptée aux ressources disponibles dans les objets connectés.

Algorithmes classiques Afin d'obtenir des points de comparaison avec les publications existantes, nous utilisons un panel d'algorithmes régulièrement utilisés par les chercheurs du domaine et qui possèdent des modes de fonctionnement différents.

Deux algorithmes, intimement liés, ont été sélectionnés : un arbre de décision (DT) et une forêt aléatoire (RF). Différents types d'arbre de décision existent tels que ID3, C4.5, et CART. Parmi eux, l'algorithme choisi produit un arbre binaire de type CART en se basant sur le critère d'impureté de Gini. Il a été montré que l'impureté de Gini et le gain d'information sont des critères très couramment utilisés pour construire des arbres de décision. D'autre part, le choix de l'un ou l'autre ne conduit pas à des différences significatives sur les performances de classification (Raileanu & Stoffel, 2004). Le classifieur RF est un ensemble contenant cent arbres binaires entraînés, eux aussi, de façon à minimiser le critère d'impureté de Gini.

Les algorithmes LDA et QDA sont capables de classifier des données en générant respectivement des séparations linéaires et quadratiques entre les classes. L'algorithme SVM, quant à lui, peut-être vu comme une variante des deux précédents avec la particularité de maximiser la distance séparant les classes. Le nombre de vecteurs de support impacte fortement la performance de l'algorithme SVM, à la fois durant la phase d'apprentissage et la phase d'inférence. Moins le nombre de vecteurs de support est grand, plus l'algorithme est rapide (Bottou et al., 2004). D'autre part, ce nombre augmente linéairement avec la taille du jeu d'entraînement. En considérant que notre jeu de données d'entraînement contient plus de 550 000 instances, nous pouvons nous attendre à une lenteur de l'algorithme. Si à cela, nous ajoutons une complexité accrue par l'utilisation d'un noyau non-linéaire, tel qu'un noyau polynomial ou de base radiale, le modèle risque d'être très lent et inapproprié pour un objet connecté.

Aussi, nous décidons d'utiliser un SVM à noyau linéaire. Nous pouvons voir ces algorithmes comme un ensemble de méthodes de même type amenant chacun des particularités et permettant une analyse plus fine des performances mesurées.

Enfin, nous complétons notre panel par un algorithme totalement différent utilisant la méthode des k plus proches voisins (k-NN), basée sur le principe bien connu de Cicéron *pares cum paribus facillime congregantur* - qui se ressemble, s'assemble. En effet, la classe affectée à une nouvelle instance est la classe majoritaire de ses k plus proches voisins.

Pour spécifier l'algorithme, il est nécessaire figer la valeur de k ainsi que la fonction de distance utilisée. Pour notre étude, nous avons fixé $k = 5$. Parmi celles régulièrement utilisées, quatre fonctions de distance d_1, d_2, d_3 et d_4 respectivement les distances Manhatan, euclidienne, Minkowski et Mahalanobis entre deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ de dimension n, donnée par les équations 4.2 à 4.5 sont considérées. La distance Mahalanobis fait intervenir un vecteur $\boldsymbol{\mu}$, de dimension n, des valeurs moyennes des caractéristiques et une matrice de covariance $\boldsymbol{\Sigma}$.

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.2}$$

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.3}$$

$$d_3(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{4.4}$$

$$d_4(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \tag{4.5}$$

On note que la distance Minkowski (d_3) est une généralisation des distances Manahattan (d_1) avec $p = 1$ et euclidienne (d_2) avec $p = 2$. La distance Mahalanobis (d_4) prend en compte d'éventuelles corrélations entre les composantes des vecteurs. Dans le cas où la matrice de covariance est une matrice diagonale unitaire, c'est-à-dire lorsque les composantes sont centrées sur zéro, indépendantes et d'écart-type unitaire, la distance Mahalanobis devient une distance euclidienne. Elle présente toutefois un inconvénient puisque la matrice de covariance doit être inversible. Ceci implique que les caractéristiques utilisées dans le corpus numérique ne soient pas des combinaisons linéaires d'autres caractéristiques. Il est donc peu probable que cette contrainte soit respectée, car nous pouvons supposer que des statistiques sur les conversations soient liées. Intuitivement, on peut imaginer que le nombre total de paquets Ethernet soit corrélé au nombre total d'octets échangés.

Pour cette raison, nous ne retenons pas cette fonction de distance. Dans le but de limiter la complexité des calculs, le compromis est d'utiliser la distance euclidienne.

Réseaux de neurones Un grand nombre de familles de réseaux de neurones existe, certains servant à la classification, d'autre servant à la génération de données ou encore à la détection d'anomalies. Pour la classification, nous considèrons les trois grandes familles formées par les perceptrons multi-couches (MLP), les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN).

Les CNN sont généralement utilisés pour classer des images. L'architecture typique de ces réseaux contient des couches convolutives réalisant une extraction des caractéristiques, suivies par un MLP. Pour la détection d'intrusions, le vecteur de caractéristiques peut être transformé en une matrice traitée comme une image (Lin et al., 2018 ; Vinayakumar et al., 2017). L'inconvénient majeur de ce type d'approche est le coût en termes de calculs. En effet, un CNN est beaucoup plus complexe qu'un simple MLP.

Les RNN, quant à eux, sont typiquement utilisés sur les séquences et conviennent particulièrement bien au traitement automatique du langage ou encore pour traiter les séries de données temporelles. Dans notre cas, les conversations de notre corpus prennent déjà en compte un aspect temporel en regroupant les trames Ethernet correspondant à une séquence entre un émetteur et son destinataire. D'autre part, ce type de réseau

semble moins performant qu'un CNN sur des tâches de détection d'intrusions à partir de conversations (Kim et al., 2019).

Les architectures CNN et RNN étant écartées pour les raisons mentionnées précédemment, nous choisissons d'étudier les performances d'un perceptron multi-couches, un réseau de neurones à propagation avant pouvant être utilisé comme classifieur (Goodfellow et al., 2016). Le modèle proposé prend en entrée les valeurs standardisées des caractéristiques sélectionnées. Il contient deux couches cachées de deux cent cinquante-six nœuds. Le choix du nombre de couches et de nœuds par couche résulte d'une analyse de performance et d'un compromis minimisant le total des erreurs dues au biais et à la variance. En effet, un modèle trop simple n'arrive pas à capturer la relation entre les données d'entrée et de sortie (erreurs importantes à la fois sur les données d'entraînement et de validation croisée) alors qu'un modèle trop complexe engendre un sur-ajustement (erreurs faibles sur les données d'entraînement et importantes sur celles de validation croisée). Bien qu'un modèle à une seule couche cachée soit un approximateur universel (Hornik, 1991), il est généralement plus efficace d'augmenter le nombre de couches plutôt que d'augmenter le nombre de nœuds d'une couche unique.

Compte tenu du temps d'entraînement nécessaire au bon dimensionnement du réseau et à l'analyse du compromis biais-variance, l'espace de recherche a été contraint à des nombres de nœuds multiples de seize. Ce choix est lié au type de processeurs utilisés dans les IoT. En effet, ces plateformes sont souvent basées sur des cœurs ARM pour lesquels une ligne de cache du processeur en charge des inférences peut contenir seize valeurs flottantes de 32 bits.

L'architecture du modèle sélectionné est décrite par la Figure 4.3.

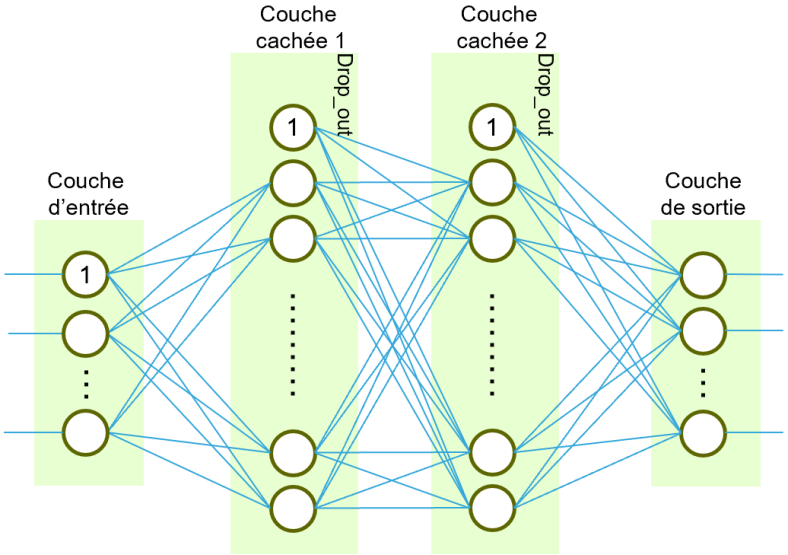


Figure 4.3 – Architecture du modèle MLP sélectionné.

Le classifieur possède quinze sorties pour les quatorze types d'attaques et le trafic

bénin. La technique du *dropout* est utilisée sur les couches cachées pour éviter le sur-ajustement aux données d'apprentissage en éteignant des nœuds aléatoirement avec une probabilité *droprate*.

La sortie du réseau de neurones $\boldsymbol{h}^{(3)}$ est calculée en chaînant les sorties des différentes couches comme décrit dans les équations (4.6), (4.7) et (4.8) dans lesquelles $\boldsymbol{x}$, $\boldsymbol{W}^{(i)}$ et $\boldsymbol{b}^{(i)}$ sont respectivement le vecteur d'entrée contenant les caractéristiques sélectionnées, la matrice de poids et le vecteur de biais pour la couche i.

Compte tenu de la topologie du réseau, les variables sont définies de la façon suivante : $\boldsymbol{x} \in \mathbb{R}^{70}$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{70\times256}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{256}$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{256\times256}$, $\boldsymbol{b}^{(2)} \in \mathbb{R}^{256}$, $\boldsymbol{W}^{(3)} \in \mathbb{R}^{256\times15}$ et $\boldsymbol{b}^{(3)} \in \mathbb{R}^{15}$.

$$\boldsymbol{h}^{(1)} = g_1\left(\boldsymbol{W}^{(1)T}.\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) \tag{4.6}$$

$$\boldsymbol{h}^{(2)} = g_1\left(\boldsymbol{W}^{(2)T}.\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right) \tag{4.7}$$

$$\boldsymbol{h}^{(3)} = g_2\left(\boldsymbol{W}^{(3)T}.\boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)}\right) \tag{4.8}$$

Les fonctions d'activation g_1 et g_2 apportent de la non-linéarité dans le réseau. Comme le MLP ne fournit pas une normalisation intrinsèque de ses sorties, la fonction d'activation SELU g_1, dont la formule est donnée en (4.9), est utilisée pour les couches cachées avec les valeurs $\lambda = 1.0507$ et $\alpha = 1.6733$ comme définies dans (Klambauer et al., 2017). Les valeurs de λ et α permettent d'obtenir à la sortie de chaque couche une distribution centrée en zéro avec un écart-type unitaire. Quant à la couche de sortie, la fonction softmax g_2, définie en (4.10), est utilisée afin que chaque sortie puisse être interprétée comme la probabilité de prédire une classe donnée. Le label prédit $\hat{y}$ est obtenu par $\hat{y} = argmax\ \boldsymbol{h}^{(3)}$.

$$g_1(z) = \lambda.\begin{cases}\alpha.(e^z - 1) & \text{pour } z < 0\\ z & \text{pour } z \geq 0\end{cases} \tag{4.9}$$

$$g_2\left(z\right)_j = \frac{e^{z_j}}{\sum_{k=1}^{15} e^{z_k}} \text{ pour } j \in [1;15] \tag{4.10}$$

4.1.3.2 Entraînement des modèles

Tous les modèles sont entraînés sur un PC utilisant un processeur Intel Core i7-8750H à 2.2 GHz, avec un logiciel développé en Python.

Algorithmes classiques Notre objectif étant de montrer qu'un réseau de neurones peut donner des performances de classification aussi bonnes que les algorithmes classiques, nous utilisons des implémentations de référence en utilisant le paquet scikit-learn (Pedregosa et al., 2011). De cette manière, nous avons la certitude de ne pas introduire de dysfonctionnements qui pourraient laisser croire, à tort, que notre réseau de neurones sur-performe les algorithmes classiques.

Réseau de neurones Le MLP a été implémenté en Python en utilisant l'infrastructure logicielle TensorFlow (TF) (Geron, 2019) pour la bibliothèque d'apprentissage profond. Les données d'entraînement ont été divisées en mini-lots de 32 instances. La taille du ceux-ci est choisie en fonction de la taille du cache de données de la machine servant à entraîner le réseau. Trente-deux instances correspondent à la plus grande puissance de deux permettant de contenir le mini-lot, les poids et les sorties de la première couche du réseau dans le cache de niveau 1 avec suffisamment de marge (40%) pour limiter les risques d'éviction de nos données pendant les traitements. Le MLP apprend à classifier en ajustant les poids entre les nœuds du réseau afin de réduire une fonction de coût $\mathcal{L}(w)$. Celle-ci est basée sur l'entropie croisée définie par l'équation (4.11) dans laquelle y est le label correct et $\hat{y}$ la classe prédite.

Plusieurs algorithmes permettent l'optimisation de la fonction de coût via une descente du gradient et une comparaison de différentes alternatives montre qu'un algorithme appelé Adam est généralement plus performant (Kingma & Ba, 2015). La fonction de coût $\mathcal{L}(\boldsymbol{w})$ est optimisée avec cet algorithme dont les trois paramètres α, β_1 et β_2 permettent une configuration fine.

$$\mathcal{L}(\boldsymbol{w}) = -\left[y.\log\left(\hat{y}\right) + (1-y).\log\left(1-\hat{y}\right)\right] \tag{4.11}$$

Le niveau de sur-ajustement du réseau est contrôlé en évaluant les performances de prédiction sur le jeu de validation croisée. Le paramètre de *dropout*, appelé *droprate*, est affiné pour garder la différence de performance entre les jeux d'entraînement et de validation croisée la plus faible possible. Le réglage des hyper-paramètres est la tâche délicate qui consiste à trouver le bon ensemble de valeurs pour obtenir les meilleures performances. Une stratégie de recherche aléatoire (Bergstra & Bengio, 2012) a été appliquée afin de régler *droprate*, α et β_1 tout en gardant la valeur par défaut $\beta_2 = 0.999$.

4.1.4 Métriques d'évaluation

Plusieurs informations clefs peuvent être extraites en comparant les prédictions aux labels fournis dans le corpus. Le nombre de vrais positifs (TP) est le nombre d'attaques correctement prédites comme étant des attaques. Le nombre de vrais négatifs (TN) est le nombre d'instances normales classées comme trafic normal alors que les faux positifs (FP) et faux négatifs (FN) sont respectivement le nombre d'instances normales classées comme attaques et le nombre d'attaques prédites comme trafic normal. Ces quatre métriques forment la matrice de confusion d'un classifieur binaire et peuvent être utilisées pour dériver des métriques supplémentaires.

Comme les travaux publiés sur le corpus CIC-IDS2017 utilisent un sous-ensemble de métriques, nous proposons de baser notre évaluation sur un ensemble plus complet permettant une comparaison avec les études existantes et futures sur ce corpus.

Le taux de vrai positif (TNR), donné par l'équation 4.12, est le pourcentage de trafic classé comme bénin par rapport au nombre effectif d'instances de trafic bénin.

$$TNR = \frac{TN}{TN + FP} \tag{4.12}$$

Le taux de faux positif (FPR) dans l'équation (4.13) est le pourcentage de trafic bénin classé comme attaques.

$$FPR = \frac{FP}{TN + FP} \tag{4.13}$$

La précision (Prec) et le rappel (Rec), donnés par les équations 4.14 et 4.15, représentent respectivement le pourcentage d'attaques correctement détectées par rapport au nombre d'instances prédites comme attaques et le pourcentage d'attaques correctement détectées par rapport au nombre effectif d'attaques.

$$précision = \frac{TP}{TP + FP} \tag{4.14}$$

$$rappel = \frac{TP}{TP + FN} \tag{4.15}$$

Le score f_1 est la moyenne harmonique de la précision et du rappel.

$$score f_1 = \frac{2 \times précision \times rappel}{précision + rappel} \tag{4.16}$$

La justesse (Acc) mesure la proportion du nombre total de classifications correctes, telle que donnée par l'équation 4.17.

$$justesse = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.17}$$

Le coefficient de corrélation de Matthews (MCC) a été introduit par B.W. Matthews (Matthews, 1975) dans un contexte de recherche biomédicale. C'est une métrique intéressante prenant en compte tous les éléments de la matrice de confusion de manière correcte dans le cas de corpus déséquilibrés, contrairement à la justesse qui peut donner des valeurs élevées même lorsque la totalité de la classe minoritaire est mal prédite. Puisque les corpus de détection d'intrusions sont intrinsèquement déséquilibrés, c'est une métrique judicieuse pour notre application. Le MCC est également connu pour être une métrique robuste quand les cas positifs et négatifs sont d'égale importance (Chicco et al., 2021).

Ce coefficient de corrélation est calculé avec l'équation 4.18.

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{4.18}$$

MCC prend une valeur comprise entre -1 et $+1$. Une prédiction parfaite correspond à $MCC = 1$. *A contrario*, $MCC = -1$ dénote un désaccord total entre les prédictions et les classes réelles. Une prédiction aléatoire correspond à $MCC = 0$.

4.2 Expérimentations et résultats

En suivant l'ensemble des étapes décrites dans la section 4.1, nous sommes en mesure de tester les différents modèles et de comparer leurs performances. Dans un premier temps, nous évaluons les résultats en utilisant le corpus CIC-IDS2017 en section 4.2.1. Ensuite, nous vérifions la généricité de la méthode avec un autre corpus en section 4.2.2.

4.2.1 Évaluation avec CIC-IDS2017

Après l'entraînement d'un modèle de référence pour le MLP, l'ensemble de test est utilisé pour mesurer les performances du MLP en tant que classifieur à quinze classes. La matrice de confusion résultante est simplifiée dans la Table 4.4 pour montrer la classe estimée dans trois colonnes : bénin, attaque correcte et autres attaques.

Dans une classification multi-classes, il est possible qu'une attaque soit mal prédite, tout en restant classée comme une autre sorte d'attaques. Cela pose un problème si la classe correcte est requise, en particulier si la réponse à l'attaque dépend précisément de sa catégorie. Dans notre cas, nous considérons acceptable une telle erreur de classification, car elle est toujours considérée comme une attaque. Plutôt que de concevoir une classification binaire, nous gardons l'ensemble des types d'attaques de façon à pouvoir analyser celles qui pourraient être mal détectées. Par conséquent, les prédictions de tous les types d'attaques ont été fusionnées en une seule classe pour obtenir un classifieur binaire et calculer les métriques dans la Table 4.5.

La Table 4.4 montre que le MLP est performant dans presque toutes les classes, à l'exception des classes *Bot*, *Infiltration* et *WebAttack*. Deux raisons expliquent les mauvais résultats sur ces attaques. Tout d'abord, elles contiennent toutes un nombre extrêmement faible d'instances. Le réseau neuronal peut ne pas être en mesure d'apprendre à partir des quelques instances disponibles dans le jeu de données d'apprentissage. Il est bien connu que la quantité de données est un point clef pour le succès de l'apprentissage automatique. Deuxièmement, certaines attaques peuvent avoir des caractéristiques de conversations très similaires au trafic normal. Ce point spécifique est discuté dans la section 4.2.3.

La Table 4.5 démontre que notre MLP est correctement entraîné et ne souffre ni de sous-apprentissage, ni de sur-apprentissage. Le modèle généralise parfaitement aux données qui n'ont pas été vues durant l'entraînement, le *dropout* est bien réglé et joue parfaitement son rôle.

4.2.1.1 Évaluation des algorithmes classiques de machine learning

Les mêmes jeux de données d'entraînement, de validation croisée et de test ont été utilisés pour divers algorithmes classiques de ML disponibles dans la bibliothèque python scikit-learn.

La Table 4.6 montre les résultats de performances de l'ensemble des algorithmes.

TABLE 4.4 – Matrice de confusion simplifiée pour le MLP sur CIC-IDS2017.

		Prédictions		
		Bénins	***Attaques correctes***	***Autres attaques***
Classes réelles	BENIGN	138,460	-	675
	Bot	184	296	9
	DDoS	38	31 964	4
	DoS GoldenEye	43	2 528	2
	DoS Hulk	106	57 425	0
	DoS Slowhttptest	16	1 342	16
	DoS Slowloris	7	1 431	11
	FTP-PATATOR	6	1 968	9
	Heartbleed	0	2	0
	Infiltration	9	0	0
	PortScan	2	39 675	24
	SSH-PATATOR	16	1 450	8
	WebAttack BruteForce	258	102	16
	WebAttack SQL Injection	2	0	3
	WebAttack XSS	131	4	32

TABLE 4.5 – Performances du MLP sur CIC-IDS2017.

Métriques	**Données d'entraînement**	**Données de test**
TNR (%)	99.52	99.51
FPR (%)	0.48	0.49
Rappel (%)	99.40	99.41
Précision (%)	99.52	99.51
Justesse (%)	99.46	99.46
Score f_1 (%)	99.46	99.46
MCC	0.9892	0.9893

TABLE 4.6 – Comparaison du MLP avec les algorithmes classiques sur CIC-IDS2017.

Algorithme	**Acc (%)**	**Prec (%)**	**Rec (%)**	**FPR (%)**	**MCC**
Notre MLP	99.46	99.51	99.41	0.49	0.989
DT	99.87	99.87	99.89	0.13	0.998
RF	99.87	99.86	99.88	0.14	0.997
k-NN	99.31	98.91	99.72	1.10	0.986
QDA	97.67	95.99	99.49	0.42	0.954
SVM	96.85	95.73	98.06	4.37	0.937
LDA	87.43	80.84	98.10	23.24	0.766

Le plus mauvais classifieur, LDA, présente un taux de faux positifs très élevé. Les résultats inférieurs de cet algorithme peuvent s'expliquer par deux raisons. Premièrement, LDA utilise une combinaison linéaire de caractéristiques pour séparer les classes. L'ensemble de données peut ne pas être complètement séparable linéairement. Deuxièmement, LDA se base sur une hypothèse d'homoscédasticité. Cette hypothèse conduit à une matrice de covariance commune à toutes les classes dans un ensemble de données, différence majeure par rapport à QDA pour lequel chaque classe peut avoir sa propre matrice de covariance. La différence de performance entre QDA et LDA montre que l'hypothèse d'une matrice de covariance commune n'est pas remplie dans notre ensemble de données. SVM et QDA affichent des performances légèrement supérieures à LDA et la précision de l'algorithme SVM se situe entre LDA et QDA. Comme nous avons utilisé un noyau linéaire, la non-linéarité de la séparation des classes est confirmée.

Le k-NN obtient de bons résultats, mais présente toutefois un taux de faux positifs légèrement plus élevé.

Les meilleurs scores sur l'ensemble des métriques sont obtenus par les algorithmes d'arbre de décision et de forêt aléatoire. Ces algorithmes se comportent comme des systèmes de détection d'intrusions basés sur des règles, connus pour produire de bons résultats. Deux de leurs inconvénients sont la tendance à sur-ajuster les données d'entraînement et la difficulté à généraliser, problèmes non détectés dans nos expérimentations.

Notre proposition de MLP produit des résultats très proches des meilleurs scores.

La FIGURE 4.4 détaille les performances des différents classifieurs pour chaque classe. Ce graphique témoigne de la difficulté de détection des attaques de type *Bot*, *Infiltration* et *WebAttack*.

4.2.1.2 Comparaison avec les résultats antérieurs

La TABLE 4.7 compare les résultats avec les articles de référence. Différentes métriques étant utilisées, toutes les cellules du tableau ne peuvent pas être remplies. Outre les solutions de réseaux de neurones, les algorithmes traditionnels d'apprentissage automatique sont également pris en compte dans cette comparaison.

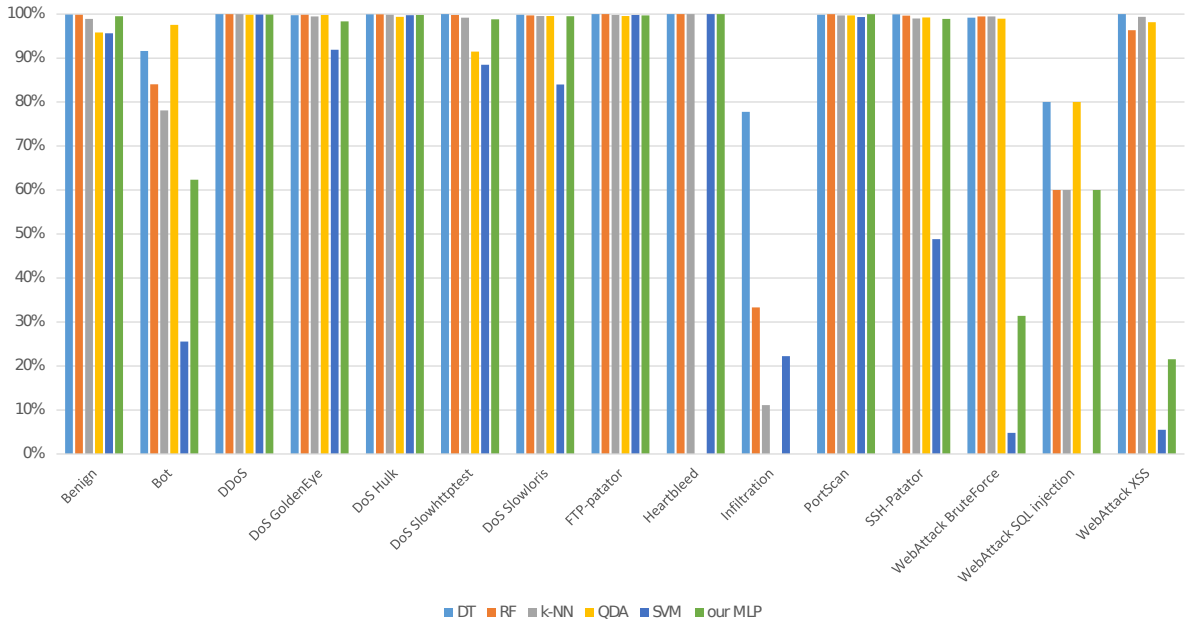


FIGURE 4.4 – Vue détaillée de la justesse (en %) des différents classifieurs.

TABLE 4.7 – Comparaison des performances du MLP avec les résultats antérieurs sur CIC-IDS2017.

Publication	**Algorithme**	**Acc (%)**	**Prec (%)**	**Rec (%)**	**FPR (%)**
Notre MLP	MLP	99.46	99.51	99.41	0.49
SHARAFALDIN et al., 2018	MLP	-	77	83	-
	QDA	-	97	88	-
	k-NN	-	96	96	-
JIANG et al., 2018	MLP	99.23	99.87	99.60	0.77
ULLAH et MAHMOUD, 2019	DT+RF	-	100	100	-
USTEBAY et al., 2018	MLP	91	-	-	-
GAMAGE et SAMARABANDU, 2020	MLP	99.58	99.56	99.52	-
	AE	98.55	98.53	98.55	-
	DBN	99.56	99.54	99.50	-
	LSTM	99.57	99.55	99.53	-
	RF	99.86	99.86	99.84	-

Notre modèle surpasse tous les résultats rapportés par les créateurs de CIC-IDS2017 (Sharafaldin et al., 2018), tant pour les réseaux de neurones que pour les techniques traditionnelles d'apprentissage automatique.

D'autres études obtiennent des performances similaires à notre solution (Jiang et al., 2018). Il convient toutefois de noter que le type d'attaques était limité aux DoS de la couche application (slowloris, slowhttptest, hulk, GoldenEye). Comme le montre la Table 4.4, ces attaques ne sont pas les plus difficiles à détecter. Nous pourrions nous attendre à une diminution de leur performance globale si toutes les autres attaques étaient prises en compte, notamment celles difficiles à détecter.

Une approche en deux étapes utilisant des arbres de décision proposée par Ullah et Mahmoud rapporte des métriques ayant une moyenne de 100%. Un examen plus approfondi des détails révèle que quatre types d'attaques ne sont pas parfaitement détectés par les arbres de décision. Ce sont exactement les mêmes classes avec lesquelles notre classifieur a quelques difficultés. Le manque de chiffres significatifs dans Ullah et Mahmoud ne permet pas une analyse fine, mais la comparaison globale présente que notre solution basée sur les MLP atteint le même niveau de performance que les algorithmes d'apprentissage automatique traditionnels.

Ustebay et al. ont également utilisé un MLP mais n'ont pas obtenu de hautes performances. Cependant, leurs résultats ne peuvent pas être directement comparés à ceux de Jiang et al. et Ullah et Mahmoud car ces derniers utilisent les adresses IP des machines menant les attaques. Dans un scénario réel, les adresses des pirates ne sont pas connues et ne peuvent pas être utilisées pour la détection des intrusions.

Plus récemment, une étude (Gamage & Samarabandu, 2020) sur quatre réseaux neuronaux et une forêt aléatoire fournit d'excellents résultats pour le corpus numérique CIC-IDS2017. Cependant, les auteurs ont conservé la proportion fortement déséquilibrée de l'ensemble de données original qui pourrait conduire à un biais en faveur de la classe majoritaire. De plus, ils ont supprimé trois classes d'attaques : infiltration, injection SQL et heartbleed. Même si leurs résultats sont très proches des nôtres, une comparaison équitable est difficile.

4.2.2 Validation avec CSE-CIC-IDS2018

Bien que l'approche que nous avons définie dans la section 4.1 soit fructueuse, il est intéressant de vérifier la généricité de notre méthode en l'appliquant à un corpus différent. Même si CIC-IDS2017 et CSE-CIC-IDS2018 présentent des similitudes, l'infrastructure du réseau de ce dernier, décrite dans la sous-section 2.3.3.3, est plus complexe et le *no free lunch theorem* (Wolpert, 1996) peut faire échouer notre solution MLP sur ce nouveau jeu de données. Dans un premier temps, nous exposons les différences entre les deux ensembles de données avant de couvrir la manière dont nous avons validé notre méthodologie. Enfin, pour finir cette section, les résultats obtenus sont présentés.

4.2.2.1 Différences entre CSE-CIC-IDS2018 et CIC-IDS2017

Les TABLES 4.8 et 4.9 décrivent le contenu des deux corpus numériques.

TABLE 4.8 – Contenu de CIC-IDS2017.

Labels	**Instances**	**Outils utilisés**
BENIGN	2 273 097	-
Bot	1 966	Ares
DDoS	128 027	HOIC, LOIC
DoS GoldenEye	10 293	GoldenEye
DoS Hulk	231 073	Hulk
DoS Slowhttptest	5 499	SlowHttpTest
DoS Slowloris	5 796	Slowloris
FTP-PATATOR	7 938	Patator (brute force)
Heartbleed	11	Heartleech
Infiltration	36	Metasploit
PortScan	158 930	nmap
SSH-PATATOR	5 897	Patator (brute force)
WebAttack BruteForce	1 507	Selenium
WebAttack SQL Injection	21	Selenium
WebAttack XSS	652	Selenium
Total	2 830 743	

Ceux-ci comprennent quatorze types d'attaques en plus du trafic normal et ont été étiquetés manuellement. Bien que la plupart des types d'attaques soient similaires dans les deux ensembles de données, ils sont classés différemment et ne peuvent pas être directement comparés. Par exemple, le scan de port est considéré comme une attaque spécifique dans CIC-IDS2017 alors qu'il est fusionné avec l'infiltration dans CSE-CIC-IDS2018. Même si les deux ensembles de données utilisent exactement les mêmes outils pour les DDoS, il existe une seule étiquette en 2017 tandis qu'elle est séparée en quatre étiquettes différentes dans l'ensemble de données 2018. Alors que le site Web du CIC (CANADIAN INSTITUTE FOR CYBERSECURITY, 2018) mentionne explicitement Heartleech comme outil d'attaque Heartbleed, il n'y a pas d'étiquette correspondante dans le corpus numérique CSE-CIC-IDS2018. Cela signifie qu'il peut contenir des erreurs d'étiquetage.

4.2.2.2 Méthode de validation

Préparation des données Les caractéristiques extraites des fichiers PCAP et fournies en CSV pour CSE-CIC-IDS2018 sont les mêmes que celles de CIC-IDS2017, à l'exception de l'identifiant de conversation, des adresses IP et ports source et destination qui ne sont pas disponibles. Ne les ayant pas utilisées avec CIC-IDS2017, leur absence ne pose aucun problème. La procédure de nettoyage appliquée est exactement celle décrite dans la

Table 4.9 – Contenu de CSE-CIC-IDS2018.

Labels	**Instances**	**Outils utilisés**
Benign	13 484 708	-
Bot	286 191	Ares, Zeus
BruteForce-Web	611	Selenium
BruteForce-XSS	230	Selenium
DDoS attack-HOIC	686 012	HOIC
DDoS attack-LOIC-UDP	1 730	LOIC
DDoS attack-LOIC-HTTP	576 191	LOIC
DoS attacks-GoldenEye	41 508	GoldenEye
DoS attacks-Hulk	461 912	Hulk
DoS attacks-Slowhttptest	139 890	SlowHttpTest
DoS attacks-Slowloris	10 990	Slowloris
FTP-BruteForce	193 360	Patator
Infiltration	161 934	Metasploit + nmap
SQL Injection	87	Selenium
SSH-BruteForce	187 589	Patator
Total	16 232 943	

section 4.1.2. Les enregistrements de trois jours présentent quelques instances contenant une étiquette appelée 'Label'. Ne connaissant pas le type de trafic correspondant à ces instances, elles ont été supprimées de l'ensemble de données. Comme pour CIC-IDS2017, 'NaN' et 'Infini' apparaissent dans certains champs pour le trafic bénin (35 006 instances), l'infiltration (1 295 instances) et les attaques FTP par force brute (6 instances). Ces données ont été supprimées.

CSE-CIC-IDS2018 contient davantage d'instances que le corpus de 2017 pour la plupart des attaques et nous avons pu définir un jeu de données d'entraînement plus équilibré que pour CIC-IDS2017. Un pourcentage de chaque classe a été sélectionné aléatoirement sans remplacement, de sorte que les jeux d'entrainement, validation croisée et test contiennent 50 % de trafic normal et 50 % d'attaques. Neuf classes contiennent le même nombre d'instances. La composition exacte des jeux de données d'entraînement, de validation croisée et de test est fournie dans le tableau 4.10.

Nous remarquons que les attaques de type *Bot* et *Infiltration* qui étaient peu représentées dans le corpus de 2017 sont plus répandues dans celui de 2018. Les attaques de serveurs d'applications (force brute, injection SQL) sont encore, quant à elles, en quantité limitée.

Modèle et entraînement Pour les deux corpus numériques de 2017 et 2018, nous utilisons le même ensemble de caractéristiques et le même nombre de classes. En conséquence, nous pouvons réutiliser les modèles définis dans la section 4.1.3 et les entraîner en suivant

TABLE 4.10 – Contenu des jeux d'entraînement, de validation croisée et de test de CSE-CIC-IDS2018.

Trafic	**Jeu d'entraînement**	**Jeu de validation croisée**	**Jeu de test**
Benign	187 926	46 981	46 981
Bot	19 999	4 999	4 999
BruteForce-Web	306	76	76
BruteForce-XSS	115	28	28
DDoS attack-HOIC	19 997	4 999	57 531
DDoS attack-LOIC-UDP	865	216	1 374
DDoS attack-LOIC-HTTP	20 000	4 999	1 449
DoS GoldenEye	20 000	5 000	1 983
DoS Hulk	20 001	5 000	2
DoS Slowhttptest	20 000	5 000	9
DoS Slowloris	6 594	1 648	39 701
FTP-BruteForce	20 001	5 000	1 474
Infiltration	20 000	4 999	376
SQL Injection	44	10	5
SSH-BruteForce	20 001	5 000	163
Total	**375 847**	**93 955**	**93 955**

la même procédure.

4.2.2.3 Résultats avec CSE-CIC-IDS2018

Comme pour le corpus de 2017, nous présentons d'abord les résultats obtenus avec notre modèle, puis nous les comparons aux algorithmes ML traditionnels et aux résultats publiés par d'autres chercheurs.

Évaluation des performances du MLP Les performances de notre MLP sont évaluées avec le jeu de données de test. De manière similaire au travail sur CIC-IDS2017, les inférences prédisent une des quinze classes. La matrice de confusion simplifiée est présentée dans le tableau 4.11.

Les prédictions de tous les types d'attaques ont été fusionnées en une seule classe pour obtenir un classifieur binaire. Le tableau 4.12 présente les métriques d'évaluation de notre modèle.

À première vue, notre modèle semble fournir de moins bons résultats sur CSE-CIC-IDS2018 avec une justesse de 95,44 %, soit 4 % de moins que la justesse de CIC-IDS2017. Nous pouvons observer dans le tableau 4.11 que ce résultat est dû aux très mauvaises prédictions des infiltrations. Nous rappelons ici que le jeu de données de 2017 ne contenait que neuf instances d'infiltrations, ce qui était clairement insuffisant. Lorsque les infiltrations ne sont pas prises en compte, notre MLP montre une précision supérieure à 99 %. Ceci est cohérent avec les résultats de CIC-IDS2017. De l'expérience avec CSE-CIC-IDS2018, nous pouvons tirer comme conclusion que la mauvaise détection des attaques d'infiltration

TABLE 4.11 – Matrice de confusion simplifiée pour CSE-CIC-IDS2018.

		Prédictions		
		Bénins	***Attaques correctes***	***Autres attaques***
Classes réelles	Benign	46 668	-	313
	Bot	5	4 994	0
	BruteForce-Web	40	36	0
	BruteForce-XSS	15	13	0
	DDoS attack-HOIC	0	4 999	0
	DDoS attack-LOIC-UDP	0	216	0
	DDoS attack-LOIC-HTTP	2	4 991	7
	DoS GoldenEye	0	4 998	2
	DoS Hulk	0	5 000	0
	DoS Slowhttptest	0	2 601	2 403
	DoS Slowloris	1	1 647	0
	FTP-BruteForce	0	4 321	676
	Infiltration	3 873	1 100	14
	SQL Injection	5	1	4
	SSH-BruteForce	1	4 996	3

TABLE 4.12 – Performance du MLP pour CSE-CIC-IDS2018.

Métriques	**Données d'entraînement**	**Données de test**
TNR (%)	99.37	99.33
FPR (%)	0.62	0.67
Recall (%)	91.59	91.61
Precision (%)	99.32	99.27
Accuracy (%)	95.48	95.47
f_1 score (%)	95.30	95.29
MCC	0.9124	0.9121

n'est pas due à un manque d'instances. Ce point est discuté plus en détail dans la section 4.2.3.

En accord de l'observation faite avec l'ensemble de données CIC-IDS2017, les attaques de serveurs d'applications ne sont pas correctement détectées. La justesse est d'environ 50%, un résultat équivalent à une estimation aléatoire.

Comparaisons avec les algorithmes classiques de ML Notre solution MLP est comparée aux algorithmes classiques d'apprentissage automatique comme ceux utilisés avec CIC-IDS2017 et les métriques obtenus sont données dans la Table 4.13 après réentraînement des modèles avec les données de 2018.

Table 4.13 – Comparaison du MLP avec les algorithmes classiques de ML sur CSE-CIC-IDS2018.

Algorithme	**Acc (%)**	**Prec (%)**	**Rec (%)**	**FPR (%)**	**MCC**
notre MLP	95.47	99.27	91.61	0.67	0.989
DT	93.49	93.71	93.24	6.26	0.998
RF	95.02	97.17	92.73	2.70	0.997
k-NN	95.19	97.70	92.55	2.17	0.986
SVM	93.64	97.14	90.38	2.26	0.937
LDA	86.12	87.67	84.05	0.12	0.723
QDA	64.68	57.79	98.19	68.81	0.396

Alors que DT et RF étaient légèrement meilleurs sur toutes les métriques que le MLP entraîné sur CIC-IDS2017, le tableau 4.13 met en évidence une situation différente avec CSE-CIC-IDS2018, le MLP obtenant une meilleure justesse et un taux de faux positifs plus faible. Notre solution est la seule à atteindre une justesse élevée et un taux de faux positifs inférieur à 0,7%. QDA qui avait des résultats corrects sur le jeu de données de 2017 échoue complètement à prédire les classes du jeu de données 2018.

Par ces résultats, nous démontrons que la méthode définie dans la section 4.1 fonctionne bien sur CSE-CIC-IDS2018.

La Table 4.14 fournit une vue détaillée des performances des algorithmes via la matrice de confusion simplifiée. Nous constatons que l'ensemble des algorithmes, à l'exception de QDA, ne parvient pas à classer les attaques par infiltration. QDA classifie correctement environ 80% de ce type d'attaques, mais ne parvient pas à détecter le trafic normal et entraîne un taux énorme de faux positifs.

Comparaison avec les résultats antérieurs Le nombre de publications liées au corpus CSE-CIC-IDS2018 reste encore limité. Toutefois, nous avons étudié les résultats de plusieurs d'entre elles. Lorsqu'ils contiennent une métrique pour l'ensemble des données, Les scores des études sont indiqués dans le tableau 4.15. Le contour des études antérieures

TABLE 4.14 – Performances des algorithmes classiques de ML sur CSE-CIC-IDS2018.

		Prédictions DT			Prédictions RF			Prédictions k-NN		
		Bénins	***Attaques correctes***	***Autres attaques***	***Bénins***	***Attaques correctes***	***Autres attaques***	***Bénins***	***Attaques correctes***	***Autres attaques***
Classes réelles	Benign	44 041	-	2 940	45 712	-	1 269	45 960	-	1 021
	Bot	2	4 997	0	8	4 991	0	2	4 997	0
	BruteForce-Web	9	67	0	28	47	1	11	58	7
	BruteForce-XSS	1	26	1	7	19	2	3	18	7
	DDoS attack-HOIC	0	4 999	0	3	4 996	0	0	4 999	0
	DDoS attack-LOIC-UDP	0	214	2	0	216	0	0	216	0
	DDoS attack-LOIC-HTTP	2	4 995	2	4	4 990	5	2	4 990	7
	DoS GoldenEye	1	4 999	0	1	4 999	0	1	4 998	1
	DoS Hulk	0	5 000	0	0	5 000	0	0	4 998	2
	DoS Slowhttptest	0	2 580	2 420	0	4 871	129	0	2 737	2 263
	DoS Slowloris	0	1 647	1	0	1 648	0	1	1 647	0
	FTP-BruteForce	0	4 400	600	0	2 034	2 966	0	4 750	250
	Infiltration	3 162	1 835	2	3 355	1 638	6	3 475	1 519	5
	SQL Injection	0	6	4	5	1	4	3	1	6
	SSH-BruteForce	0	4 996	4	2	4 995	3	1	4 996	3

(a) Matrices de confusion simplifiées pour les algorithmes DT, RF, k-NN.

		Prédictions SVM			Prédictions LDA			Prédictions QDA		
		Bénins	***Attaques correctes***	***Autres attaques***	***Bénins***	***Attaques correctes***	***Autres attaques***	***Bénins***	***Attaques correctes***	***Autres attaques***
Classes réelles	Benign	45 712	-	14 651	41 428	-	5 553	14 651	-	32 330
	Bot	8	4 991	0	2 599	2 398	2	14	2 969	16
	BruteForce-Web	54	16	6	5	32	2	2	50	24
	BruteForce-XSS	12	12	4	1	13	14	9	18	2
	DDoS attack-HOIC	0	4 999	0	0	4 999	0	0	4 997	2
	DDoS attack-LOIC-UDP	0	216	0	1	215	0	0	216	0
	DDoS attack-LOIC-HTTP	0	4 982	17	1 482	3 449	68	0	4 990	9
	DoS GoldenEye	0	4 996	4	0	3 365	1 635	2	4 989	9
	DoS Hulk	0	4 998	2	0	4 719	281	45	4 941	14
	DoS Slowhttptest	0	2 748	2 252	0	2 748	2 252	0	2 706	2 294
	DoS Slowloris	2	1 646	0	0	1 112	536	4	1 636	8
	FTP-BruteForce	0	3 883	1 117	0	3 883	1 117	0	3 955	1 045
	Infiltration	4 432	418	149	3 402	948	649	772	4 196	31
	SQL Injection	9	1	0	1	1	8	2	0	8
	SSH-BruteForce	0	4 996	4	0	4 997	3	6	4 991	3

(b) Matrices de confusion simplifiées pour les algorithmes SVM, LDA, QDA.

différant souvent du nôtre, une analyse suffisamment approfondie est nécessaire pour une bonne interprétation des résultats.

TABLE 4.15 – Comparaison des performances du MLP avec les résultats antérieurs sur CSE-CIC-IDS2018.

Publication	**Algorithme**	**Acc (%)**	**Prec (%)**	**Rec (%)**	**FPR (%)**
notre MLP	MLP	95.47	99.27	91.61	0.67
PRASAD et al., 2019	SGB	100	100	100	-
	RF	99.95	99.96	99.95	-
	DT	99.94	99.92	99.96	-
	k-NN	99.94	99.95	99.95	-
	NB	91.91	91.37	89.93	-
KARATAS et al., 2020	Adaboost	99.32	-	-	-
	DT	98.56	-	-	-
	RF	99.19	-	-	-
	k-NN	95.30	-	-	-
	GB	99.38	-	-	-
	LDA	83.62	-	-	-
HUA, 2020	LightGBM	98.37	98.14	98.37	-
	SVM	92.73	92.06	92.73	
	RF	98.09	97.92	98.09	-
	Adaboost	86.66	76.03	86.66	-
	MLP	97.58	97.23	97.58	-
GAMAGE et SAMARABANDU, 2020	MLP	98.38	-	-	-
	AE	98.22	-	-	-
	DBN	98.31	-	-	-
	LSTM	98.88	-	-	-
	RF	98.34	-	-	-

PRASAD et al. se sont concentrés sur les attaques DoS distribuées avec des algorithmes ML traditionnels. Cette étude montre des résultats supérieurs à 99,9% pour le boosting de gradient stochastique, DT, RF et k-NN. Ces résultats sont très similaires à ceux obtenus sur les attaques DoS avec notre MLP, qui ne produit que deux erreurs de classification sur plus de 10 000 inférences.

KARATAS et al. ont proposé d'équilibrer l'ensemble de données en utilisant la technique de sur-échantillonnage synthétique minoritaire (CHAWLA et al., 2002) et ont mesuré les performances des algorithmes ML classiques. Les attaques ont été regroupées en cinq catégories. Malheureusement, il n'y a aucune explication sur la méthode d'agrégation et la technique d'échantillonnage, ce qui rend difficile la comparaison de nos résultats avec leur étude.

HUA a obtenu de bons résultats avec un MLP. Malheureusement, la structure du MLP n'est pas décrite et sa complexité ne peut pas être comparée à la notre. De plus, comme pour tous les autres cas mentionnés dans la TABLE 4.15, le taux de faux positif n'est pas disponible.

La comparaison des travaux de Gamage et Samarabandu à l'art antérieur se base uniquement sur la justesse moyennée à partir de réseaux de neurones de structures différentes. De plus, ils ont travaillé en préservant la proportion de chaque classe. Sachant que plus de 80 % des cas sont normaux, l'utilisation de la justesse comme unique métrique n'est pas la plus appropriée.

Kim et al. ont étudié les performances d'un CNN et d'un RNN. Les meilleurs résultats ont été obtenus avec CNN. La justesse pour chaque type d'attaque est très similaire à nos résultats. Toutefois, la méthodologie employée rend les résultats difficiles à interpréter. D'une part, le corpus a été divisé en plusieurs sous-corpus contenant chacun du trafic bénin et une famille d'attaques et les résultats sont donnés par sous-corpus. D'autre part, au moins un des sous-ensembles est composé d'une faible quantité d'attaques par rapport à du trafic normal. Ce point ajoute un biais et rend l'utilisation de la justesse moins appropriée. Enfin, la description du CNN ne contient pas le nombre de nœuds dans la couche dense du CNN, nous ne pouvons pas comparer la complexité et le temps d'entrainement des réseaux neuronaux.

Un IDS a été proposé pour une infrastructure 5G et les performances des algorithmes SVM, DT, CNN et du RNN ont été étudiées (Ferrag & Maglaras, 2019). Leurs attaques sont agrégées et leurs résultats ne peuvent pas être intégrés dans la Table 4.15. La comparaison avec notre travail est difficile mais on peut observer que les meilleurs résultats sont obtenus avec les réseaux neuronaux. Les attaques Web semblent mieux classées, mais cela est dû à une erreur dans l'étiquetage des attaques. En effet, nous avons identifié à partir du contenu de l'ensemble de données que le nombre d'attaques par force brute et d'attaques Web est inversé dans leur étude. En tenant compte de cela, la force brute FTP et SSH est mieux détectée que les attaques Web. Ceci est cohérent avec nos résultats.

Une enquête sur les articles étudiant le corpus numériques CSE-CIC-IDS2018 (Leevy & Khoshgoftaar, 2020) constate que les scores de performance sont étonnamment élevés et que la plupart des articles sur ce corpus numérique utilisent des classes déséquilibrées et ne détaillent pas la préparation et le nettoyage des données. Notre analyse confirme leurs conclusions.

Nos scores globaux de justesse et de rappel pour les différents algorithmes sont légèrement inférieurs aux publications antérieures du tableau 4.15, mais sont obtenus avec un jeu de données mieux équilibré et une description détaillée de notre méthode permettant la reproductibilité de nos expériences.

4.2.3 Discussion

Au-delà de la confirmation qu'une solution de détection d'intrusions réseau basée sur le perceptron multi-couches peut atteindre une performance similaire à celle de l'apprentissage automatique traditionnel, cette étude met en évidence certains éléments à approfondir.

Attaque des serveurs d'applications Le projet OWASP top ten (OWASP, 2017) a signalé en 2017 que l'injection est en tête des risques de sécurité les plus critiques pour les serveurs d'applications, comme c'était déjà le cas dans la version 2013. Malgré la popularité de cette attaque, elle est largement sous-représentée dans les ensembles de données. CIC-IDS2017 et CSE-CIC-IDS2018 contiennent moins de cent conversations d'injection SQL sur un total de plus de seize millions de conversations. Ce problème existe pour d'autres attaques d'applications Web telles que la force brute et le cross-site scripting. Le manque d'instances empêche un apprentissage automatique efficace, mais ce n'est pas le seul problème.

Injection SQL En effet, outre le problème de la quantité, les attaques SQL présentent une autre difficulté. Les attaques reposent sur des requêtes SQL mal formées. Un exemple de base consiste à transmettre des identifiants de connexion erronés (V.Harika Naidu, 2018). Le nom de connexion est remplacé par un test qui est toujours vrai et le mot de passe est simplement ignoré en raison de deux caractères spéciaux dans la requête. À partir de cet exemple, nous pouvons facilement comprendre que les statistiques de conversations ne sont pas vraiment affectées, rendant extrêmement difficile de différencier une injection SQL d'une requête SQL légitime et plus généralement d'un trafic normal. Dans de tels cas, les IDS basés sur les conversations pourraient être moins efficaces que les techniques basées sur les paquets permettant une analyse de la charge utile.

Infiltration Comme mentionné sur le site Web présentant le corpus numérique CSE-CIC-IDS2018 (Canadian Institute for Cybersecurity, 2018), l'attaque par infiltration est exécutée en deux étapes. Un fichier malveillant est de prime abord téléchargé par l'ordinateur victime, puis une porte dérobée est utilisée pour analyser le réseau interne. Puisque les détails de l'étiquetage manuel ne sont pas fournis, il est impossible de savoir quelle partie de l'attaque est étiquetée comme infiltration. Compte tenu du nombre énorme de conversations, nous pourrions deviner qu'elle contient les deux parties. Comme les attaques Portscan de l'ensemble de données de 2017 sont correctement détectées, on pourrait s'attendre à pouvoir atteindre un bon niveau de détection pour les attaques d'infiltration de 2018. En mélangeant le téléchargement de fichiers malveillants et le balayage du réseau dans une seule classe, notre classifieur pourrait ne pas être en mesure de détecter une telle complexité. Des investigations supplémentaires sont nécessaires pour mieux comprendre la faible justesse de classification de cette attaque.

L'analyse des adresses IP dans CIC-IDS2017 montre clairement que toutes les conversations d'infiltration correspondent à la communication entre les machines de l'attaquant et de la victime et ne couvrent pas le scan du réseau interne. Par conséquent, seul le téléchargement de fichiers est considéré comme une attaque d'infiltration. Comme pour l'injection SQL, il n'est pas possible d'identifier un téléchargement de fichier légitime d'un téléchargement contenant un morceau de code malveillant sur la base des statistiques de conversations. Une telle détection nécessiterait alors une analyse des charges utiles des paquets.

4.3 Synthèse

Dans ce chapitre, nous avons tout d'abord défini notre méthodologie permettant une comparaison entre un réseau de neurones et un ensemble d'algorithmes classiques d'apprentissage automatique. Cette approche a permis de sélectionner deux corpus : le premier permettant de tester notre méthode et le second permettant de valider la généralisation des résultats obtenus.

les phases de préparation des données, de création et d'entraînement des modèles ont ensuite été clarifiées. Enfin, les métriques ont été définies de façon à comparer les résultats.

Les tests réalisés sur le corpus CIC-IDS2017 ont montré qu'un réseau de neurones simple, de type MLP, est capable d'atteindre des performances similaires à celles d'algorithmes classiques de machine learning lorsqu'il est correctement entraîné. Nos mesures avec ces algorithmes sont en cohérence avec les mesures disponibles dans les publications antérieures.

Pour finir, nous avons réutilisé la même méthode avec un second corpus. Même si les résultats sont un peu moins bons, les performances du MLP s'approchent encore des algorithmes classiques lorsqu'on supprime les attaques d'infiltrations. Si, *a priori*, cette considération semble étrange, elle prend tout son sens considérant la façon dont a été construit le corpus.

Grâce à la méthode appliquée et les expériences menées, nous avons validé notre première hypothèse énoncée page 20 : « *Un réseau de neurones potentiellement exécutable sur un accélérateur matériel modeste peut atteindre d'aussi bonnes performances de détection d'intrusions réseau que les algorithmes d'apprentissage automatique classiques* ».

Ces travaux ont conduit à des publications dans des conférences et dans une revue (Rosay et al., 2020a, 2020b, 2020c ; Rosay et al., 2022).

Nous pouvons à présent étudier la deuxième hypothèse sur la fiabilisation d'un corpus numérique de détection d'intrusions dans le chapitre suivant.

Références

Bergstra, J. & Bengio, Y., (2012), Random Search for Hyper-parameter Optimization, *The Journal of Machine Learning Research*, *13*, 281-305.

Bottou, L., Weston, J. & Bakir, G., (2004), Breaking SVM Complexity with Cross-Training, In L. Saul, Y. Weiss & L. Bottou (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/2004/file/dbbf603ff0e99629dda5d75b6f75f966-Paper.pdf

Canadian Institute for Cybersecurity, (2018), CSE-CIC-IDS2018 on AWS, A collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2018.html

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P., (2002), SMOTE : Synthetic Minority over-Sampling Technique, *J. Artif. Intell. Res.*, *16*, 321-357.

Chicco, D., Tötsch, N. & Jurman, G., (2021), The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation, *BioData Mining*, *14 1*, 13, https://doi.org/10.1186/s13040-021-00244-z

Ferrag, M. A. & Maglaras, L., (2019), DeliveryCoin : An IDS and Blockchain-Based Delivery Framework for Drone-Delivered Services, *Computers*, *8 3*, https://doi.org/10.3390/computers8030058

Gamage, S. & Samarabandu, J., (2020), Deep learning methods in network intrusion detection : A survey and an objective comparison, *Journal of Network and Computer Applications*, *169*, 102767, https://doi.org/10.1016/j.jnca.2020.102767

Garcia, S., Parmisano, A. & Erquiaga, M. J., (2020), IoT-23 : A labeled dataset with malicious and benign IoT network traffic, https://doi.org/10.5281/zenodo.4743746

Geron, A., (2019), *Hands-on machine learning with scikit-learn, keras, and TensorFlow : Concepts, tools, and techniques to build intelligent systems* (2e éd.), O'Reilly Media.

Goodfellow, I., Bengio, Y. & Courville, A., (2016), *Deep Learning*, MIT Press.

Hornik, K., (1991), Approximation capabilities of multilayer feedforward networks, *Neural Networks*, *4 2*, 251-257, https://doi.org/10.1016/0893-6080(91)90009-T

Hua, Y., (2020), An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM, *Information Communication Technologies Conference (ICTC)*, 125-130, https://doi.org/10.1109/ICTC49638.2020.9123302

Jiang, J., Yu, Q., Yu, M., Li, G., Chen, J., Liu, K., Liu, C. & Huang, W., (2018), ALDD : A Hybrid Traffic-User Behavior Detection Method for Application Layer DDoS, *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 1565-1569, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00225

Karatas, G., Demir, O. & Sahingoz, O. K., (2020), Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset, *IEEE Access*, *8*, 32150-32162, https://doi.org/10.1109/ACCESS.2020.2973219

Kim, J., Shin, Y. & Choi, E., (2019), An Intrusion Detection Model based on a Convolutional Neural Network, *Journal of Multimedia Information System*, *6 4*, 165-172, https://doi.org/10.33851/JMIS.2019.6.4.165

Kingma, D. P. & Ba, J., (2015), Adam : A Method for Stochastic Optimization, *3rd International Conference for Learning Representations*.

Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S., (2017), Self-Normalizing Neural Networks, *Advances in Neural Information Processing Systems*, 971-980.

Leevy, J. L. & Khoshgoftaar, T. M., (2020), A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data, *Journal of Big Data*, *7 1*, 104, https://doi.org/10.1186/s40537-020-00382-x

Lin, W., Lin, H., Wang, P., Wu, B. & Tsai, J., (2018), Using Convolutional Neural Networks to Network Intrusion Detection for Cyber Threats, *IEEE International Conference on Applied System Invention (ICASI)*, 1107-1110, https://doi.org/10.1109/ICASI.2018.8394474

Matthews, B., (1975), Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme, *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405 2*, 442-451, https://doi.org/10.1016/0005-2795(75)90109-9

McHugh, J., (2000), Testing Intrusion Detection Systems : A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory, *ACM Trans. Inf. Syst. Secur.*, *3 4*, 262-294, https://doi.org/10.1145/382912.382923

Moustafa, N. & Slay, J., (2015), UNSW-NB15 : A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set), *Military Communications and Information Systems Conference (MilCIS)*, 1-6, https://doi.org/10.1109/MilCIS.2015.7348942

OWASP, (2017), OWASP top ten 2017, Récupérée 24 juillet 2022, à partir de https://owasp.org/www-project-top-ten/OWASP%5C_Top%5C_Ten%5C_2017/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al., (2011), Scikit-learn : Machine learning in Python, *the Journal of machine Learning research*, *12*, 2825-2830.

Prasad, M. D., V, P. B. & Amarnath, C., (2019), Machine Learning DDoS Detection Using Stochastic Gradient Boosting, *International Journal of Computer Sciences and Engineering*, *7*, 157-166, https://doi.org/10.26438/ijcse/v7i4.157166

Raileanu, L. E. & Stoffel, K., (2004), Theoretical comparison between the gini index and information gain criteria, *Annals of Mathematics and Artificial Intelligence*, *41 1*, 77-93, https://doi.org/10.1023/B:AMAI.0000018580.96245.c6

Rosay, A., Carlier, F. & Leroux, P., (2020a), Feed-forward neural network for Network Intrusion Detection, *2020 IEEE 91st Vehicular Technology Conference*, https://doi.org/10.1109/VTC2020-Spring48590.2020.9129472

Rosay, A., Carlier, F. & Leroux, P., (2020b), MLP4NIDS : Application pratique d'un réseau de neurones pour la détection d'intrusions réseau dans les voitures connectées, *Applications Pratiques de l'Intelligence Artificielle (APIA 2020)*, 27-34, http://pfia2020.fr/wp-content/uploads/2020/08/Actes_CH_PFIA2020_V3.pdf

Rosay, A., Carlier, F. & Leroux, P., (2020c), MLP4NIDS : An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset, In S. Boumerdassi, É. Renault & P. Mühlethaler (Éd.), *Machine Learning for Networking* (p. 240-254), Springer International Publishing, https://doi.org/10.1007/978-3-030-45778-5_16

Rosay, A., Riou, K., Carlier, F. & Leroux, P., (2022), Multi-layer perceptron for network intrusion detection, *Annals of Telecommunications*, *77 5*, 371-394, https://doi.org/10.1007/s12243-021-00852-0

Sharafaldin, I., Lashkari, A. H., Hakak, S. & Ghorbani, A. A., (2019), Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy, *International Carnahan Conference on Security Technology (ICCST)*, 1-8, https://doi.org/10.1109/CCST.2019.8888419

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., (2018), Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, 108-116, https://doi.org/10.5220/0006639801080116

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A., (2009), A Detailed Analysis of the KDD CUP 99 Data Set, *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6, https://doi.org/10.1109/CISDA.2009.5356528

Ullah, I. & Mahmoud, Q. H., (2019), A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks, *16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 1-6, https://doi.org/10.1109/CCNC.2019.8651782

Ullah, I. & Mahmoud, Q. H., (2020), A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks, In C. Goutte & X. Zhu (Éd.), *Advances in Artificial Intelligence* (p. 508-520), Springer International Publishing, https://doi.org/10.1007/978-3-030-47358-7_52

Ustebay, S., Turgut, Z. & Aydin, M. A., (2018), Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier, *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 71-76, https://doi.org/10.1109/IBIGDELFT.2018.8625318

V.Harika Naidu, M. V., Shreya Valluri, (2018), Exposure Of Sql Injection In Packet Stream, *International Journal of Engineering and Computer Science*, *5 11.*

Vinayakumar, R., Soman, K. P. & Poornachandran, P., (2017), Applying Convolutional Neural Network for Network Intrusion Detection, *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1222-1228, https://doi.org/10.1109/ICACCI.2017.8126009

Wolpert, D. H., (1996), The Lack of A Priori Distinctions Between Learning Algorithms, *Neural Computation*, *8 7*, 1341-1390, https://doi.org/10.1162/neco.1996.8.7.1341

Chapitre 5

Méthodologie de fiabilisation d'un corpus de détection d'intrusions

---

> « *Il semble que la perfection soit atteinte, non quand il n'y a plus rien à ajouter mais quand il n'y a plus rien à retrancher.* »
>
> Antoine de Saint-Exupéry
> *Écrivain, poète et aviateur (1900 – 1944)*

Les systèmes embarqués contraints ne souffrent pas le superflu. Aussi est-il nécessaire pour les concepteurs de ce type d'objet d'éliminer tout ce qui n'est pas indispensable, faute de quoi nous nous éloignons de la perfection en surdimensionnant inutilement le système. C'est dans cet esprit que ce chapitre illustre comment la fiabilisation du corpus numérique nous conduit à la recherche d'une sobriété dans l'exploitation des ressources matérielles.

Nous avons remarqué dans la section 4.1.2 que huit caractéristiques n'étaient pas porteuses d'information utile pour la détection d'intrusions, car leurs valeurs sont identiques pour l'ensemble des conversations. La suppression de ces caractéristiques à l'entrée de nos différents algorithmes de classification a permis une réduction des calculs à effectuer. Il semble légitime de penser que d'autres caractéristiques pourraient également être écartées afin de réduire davantage l'utilisation de ressources lors des opérations de classification, sans pour autant sacrifier les performances de détection d'intrusions.

Une analyse rapide du corpus numérique permet d'illustrer cette possibilité avec le cas simple et intéressant d'une caractéristique déclarée non-informative dans la section 4.1.2.3. Parmi les caractéristiques, six sont des ratios calculés sur la notion de *bulk*. Un *bulk* est composé d'au moins quatre paquets allant dans le même sens et avec moins d'une seconde entre deux paquets consécutifs. Intuitivement, une valeur nulle pour les six caractéristiques basées sur les *bulks* est suspecte.

Ces observations lèvent un doute sur la fiabilité du corpus. La volonté de réduire la complexité du modèle de classification tout en obtenant des performances de détection à un bon niveau nous conduit à fiabiliser les données utilisées pour entrainer nos modèles de classification.

Dans un premier temps, la section 5.1 propose une méthodologie d'analyse d'un corpus de détection d'intrusions. La section 5.2 fournit, quant à elle, une analyse de corpus numérique conformément à la méthodologie. Nos propositions de fiabilisation du corpus

sont ensuite expliquées dans la section 5.3 et sont suivies d'une méthodologie d'expérimentation en section 5.4. Enfin, la section 5.5 contient les résultats et une discussion.

5.1 Méthodologie proposée

Pour aboutir à un corpus de bonne qualité, nous proposons une méthodologie de fiabilisation décrite par la FIGURE 5.1. Cette méthode n'est applicable qu'aux corpus proposant des données brutes, par exemple les trames Ethernet échangées sur un réseau, et des caractéristiques de conversations générées depuis ces données brutes. Nous allons l'illustrer avec le corpus numérique CIC-IDS2017.

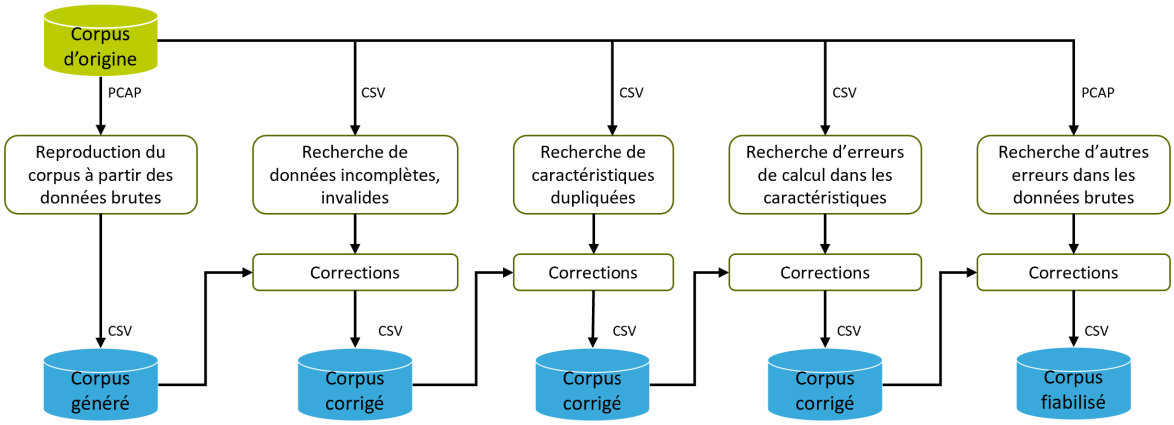


FIGURE 5.1 – Méthodologie de fiabilisation.

Dans l'idéal, la totalité des étapes conduisant à un corpus numérique de détection d'intrusions devrait être entièrement reproductible. Disposer d'un réseau et de machines identiques, d'une configuration exacte et complète de chaque appareil connecté au réseau est en réalité extrêmement difficile à réaliser.

En pratique, les corpus numériques d'IDS contiennent les caractéristiques extraites des paquets réseau et, pour certains d'entre eux, les données brutes sous forme de fichiers PCAP. CIC-IDS2017 se place dans cette catégorie et permet d'appliquer la méthodologie proposée pour vérifier et accroître sa fiabilité.

Nous proposons de commencer par la reproduction du corpus numérique à partir des données brutes. Des signes évidents peuvent apparaître dès cette étape, par exemple si le fichier généré diffère du fichier d'origine.

Une liste de recherche de différents problèmes et de vérifications permet leur détection et leur correction. Parmi ceux-ci, les points particuliers à étudier sont :

— la recherche de données incomplètes, par exemple avec des lignes vides ou des caractéristiques manquantes, ou de données invalides telles que des chaînes de caractères dans des champs numériques ;

— les caractéristiques dupliquées ;

— la vérification des valeurs des caractéristiques calculées ;
— la recherche de différents types d'informations à partir des données brutes, comme les protocoles ou les services utilisés, mais aussi les comportements lors des différents types de trafic afin de détecter d'éventuelles erreurs de labellisation.

Bien que la FIGURE 5.1 établisse une séquence d'étude, l'ordre peut être adapté. L'important reste de détecter les différentes erreurs afin de les corriger et d'obtenir un corpus numérique d'IDS avec un niveau de confiance maximum quant à la qualité de son contenu.

La détection et la correction d'erreurs présentent une difficulté majeure. En effet, les erreurs existantes ne doivent pas être ignorées et les modifications ne doivent pas introduire de nouveaux problèmes. À cette fin, nous avons mis en œuvre tout un panel de méthodes :

— l'analyse de l'outil CICFlowMeter utilisé pour la génération du corpus numérique CIC-IDS2017 ;
— l'analyse grâce à des outils alternatifs existants tels que des librairies pour le calcul de caractéristiques ;
— la reproduction des attaques avec les outils et étude de leurs comportements avec le logiciel libre Wireshark (BANERJEE et al., 2010 ; NDATINYA et al., 2015) ;
— l'extraction du corpus original pour travailler sur de petits morceaux et plus facilement étudier les erreurs d'analyses faites par CICFlowMeter ;
— le croisement des sources d'informations venant de publications, du site web correspondant au corpus, ou des fichiers fournis.

Malgré toutes les précautions prises, des problèmes peuvent subsister. La publication de nos outils de corrections et du corpus fiabilisé permet à la communauté scientifique de reproduire, vérifier et éventuellement améliorer nos apports.

5.2 Analyse du corpus numérique CIC-IDS2017

La détection de problèmes dans un corpus numérique d'IDS tel que CIC-IDS2017 est fastidieux. L'analyse porte sur 50 Go de données brutes sous la forme de cinq fichiers PCAP. Ces données ont été transformées par un outil Java en 2 830 743 lignes de quatre-vingt-quatre caractéristiques et classées en quinze catégories.

Expliquer en détail chacune de nos analyses serait tout aussi fastidieux. Aussi, cette section résume nos résultats d'analyse.

5.2.1 Reproduction du corpus à partir des données brutes

Les caractéristiques du corpus original ont été extraites des fichiers PCAP avec un outil appelé CICFlowMeter, anciennement ISCXFlowMeter (CANADIAN INSTITUTE FOR CYBERSECURITY, 2017a ; DRAPER-GIL. et al., 2016 ; LASHKARI et al., 2017). Le code source de cet outil est disponible, mais a toutefois évolué au fil du temps sans gestion de

TABLE 5.1 – Noms des fichiers formant les corpus CIC-IDS2017 et LYCOS-IDS2017.

Fichiers PCAP	Fichiers originaux	Fichiers générés par CICFlowMeter
Monday-WorkingHours.pcap	Monday-WorkingHours.pcap_ISCX.csv	Monday-WorkingHours.pcap_Flow.csv
Tuesday-WorkingHours.pcap	Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday-WorkingHours.pcap_Flow.csv
Wesnesday-WorkingHours.pcap	Wednesday-workingHours.pcap_ISCX.csv	Wednesday-workingHours.pcap_Flow.csv
Thursday-WorkingHours.pcap	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Thursday-WorkingHours.pcap_Flow.csv
	Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	
Friday-WorkingHours.pcap	Friday-WorkingHours-Morning.pcap_ISCX.csv	Friday-WorkingHours.pcap_Flow.csv
	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	
	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	

version claire, rendant impossible de connaître la version exacte à utiliser pour reproduire les caractéristiques extraites de l'ensemble de données. Le nombre de fichiers générés par CICFlowMeter, ainsi que leurs noms, ne correspondent pas à ceux du corpus original comme le montre la TABLE 5.1.

Aucune version de CICFlowMeter mise en ligne ne permet de générer des fichiers en nombre et en nom identiques au corpus numérique d'origine. Nous supposons que les fichiers ont été générés avec une version antérieure de l'outil dont le code source n'est pas disponible. Il nous est donc impossible de reproduire les fichiers CSV à partir des fichiers PCAP fournis.

Les comparaisons entre les fichiers CSV générés par CICFlowMeter, ci-après dénommés fichiers CFM, et les fichiers CSV originaux, ci-après dénommés fichiers ISCX, ont révélé certaines divergences. Nous avons constaté que certains problèmes apparaissent à la fois dans les fichiers CFM et ISCX, tandis que d'autres n'apparaissent que dans l'un d'entre eux. Ils sont classés en cinq catégories et analysés dans les sections suivantes :

— duplication de caractéristiques ;
— erreurs de calcul de caractéristiques ;
— détection erronée de protocoles ;
— terminaison inadéquate de sessions TCP ;
— problème de labellisation.

Bien que les trois dernières catégories correspondent à la même étape de la méthodologie proposée, appelée la recherche d'autres erreurs dans les données brutes, nous les avons traités dans des sections séparées pour faciliter la compréhension des problèmes identifiés.

Avant d'entrer dans le détail, posons quelques définitions des mots anglais utilisés dans les caractéristiques du corpus et utilisés dans notre texte :

— forward et uplink : sens de transmission depuis la source vers la destination ;
— backward et downlink : sens de transmission depuis la destination vers la source ;
— subflow : ensemble de paquets appartenant à la même conversation, reçus avec moins d'une seconde entre deux paquets consécutifs, quelle que soit leur direction ;
— bulk : ensemble de paquets appartenant à la même conversation, reçus avec moins d'une seconde entre deux paquets consécutifs, dans la même direction.

La définition de bulk et subflow n'est pas donné par les créateurs du corpus et a été déduite du code de CICFlowMeter.

5.2.2 Caractéristiques dupliquées

Quatre caractéristiques dupliquées ont été identifiées dans les fichiers ISCX ;
— la longueur moyenne des paquets par conversation, calculée pour les deux caractéristiques : *Average Packet Size* et *Packet Length Mean* ;
— la longueur moyenne des paquets forward transmis dans la conversation : *Fwd Packet Length Mean* et *Fwd Segment Size Avg*.
— la longueur moyenne des paquets backward transmis dans la conversation : *Bwd Packet Length Mean* et *Bwd Segment Size Avg*.
— la quatrième caractéristique dupliquée dans les fichiers ISCX, corrigée dans le code CICFlowMeter, est la *Fwd Header Length* avec *Fwd Header Length.1*.

Comme le code utilisé pour générer les fichiers ISCX n'est pas disponible, nous expliquons seulement les erreurs de calcul restantes dans les fichiers CFM en utilisant le code CICFlowMeter.

5.2.3 Caractéristiques mal calculées

Trente-quatre caractéristiques mal calculées ont été trouvées dans les fichiers ISCX. Très probablement en raison de mises à jour du code de CICFlowMeter, certaines caractéristiques ont été corrigées dans la dernière version de l'outil (téléchargée le 23-03-2021) sur laquelle nous avons basé notre étude. Vingt-trois erreurs restantes ont été trouvées dans le calcul des caractéristiques, nombre tombant à vingt-et-un si nous gardons seulement les caractéristiques non dupliquées.

Les problèmes de calcul sont regroupés en cinq familles.
— L'erreur la plus courante est due à un problème de type de calcul (division en nombres entiers au lieu de division en nombres flottants). Cela affecte onze fonctionnalités : six liées aux bulks, quatre liées aux subflows et une liée au ratio downlink/uplink.
— Pour chaque nouveau paquet, les caractéristiques de *backward bulk* sont mises à jour, indépendamment de la direction du paquet. De la même manière, les caractéristiques de *forward bulk* ne sont jamais mises à jour. Cela conduit à des valeurs erronées pour les six caractéristiques liées au *bulk*.
— Le nombre de *subflow* est mis à jour en fonction d'un test sur les valeurs d'horodatage. En raison de parenthèses mal placées, le test est toujours vrai et augmentant le nombre de *subflow* pour chaque nouveau paquet reçu.
— Le premier paquet de chaque conversation est utilisé deux fois pour mettre à jour les caractéristiques de longueur de paquet, impactant ainsi trois caractéristiques : la moyenne, l'écart type et la longueur totale du paquet.
— Les caractéristiques liées au TCP contiennent des erreurs ; Les comptages des drapeaux SYN et PSH sont inversés, tout comme les comptages des drapeaux FIN et URG. Les comptages *forward* et *backward* pour PSH et URG ne sont jamais mis à jour. La taille initiale de la fenêtre TCP dans le sens *backward* est mise à jour avec chaque paquet reçu et contient donc la dernière valeur au lieu de la valeur initiale.

5.2.4 Mauvaise détection du protocole

CICFlowMeter traite les paquets en fonction du protocole qui est détecté. Malheureusement, l'outil ne parvient pas toujours à l'identifier. Deux types de problèmes de détection de protocole dus à une mauvaise analyse des trames Ethernet ont été détectés.

Un premier niveau de détection devrait dépendre du champ Ethertype de l'en-tête Ethernet. CICFlowMeter attribue tous les paquets du fichier PCAP à IPv4 ou IPv6. Au lieu d'utiliser le champ Ethertype, l'outil vérifie si le paquet contient un en-tête IPv4 ou IPv6. Si une séquence d'octets dans la charge utile ressemble à un en-tête IPv4 ou IPv6, le paquet est classé en conséquence, quel que soit le champ Ethertype. Une analyse du contenu du fichier PCAP montre que des paquets ARP (Address Resolution Protocol), LLDP (Link Layer Discovery Protocol) et CDP (Cisco Discovery Protocol) sont présents et mal classés. Pour illustrer cela, certains paquets ARP sont interprétés comme des paquets IPv4 avec les adresses IP 8.0.6.4 et 8.6.0.1 correspondant aux valeurs des champs d'en-tête ARP.

Un deuxième niveau d'analyse doit ensuite être basé sur le champ de protocole de l'en-tête IPv4. De la même manière que pour la détection d' IPv4 ou IPv6, CICFlowMeter vérifie si le paquet contient un en-tête TCP ou UDP. Une séquence d'octets dans la charge utile peut être interprétée à tort comme un en-tête TCP ou UDP. Si l'outil ne détecte pas l'un de ces deux protocoles, le paquet est classé dans un autre groupe qui ne correspond à aucun protocole IPv4. L'analyse montre que les fichiers PCAP contiennent des paquets ICMP (Internet Control Message Protocol), IGMP (Internet Group Management Protocol) et SCTP (Stream Control Transmission Protocol). Les paquets TCP et UDP sont classés selon les numéros de protocole IANA (Arkko & Bradner, 2008 ; IANA, 2021), tandis que la plupart des autres paquets sont associés à un numéro de protocole zéro. Nous avons observé que CICFlowMeter classe les paquets ICMP parfois comme TCP, parfois comme UDP ou comme aucun des deux. Un autre effet secondaire apparaît avec les trames fragmentées. Excepté le premier, les fragments UDP ne sont pas reconnus comme UDP et sont traités comme d'autres protocoles avec un numéro de protocole zéro. Ces fragments constituent une autre conversation, rendant les caractéristiques de la conversation UDP erronées.

5.2.5 Terminaison incohérente des sessions TCP

Dans CICFlowMeter, les conversations sont fermées soit lorsque leur durée dépasse un délai d'attente prédéfini de 120 s, soit si le paquet reçu porte un drapeau FIN. TCP est un protocole connecté avec poignée de main. Un exemple typique est décrit dans la Figure 5.2.

Lorsqu'une nouvelle communication TCP est lancée, une première conversation est créée. Elle contiendra alors tous les paquets d'établissement de la session (*3-way handshake*) et du transfert de données. Cette conversation est fermée lorsque le premier paquet de la phase de clôture (*4-way handshake*) est reçu. Une deuxième conversation est produite

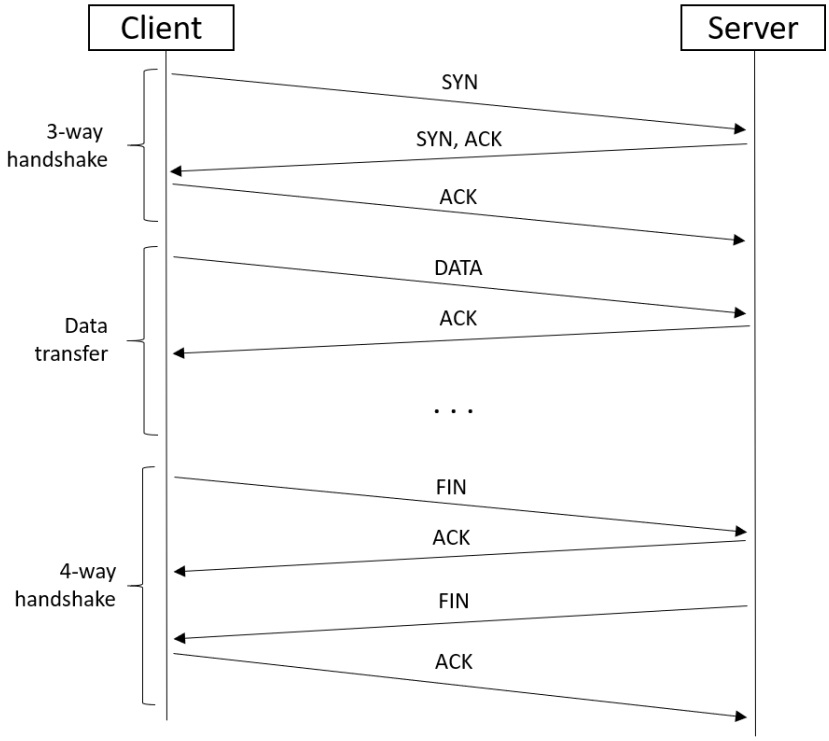


FIGURE 5.2 – Diagramme de séquence TCP.

contenant les deuxième et troisième paquets du *4-way handshake*. Enfin, le dernier paquet crée une troisième conversation. La machine d'état TCP est complexe et de nombreux autres cas de fin de session peuvent se produire.

Ce comportement entraîne trois problèmes. Premièrement, si aucune autre communication utilisant les mêmes adresses et ports n'est lancée, cette troisième conversation est fermée après un délai d'expiration. Dans le cas contraire, le dernier paquet de la conversation est agrégé avec ceux de la nouvelle communication. Deuxièmement, puisque les caractéristiques basées sur les conversations sont similaires pour les *4-way handshake* dans le cadre d'un trafic normal et d'une attaque, elles peuvent avoir un impact négatif sur les performances de l'apprentissage automatique en fonction de l'étiquette attachée aux deuxième et troisième conversations. En outre, une incohérence a été observée dans les étiquettes d'attaques des fichiers ISCX : certaines deuxièmes et troisièmes conversations sont étiquetées comme bénignes tandis que d'autres sont étiquetées comme des attaques, créant ainsi davantage de confusion pour les algorithmes de machine learning. Troisièmement, si une communication TCP est interrompue par un drapeau RST, CICFlowMeter ne considère pas que la conversation est fermée. Toute autre communication avec le même identifiant sera fusionnée avec la conversation qui aurait dû être fermé par le drapeau RST.

5.2.6 Doutes sur la labellisation

Une comparaison entre les informations disponibles sur le site Web présentant le corpus CIC-IDS2017 (Canadian Institute for Cybersecurity, 2017b) et le contenu du jeu de données lui-même a révélé certaines divergences, soulevant ainsi des doutes quant à l'exactitude de la labellisation du corpus.

Ce point est illustré par les attaques de Portscan. Le site Web mentionne que les attaques Portscan se produisent avec et sans pare-feu activé, tandis que l'ensemble des attaques Portscan dans ISCX utilisent l'adresse IP du pare-feu (172.16.0.1). Certaines conversations utilisant l'adresse IP des attaquants (205.174.165.73) lorsque le pare-feu est désactivé sont étiquetées comme bénignes. Ces conversations ont été analysées avec Wireshark (Ndatinya et al., 2015) et le modèle de trafic est très similaire à l'attaque Portscan. En effet, beaucoup de conversations courtes sur de nombreux ports différents ont été trouvées et se terminent par une réinitialisation TCP. Cela semble confirmer que les informations sur le site Web sont correctes, mais que certaines des étiquettes sont erronées.

Nous avons contacté l'un des créateurs de CIC-IDS2017. L'auteur a confirmé que s'il y a des divergences entre le site Web et les labels ISCX, la priorité doit être donnée aux labels. Il a également confirmé que la labellisation a été réalisée à l'aide d'un logiciel propriétaire non accessible au public. Bien que les modèles dans les fichiers PCAP montrent clairement que certaines conversations étiquetées comme bénignes correspondent à un comportement de Portscan, il n'est pas possible de confirmer formellement la question de la labellisation.

5.2.7 Synthèse des problèmes trouvés

En résumé, cinq problèmes ont été clairement identifiés. La duplication des caractéristiques est un problème mineur et peut simplement être corrigé en supprimant les doublons.

L'agrégation des paquets pour créer des conversations est incorrecte en raison d'une mauvaise détection des protocoles. De nombreuses caractéristiques sont mal calculées. La terminaison incorrecte des sessions TCP crée des conversations similaires avec des étiquettes différentes. Des doutes subsistent sur le label de certaines conversations. Ces problèmes majeurs ont un impact sur les performances des algorithmes d'apprentissage automatique qui sont couramment utilisés pour la détection des intrusions réseau.

5.3 Propositions

Compte tenu des problèmes levés dans la section 5.2, nous avançons l'idée que la fiabilisation du corpus numérique nous conduit à optimiser notre système de détection d'intrusions.

Nous proposons, d'une part, de créer un extracteur de caractéristiques de conversation appelé LycoSTand tel que décrit dans la section 5.3.1, et d'autre part, de créer un nouveau corpus fiabilisé appelé LYCOS-IDS2017 comme expliqué dans la section 5.3.2. Cette étape permet d'apporter l'ensemble des corrections indiquées dans notre méthodologie sur la Figure 5.1.

Dans les deux propositions, le mot grec *lycos* apparaît et se traduit en anglais par *wolf*, lui-même correspondant au mot *flow* en renversant l'ordre des lettres. *Flow* est le terme utilisé au sens de conversation dans les systèmes de détection d'intrusions (flow-based intrusion detection system). LycoSTand est une contraction de *lycos* et *understand* et fait sens car pour créer cet outil, il a fallu étudier et comprendre ce qu'est une conversation et comment la former. La capitalisation des lettres ST au centre du nom est une référence à l'entreprise STMicroelectronics, souvent appelée ST.

5.3.1 LycoSTand : un extracteur de caractéristiques de conversations

Étant donné l'ampleur de tous les problèmes à résoudre et de l'absence d'une base de référence claire pour CICFlowMeter, nous avons décidé de créer et de diffuser publiquement un programme C appelé LycoSTand afin qu'il puisse être utilisé pour reproduire nos résultats ou pour traiter les fichiers PCAP d'autres corpus numériques comme alternative à CICFlowMeter. Il s'appuie sur la bibliothèque libpcap pour analyser les paquets Ethernet. Cet outil a été développé et validé sur Ubuntu 18.04, en utilisant libpcap-dev version 1.8.1-6ubuntu1.18.04.2 et le compilateur gcc version 7.5.0.

Ce programme prend les fichiers PCAP, analyse les paquets et calcule quatre-vingt-deux caractéristiques basées sur la conversation. Nous avons principalement réutilisé les caractéristiques de CIC-IDS2017, à l'exception des erreurs de calcul mentionnées dans la section 5.2.3. LycoSTand crée un fichier CSV contenant toutes ces caractéristiques pour chaque fichier PCAP. La liste complète des caractéristiques est décrite dans la Table 5.2.

Une différence majeure entre LycoSTand et CICFlowMeter concerne la gestion des terminaisons de session TCP. Les conversations UDP sont exclusivement fermées avec un délai d'expiration de 120 s après la réception du premier paquet (configurable dans LycoSTand). Pour TCP, nous avons décidé de fermer les conversations dès qu'un paquet avec le drapeau RST ou FIN est reçu. Dans ce cas, le *flowID* est stocké dans une liste *TCP_terminated*. Lorsqu'un paquet TCP est reçu, nous vérifions si son *flowID* se trouve dans cette liste. S'il l'est, les paquets appartenant à la terminaison de la session TCP sont abandonnés jusqu'à ce qu'un paquet avec un drapeau SYN soit reçu, indiquant qu'une nouvelle conversation commence. La liste *TCP_terminated* est nettoyée à chaque nouveau paquet en comparant la différence d'horodatage avec la durée maximale d'une fin de session TCP. Le traitement des paquets est décrit dans les algorithmes 1 et 2.

TABLE 5.2 – Liste des caractéristiques générées par LycoSTand.

Caractéristiques	Description	Caractéristiques	Description
flow_id	Flow identifier	fwd_iat_max	Maximum time between two packets sent in forward direction
src_addr	IP address of the initiator	fwd_iat_min	Minimum time between two packets sent in forward direction
src_port	Port of the initiator of the flow	fwd_iat_mean	Mean time between two packets sent in forward direction
dst_addr	IP address of the receiver	fwd_iat_std	Standard deviation time between two packets sent in forward direction
dst_port	Port of the receiver of the flow	bwd_iat_tot	Total time between two packets sent in backward direction
ip_prot	IP Protocol number	bwd_iat_max	Maximum time between two packets sent in backward direction
timestamp	Timestamp of the first packet of the flow	bwd_iat_min	Minimum time between two packets sent in backward direction
flow_duration	Duration of the flow in microsecond	bwd_iat_mean	Mean time between two packets sent in backward direction
down_up_ratio	Downlink and uplink ratio	bwd_iat_std	Standard deviation time between two packets sent in backward direction
pkt_len_max	Maximum length of a packet	active_max	Maximum time a flow was active before becoming idle
pkt_len_min	Minimum length of a packet	active_min	Minimum time a flow was active before becoming idle
pkt_len_mean	Mean length of a packet	active_mean	Mean time a flow was active before becoming idle
pkt_len_var	Variance length of a packet	active_std	Standard deviation time a flow was active before becoming idle
pkt_len_std	Standard deviation length of a packet	idle_max	Maximum time a flow was idle before becoming active
bytes_per_s	Number of flow bytes per second	idle_min	Minimum time a flow was idle before becoming active
pkt_per_s	Number of flow packets per second	idle_mean	Mean time a flow was idle before becoming active
fwd_pkt_per_s	Number of flow packets per second in forward direction	idle_std	Standard deviation time a flow was idle before becoming active
bwd_pkt_per_s	Number of flow packets per second in backward direction	flag_syn	Number of packets with SYN flag
fwd_pkt_cnt	Number of packets in forward direction	flag_fin	Number of packets with FIN flag
fwd_pkt_len_tot	Total size of packets in forward direction	flag_rst	Number of packets with RST flag
fwd_pkt_len_max	Maximum size of packets in forward direction	flag_ack	Number of packets with ACK flag
fwd_pkt_len_min	Minimum size of packets in forward direction	flag_psh	Number of packets with PUSH flag
fwd_pkt_len_mean	Mean size of packets in forward direction	fwd_flag_psh	Number of packets with PUSH flag in forward direction
fwd_pkt_len_std	Standard deviation size of packets in forward direction	bwd_flag_psh	Number of packets with PUSH flag in backward direction
fwd_pkt_hdr_len_tot	Total number of bytes used for headers in forward direction	flag_urg	Number of packets with URG flag
fwd_pkt_hdr_len_min	Minimum number of bytes used for headers in forward direction	fwd_flag_urg	Number of packets with URG flag in forward direction
fwd_non_empty_pkt_cnt	Number of packets with ≥ 1 byte of TCP payload in forward direction	bwd_flag_urg	Number of packets with URG flag in backward direction
bwd_pkt_cnt	Number of packets in backward direction	flag_cwr	Number of packets with CWR flag
bwd_pkt_len_tot	Total size of packets in backward direction	flag_ece	Number of packets with ECE flag
bwd_pkt_len_max	Maximum size of packets in backward direction	fwd_bulk_bytes_mean	Average number of bytes/bulk in forward direction
bwd_pkt_len_min	Minimum size of packets in backward direction	fwd_bulk_pkt_mean	Average number of packets/bulk in forward direction
bwd_pkt_len_mean	Mean size of packets in backward direction	fwd_bulk_rate_mean	Average number of bulk rate in forward direction
bwd_pkt_len_std	Standard deviation size of packets in backward direction	bwd_bulk_bytes_mean	Average number of bytes/bulk in backward direction
bwd_pkt_hdr_len_tot	Total number of bytes used for headers in backward direction	bwd_bulk_pkt_mean	Average number of packets/bulk in backward direction
bwd_pkt_hdr_len_min	Minimum number of bytes used for headers in backward direction	bwd_bulk_rate_mean	Average number of bulk rate in backward direction
bwd_non_empty_pkt_cnt	Number of packets with ≥ 1 byte of TCP payload in backward direction	fwd_subflow_bytes_mean	Average number of bytes in a sub flow in forward direction
iat_max	Maximum time between two packets sent in flow	fwd_subflow_pkt_mean	Average number of packets in a sub flow in forward direction
iat_min	Minimum time between two packets sent in flow	bwd_subflow_bytes_mean	Average number of bytes in a sub flow in backward direction
iat_mean	Mean time between two packets sent in flow	bwd_subflow_pkt_mean	Average number of packets in a sub flow in backward direction
iat_std	Standard deviation time between two packets sent in flow	fwd_tcp_init_win_bytes	Total number of bytes sent in initial TCP window in forward direction
fwd_iat_tot	Total time between two packets sent in forward direction	bwd_tcp_init_win_bytes	Total number of bytes sent in initial TCP window in backward direction

Algorithme 1 : Traitement des paquets.

```
while True do
    wait for next packet
    clean TCP terminated list
    extract flow identifiers from received packet
    search flow in list of flows
    if IPv4 packet then
        if flow is not in list of flows then
            if TCP packet then
                if flow is not in TCP terminated list then
                    if packet hasn't FIN nor RST flag then
                        create new flow
                        calculate its initial statistics
                    else
                        add ID to TCP terminated list
                else
                    if packet has SYN flag then
                        create new flow
                        calculate its initial statistics
                        remove ID from TCP terminated list
            else
                create new flow
                calculate its initial statistics
        else
            if flow not terminated then
                update running statistics
            else
                closeFlow(packet, flow)
```

Algorithme 2 : Fermeture des conversations.

```
Procedure closeFlow(packet, flow)
    close existing flow
    if flow ended by timeout then
        if TCP packet then
            if packet has FIN or RST flag then
                add ID to TCP terminated list if it is not there yet
            else
                create new flow
                calculate its initial statistics
        else
            create new flow
            calculate its initial statistics
    else
        add ID to TCP terminated list if it is not there yet
    return
```

Lorsqu'une conversation commence, toutes ses caractéristiques sont initialisées à zéro. Elles sont mises à jour lorsqu'un paquet ultérieur appartenant à la même conversation est reçu.

5.3.2 LYCOS-IDS2017 : un corpus fiabilisé

Pour confirmer l'impact des corrections apportées à l'extracteur de caractéristiques sur l'apprentissage automatique, nous avons créé un nouveau jeu de données en traitant les fichiers PCAP de CIC-IDS2017 avec LycoSTand. Comme le jeu de données original était étiqueté avec un logiciel propriétaire, nous avons développé notre propre solution. Ce programme est écrit en Python. Notre solution d'étiquetage repose sur l'analyse du trafic que nous avons effectuée avec Wireshark. Celle-ci a révélé que les étiquettes peuvent être attribuées à l'aide des adresses et des ports source et destination couplés à des horodatages. Nous avons utilisé les informations temporelles fournies sur le site Web du CIC, mais avons ajusté l'heure en fonction de l'analyse du trafic.

L'ensemble de données et le programme qui en résultent sont accessibles à tous à l'adresse https://lycos-ids.univ-lemans.fr/. Nous utilisons différentes techniques pour attacher une étiquette à chaque flux :

— la référence (Canadian Institute for Cybersecurity, 2017b) fournit des informations sur la période et les adresses ou ports impliqués dans les différentes attaques ;
— nous avons analysé les fichiers PCAP avec Wireshark pour confirmer les étiquettes

et, en observant les modèles d'attaques, nous avons pu ajuster les périodes utilisée pour les lancer ;
— nous avons découvert que de nombreuses conversations enregistrées le jeudi après-midi ont été étiquetés comme bénignes dans CIC-IDS2017, alors que l'étiquette réelle est très probablement une attaque de Portscan.

Le dernier item correspond à des attaques par infiltration, ce comportement peut être expliqué par le fait que cette attaque est composée de deux phases. Selon les informations données dans « Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization », un fichier contenant une vulnérabilité est téléchargé et ensuite, l'attaquant exécute une attaque Portscan. Nous pensons que les auteurs de CIC-IDS2017 ont qualifié la deuxième phase de trafic bénin. En l'absence d'informations détaillées de la part des auteurs, il est impossible d'identifier avec une certitude absolue quels flux sont réellement du trafic bénin et lesquels auraient dû être étiquetés comme de l'infiltration. Pour éviter les erreurs dans nos étiquettes, nous avons décidé d'écarter tous les paquets du jeudi après-midi. Par conséquent, LYCOS-IDS2017 ne contient aucune attaque par infiltration.

La liste des attaques et le nombre d'instances pour chaque ensemble de données sont présentés dans la Table 5.3.

Table 5.3 – Nombre d'instances des corpus CIC-IDS2017 et LYCOS-IDS2017.

Instances	CIC-IDS2017	LYCOS-IDS2017
BENIGN	2 273 097	1 395 659
Bot	1 966	735
DDoS	128 027	95 683
DoS GoldenEye	10 293	6 765
DoS Hulk	231 073	158 988
DoS Slowhttptest	5 499	4 866
DoS Slowloris	5 796	5 674
FTP-PATATOR	7 938	4 003
Heartbleed	11	11
Infiltration	36	0
PortScan	158 930	160 106
SSH-PATATOR	5 897	2 959
WebAttack BruteForce	1 507	1 360
WebAttack SQL Injection	21	12
WebAttack XSS	652	661
Total	2 830 743	1 837 500

À l'exception de Portscan et WebAttack XSS, chaque label a moins d'instances dans LYCOS-IDS2017 que dans CIC-IDS2017. Ce phénomène s'explique par les terminaisons de session TCP qui ne créent pas de conversation spécifique pour les terminaisons de session TCP dans notre nouveau corpus.

Pour Portscan, beaucoup de sessions TCP qui se terminent avec un drapeau RST, suivies par une autre commençant avec un drapeau SYN pour le même identifiant de conversation. Ce comportement conduit à plus de flux dans le fichier Lycos que dans le fichier ISCX car CICFlowMeter ne prend pas en compte le drapeau RST pour mettre fin à une conversation.

Une incertitude demeure pour notre programme de labellisation dans un cas précis. Le nombre de WebAttack XSS dans le fichier Lycos est élevé par rapport à ISCX. Nous soupçonnons que CIC-IDS2017 contient du trafic bénin utilisant les mêmes adresses IP et ports que l'attaque. Ici encore, nous regrettons de ne pas avoir accès au programme de labellisation original. Malgré des doutes subsistant sur nos étiquettes pour cette attaque, nous avons décidé de la garder dans l'ensemble de données sachant qu'elle ne représente que neuf conversations. En publiant nos outils, nous espérons que d'autres personnes pourront étudier et éventuellement améliorer notre travail.

5.4 Méthodologie d'expérimentation

Maintenant que nous avons en notre possession le corpus corrigé LYCOS-IDS2107, il est nécessaire de mesurer les performances de détection d'intrusions et, dans la mesure du possible, les comparer au corpus original CIC-IDS2017.

Nous reprenons globalement les mêmes étapes que celles décrites dans la section 4.1. Les données sont préparées de la même manière et les jeux de données d'entraînement, de validation croisée et de test sont générés selon les mêmes principes, de façon à équilibrer le nombre de conversations d'attaques et le nombre de conversations bénignes. Toutefois, comme CIC-IDS2017 et LYCOS-IDS2017 ne contiennent pas exactement le même nombre d'attaques, nous avons ajusté la composition de ces différents jeux de données pour obtenir le même nombre d'instances et permettre une comparaison équitable. Ensuite, chaque ensemble est complété par des instances de trafic normal sélectionnées au hasard, sans remplacement. Le résultat est un ensemble de données équilibré en termes de trafic normal par rapport aux attaques, mais toujours déséquilibré en termes de types d'attaques. La composition exacte des jeux de données est indiquée dans le tableau 5.4.

Seuls les jeux de données d'entraînement et de validation croisée sont utilisés pendant la phase d'apprentissage. La performance finale est mesurée avec le jeu de données de test qui n'a jamais été utilisé pendant l'entraînement. Les comparaisons sont réalisées en utilisant un sous-ensemble des mêmes algorithmes et les mêmes métriques. Nous gardons pour notre analyse les algorithmes classiques les plus performants ainsi que le moins performant en complément de notre MLP. Le sous-ensemble se compose donc des algorithmes LDA, k-NN, DT, RF et MLP.

Les modèles entraînés diffèrent sur un point notable par rapport à la section 4.1. Comme le corpus LYCOS-IDS2017 ne contient pas d'attaque d'infiltration, ces attaques ont été également supprimées de CIC-IDS2017 et les modèles font un apprentissage sur seulement quatorze classes au lieu de quinze.

TABLE 5.4 – Contenu des jeux de données pour la comparaison de CIC-IDS2017 et LYCOS-IDS2017.

Trafic	**Jeu d'entraînement**	**Jeu de validation croisée**	**Jeu de test**
benign	220 316	110 158	110 158
bot	367	183	183
ddos	47 841	23 920	23 920
dos_goldeneye	3 382	1 691	1 691
dos_hulk	79 494	39 747	39 747
dos_slowhttptest	2 433	1 216	1 216
dos_slowloris	2 837	1 418	1 418
ftp_patator	2 001	1 000	1 000
heartbleed	5	2	2
Portscan	79 465	39 732	39 732
ssh_patator	1 479	739	739
webattack_bruteforce	680	340	340
webattack_sql_injection	6	3	3
webattack_xss	317	158	158
Total	440 632	220 312	220 312

Enfin, il nous faut revenir à notre objectif. Nous souhaitons démontrer que la fiabilisation du corpus permet l'optimisation de notre modèle MLP. Pour traiter cet aspect, nous réduisons la complexité de façon drastique en réalisant une sélection des caractéristiques les plus importantes et en limitant le nombre de nœuds de chaque couche du MLP. Les deux couches cachées sont réduites à soixante-quatre et trente-deux nœuds au lieu de deux cent cinquante-six comme définies dans la section 4.1.3. Notre analyse porte sur quatre cas :

— l'utilisation de toutes les caractéristiques nettoyées de CIC-IDS2017 ;
— l'utilisation de toutes les caractéristiques de LYCOS-IDS2017 ;
— l'utilisation des dix caractéristiques les plus importantes de CIC-IDS2017 ;
— l'utilisation des dix caractéristiques les plus importantes de LYCOS-IDS2017.

La sélection de caractéristiques est un exercice difficile et le devient davantage avec des prédicteurs fortement corrélés. Il a été démontré que la corrélation a un impact sur la sélection des caractéristiques avec l'algorithme RF et l'élimination récursive des caractéristiques (RFE) est recommandée lorsqu'il existe des corrélations (GREGORUTTI et al., 2017). Pour minimiser le nombre de caractéristiques, nous avons d'abord étudié les corrélations entre elles deux par deux. Deux coefficients de corrélation, celui de Pearson et celui de Spearman, ont été utilisés. Nous avons choisi d'inclure uniquement les caractéristiques ayant une corrélation inférieure à 97% avec chacun des deux coefficients de corrélation. Avec cette analyse, trente caractéristiques de CIC-IDS2017 ont été supprimées et trente-quatre caractéristiques de LYCOS-IDS2017.

Après avoir éliminé les caractéristiques fortement corrélées, une sélection des caractéristiques restantes a été réalisée en fonction de leur importance telle que rapportée par

l'algorithme RFE, un algorithme largement utilisé dans les mécanismes d'apprentissage automatique. Cette méthode fournit le classement des caractéristiques par importance, selon leur contribution aux performances d'un algorithme d'apprentissage automatique donné. Elle consiste à entraîner un algorithme de ML en supprimant une caractéristique à la fois. La caractéristique diminuant le plus ou le moins les performances, selon ce qui convient le mieux, est ensuite ajoutée au classement. Nous avons choisi d'utiliser un classifieur RF pour effectuer l'élimination récursive de caractéristiques. Finalement, comme le montre le tableau 5.5, dix caractéristiques ont été retenues pour effectuer une comparaison de performances des différents algorithmes ML.

TABLE 5.5 – Liste des dix caractéristiques par ordre d'importance.

Rang	**Caractéristiques**
1	Port destination
2	Protocole IP
3	Durée de la conversation
4	Nombre de paquets en direction forward de la conversation
5	Taille totale des paquets forward de la conversation
6	Longueur minimale des paquets forward de la conversation
7	Longueur moyenne des paquets forward de la conversation
8	Ecart-type de la longueur des paquets forward de la conversation
9	Longueur maximale des paquets backward de la conversation
10	Longueur minimale des paquets backward de la conversation

(a) CIC-IDS2017.

Rang	**Caractéristiques**
1	Port destination
2	Protocole IP
3	Durée de la conversation
4	Ratio Downlink/uplink
5	Longueur maximale des tous les paquets de la conversation
6	Nombre d'octets par seconde de la conversation
7	Nombre de paquet par seconde de la conversation
8	Nombre de paquets en direction forward de la conversation
9	Taille totale des paquets forward de la conversation
10	Longueur maximale des paquets forward de la conversation

(b) LYCOS-IDS2017.

Nous observons que les trois premières caractéristiques sont les mêmes entre les deux ensembles de données. La destination est utile, car elle indique par exemple les transactions FTP ou SSH liées à des attaques spécifiques. Le protocole est également une source

d'information importante pour cet ensemble de données particulier, car toutes les attaques se produisent en TCP. Cependant, l'ordre est différent pour les autres caractéristiques. En effet, des erreurs dans CIC-IDS2017 génèrent des informations incorrectes. Par exemple, le rapport Downlink/Uplink n'est pas très informatif dans le corpus CIC-IDS2017, alors qu'il constitue l'une des caractéristiques les plus informatives dans notre corpus fiabilisé.

5.5 Résultats

Contrairement à ce que nous avons fait au chapitre 4, nous n'allons pas détailler les performances du MLP avant de donner les résultats des algorithmes classiques d'apprentissage automatique. Les métriques calculées pour chaque ensemble de données, pour chaque algorithme ainsi que le temps d'entraînement et le temps de prédiction pour l'ensemble de test complet contenant 220 312 instances sont détaillés dans la Table 5.6. Les résultats sont divisés en quatre quadrants permettant de comparer CIC-IDS2017 avec LYCOS-IDS2017 en utilisant toutes les caractéristiques ou seulement les dix plus importantes.

Table 5.6 – Comparaison de performances entre CIC-IDS2017 et LYCOS-IDS2017.

		Toutes les caractéristiques					**10 caractéristiques**				
Corpus	**Métrique**	**LDA**	**k-NN**	**DT**	**RF**	**MLP**	**LDA**	**k-NN**	**DT**	**RF**	**MLP**
CIC-IDS2017	Acc (%)	87.88	99.30	99.89	99.91	99.40	80.37	99.30	99.25	99.26	97.58
	Prec (%)	81.31	98.87	99.88	99.90	99.07	73.87	99.25	98.98	98.99	98.50
	Rec (%)	98.36	99.74	99.90	99.92	99.23	93.96	99.35	99.53	99.54	96.63
	FPR (%)	22.61	1.14	0.12	0.10	0.94	33.23	0.75	1.03	1.00	1.47
	MCC	0.7748	0.9860	0.9978	0.9983	0.9880	0.6310	0.9860	0.9850	0.9853	0.9518
	Training (s)	4.600	-	12.846	22.255	895.541	0.465	-	1.283	9.677	952.686
	Prédiction (s)	0.101	1793.22	0.046	1.085	0.394	0.52	26.300	0.021	0.965	0.220
LYCOS-IDS2017	Acc (%)	96.59	99.93	99.92	99.96	99.72	93.41	99.60	99.57	99.74	99.54
	Prec (%)	94.00	99.90	99.92	99.95	99.89	89.06	99.69	99.55	99.93	99.69
	Rec (%)	99.54	99.96	99.93	99.98	99.54	98.98	99.50	99.59	99.56	99.39
	FPR (%)	6.35	0.10	0.08	0.06	0.11	12.15	0.31	0.45	0.07	0.31
	MCC	0.9335	0.9986	0.9985	0.9992	0.9943	0.8737	0.9919	0.9914	0.9949	0.9907
	Training (s)	5.574	-	10.014	19.057	819.460	0.452	-	1.747	9.866	1135.500
	Prédiction (s)	0.101	1765.34	0.037	0.941	0.499	0.032	22.437	0.018	0.849	0.220

Pour CIC-IDS2017, nous observons que la réduction de soixante-dix à dix caractéristiques a un impact très mineur sur la plupart des métriques pour les algorithmes classiques DT, RF et k-NN. Bien que DT et RF soient toujours les meilleurs algorithmes, leur taux de faux positifs a considérablement augmenté, passant respectivement de 0.12 % et 0.10 % à 1.03 % et 1.00 %.

Toutes métriques confondues, LYCOS-IDS2017 fournit de meilleurs résultats que CIC-IDS2017. La différence est particulièrement significative pour LDA, qui s'est le plus amélioré, même s'il reste tout de même le plus mauvais algorithme. Sa précision est passée de 88,10 % à 96,59 %, tandis que le taux de faux positifs a diminué de 22,61 % à 6,35 %. La résolution des problèmes du CIC-IDS2017 a très probablement amélioré la capacité à

séparer linéairement les classes. Toutefois, l'ensemble de données n'est toujours pas complètement séparable de façon linéaire. L'arbre de décision et la forêt aléatoire fournissent les meilleurs résultats. Les métriques du réseau neuronal sont très légèrement inférieures aux deux algorithmes mentionnés ci-dessus.

La sélection des caractéristiques a un impact très limité sur LYCOS-IDS2017. Avec seulement dix caractéristiques, la comparaison avec CIC-IDS2017 affiche de meilleurs résultats pour toutes les métriques, quel que soit l'algorithme.

LYCOS-IDS2017 avec dix caractéristiques est même meilleur que CIC-IDS2017 avec toutes les caractéristiques sur la plupart des mesures. RF reste l'algorithme donnant les meilleurs résultats.

La sélection des dix meilleures caractéristiques a permis de réduire le temps d'apprentissage et de prédiction, un critère important pour permettre le déploiement du classifieur dans les appareils connectés disposant de ressources informatiques limitées. Parmi les algorithmes qui obtiennent les meilleures métriques, nous observons que k-NN est le pire en termes de temps d'inférence. k-NN est un classifieur, dit paresseux, qui ne nécessite pas de phase d'apprentissage, mais mesure la distance entre l'instance à classer et toutes les autres instances afin de trouver la classe majoritaire des k plus proches voisins. Pour cette raison, cet algorithme n'est pas adapté aux grands ensembles de données comme le nôtre.

Les métriques du réseau de neurones sont très légèrement inférieures à celles des meilleurs algorithmes lorsque toutes les caractéristiques sont utilisées. La sélection des caractéristiques sur CIC-IDS2017 montre des performances inférieures à DT et RF pour MLP, alors qu'elle reste à un très bon niveau de performances avec LYCOS-IDS2017.

En fonction des critères les plus importants, qui peuvent varier selon l'usage, RF peut être sélectionné pour obtenir les meilleures performances au prix d'un temps d'inférence plus long. Cette contrainte est particulièrement préoccupante pour un système qui doit analyser un réseau à très haut débit. Dans ce cas, l'arbre de décision est une alternative séduisante, mais il entraîne un taux de faux positifs plus élevé. Le MLP est un excellent compromis, avec un taux de faux positifs plus faible que DT et une exécution plus rapide que RF. Il s'agit donc d'un point intéressant puisque l'utilisation d'un arbre de décision semblait être la meilleure option avec CIC-IDS2017.

5.6 Discussion

Au-delà des résultats, deux aspects méritent d'être abordés. Nous avons vu dans ce chapitre qu'un corpus, pourtant largement utilisé, contient un nombre significatif d'erreurs. Nous avons alors proposé une version fiabilisée, dérivée des mêmes données brutes et comparé les performances de classification. Il est légitime de s'interroger, d'une part, sur la valeur de la comparaison, et d'autre part, sur l'impact des erreurs détectées sur d'autres corpus fournis par le CIC.

5.6.1 Limite des comparaisons

Pour que la comparaison soit indiscutable, il faudrait que les conversations du jeu de test soient les mêmes pour CIC-IDS2017 et LYCOS-IDS2017. Malheureusement, cet idéal est impossible dans notre cas. Les problèmes détectés sur le corpus d'origine font que, d'une part, pour un même ensemble de trames Ethernet, il n'y a pas le même nombre de conversations dans LYCOS-IDS2017 et CIC-IDS2017 comme indiqué sur la Figure 5.3, et d'autre part, les erreurs de labellisation sur les conversations correspondant aux fins de session TCP ne sont pas systématiques.

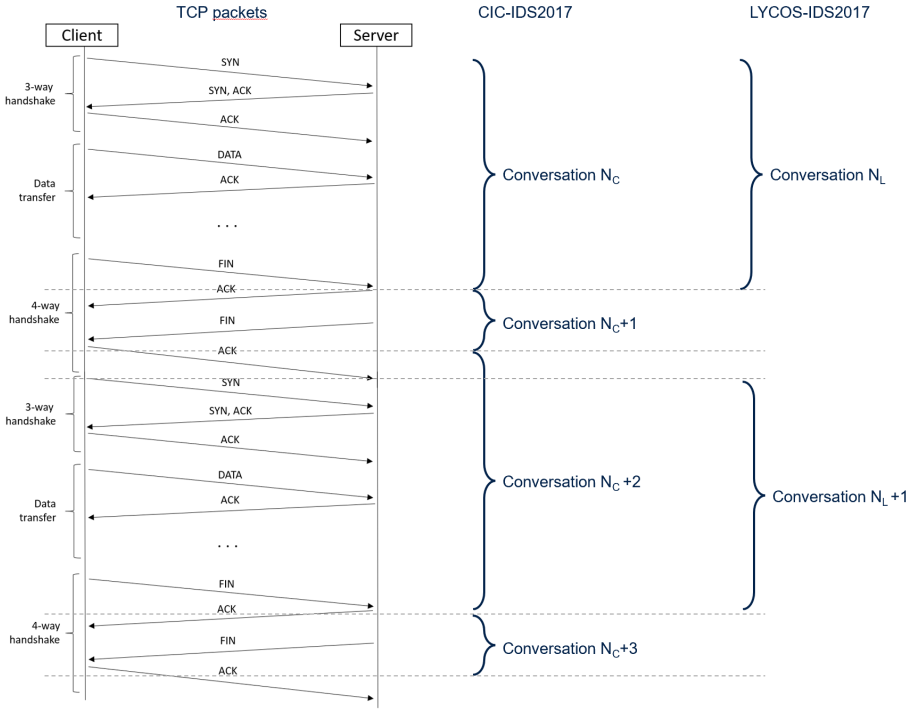
La Figure 5.3a montre que certaines conversations de CIC-IDS2017 n'ont pas de correspondance dans LYCOS-IDS2017. Supprimer ces conversations des tests reviendrait à corriger implicitement une partie des erreurs du corpus d'origine et ne permettrait pas une comparaison objective. La Figure 5.3b illustre le cas où à une seule conversation CIC-IDS2017 correspondent plusieurs conversations de LYCOS-IDS2017. Une conversation se terminant par un drapeau RST - comportement typique de certaines attaques - peut être regroupée avec la connexion TCP suivante, si les adresses et ports source et destination sont identiques. Il nous est impossible d'établir une relation bijective entre l'ensemble des conversations du corpus d'origine et notre propre corpus.

Pour augmenter le niveau de confiance de nos résultats, cinq jeux de données d'entrainement, de validation croisée et de test différents ont été créés en initialisant la graine du générateur de nombres aléatoires avec des valeurs distinctes. Ainsi, les données ont été extraites de CIC-IDS2017 et LYCOS-IDS2017 en suivant les séquences de nombre aléatoires différentes. Il est nécessaire de ré-entraîner l'ensemble des modèles pour chaque cas.

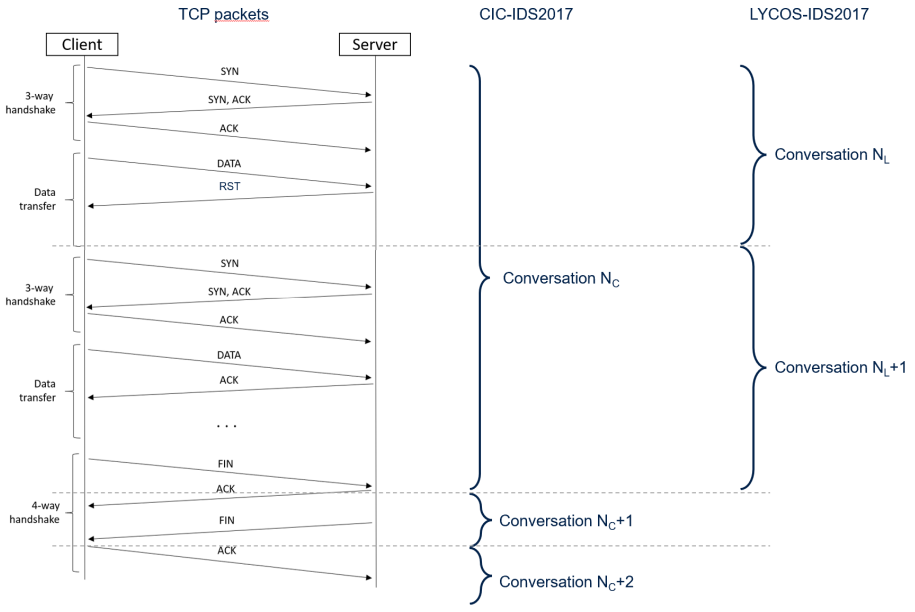
La Table 5.7 regroupe les valeurs moyennes et les coefficients de variation (CV) en pourcentage sur une partie de nos modèles, limités pour faciliter la lecture. Les valeurs non présentes ont un comportement identique à celles affichées dans le tableau. Le coefficient de variation est une mesure de dispersion relative et il correspond au ratio de l'écart-type sur la moyenne.

Tout d'abord, la Table 5.7 montre que les valeurs moyennes sur ce petit échantillon sont très proches des valeurs de la Table 5.6. Ensuite, nous observons que les coefficients de variation sont systématiquement plus grands avec le corpus original. Cette variation peut s'expliquer par la quantité d'instances erronées incorporée de façon aléatoire dans les jeux de données. Les coefficients de variation sont également réduits lorsqu'on diminue le nombre de caractéristiques. L'explication réside dans le fait qu'une grande partie des caractéristiques mal calculées ont été supprimées.

Globalement, même s'il n'est pas possible de faire une comparaison exacte des performances des deux corpus, nous notons que le niveau de confiance dans les résultats obtenus est relativement bon.



(a) Fin de session mal gérée.



(b) RST mal géré.

FIGURE 5.3 – Correspondances entre paquet Ethernet, conversations CIC-IDS2017 et conversations LYCOS-IDS2017.

TABLE 5.7 – Dispersion des performances entre CIC-IDS2017 et LYCOS-IDS2017.

		Toutes les caractéristiques				10 caractéristiques			
		RF		MLP		RF		MLP	
Corpus	**Métrique**	**Moyenne**	**CV**	**Moyenne**	**CV**	**Moyenne**	**CV**	**Moyenne**	**CV**
CIC	Acc (%)	99.92	0.023	99.50	0.158	99.26	0.003	97.87	0.08
	Prec (%)	99.91	0.017	99.40	0.265	98.99	0.004	98.41	0.13
	Rec (%)	99.92	0.030	99.60	0.182	99.54	0.002	97.32	0.29
	FPR (%)	0.09	18.971	0.60	44.362	1.01	0.403	1.57	8.19
	MCC	0.9984	0.046	0.9900	0.317	0.9852	0.0001	0.9575	0.17
LYCOS	Acc (%)	99.96	0.001	99.77	0.030	99.74	0.001	99.51	0.02
	Prec (%)	99.94	0.002	99.78	0.055	99.93	0.002	99.72	0.03
	Rec (%)	99.97	0.002	99.76	0.106	99.56	0.001	99.29	0.02
	FPR (%)	0.06	2.682	0.22	25.244	0.07	1.043	0.27	10.39
	MCC	0.9992	0.001	0.9954	0.061	0.9949	0.002	0.9902	0.03

5.6.2 Extension à des corpus supplémentaires

Nos résultats prouvent qu'une extraction correcte des caractéristiques des données brutes de CIC-IDS2017 apporte un avantage significatif. Sachant que le CIC met à disposition des chercheurs un grand nombre de corpus numériques de détection d'intrusions, il est légitime de considérer les impacts de notre étude sur ceux-ci.

CICFlowMeter a été utilisé pour générer cinq des dix-neuf corpus numériques fournis sur le site Web du CIC : (1) CIC-IDS2017, (2) CIC-AndMal2017 (LASHKARI et al., 2018), (3) CSE-CIC-IDS2018, (4) CIC-InvesAndMal2019 (TAHERI et al., 2019) et (5) CIC-DDoS2019 (SHARAFALDIN et al., 2019).

Nous pouvons nous attendre à ce que les problèmes constatés en (1) soient présents dans tous les ensembles de données générés par CICFlowMeter. Une analyse complète des fichiers PCAP pour chacun de ces ensembles de données serait une tâche énorme. Au lieu de cela, nous nous sommes concentrés sur l'identification de certains artefacts dans les fichiers CSV. Nous avons pu identifier quelques erreurs de calcul, des problèmes dans la détection des protocoles et la fin incohérente des sessions TCP sur tous ces ensembles de données générés. Comme les auteurs de ces corpus numériques ont fourni des données brutes, LycoSTand constitue une alternative intéressante pour l'extraction de caractéristiques basées sur le flux à partir de fichiers PCAP existants.

En plus des jeux de données fournis par le CIC, certains chercheurs utilisent CICFlowMeter pour créer des jeux de données à partir de leurs propres fichiers PCAP (GOHARI et al., 2021 ; ULLAH & MAHMOUD, 2020). Bien que ces ensembles de données n'aient pas été vérifiés, les mêmes faiblesses sont probablement présentes.

5.7 Synthèse

Dans ce chapitre, nous avons tout d'abord proposé une méthodologie de fiabilisation d'un corpus numérique de détection d'intrusions basée sur les conversations.

Ensuite, nous avons analysé le corpus CIC-IDS2017 et mis en évidence de sérieux défauts à plusieurs niveaux : des caractéristiques mal calculées, des identifications hasardeuses de protocoles IP, une mauvaise gestion des conversations et des doutes sur la labellisation des données.

Nous avons proposé deux contributions à l'amélioration du corpus. Un nouvel extracteur de caractéristiques des conversations corrige les problèmes identifiés. Un programme de labellisation fournit un corpus fiabilisé grâce à une analyse des échanges réseau et l'éviction de certaines conversations sur lesquelles un doute persistait.

Une méthodologie d'expérimentation a été décrite pour comparer les performances du corpus original et du corpus fiabilisé. Les résultats ont montré qu'avec un modèle optimisé en termes de complexité de calculs, nous obtenons de biens meilleurs résultats sur le corpus amélioré. Les outils ainsi que le corpus résultant de nos travaux ont été mis à disposition sur le site de Le Mans Université (Rosay & Carlier, 2022) afin qu'ils puissent être analysés et éventuellement améliorés par la communauté scientifique travaillant sur la détection d'intrusions réseau.

Enfin, la discussion a permis de jeter un œil critique, d'une part, sur la comparaison des corpus et, d'autre part, sur les autres corpus générés à partir des outils du laboratoire canadien CIC.

Pour conclure ce chapitre, notre méthodologie de fiabilisation de corpus numérique spécifique à la détection d'intrusions, ainsi que les propositions d'amélioration avec l'extracteur LycoSTand et le programme de labellisation ont permis de générer un modèle de réseau neuronal simplifié, tout en obtenant des performances de détection supérieures. Ceci valide notre deuxième hypothèse : « *La fiabilisation d'un corpus de détection d'intrusions conduit à améliorer la performance de détection tout en garantissant un modèle plus simple* ».

Ces travaux ont conduit à des publications dans deux conférences internationales (Rosay et al., 2021 ; Rosay et al., 2022).

Maintenant que nous avons la possibilité d'utiliser un réseau de neurones simple, entraîné sur un corpus fiabilisé, nous nous penchons dans le chapitre suivant sur la dernière hypothèse. En effet, un système de détection d'intrusions ne résume pas à un modèle de détection et il reste à montrer qu'un tel système soit implémentable sur un objet connecté.

Références

Arkko, J. & Bradner, S., (2008), *IANA Allocation Guidelines for the Protocol Field* (BCP N° 37), RFC Editor, http://www.rfc-editor.org/rfc/rfc5237.txt

Banerjee, U., Vashishtha, A. & Saxena, M., (2010), Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection [Published By Foundation of Computer Science], *International Journal of Computer Applications*, *6 7*, 1-5, https://doi.org/doi.org/10.5120/1092-1427

Canadian Institute for Cybersecurity, (2017a), Applications - CICFlowMeter (formerly ISCXFlowMeter), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/research/applications.html

Canadian Institute for Cybersecurity, (2017b), Intrusion Detection Evaluation Dataset (CICIDS2017), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2017.html

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I. & Ghorbani., A. A., Characterization of Encrypted and VPN Traffic using Time-related Features, in : *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, INSTICC, SciTePress, 2016, 407-414, isbn : 978-989-758-167-0, https://doi.org/10.5220/0005740704070414.

Gohari, M., Hashemi, S. & Abdi, L., (2021), Android Malware Detection and Classification Based on Network Traffic Using Deep Learning, *2021 7th International Conference on Web Research (ICWR)*, 71-77, https://doi.org/10.1109/ICWR51868.2021.9443025

Gregorutti, B., Michel, B. & Saint-Pierre, P., (2017), Correlation and variable importance in random forests, *Statistics and Computing*, *27 3*, 659-678, https://doi.org/10.1007/s11222-016-9646-1

IANA, (2021), Protocol Numbers, Récupérée 7 août 2022, à partir de https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Lashkari, A. H., Kadir, A. F. A., Taheri, L. & Ghorbani, A. A., (2018), Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification, *2018 International Carnahan Conference on Security Technology (ICCST)*, 1-7, https://doi.org/10.1109/CCST.2018.8585560

Lashkari, A. H., Gil, G. D., Mamun, M. S. I. & Ghorbani, A. A., (2017), Characterization of Tor Traffic using Time based Features, *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1 : ICISSP,* 253-262, https://doi.org/10.5220/0006105602530262

Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K. & Xiao, Y., (2015), Network forensics analysis using Wireshark, *International Journal of Security and Networks*, *10 2*, 91-106, https://doi.org/10.1504/IJSN.2015.070421

Rosay, A. & Carlier, F., (2022), Intrusion Detection Evaluation Dataset (LYCOS-IDS2017), Récupérée 26 juillet 2022, à partir de https://lycos-ids.univ-lemans.fr/

Rosay, A., Carlier, F., Cheval, E. & Leroux, P., (2021), From CIC-IDS2017 to LYCOS-IDS2017 : A Corrected Dataset for Better Performance, *IEEE/WIC/ACM International Conference on Web Intelligence*, 570-575, https://doi.org/10.1145/3486622.3493973

Rosay, A., Cheval, E., Carlier, F. & Leroux, P., Network Intrusion Detection : A Comprehensive Analysis of CIC-IDS2017 [Best student paper award], in : *Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP,* Best student paper award, INSTICC, SciTePress, 2022, 25-36, isbn : 978-989-758-553-1, https://doi.org/10.5220/0010774000003120.

Sharafaldin, I., Lashkari, A. H., Hakak, S. & Ghorbani, A. A., (2019), Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy, *International Carnahan Conference on Security Technology (ICCST)*, 1-8, https://doi.org/10.1109/CCST.2019.8888419

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., (2018), Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, 108-116, https://doi.org/10.5220/0006639801080116

Taheri, L., Kadir, A. F. A. & Lashkari, A. H., (2019), Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls, *International Carnahan Conference on Security Technology (ICCST)*, 1-8, https://doi.org/10.1109/CCST.2019.8888430

Ullah, I. & Mahmoud, Q. H., (2020), A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks, In C. Goutte & X. Zhu (Éd.), *Advances in Artificial Intelligence* (p. 508-520), Springer International Publishing.

Chapitre 6

Approche système embarqué d'un IDS

« Prenez suffisamment de recul et votre travail vous apparaîtra plus petit, et vous pourrez plus facilement l'embrasser d'un seul regard et voir si l'harmonie ou les proportions de l'ensemble laissent à désirer. »

Léonard de Vinci
Philosophe (1452 – 1519)

Maintenant que nous avons montré dans les chapitres 4 et 5 qu'un réseau de neurones de taille modeste peut être implémenté pour classifier les différentes tentatives d'intrusions, nous pourrions continuer l'optimisation en déployant tout un panel de techniques utilisées dans les réseaux de neurones complexes. En effet, les réseaux neuronaux de classification ou de segmentation d'images à base de CNN mettent en œuvre des techniques de quantification des poids sous forme d'entiers 8 bits (Zhu et al., 2020), ou même sur 2 ou 1 bit (Kang et al., 2019 ; Rastegari et al., 2016 ; Redfern et al., 2021). D'autres techniques permettent d'obtenir des réseaux de neurones parcimonieux (Tartaglione et al., 2018), pour lesquels un grand nombre de poids est nul et pouvant être représentés par des matrices creuses, par exemple à l'aide de méthodes d'élagage (Ding et al., 2019). Toutes ces techniques présentent deux objectifs : d'une part, réduire l'empreinte mémoire des coefficients du réseau de neurones, et d'autre part, réduire la complexité et le temps d'inférence.

À première vue, le support de détection d'intrusions réseau sur un objet connecté aurait tout intérêt à tirer parti de ces méthodes d'optimisation. Néanmoins, les systèmes de détection d'intrusions réseau ne se limitent pas aux algorithmes de détection eux-mêmes. Ils font partie d'un tout, d'un système plus global qu'il convient de prendre dans son ensemble. En prenant le recul nécessaire, une approche holistique permet d'étudier notre domaine en décomposant la conception d'un système de détection d'intrusions dans la section 6.1, et d'identifier la partie du système devant être optimisé dans la section 6.2. Basé sur le constat qu'il est aussi primordial d'optimiser l'extraction des caractéristiques avant de les injecter dans le réseau de neurones, nous présentons notre proposition dans la section 6.3 et les résultats obtenus dans la section 6.4.

6.1 Conception du système de détection d'intrusions réseau

La détection d'intrusions réseau dans le domaine des IoT s'exécute sur un point d'observation du réseau, aussi bien dans une passerelle qu'au sein d'un nœud. La différence principale réside dans le fait qu'un nœud analysera uniquement les messages réseau le concernant afin de se protéger lui-même, tandis qu'une passerelle analysera tout le trafic réseau pour protéger l'ensemble des nœuds qui lui sont connectés.

D'un point de vue conceptuel, l'objet connecté est composé, comme indiqué sur la Figure 6.1, d'une pile de protocoles de communication dans le but de se connecter au réseau et d'une fonction primaire. Cette dernière gère des capteurs et actionneurs pour interagir avec son environnement dans le cas d'un nœud. Elle distribuera les messages réseau à d'autres nœuds dans le cas d'une passerelle.

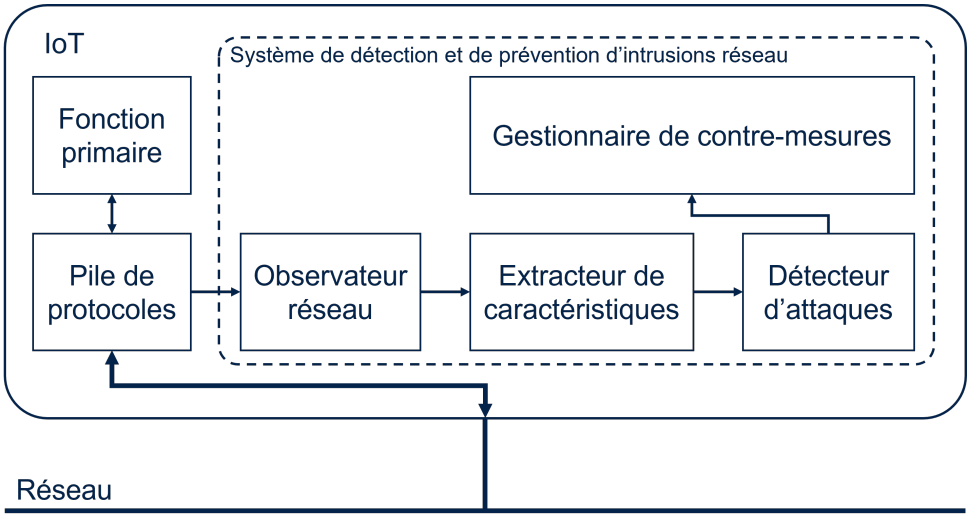


Figure 6.1 – Décomposition d'un objet connecté avec IDS.

À cela, s'ajoutent d'autres blocs pour la détection et éventuellement la prévention d'intrusions réseau :

— l'observateur réseau capture le trafic et extrait les données utiles pour le bloc suivant à chaque trame réseau reçue ou envoyée ;
— l'extracteur de caractéristiques prend les données issues de l'observateur réseau, les associe à une conversation et met à jour les différentes caractéristiques statistiques correspondant à cette conversation à chaque trame réseau reçue ou envoyée ;
— le détecteur d'attaques procède à la prédiction du trafic à chaque fois qu'une conversation est terminée, que ce soit par une fin de session de communication ou par l'expiration d'un délai prédéfini ;
— le gestionnaire de contre-mesure, présent seulement dans le cas d'un système de détection et de prévention d'intrusions réseau, décide des actions à prendre lorsqu'une

attaque est détectée, par exemple en bloquant le trafic d'une machine distante particulière.

L'observateur réseau et le gestionnaire de contre-mesures sont indépendants de la méthode de détection des intrusions. L'extraction des caractéristiques des trames échangées sur le réseau et la détection des attaques sont les éléments les plus importants et les plus complexes d'un système de détection d'intrusion réseau. Ainsi, pour fonctionner sur des dispositifs à ressources limitées, il est nécessaire de réduire la complexité de ces composants tout en maintenant de bonnes capacités de détection des attaques.

Contrairement à une approche basée sur des conversations, celle basée sur les paquets n'a pas besoin de l'extracteur de caractéristiques car l'ensemble du paquet est utilisé par le détecteur. Celui-ci devra être activé bien plus souvent, à chaque envoi ou réception d'un paquet. D'un côté, une telle approche réduit la complexité, mais augmente en contrepartie le nombre de traitements du détecteur d'attaques.

6.1.1 Extracteur de caractéristiques

Les traitements réalisés par ce bloc sont détaillés dans l'Algorithme 1 présenté en section 5.3.1. Deux points essentiels sont à retenir. D'une part, certaines des caractéristiques à extraire sont des valeurs statistiques devant être calculées. D'autre part, la gestion des conversations nouvelles, en cours et terminées est essentielle du point de vue des performances.

6.1.1.1 Calcul des caractéristiques

Le calcul des statistiques courantes est basé sur une technique incrémentale (Welford, 1962) évitant de garder un historique de toutes les valeurs et de résoudre à la fois un problème d'imprécisions dans les calculs en nombre flottants. Pour un ensemble de valeurs x_i avec $i \in [0; k]$, on définit m_n comme la moyenne des premiers éléments selon l'équation 6.1 et S_n comme la somme des carrés des différences par rapport à la moyenne actuelle selon l'équation 6.2 pour $n = 1, \ldots, k$.

$$m_n = \sum_{i=1}^{n} \frac{x_i}{n} \tag{6.1}$$

$$S_n = \sum_{i=1}^{n} (x_i - m_n)^2 \tag{6.2}$$

Ces formules de départ, sous forme de récurrence, deviennent les équations 6.3 et 6.4 :

$$m_n = m_{n-1} + \frac{x_n - m_{n-1}}{n} \tag{6.3}$$

$$S_n = S_{n-1} + \left(\frac{n-1}{n}\right) (x_n - m_{n-1})^2 \tag{6.4}$$

La formule 6.4 se simplifie selon l'équation 6.5 pour limiter les divisions par n.

$$S_n = S_{n-1} + (x_n - m_{n-1})(x_n - m_n) \tag{6.5}$$

Ainsi, à chaque instant, on retrouve facilement la moyenne $\mu = m_n$ et la variance $v = \frac{Sn}{n-1}$.

6.1.1.2 Gestion des conversations

L'Algorithme 1 montre que deux listes sont nécessaires. La première contient la liste de toutes les conversations en cours. Lorsqu'un paquet est reçu, ses identifiants sont utilisés pour vérifier si la conversation est d'ores et déjà en cours. Si ce n'est pas le cas, une liste des conversations terminées utilisant le protocole TCP est également inspectée dans le but de vérifier si le paquet reçu appartient ou non à la fin d'une session TCP.

Dans le cas de certaines attaques du type DoS, DDoS et Portscan qui inondent leur cible de paquets, ces recherches dans les listes peuvent être chronophages.

6.1.2 Détecteur d'attaques

Cette partie du système a déjà été largement couverte dans le chapitre 5 avec un ensemble d'algorithmes, leurs performances de détection, mais aussi leurs temps d'inférence. Nous n'analyserons pas plus en détail le détecteur. Toutefois, nous sélectionnons le MLP et gardons à l'esprit que le temps d'inférence du réseau de neurones pourra être amélioré sur les objets connectés possédant des accélérateurs matériels si nécessaire.

6.2 Goulet d'étranglement : la recherche des conversations

Les systèmes de détection d'intrusions sont classiquement implémentés sur des machines puissantes. Dans notre cas, nous cherchons à réaliser un tel système sur un objet connecté disposant de ressources limitées. Il est nécessaire de procéder à une analyse de performance du système qui passe généralement par une identification d'un ou plusieurs goulets d'étranglement. En effet, les capacités de l'ensemble du système sont déterminées par la capacité maximale du sous-système le plus faible. Le goulet d'étranglement correspond au sous-système limitant la capacité du système complet. En conséquence, les efforts d'améliorations doivent commencer par l'identification du goulet d'étranglement (Tang, 2019).

Sa recherche est une étape importante pour obtenir un système de détection d'intrusions réseau. Si le temps d'inférence est significativement plus long que le temps nécessaire à l'extraction des caractéristiques, il faut optimiser le calcul du réseau neuronal, soit en le simplifiant, soit en utilisant des techniques de quantification et d'élagage. À l'inverse, si

l'extraction des caractéristiques demande davantage de temps que l'inférence, il faut alors optimiser l'extracteur.

Pour procéder à cette recherche, une première analyse est menée avec le modèle de MLP défini dans le chapitre 4 avec le corpus numérique CIC-IDS2017. Le comportement de l'extracteur de caractéristiques serait identique avec notre corpus amélioré. En effet, la charge du système embarqué dépend principalement du nombre de paquets à traiter. Ensuite, le modèle optimisé du chapitre 5 est utilisé avec un système embarqué moins performant pour vérifier le fonctionnement global de l'IDS.

6.2.1 Plateforme d'expérimentation

Pour une première recherche du goulet d'étranglement, un système de détection d'intrusions est déployé sur une carte électronique basée sur un système sur puce (SoC) de la société STMicroelectronics. Notre IDS est conforme à la Figure 6.1 et les blocs d'extraction de caractéristiques et de détection d'attaques sont implémentés chacun dans une tâche.

Le SoC s'appelle TelemaCo3P et fournit une solution pour assurer une connexion entre le véhicule et des serveurs distants. Son architecture multi-cœur asymétrique fournit de puissants processeurs d'application (deux Cortex-A7) et un sous-système de contrôle CAN indépendant basé sur un Cortex-M3. Sa qualification de qualité automobile en fait un bon candidat pour la mise en œuvre d'un large éventail d'applications télématiques sécurisées prenant en charge une connectivité sans fil à haut débit et pouvant nécessiter une détection des intrusions dans le réseau. Un exemple de solution typique est représenté sur la Figure 6.2.

Telemaco3P contient deux cœurs ARM Cortex-A7 à 600 MHz avec NEON (Smith, 2020) ainsi que des extensions d'unités à virgule flottante. Ce SoC n'embarque pas d'accélérateur matériel dédié aux réseaux de neurones, mais permet toutefois d'analyser les performances d'un système embarqué existant correspondant au cas d'utilisation visé. Le système d'exploitation Linux sert de base au logiciel embarqué.

En outre, comme les processeurs ARM sont les mêmes que ceux embarqués sur une carte Raspberri Pi (Johnston et al., 2018) et que ces cartes sont utilisées dans le mur d'écrans, notre étude est applicable dans le contexte de nouveaux dispositifs numériques pour l'éducation.

6.2.2 Extracteur de caractéristiques

L'algorithme 1 est implémenté en langage C et la gestion des conversations est implémentée avec des listes chaînées.

L'extracteur devant garder la trace de toutes les conversations en cours ou récemment terminées, les besoins en traitement et en mémoire varient en fonction du trafic. Ce problème est illustré par la Figure 6.3.

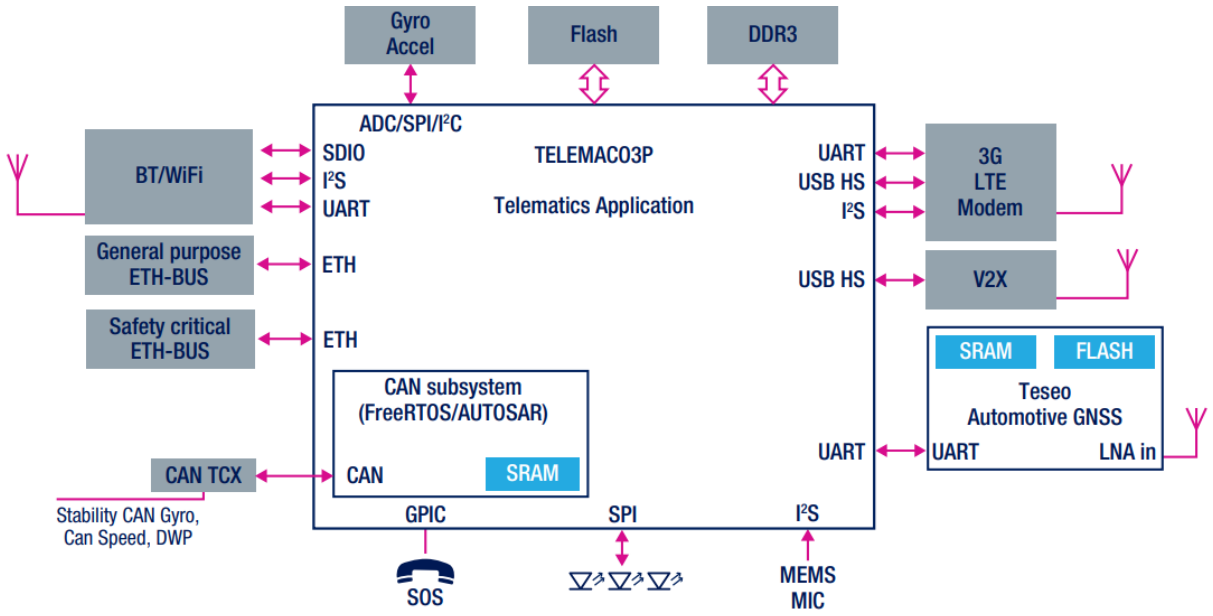


FIGURE 6.2 – Exemple de solution basée sur Telemaco3P.

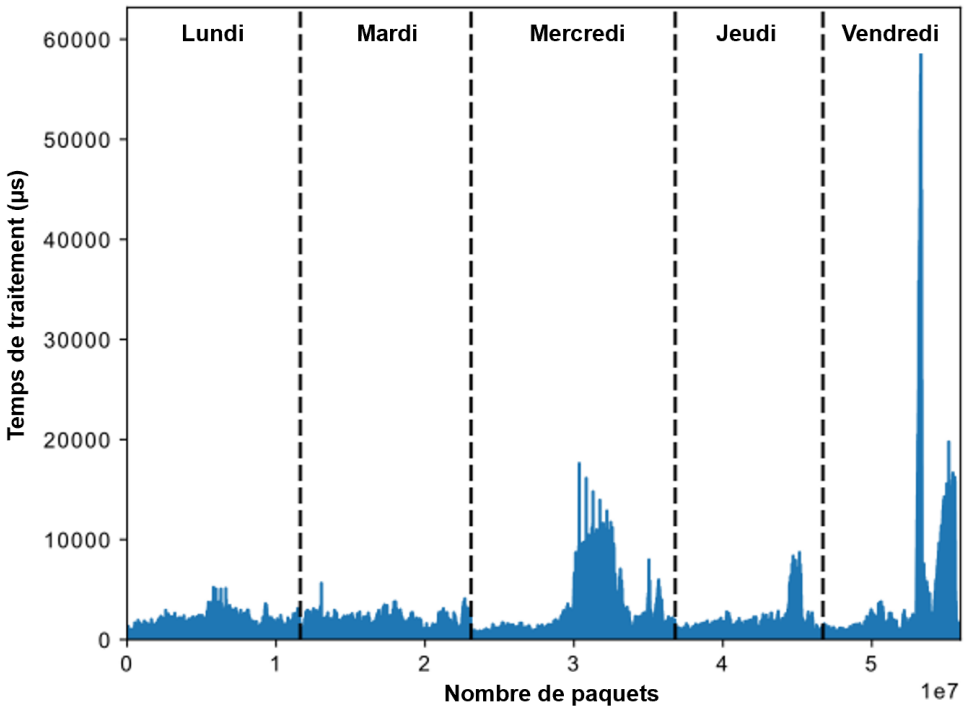


FIGURE 6.3 – Temps de traitement de l'extracteur de caractéristiques.

Cette figure montre le temps d'exécution de l'Algorithme 1 implémenté sur un seul cœur de Telemaco3P, fonctionnant à une fréquence constante pour l'ensemble des données brutes du corpus CIC-IDS2017.

Le temps de traitement du lundi démontre quelques variations correspondant à une utilisation normale du réseau et peut être utilisé comme référence pour les autres jours. Le mardi contient des attaques visant à craquer un mot de passe, soit pour FTP, soit pour SSH. Le temps de traitement est très similaire à celui du jour de référence. Lors de ces attaques, une connexion TCP est lancée et en cas de mot de passe erroné, la session TCP est terminée. Par conséquent, le nombre de flux n'augmente pas de manière significative pendant les attaques. Les trois autres jours contiennent des pics correspondant aux attaques DoS (mercredi), aux attaques DDoS (vendredi) et au scan de port (jeudi et vendredi). Pendant ces attaques DoS et DDoS, nous observons un nombre toujours croissant de conversations ouvertes tandis que le scan de port vérifie de nombreuses adresses IP et ports différents et génère donc de nombreuses conversations différentes. Celles-ci se terminent uniquement lorsque le délai d'attente expire. Lorsqu'un paquet est reçu pendant les attaques, l'algorithme doit vérifier de nombreuses conversations afin de vérifier s'il correspond à une conversation existante, expliquant l'augmentation considérable du temps de traitement.

En moyenne, l'analyse des paquets est exécutée en $\sim$127 µs, mais peut prendre jusqu'à plusieurs dizaines de millisecondes lors de certaines attaques.

6.2.3 Détecteur d'intrusions

Comme le système d'exploitation est basé sur Linux, il est possible d'utiliser des scripts Python et d'utiliser TensorFlow-Lite (TF-Lite), une version de TensorFlow (TF) dédiée aux calculs d'inférence (Geron, 2019) pour implémenter le détecteur d'intrusions. Bien que TF-Lite ne supporte qu'un sous-ensemble des opérations de TF, toutes celles nécessaires à notre modèle sont supportées. Comme le format des nombres en virgule flottante de TF-Lite est sur 32 bits, notre modèle de réseau de neurones a été entraîné en simple précision au lieu de double précision, format utilisé par défaut en Python. Même si ce n'est pas l'objet ici, nous notons que ce changement ne conduit pas à un changement significatif des capacités de détection des intrusions.

L'inférence est exécutée avec un script Python dans un seul thread et charge donc un seul cœur. TF-Lite utilise les bibliothèques d'algèbre linéaire disponibles dans le système Linux et optimisées pour utiliser l'accélérateur vectoriel NEON (Smith, 2020). Le temps de traitement moyen pour une seule inférence est de 1,08 ms.

6.2.4 Analyse

Deux aspects ressortent des expérimentations qui ont été menées et méritent une analyse plus poussée.

D'une part, la quantité de mémoire et les temps de traitements nécessaire pour l'extracteur de caractéristiques augmentent considérablement lors des attaques de type déni de service et scan de port. Le temps d'exécution est bien évidemment un problème, mais au-delà de cet aspect, l'extracteur et par extension le système de détection d'intrusions peut devenir la victime d'un déni de service. Ce serait un comble pour un système cherchant à protéger un appareil ou un réseau de ce type d'attaques.

D'autre part, nous remarquons que le temps d'inférence est d'un ordre de grandeur plus grand que le temps moyen d'analyse des paquets. Toutefois, les inférences se font sur les conversations et non sur les paquets. Dans le cas des conversations UDP, l'inférence n'a lieu qu'à l'expiration d'un délai de 120 s, temps largement supérieur au temps de calcul de l'inférence. Pour toute conversation TCP composée de plus de neuf paquets, le temps d'inférence est inférieur au temps moyen d'extraction des caractéristiques. En gardant en tête qu'une connexion TCP possède généralement sept paquets pour l'établissement et la fermeture de la session, il ne reste plus qu'un paquet de donnée et un paquet d'accusé de réception pour compléter une conversation de neuf paquets. Enfin, avec l'utilisation du corpus LYCOS-IDS2017, le réseau de neurones est considérablement réduit avec un MLP comportant 1 361 poids au lieu des 87 823 poids utilisés pour le calcul du temps d'inférence de 1.08 ms. La réduction de ce temps d'inférence est moins critique que celui de l'extracteur de caractéristiques. En conséquence, notre analyse se focalisera sur le goulet d'étranglement le plus important : l'extraction des caractéristiques.

6.2.4.1 Limitations dues aux structures de données

Le nombre de conversations à gérer dans l'extracteur de caractéristiques est impossible à déterminer *a priori* de manière réaliste. L'exemple commun consiste à utiliser une combinaison des adresses IP source et destination, des ports de transport source et destination et du protocole de transport comme clef d'identification d'un groupe de paquet. Cet exemple conduit à une clef de 104 bits et donc 2^{104} combinaisons. À titre de comparaison, un disque dur de 20 To, une des tailles les plus grandes en 2022, ne contiendrait qu'une infime partie des combinaisons possibles. Dans une implémentation logicielle, afin de garder un maximum de flexibilité et ne pas réserver inutilement de la mémoire, il convient de stocker les conversations dans une structure de données dynamique plutôt que dans une structure de taille statique.

La structure de données choisie pour notre implémentation est une liste chaînée. Bien que nous ayons tous une compréhension intuitive de ce qu'est une liste, il peut être utile de se référer au concept de liste chaînée utilisée en informatique (Shaffer, 2011, section 4.1.2). Ce type de structure utilise des allocations dynamiques de mémoire. La liste est composée de nœuds contenant un élément dont la structure peut être complexe et un pointeur vers le nœud suivant. Dans le cas d'une liste doublement chaînée, le nœud possède également un pointeur vers l'élément précédent.

Afin d'optimiser l'insertion d'un élément, il est possible de l'ajouter systématiquement en tête ou en queue de la liste. La recherche d'un nœud est moins optimale. Comme

indiqué sur l'exemple fourni par la FIGURE 6.4, la recherche d'un nœud particulier, ici contenant la valeur 79, peut nécessiter le parcours complet de la liste chaînée. En se

FIGURE 6.4 – Recherche d'un élément dans une liste chaînée.

référant à l'analyse asymptotique (SHAFFER, 2011, p. 65-66), le temps de recherche d'un élément est en $\mathcal{O}(n)$. Quand le nombre n de nœud augmente, le temps maximum de recherche augmente de façon proportionnelle à n.

Les listes chaînées peuvent être ordonnées ou non. Dans le cas de la détection d'intrusions, la clef d'identification d'une conversation est composée d'éléments hétérogènes avec deux adresses IP, représentées par exemple sous la forme décimale 205.174.165.73, de deux ports et d'un identifiant de protocole IP. Une telle clef ne possède pas de relation d'ordre (notion clarifiée dans la section 6.3) et la liste ne peut pas être ordonnée.

Il est particulièrement intéressant de remarquer qu'une attaque comme le Portscan fait varier la clef d'identification de la conversation pour chaque port testé. Lorsque l'Algorithme 1 de la section 5.3.1 recherche si la conversation existe déjà, il ne la trouvera pas et parcourra la totalité de la liste chaînée avant de décider la création d'une nouvelle conversation. Ainsi, nous passons à de multiples reprises dans le pire cas en termes de temps d'exécution. Ce comportement est conforme à celui que nous observons sur la FIGURE 6.3. Cette limitation doit être levée pour diminuer le temps passé à rechercher les conversations.

6.2.4.2 Limitations dues au temps d'inférence

Les paquets Ethernet étant regroupés en conversation, la classification des conversations n'est nécessaire que lorsqu'une conservation est terminée. La comparaison du temps d'inférence avec le temps nécessaire à l'extraction des caractéristiques et à la mise à jour des métadonnées n'est pas immédiat. Ce point est d'autant plus vrai qu'il dépend de la taille des paquets Ethernet. Pour un débit donné, le nombre de paquets maximum à traiter augmente lorsque leur taille diminue. Il est usuel, notamment dans l'industrie automobile, d'utiliser une distribution particulière pour la taille de paquet, appelée Internet Mix, profils de trafic réseau sur Internet (IMIX) (WIKIPEDIA, 2017). Des distributions alternatives de tailles de paquets sont proposées dans les outils de test de réseau (AGILENT TECHNOLOGIES, 2007, p. 1-9).

Dans le monde de l'automobile, il n'y a pas de consensus sur ce type de distribution à cause d'une dépendance trop forte à l'architecture électrique et électronique des véhicules. En effet, certains constructeurs estiment qu'une quantité importante de gros paquets sera utilisée, par exemple pour remonter des informations brutes à une unité de calcul centrale

ou pour faire des mises à jour des programmes des boitiers électroniques. Au contraire, d'autres constructeurs considèrent qu'Ethernet servira pour transférer une multitude de petits paquets correspondant à des messages CAN, limités en taille par nature. Dans tous les cas, la distribution *simple IMIX* reste intéressante pour estimer le coût de calcul des inférences dans un système fonctionnant dans un mode normal, sans attaque. Si ce coût devait être important, le système embarqué devrait être dimensionné en conséquence et potentiellement, sa viabilité économique pourrait être discutée.

Dans cette section, nous utiliserons une distribution appelée *Simple IMIX*, le cas le plus favorable avec seulement des paquets de taille maximale ainsi que le plus défavorable avec des paquets de taille minimale. Les informations sur ces distributions sont données dans la Table 6.1.

Table 6.1 – Distribution des paquets Ethernet utilisés pour notre analyse.

Cas	**Taille des paquets en octet**	**Nombre de paquets**	**Distribution en %**	**Taille moyenne des paquets en bit**
Simple IMIX	40	7	58,22	2 722,7
	576	4	33,33	
	1 500	1	8,33	
Taille max de paquets	1 500	1	100,00	12 000
Taille min de paquets	40	1	100,00	320

Par des calculs à partir de la Table 6.1, un lien Ethernet à 100 Mbps utilisé à 100 % de sa capacité transportera 36 729 paquets/s (pkt/s), en considérant les tailles moyennes de paquets du cas *simple IMIX*.

La Table 6.2 fournit un calcul de la charge du processeur pour réaliser les inférences avec TF-Lite en fonction du nombre de paquets par conversation (pkt/conv).

Table 6.2 – Charge CPU estimée pour différents cas de conversations sur TelemaCo3P.

	pkt/s	**charge CPU (%)**		
		6 pkt/conv	20 pkt/conv	30 pkt/conv
Simple IMIX	36 729	661,12	198,22	132,22
Taille max de paquets	8 333	5 625,00	45,00	30,00
Taille min de paquets	312 500	150,00	1 687,50	1 125,00

La Table 6.2 fait clairement apparaître des cas de charge supérieure à 100 %. Avec un temps d'inférence de 1.08 ms, seules de longues conversations composées de paquets de grandes tailles peuvent être classées sans engorgement du CPU.

En considérant un lien Ethernet à 1 000 Mbps, la situation se dégrade d'un facteur dix et l'engorgement a lieu dans tous les cas. Dans le cas des cyberattaques, il est commun d'avoir des conversations composées de petits paquets. Ceci justifie pleinement le besoin d'avoir des temps d'inférence plus courts.

Pour résumer cette section, deux goulets d'étranglement ont été identifiés avec des importances relatives. L'extraction des caractéristiques est le problème le plus important pour éviter que le détecteur soit victime de déni de service et ramener le temps d'extraction dans des bornes réduites. Lorsque l'extracteur sera amélioré, nous devrons alors réduire le temps passé dans le détecteur d'intrusions en accélérant les calculs d'inférence de façon à ce que le temps d'inférence soit inférieur au temps moyen d'extraction des caractéristiques. Seul le problème le plus important a été étudié et l'analyse montre que l'utilisation de liste chaînée est inadaptée à la détection d'intrusions basée sur des conversations.

6.3 Proposition

L'analyse explicitée dans la section 6.2.4 nous indique la voie à suivre : la recherche des conversations ne doit pas se faire dans une structure de donnée dont le temps de recherche est en $\mathcal{O}(n)$. Il est nécessaire de se tourner vers des structures répondant à deux critères : d'une part, permettre une gestion dynamique de la mémoire, et d'autre part, avoir des temps de recherche et d'insertion courts.

6.3.1 Clarification de l'objectif

Parmi les structures de données permettant une gestion dynamique, beaucoup se basent sur la nécessité de suivre des liens vers les nœuds précédents ou suivants. Contrairement à ces structures, la table de hachage sort du lot en permettant un accès direct à l'élément, permettant ainsi un temps de recherche en $\mathcal{O}(1)$. Le concept d'une telle table repose sur une fonction de hachage permettant la conversion de la clef du nœud à chercher en un index de tableau. Le résultat de la fonction de hachage étant généralement plus grand que l'index maximum du tableau, la conversion en index se fait modulo la taille de la table.

Le hachage comme le modulo peuvent conduire à des collisions, avec deux clefs différentes converties en un même index. Dans un tel cas, la donnée correspondant à cet index est généralement associée à une liste chaînée. Lorsque la table de hachage atteint un seuil de remplissage, la taille de la table est doublée et les conversions des clefs vers les index sont recalculées pour prendre en compte ce changement de taille.

Le temps d'insertion est rapide excepté au moment où la taille de la table doit être doublée et en conséquence en $\mathcal{O}(1)$ en temps amorti. Dans un système de détection d'intrusion, un blocage dû au pic d'activité lié au doublement de la table de hachage est d'autant plus problématique que ce temps augmente à chaque fois que la table est agrandie.

D'autre part, dès que la taille de la clef devient supérieure au plus grand entier pouvant être contenu dans un mot du processeur (en général entre 8 et 32 bits sur les systèmes embarqués), les calculs nécessaires dans les fonctions de hachage deviennent inefficaces (Mehta & Sahni, 2004, p. 118). Avec une clef d'identification de conversation contenant deux adresses IP, deux ports et un protocole, un processeur 32 bits ne permettraient pas une implémentation optimale.

Dans un système puissant, par exemple sur un serveur, ces pics d'activité et d'inefficacité ne sont pas des problèmes bloquants - une table de hachage est d'ailleurs utilisée dans l'outil CICFlowMeter - mais le deviennent dans un système embarqué contraint. Pour ces raisons, l'utilisation de table de hachage n'est pas la solution à privilégier.

À défaut d'une structure de données permettant un temps de recherche en $\mathcal{O}(1)$, nous devons considérer les algorithmes avec des temps de recherche en $\mathcal{O}(\log n)$. Parmi ces algorithmes, nous retrouvons les arbres binaires équilibrés, par exemple l'arbre B ou l'arbre rouge-noir (Mehta & Sahni, 2004) mais aussi des structures de données probabilistes comme les listes à enjambements. Ce type de listes possède des propriétés similaires aux arbres binaires équilibrés et leur simplicité les rend généralement plus simple à implémenter (Pugh, 1990b).

Ces algorithmes ont tous un point commun : il nécessite de travailler sur un ensemble possédant une relation d'ordre selon la définition 6.

Définition 6: Relation d'ordre

Soit E un ensemble. Une relation binaire $\mathcal{R}$ dans E est une relation d'ordre si elle est :

- réflexive : $\forall x \in E, x\mathcal{R}x$;
- antisymétrique : $\forall x \in E, \forall y \in E, (x\mathcal{R}y \text{ et } y\mathcal{R}x) \Rightarrow x = y$;
- transitive : $\forall x \in E, \forall y \in E, \forall z \in E, (x\mathcal{R}y \text{ et } y\mathcal{R}z) \Rightarrow x\mathcal{R}z$.

À titre d'exemple, les relations binaires $\leq$, $\geq$, $=$ sont des relations d'ordre sur les ensembles $\mathbb{N}$, $\mathbb{R}$. Lorsque l'ordre est total, $\forall x \in E, \forall y \in E, (x\mathcal{R}y \text{ ou } y\mathcal{R}x)$, la relation d'ordre indique que deux éléments de E sont toujours comparables. Une relation d'ordre est stricte lorsqu'elle est irréflexive et transitive. Par exemple, dans l'ensemble $\mathbb{R}$, la relation $<$ est une relation d'ordre strict et total, $\forall x \in E, \forall y \in E, (x<y \text{ ou x=y ou } y < x)$.

Par nature, leurs clefs d'identification des conversations sur un réseau informatique ne sont pas ordonnées et par défaut, nous ne pouvons pas utiliser ces structures de données proposant des temps de recherche en $\mathcal{O}(\log n)$. La proposition que nous adressons dans cette section consiste à définir une solution de création de clefs d'identification permettant d'établir une relation d'ordre strict et total sur un ensemble de conversations réseau. Cette définition nous permettra alors d'utiliser des structures de données avec des temps de recherche suffisamment rapide pour éviter l'effet démontré sur la Figure 6.3.

6.3.2 Possibilités de création de clefs d'identification

Le gain de temps est un élément capital pour réaliser un système de détection d'intrusions réseau sur un système embarqué. Plusieurs façons de créer les clefs d'identification permettant d'utiliser une relation d'ordre sont définies et quelques exemples d'implémentations sont couverts afin de protéger le concept de la façon la plus large possible au travers d'un brevet déposé (Rosay et al., 2022).

Un dépôt de brevet est un investissement économique important. Afin que la protection apportée par celui-ci soit maximale, toute description en dehors du brevet doit éviter de donner la moindre piste, si elle existe, de contourner ses revendications. Dans ce document, nous exposons les éléments principaux, utiles à la compréhension de la solution proposée.

Ce brevet couvre, entre autres, une méthode comprenant la réception et transmission de données sur un réseau, le calcul d'une clef d'identification, la recherche en mémoire dans des structures de données sur des ensembles ordonnés avec une gestion dynamique de la mémoire, ainsi que la création ou la mise à jour de méta-données dans les structures de données identifiées par une clef.

Dans sa forme la plus simple et la plus générique, l'identifiant est généré par concaténation des valeurs des caractéristiques de la conversation, que nous nommerons champs dans le contexte de ce chapitre. Un exemple d'implémentation comprend un registre à décalage configuré pour recevoir en entrée une pluralité de champs reçus d'un paquet de façon sérielle et pour sortir une valeur sur plusieurs bits en parallèle formant la clef d'identification. Pour une autre implémentation, la clef est calculée de façon logicielle sur un processeur.

Dans une forme plus complexe, l'identifiant peut être généré par un circuit de traitement qui appliquera une fonction mathématique prenant comme antécédents les valeurs des champs. Dans un autre exemple d'implémentation, la fonction mathématique est appliquée en pré-traitement des champs qui peuvent être concaténés.

Une variante consiste non pas à créer une clef d'identification, mais deux clefs distinctes qui permettront dans le cas de conversations bidirectionnelles (Trammell & Boschi, 2008) de discriminer les conversations entrantes et sortantes.

La clef d'identification doit permettre une relation d'ordre. Bien que la clef puisse être décrite de manière générale, il semble judicieux pour une implémentation dans un système embarqué que cette clef soit représentée sous la forme d'un entier ou d'un nombre réel. Les relations binaires $>$ et $=$ dans $\mathbb{N}$, $\mathbb{R}$ permettent de classer les conversations, entre autres, dans des arbres binaires équilibrés ou des listes à enjambements.

Afin de rendre concret la forme la plus simple de la création d'une clef d'identification, deux exemples sont fournis combinant des champs en une valeur entière. Ces exemples, sous forme algorithmique, illustrent l'utilisation des champs conformément à la description de conversations Ethernet (Claise et al., 2013). L'Algorithme 3 crée la clef à partir d'une simple concaténation des adresses IP source et destination, des ports de transport source et destination et du protocole de transport, obtenue par des opérations de décalage et des fonctions OU logique. L'Algorithme 4 utilise les mêmes principes de création de clef mais en prenant, d'une part, un ordre différent des champs, et d'autre part, des adresses source et destination réduites par exemple pour limiter la liste des conversations à un sous-réseau.

Une multitude de variantes sont possibles, avec des implémentations purement logicielles ou purement matérielles, ou encore une combinaison des deux. Les champs utilisés

Algorithme 3 : Exemple 1 de création de clef d'identification.

Input : sAddr : 32-bit value encoding source IPv4 addresses
Input : dAddr : 32-bit value encoding destination IPv4 addresses
Input : sPort : 16-bit value encoding source ports
Input : dPort : 16-bit value encoding destination ports
Input : prot : 8-bit value encoding IP Protocol
Result : key : 104-bit integer value encoding flow indentifier

let *key* be the generated value by concatenation of the different inputs

$key \leftarrow prot << 96 \mid (dPort << 16 \mid sPort) << 64 \mid dAddr << 32 \mid sAddr$

Algorithme 4 : Exemple 2 de création de clef d'identification.

Input : sAddr : 32-bit value encoding source IPv4 addresses
Input : dAddr : 32-bit value encoding destination IPv4 addresses
Input : sPort : 16-bit value encoding source ports
Input : dPort : 16-bit value encoding destination ports
Input : prot : 8-bit values encoding IP Protocol
Result : key : 56-bit integer value encoding flow indentifier
Data : sAddr8b : 8-bit value encoding least significant bits of sAddr
Data : dAddr8b : 8-bit value encoding least significant bits of dAddr

let *key* be the generated value by concatenation of a part of the different inputs and short versions of IP addresses

$sAddr8b \leftarrow$ eight least significant bit of $sAddr$
$dAddr8b \leftarrow$ eight least significant bit of $dAddr$
$key \leftarrow (dPort << 16 \mid sPort) << 40 \mid dAddr8b << 16 \mid sAddr8b << 8 \mid prot$

pour la création de la clef, comme pour le contenu des méta-données peuvent également varier. La liste des types de structure de données nécessitant des ensembles ordonnés n'est pas exhaustive et la description du brevet couvre un panel de variations.

Cette proposition de création de clef d'identification permettant d'utiliser une relation d'ordre nécessite d'être expérimenté afin de rendre visible les bénéfices de notre apport.

6.4 Méthodologie d'expérimentations et résultats

Afin de valider notre approche, nous devons valider les deux critères énoncés en section 6.3 : permettre une gestion dynamique de la mémoire et avoir des temps de recherche et d'insertion courts en $\mathcal{O}(\log n)$ grâce à une relation d'ordre.

Le premier critère est immédiat dans le sens où les algorithmes comme les arbres binaires équilibrés ou les listes à enjambements utilisent par nature une gestion dynamique de la mémoire (Mehta & Sahni, 2004 ; Pugh, 1990b).

Pour expérimenter notre proposition, nous procédons en deux étapes. Dans un premier temps, nous travaillons uniquement sur les structures de données en utilisant une clef composées des champs habituels de la détection d'intrusion réseau, à savoir les adresses IP source et destination, des ports de transport source et destination et du protocole de transport afin de caractériser les temps d'insertion et de recherche sur un ordinateur de bureau utilisant un processeur 64 bits. La clef, comme les données qui lui sont liées, sont composées chacune de quatre mots de 32 bits. Une quantité de 524 288 éléments est placée dans les structures de données pour les insertions et recherches.

Dans un deuxième temps, nous validons notre proposition par une première implémentation dans un système embarqué testée avec le corpus CIC-IDS2017, le même que celui utilisé lors de l'étude des goulets d'étranglement, en section 6.2. La journée du vendredi étant la plus critique en termes de temps de recherche des conversations, une comparaison de l'implémentation de départ à base de liste chaînée avec un algorithme correspondant à notre proposition est présentée afin de montrer le gain obtenu sur cette journée. Enfin, une deuxième implémentation sur un microcontrôleur valide notre proposition pour des objets connectés ayant de plus fortes contraintes en ressources.

6.4.1 Études des performances de différentes structures de données

L'objectif est d'investiguer les performances de la liste chaînée, référence de notre implémentation de départ. La table de hachage est utilisée pour confirmer l'analyse mentionnée dans la section 6.3.1, en particulier pour montrer que le temps de recherche est très court, mais que le temps d'insertion d'un élément peut être très élevé lorsque la table est agrandie. Enfin, nous étudions un algorithme nécessitant une relation d'ordre. Pour cela, nous choisissons la liste à enjambements pour sa simplicité d'implémentation et l'absence de pic de traitement que nous trouvons dans les arbres binaires équilibrés.

Avant de présenter les résultats des expérimentations, prenons le temps de décrire plus en détails les algorithmes et les tests.

6.4.1.1 Liste chaînée

Le principe de la liste chaînée a déjà été expliqué en section 6.2.4 et le code utilisé pour cette étude est le même que celui utilisé dans les expérimentations de la section 6.2.1. Il s'agit d'une liste simplement chaînée pour laquelle l'insertion se fait toujours en tête. Cette structure de données est tout à fait classique et ne mérite pas de s'y attarder davantage.

6.4.1.2 Table de hachage

Une table de hachage est une table dont la taille varie dynamiquement. Les données stockées à l'intérieur sont en accès direct grâce à un index dérivé d'une clef. Le principe de fonctionnement d'une table de hachage est donnée par la FIGURE 6.5.

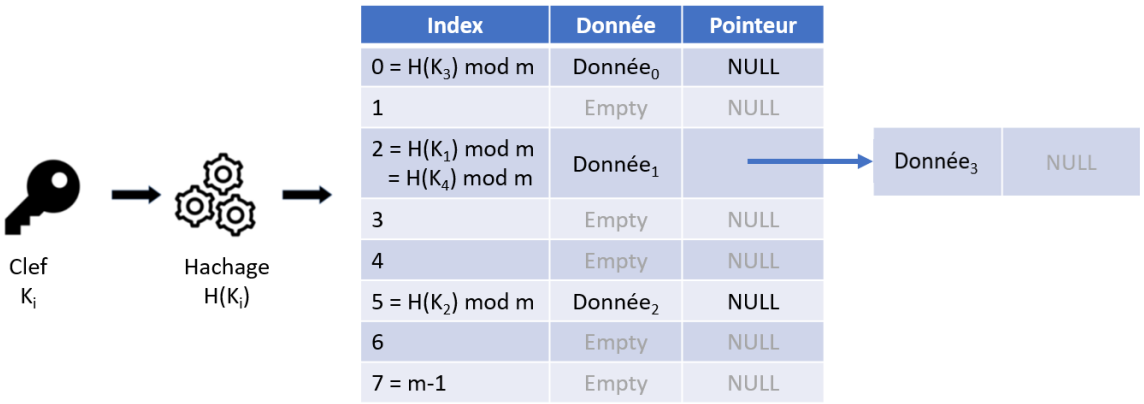


FIGURE 6.5 – Description d'une table de hachage.

Dans cet exemple, les clefs d'identification des données K_i appartenant à l'ensemble des clefs $\mathcal{K}$ sont fournies en entrée d'une fonction de hachage qui, modulo la taille de la table, donne l'index où placer la donnée. Même si nous cherchons à utiliser des fonctions de hachage générant le moins de collisions possibles, en pratique, et en particulier quand un ensemble très grand de clefs est projeté sur un ensemble beaucoup plus petit, des collisions apparaissent. Une illustration avec $K_1 \neq K_4$ et $H(K_1) = H(K_4)$ apparaît sur la FIGURE 6.5. Dans ce cas, une liste chaînée peut être utilisée pour associer des données différentes à un index unique de la table.

Pour nos expérimentations, la taille initiale m de la table est de seize entrées et se trouve doublée à chaque fois que le taux de remplissage de la table atteint 75 %. Nous avons sélectionné une fonction de hachage multiplicative H décrite par l'équation 6.6, avec 2^λ le nombre de clefs possibles, $m = 2^\mu$ la taille de la table de hachage et respectant $1 \leq \mu \leq \lambda$ et $0 < a < 2^\lambda$.

$$H : \begin{array}{l} [0, 2^\lambda - 1] \to [0, 2^\mu - 1] \\ k \mapsto \left(aK \mod 2^\lambda\right) / 2^{\lambda-\mu} \end{array} \tag{6.6}$$

Même si toute valeur de $a \in [1, 2^\mu - 1]$ peut être utilisée, il est judicieux de choisir a impair afin que la fonction $H(K)$ génère peu de collision. Dans ce cas particulier, $\forall K_1 \neq K_2$, la probabilité $P\left(H\left(k_1\right) = H\left(K_2\right)\right)) \leq \frac{2}{m}$ (Dietzfelbinger, 1996, p. 570).

Outre la faible probabilité de collision quand la table est grande, cette fonction de hachage offre l'avantage d'être facile à calculer en invoquant uniquement des opérations de multiplication et de décalage.

6.4.1.3 Liste à enjambements

Les listes à enjambements, aussi connues sous leur appellation anglaise de *skip lists*, sont des structures de données probabilistes partageant des propriétés avec les listes chaînées, tout en diminuant fortement leurs défauts. Dans cette section, nous privilégierons l'utilisation du nom anglais pour éviter d'éventuelles confusions entre les listes chaînées et à enjambements. En l'absence de précision, une liste correspond à une liste chaînée.

Bien que ce type de structure soit ancienne (Pugh, 1990a, 1990b), elle reste assez méconnue et nécessite quelques explications.

Une *skip list* est un ensemble E d'items distincts composés d'une paire (clef, élément) et possède une série de listes chaînées $L_0, L_1, \ldots, L_h$ telles que :

- chaque liste contient deux clefs spéciales correspondant aux valeurs minimale et maximale, par exemple $-\infty$ et $+\infty$;
- la liste L_0 contient tous les items de E dans l'ordre croissant des clefs ;
- chaque liste est un sous-ensemble de la liste précédente : $L_h \subset \cdots \subset L_1 \subset L_0$;
- la liste L_h contient seulement les deux clefs spéciales correspondant aux valeurs minimale et maximale, par exemple $-\infty$ et $+\infty$.

Un exemple d'une telle structure est représentée sur la Figure 6.6 avec la simplification que seule la clef est représentée.

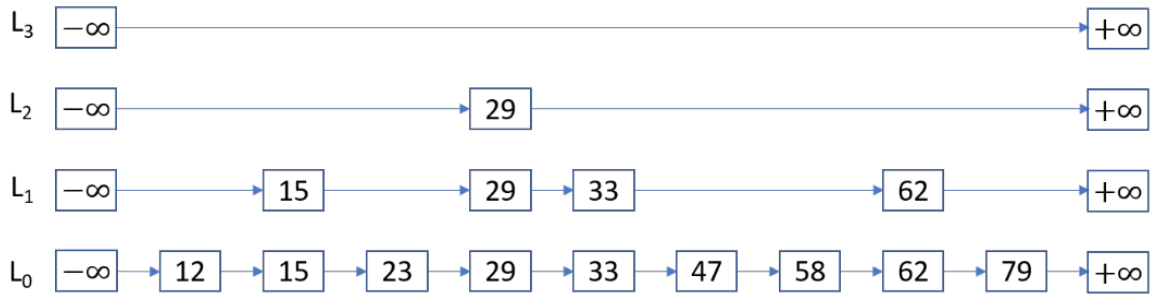


Figure 6.6 – Exemple de *skip list*.

Pour illustrer le fonctionnement d'une *skip list*, imaginez-la comme un système de transport en commun idéal où le temps d'attente pour les correspondances est nul. Un nœud de la *skip list* est un arrêt obligatoire auquel un passager peut changer de mode de transport. La couche la plus basse de la *skip list* peut être interprétée comme un bus. Il s'arrête à chacun de ses arrêts prévus, son temps de trajet est donc très lent. La couche L_1 est plus rapide et est comparabe à un métro tandis que la couche L_2 est un train. Si un passager veut aller du Mans à un concert à Paris, il ne va pas faire le trajet en bus. Il est naturel et plus rapide dans ce cas de prendre d'abord le train, puis le métro afin de se rapprocher le plus possible du concert. Si le dernier arrêt de métro ne permet pas d'accéder à la salle de concert, dans ce cas seulement, le voyageur doit prendre le bus.

La recherche d'une clef dans la *skip list* est illustrée par la FIGURE 6.7 avec le chemin pour retrouver la clef valant 79. Nous parcourons la *skip list* depuis la clef minimale de la liste chaînée la plus haute. Si la clef recherchée est inférieure à la clef suivante de la même liste chaînée, il faut alors se déplacer sur la liste de niveau inférieur. Dans le cas contraire, nous nous déplaçons en restant sur la même liste. Ces étapes se répètent jusqu'à trouver la bonne clef, à moins que la clef ne soit pas présente dans la *skip list*.

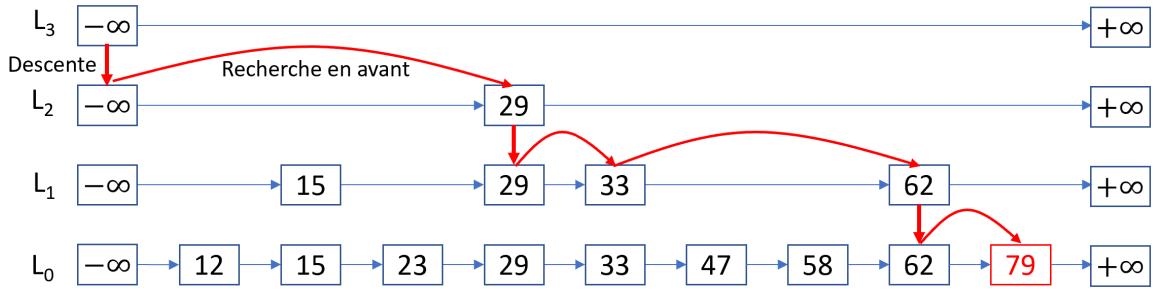


FIGURE 6.7 – Recherche dans une *skip list*.

L'insertion est plus compliquée puisqu'elle fait intervenir une fonction aléatoire pour déterminer la hauteur de l'élément à insérer :

— nous procédons à des tirages aléatoires de type pile ou face jusqu'à l'obtention du côté pile et désignons i le nombre de fois où la pièce est tombée sur face ;
— si $i > h$ avec h le nombre de listes chaînées composant la *skip list*, une nouvelle liste chaînée L_{h+1} est ajoutée, contenant seulement les deux clefs spéciales ;
— nous recherchons l'emplacement de la clef en mémorisant chaque endroit p_k où il est nécessaire de créer de nouveaux liens ;
— la nouvelle clef est ajoutée dans toutes les listes nécessaires et les liens permettant de retrouver rapidement cette clef sont mis à jour.

Ce comportement est décrit dans la FIGURE 6.8 dans laquelle la clef de valeur 32 est ajoutée après avoir tiré au sort, trois fois face avant d'obtenir pile. Ce tirage au sort fait que la nouvelle clef apparaît dans les listes L_0, L_1, L_2. Les traits rouges indiquent les liens devant être mis à jour lors de l'insertion.

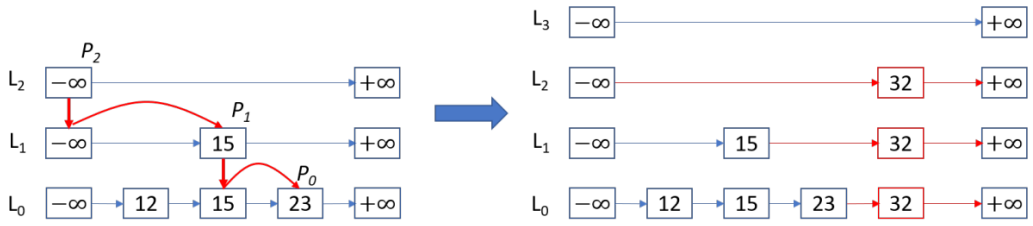


FIGURE 6.8 – Insertion dans une *skip list*.

Les propriétés de la *skip list* sont les suivantes :
— l'espérance des temps de recherche et d'insertion sont en $\mathcal{O}(\log n)$;
— dans le pire cas, les temps de recherche et d'insertion sont en $\mathcal{O}(n)$ mais sont très peu probable ;
— la probabilité que la liste $L_{c\cdot\ln n}$ possède au moins un item est $\frac{1}{n^{c-1}}$;
— la probabilité qu'une *skip list* soit au plus de hauteur h est $P(h \leq c\cdot\ln n) = 1-\frac{1}{n^{c-1}}$.

Comme souvent, il est possible d'ajuster le compromis temps/empreinte mémoire. L'illustration utilisant un tirage aléatoire de type pile ou face n'est autre qu'une expérience de Bernoulli de paramètre $p = 0.5$. En faisant varier la probabilité p de faire apparaitre une clef à un niveau supérieur, par exemple avec $p = 1/4$, la hauteur de la *skip list* sera réduite au prix d'un chemin plus long avant d'atteindre l'élément requis.

6.4.1.4 Résultats expérimentaux

Afin de mettre en évidence le comportement des trois structures de données étudiées, nous nous appuyons sur deux graphiques présentant les temps d'insertion en FIGURE 6.9 et FIGURE 6.10, et un graphique montrant les temps de recherche en FIGURE 6.10.

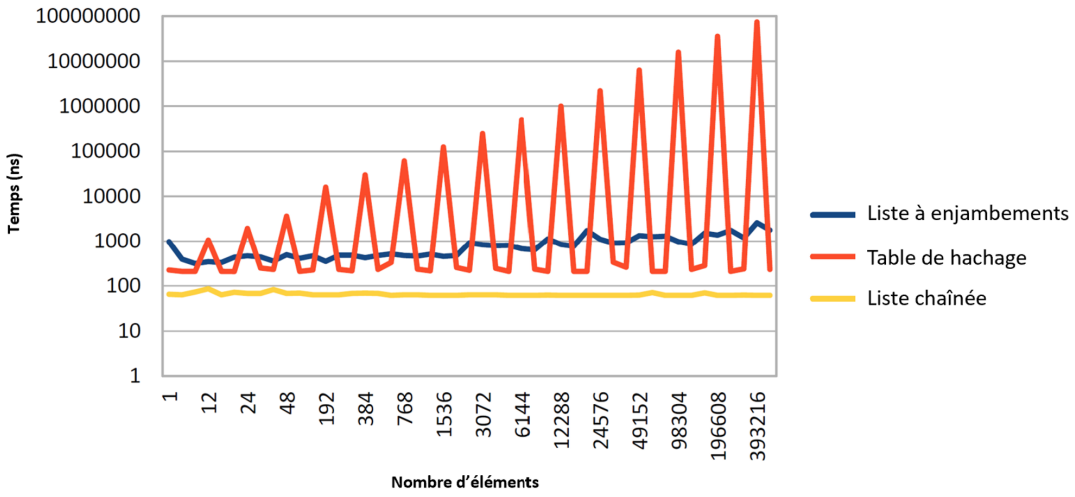


FIGURE 6.9 – Comparaison des temps d'insertion sur une échelle non linéaire du nombre d'éléments.

La FIGURE 6.9 montre la particularité de l'insertion dans les tables de hachage. Le graphique donne l'évolution du temps d'insertion en échelle logarithmique, en fonction du nombre d'éléments insérés. Afin de faciliter la lecture et la compréhension, il se concentre essentiellement sur les phases d'extension de la table de hachage. Pour cela, l'axe des abscisses n'est pas linéaire et montre des séries de trois points : le nombre d'éléments juste avant l'extension, au moment de l'extension et immédiatement après l'extension de la table de hachage. Avec une taille initiale de la table $m = 16$ et un taux de remplissage de 75 %, les séries de trois points sont $\frac{3m}{4} - 1$, $\frac{3m}{4}$ et $\frac{3m}{4} + 1$, avec m doublant à chaque extension. Ce choix de présentation des données permet de mettre en évidence la différence de temps entre une insertion sans extension et une insertion avec extension.

La FIGURE 6.9 rend visible la grande faiblesse de l'extension de la table lors d'insertions, en contraste à ses possibilités de recherche très performantes. Les temps d'insertion lors des extensions augmentent à chaque fois puisqu'il faut recalculer l'index de l'ensemble des éléments déjà présent afin de les répartir dans la nouvelle table conformément à la fonction de hachage avec sa dépendance à la taille de la table $m = 2^{\mu}$.

Sans surprise, la liste chaînée est la plus efficace avec une insertion systématique en tête de liste, avec un temps d'insertion en $\mathcal{O}(1)$.

Quant à la liste à enjambements, nous observons un temps d'insertion intermédiaire qui augmente légèrement avec le nombre d'éléments insérés. Cette courbe contient quelques pics s'expliquant par le côté stochastique de la hauteur d'insertion des éléments. Selon l'élément à insérer, le chemin à emprunter pour identifier le point d'insertion sera plus ou moins optimisé. Les temps d'insertion ont une espérance en $\mathcal{O}(\log n)$.

Le graphique de la FIGURE 6.10 donne l'évolution du temps d'insertion en échelle logarithmique, en fonction du nombre d'éléments insérés, cette fois en utilisant une échelle linéaire pour l'axe des abscisses.

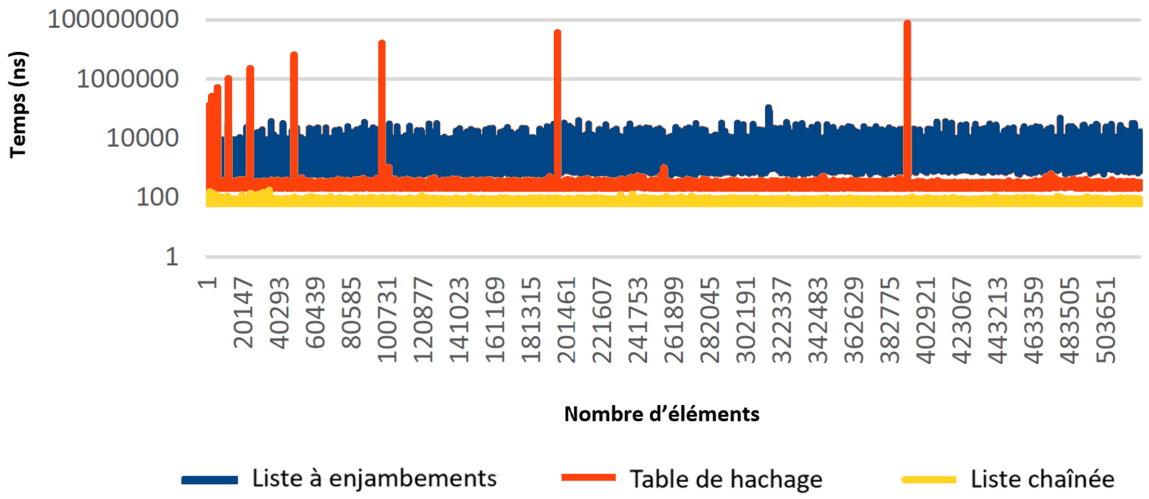


FIGURE 6.10 – Comparaison des temps d'insertion sur une échelle linéaire du nombre d'éléments.

Il montre l'espacement de plus en plus grand entre chaque extension de la table de hachage. La variabilité du temps d'insertion de la liste à enjambement est également plus visible. En portant notre attention sur l'échelle logarithmique des temps, il devient évident que l'insertion dans la table de hachage, même si elle est en général plus rapide que dans la liste à enjambement, a une telle pénalité lors des extensions qu'elle devient problématique. Le denier doublement de taille prend près de 100 ms sur un ordinateur de bureau et serait très largement supérieur dans un système embarqué contraint. Une façon de s'abstraire de ce problème serait de fixer une taille initiale de la table de hachage très grande pour éviter les extensions. Mais dans ce cas, nous reproduisons un système qui s'approche d'une gestion statique de la mémoire, ce qui est contraire à l'objectif visé.

De façon similaire, les temps de recherche ont été mesurés et apparaissent sur la FIGURE 6.11 au niveau de l'axe vertical en échelle logarithmique, en fonction du nombre d'éléments contenus dans les structures de données. Les meilleurs résultats sont obtenus avec la table de hachage dont le temps de recherche est en $\mathcal{O}(1)$ en l'absence de collision. La liste chaînée, dont le temps de recherche est en $\mathcal{O}(n)$, montre sa lenteur en fonction du nombre d'éléments présents. Enfin, la liste à enjambement est un bon compromis, comme observé pour les temps d'insertion.

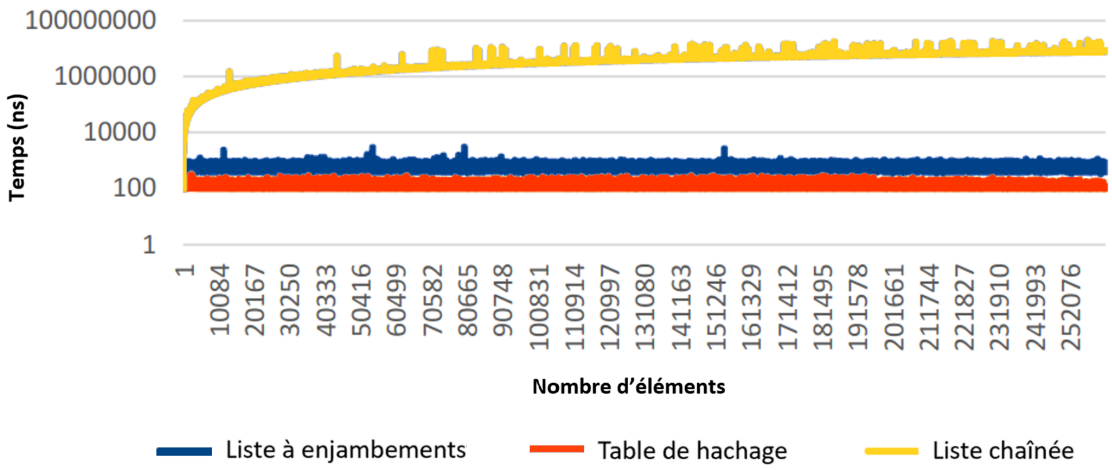


FIGURE 6.11 – Comparaison des temps de recherche sur une échelle linéaire du nombre d'éléments.

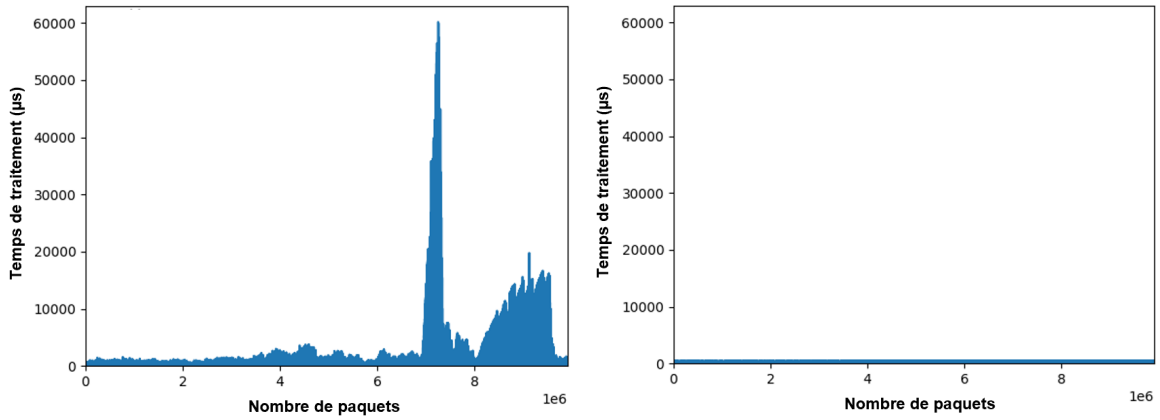
Nous notons certaines variations sur les courbes. Celles-ci semblent dues à l'environnement de l'ordinateur, beaucoup plus compliqué à maitriser que sur un système embarqué. Des activités annexes et de possibles changements dynamiques de fréquence peuvent expliquer ces variations changeantes d'une exécution à l'autre. Une vérification sur le système embarqué Raspberry Pi a donné des valeurs plus lisses, relevées avec des statistiques et non sous formes de courbes comme sur l'ordinateur de bureau. Le degré de confiance dans les résultats obtenus restent élevés et en cohérence avec les résultats attendus.

6.4.2 Validation sur le corpus CIC-IDS2017

L'analyse de la section précédente conduit à remplacer la liste chaînée utilisée dans le chapitre 4 par une liste à enjambement et étudier l'évolution du temps d'extraction des caractéristiques des conversations. L'utilisation de la liste à enjambement n'est rendue possible qu'en transformant les identifiants des conversations selon la méthode proposée en section 6.3.

Nous avons reproduit l'expérimentation mettant en évidence le goulet d'étranglement. Afin de ne pas introduire plusieurs changements en même temps, empêchant une comparaison honnête des implémentations, seule la liste chaînée a été remplacée par le code C de la liste à enjambement utilisé dans la section 6.4.1.4.

La FIGURE 6.12a montre le temps de traitement pour chaque paquet Ethernet reçu sur la journée du vendredi avec le corpus CIC-IDS2017 en utilisant une liste chaînée. Elle correspond à la partie la plus à droite de la FIGURE 6.3. La FIGURE 6.12b donne les temps de traitement obtenus lorsque que la liste chaînée est remplacée par une liste à enjambement. Les deux graphiques utilisent la même échelle de façon à visualiser facilement les différences.



(a) utilisation d'une liste chaînée.

(b) utilisation d'une liste à enjambements.

FIGURE 6.12 – Comparaison des temps de traitement de l'extracteur de caractéristiques.

Les pics d'activité liés à des attaques ont complètement disparu et le temps de traitement est constamment inférieur même en l'absence d'attaque. Le principal goulet d'étranglement identifié en section 6.2 est bien corrigé par la proposition d'introduire une relation d'ordre strict et total dans les identifiants des conversations.

Comme le remplacement d'une liste chaînée par une liste à enjambement n'affecte pas les valeurs des caractéristiques injectées dans les algorithmes de machine learning, les performances de détection d'intrusions ne sont pas répétées dans cette section. Les métriques ont néanmoins été vérifiées et sont en tout point identiques.

Cette expérimentation valide donc la possibilité d'implémenter un système IDS complet sur un système embarqué comme celui utilisé dans le mur d'écrans à base d'agents intelligents dans des Raspberry Pi.

6.4.3 Validation sur un microcontrôleur

Pour aller plus loin dans la validation de l'hypothèse 3 *« Il est possible d'implémenter sur un système embarqué un système de détection d'intrusions incluant l'extraction des caractéristiques des paquets réseau et l'algorithme de détection d'intrusions. »*, le système de détection d'intrusions décrit en FIGURE 6.1 a été porté sur un microcontrôleur MCU de la société STMicroelectronics. Ce MCU de la famille Stellar SR6G7x (STMICROELECTRONICS, 2022) adresse les besoins du domaine automobile, notamment pour la réalisation de passerelle au sein du réseau Ethernet des voitures.

Le schéma fonctionnel de ce MCU est donné sur la FIGURE 6.13 et fait apparaitre six processeurs ARM Cortex-R52 fonctionnant à 400 MHz et possédant deux extensions SIMD NEON.

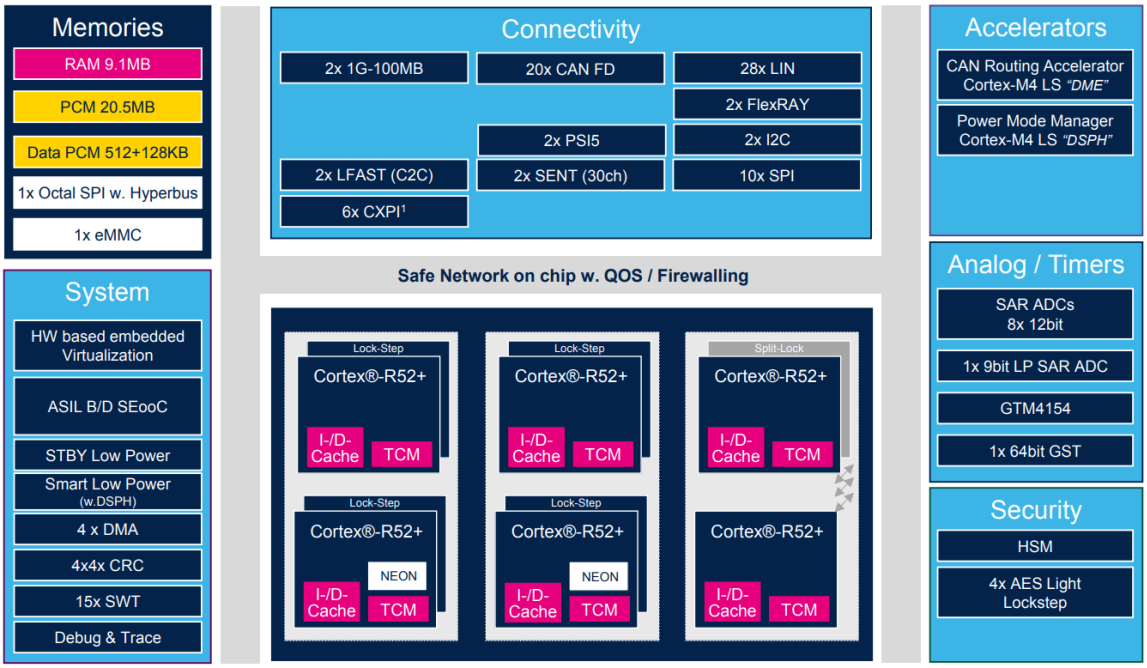


FIGURE 6.13 – Schéma fonctionnel du Stellar SR6G7x.

La plateforme Stellar sert de passerelle entre un ordinateur lançant des attaques et un système d'info-divertissement comme schématisé sur la FIGURE 6.14.

Ce système décode une vidéo reçue en streaming à travers la passerelle, et un ordinateur portable, également connecté à la passerelle, lance en parallèle trois attaques différentes vers la plateforme d'info-divertissement :

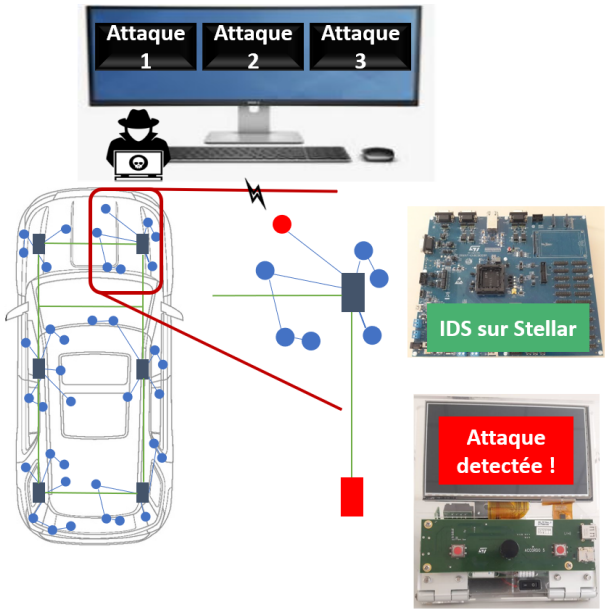


FIGURE 6.14 – Schéma du démonstrateur Stellar.

— une attaque de type portscan ;
— une attaque permettant de casser une connexion FTP en force brute ;
— une attaque permettant de casser une connexion SSH en force brute.

Une adaptation de notre implémentation basée sur la liste à enjambement et le réseau de neurones est nécessaire. Contrairement à un Raspberry Pi basé sur Linux, ce MCU utilise un système d'exploitation temps-réel FreeRTOS (GUAN et al., 2016) et ne bénéficie pas de toutes les librairies disponibles avec Linux. L'implémentation de la liste à enjambement a été complètement réutilisée. Le calcul d'inférence avec TF-Lite a été remplacé par des fonctions logicielles d'algèbre linéaire exploitant l'accélérateur NEON (SMITH, 2020) que nous avons optimisées pour un processeur sans mémoire cache de niveau 2, à la différence des considérations prises pour les librairies classiques (GOTO & GEIJN, 2008). Ces fonctions d'algèbre linéaire ont été validées au préalable sur Raspberry Pi en remplacement de TF-Lite.

Chacun des trois éléments composant notre IDS - l'observateur réseau, l'extracteur de caractéristique et le détecteur d'attaques - a été porté sur un processeur Cortex-R52 différent dans un souci de simplicité. Un mécanisme de communication inter-processeur permet à ces éléments d'échanger les informations nécessaires à chacune de ces activités. L'extracteur de caractéristiques est amélioré par l'utilisation d'une liste à enjambement. Les temps recherche moyen et maximal, mesuré dans notre cas d'utilisation avec un total de 302 953 paquets sont respectivement de 2.33 μs et de 138 μs. Le modèle de détection est, d'une part, optimisé conformément aux corrections apportées par le corpus numérique LYCOS-IDS2017, et d'autre part, accéléré par les fonctions d'algèbre linéaire exploitant le NEON. Le temps d'inférence alors obtenu pour une conversation est de 32 μs.

À titre de comparaison, une étude (Soe et al., 2020) mentionne un temps d'inférence moyen de 183 μs par conversation en utilisant un arbre de décision J48, une implémentation en Java de l'algorithme C4.5 (Quinlan, 1993). Le temps nécessaire à l'extraction des caractéristiques n'est pas inclus. Le programme s'exécute sur un processeur ARM Cortex-A53 à une fréquence de 1.2 GHz. Le Cortex-R52 de la plateforme Stellar est 4.26 fois moins performant. Notre temps d'inférence de 32 μs est donc à comparer à $4.26 \times 183 = 780$ μs, montrant un excellent niveau d'optimisation de notre modèle de détection d'intrusions.

En reprenant l'analyse faite en section 6.2.4.2 pour un lien Ethernet à 100 Mbps, nous pouvons estimer la charge CPU liée au temps d'inférence. La Table 6.3 fournit un calcul de la charge du processeur pour différents nombres de paquets par conversation (pkt/conv).

Table 6.3 – Charge CPU estimée pour différents cas de conversations sur Stellar.

	pkt/s	**charge CPU (%)**		
		6 pkt/conv	20 pkt/conv	30 pkt/conv
Simple IMIX	36 728	19,59	5,88	3,92
Taille max de paquets	8 333	4,44	25,00	0,89
Taille min de paquets	312 500	83,33	1,33	16,67

La Table 6.3 fait clairement apparaitre qu'avec un temps d'inférence de 32 μs, toutes les conversations, quelle que soit leur taille, peuvent être classées sans engorgement du CPU, la charge restant inférieure à 100 % dans tous les cas.

Les performances que nous avons relevées sur chaque CPU indiquent que la somme des charges CPU est inférieure à la charge maximale d'un seul cœur, ouvrant la possibilité à une implémentation sur un seul processeur. Ces résultats montrent qu'en utilisant un seul CPU, notre IDS fonctionnerait en laissant tous les autres processeurs pour l'exécution des fonctions primaires de la plateforme Stellar.

Une démonstration IDS sur la plateforme Stellar, dont un enregistrement vidéo est disponible [1], a été présentée en juin 2022 à Nuremberg (Allemagne) lors du salon professionnel *Embedded World*.

6.5 Synthèse

Dans un premier temps, nous avons décrit la conception d'un système de détection d'intrusions réseau. La décomposition a notamment fait apparaitre un observateur réseau, un extracteur de caractéristiques et un détecteur d'attaques.

Nous avons ensuite recherché les goulets d'étranglement dans un tel système pouvant empêcher ou limiter la possibilité de détecter des attaques sur un système embarqué. Une

1. https://www.youtube.com/watch?v=nEXRQX6iuow

implémentation sur un système sur puce utilisé dans le domaine automobile et similaire à un Raspberry Pi utilisé dans le mur d'écrans évoqué dans la section Positionnement des travaux du chapitre Introduction a permis d'identifier un goulet d'étranglement principal. Le point critique consiste, quand un paquet Ethernet est reçu, à retrouver la conversation à laquelle il appartient, si elle existe.

L'étude de plusieurs structures de données, avec une gestion dynamique de la mémoire, nous a amené à proposer une solution permettant de transformer les identifiants de conversations pour qu'ils forment un ensemble ordonné. Ainsi, il devient possible d'exploiter des structures de données dont les temps d'insertion et de recherche sont en $\mathcal{O}(\log n)$.

Une méthodologie d'expérimentation a été décrite pour comparer les performances de différentes structures de données. Parmi elles, la liste à enjambement nécessite un ensemble ordonné basé sur des identifiants supportant une relation d'ordre strict et total. Les mesures réalisées sur les listes chaînées, les tables de hachage et les listes à enjambements ont mis en évidence les différents comportements en termes de temps d'insertion et de recherche. La liste à enjambement est le meilleur compromis.

Une validation sur un système embarqué, avec le corpus numérique CIC-IDS2017 montre, d'une part, un réel avantage et supprime le risque que le détecteur d'intrusions soit lui-même la victime d'un déni de service, et d'autre part, valide qu'il est possible d'exécuter un IDS sur un système embarqué avec un niveau de ressources équivalent à un Raspberry Pi. Enfin, la validation a été complétée par une implémentation sur un microcontrôleur du domaine automobile ayant des ressources plus limitées et un ensemble logiciel très différent, cette fois, en utilisant le corpus numérique fiabilisé LYCOS-IDS2017.

Pour conclure ce chapitre, nous avons montré qu'avec une approche globale d'un système de détection d'intrusions basée sur les conversations, il n'est pas suffisant d'optimiser les temps d'inférence des algorithmes de machine learning. Il est aussi nécessaire de prendre en compte le fonctionnement du système dans son ensemble pour lever des problèmes annexes, tout aussi importants comme la recherche efficace des conversations.

La solution proposée a abouti à un dépôt de brevet. Ainsi, nous avons validé la troisième hypothèse : *Il est possible d'implémenter sur un système embarqué un système de détection d'intrusions incluant l'extraction des caractéristiques des paquets réseau et l'algorithme de détection d'intrusions.*

Références

Agilent Technologies, (2007), The journal of Internet Test Methodologies, Récupérée 15 juillet 2022, à partir de https://web.archive.org/web/20130127153505/http://www.ixiacom.com/pdfs/test_plans/agilent_journal_of_internet_test_methodologies.pdf

Claise, B., Trammell, B. & Aitken, P., (2013), *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* (STD N° 77), RFC Editor, http://www.rfc-editor.org/rfc/rfc7011.txt

Dietzfelbinger, M., (1996), Universal hashing and k-wise independent random variables via integer arithmetic without primes, In C. Puech & R. Reischuk (Éd.), *STACS 96* (p. 567-580), Springer Berlin Heidelberg.

Ding, X., ding guiguang, g., Zhou, X., Guo, Y., Han, J. & Liu, J., (2019), Global Sparse Momentum SGD for Pruning Very Deep Neural Networks, In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2019/file/f34185c4ca5d58e781d4f14173d41e5d-Paper.pdf

Geron, A., (2019), *Hands-on machine learning with scikit-learn, keras, and TensorFlow : Concepts, tools, and techniques to build intelligent systems* (2e éd.), O'Reilly Media.

Goto, K. & Geijn, R. A. v. d., (2008), Anatomy of High-Performance Matrix Multiplication, *ACM Trans. Math. Softw.*, *34 3*, https://doi.org/10.1145/1356052.1356053

Guan, F., Peng, L., Perneel, L. & Timmerman, M., (2016), Open source FreeRTOS as a case study in real-time operating system evolution, *Journal of Systems and Software*, *118*, 19-35, https://doi.org/10.1016/j.jss.2016.04.063

Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Tso, F. P., Pezaros, D., Mullins, R. D., Yoneki, E., Cox, S. J. & Singer, J., (2018), Commodity single board computer clusters and their applications, *Future Generation Computer Systems*, *89*, 201-212, https://doi.org/10.1016/j.future.2018.06.048

Kang, S., Park, H. & Park, J.-I., (2019), CNN-Based Ternary Classification for Image Steganalysis, *Electronics*, *8 11*, https://doi.org/10.3390/electronics8111225

Mehta, D. P. & Sahni, S., (2004), *Handbook Of Data Structures And Applications (Chapman & Hall/Crc Computer and Information Science Series.)*, Chapman & Hall/CRC.

Pugh, W., (1990a), *A Skip List Cookbook* (rapp. tech.), USA, University of Maryland at College Park.

Pugh, W., (1990b), Skip Lists : A Probabilistic Alternative to Balanced Trees, *Commun. ACM*, *33 6*, 668-676, https://doi.org/10.1145/78973.78977

Quinlan, J. R., (1993), *C4.5 : Programs for Machine Learning*, Morgan Kaufmann.

Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A., (2016), XNOR-Net : ImageNet Classification Using Binary Convolutional Neural Networks, In B. Leibe, J. Matas, N. Sebe & M. Welling (Éd.), *Computer Vision – ECCV 2016* (p. 525-542), Springer International Publishing.

Redfern, A. J., Zhu, L. & Newquist, M. K., (2021), BCNN : A Binary CNN With All Matrix Ops Quantized to 1 Bit Precision, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4604-4612.

Rosay, A., Carlier, F. & Leroux, P., (2022, juillet 22), *Network anomalies detection method* (pareqfr No 2207253).

Shaffer, C., (2011), *Data Structures and Algorithm Analysis*, Dover Publications.

Smith, S., (2020), Neon Coprocessor. *Programming with 64-Bit ARM Assembly Language : Single Board Computer Development for Raspberry Pi and Mobile Devices* (p. 291-306), Apress, https://doi.org/10.1007/978-1-4842-5881-1_13

Soe, Y. N., Feng, Y., Santosa, P. I., Hartanto, R. & Sakurai, K., (2020), Implementing Lightweight IoT-IDS on Raspberry Pi Using Correlation-Based Feature Selection and Its Performance Evaluation, In L. Barolli, M. Takizawa, F. Xhafa & T. Enokido (Éd.), *Advanced Information Networking and Applications* (p. 458-469), Springer International Publishing, https://doi.org/10.1007/978-3-030-15032-7_39

STMicroelectronics, (2022), Data Brief - Stellar SR6 G7 line, Récupérée 26 juillet 2022, à partir de https://www.st.com/resource/en/data_brief/sr6g7c4.pdf

Tang, H., (2019), A new method of bottleneck analysis for manufacturing systems, *Manufacturing Letters*, *19*, 21-24, https://doi.org/10.1016/j.mfglet.2019.01.003

Tartaglione, E., Lepsøy, S., Fiandrotti, A. & Francini, G., (2018), Learning sparse neural networks via sensitivity-driven regularization, In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2018/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

Trammell, B. & Boschi, E., (2008), *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)* (STD N° 5103), RFC Editor, http://www.rfc-editor.org/rfc/rfc5103.txt

Welford, B., (1962), *Technometrics*, *4 3*, 419-420.

Wikipedia, (2017), internet Mix - Wikipedia, Récupérée 15 juillet 2022, à partir de https://en.wikipedia.org/wiki/Internet_Mix

Zhu, F., Gong, R., Yu, F., Liu, X., Wang, Y., Li, Z., Yang, X. & Yan, J., (2020), Towards Unified INT8 Training for Convolutional Neural Network, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Conclusion

« Le genre humain a toujours été en progrès et continuera toujours de l'être à l'avenir : ce qui ouvre une perspective à perte de vue dans le temps. »

Emmanuel Kant
Philosophe (1724 – 1804)

Au cours de cette thèse, nous avons étudié différents systèmes de détection d'intrusions réseau basés sur des techniques d'apprentissage automatique dans les objets connectés. Dans le cadre d'un partenariat entre le laboratoire CREN et STMicroelectronics, nos travaux se sont focalisés sur les objets connectés utilisant le protocole IP dans le but de couvrir, d'une part, les dispositifs numériques dans l'éducation tels que le mur d'écrans à base d'un système multi-agents, et d'autre part, les voitures connectées.

Ce dernier chapitre présente tout d'abord les apports de la thèse. Une synthèse des travaux reprend notre problématique et nos hypothèses, les démarches que nous avons adoptées ainsi que les résultats obtenus.

Ensuite, nous abordons différentes perspectives à la fois sous la forme de perfectionnements possibles, mais aussi de pistes d'investigations pour rendre plus résilients les dispositifs connectés.

Apports de la thèse

Notre objectif consistait à répondre à la problématique suivante :

Les objets connectés utilisant des communications par protocole IP peuvent-ils être protégés des tentatives d'intrusions réseau grâce à des techniques d'apprentissage automatique supervisé ?

Pour y répondre, nous avons émis trois hypothèses :

— un réseau de neurones potentiellement exécutable sur un accélérateur matériel modeste peut atteindre d'aussi bonnes performances de détection d'intrusions réseau que les algorithmes classiques d'apprentissage automatique ;
— la fiabilisation d'un corpus de détection d'intrusions conduit à améliorer la performance de détection tout en garantissant un modèle plus simple ;
— il est possible d'implémenter sur un système embarqué un système de détection d'intrusions incluant l'extraction des caractéristiques des paquets réseau et l'algorithme de détection d'intrusions.

Synthèse des travaux

L'introduction positionne nos travaux selon deux axes. Le premier concerne les objets connectés et la cybersécurité, en particulier sur les dispositifs utilisant le protocole IP, cette catégorie représentant la majorité des objets connectés. Le second axe porte sur l'apprentissage automatique. Ce contexte nous amène à la problématique et aux trois hypothèses étudiées.

Afin de clarifier le sujet et de faciliter la compréhension de nos contributions, nous avons développé un état de l'art en trois parties. Le chapitre 1 présente les systèmes embarqués communicants avec la notion de contraintes spécifiques à ce type de dispositifs. Nous avons évoqué les aspects matériel et logiciel pouvant être utilisés afin d'adresser la grande variété d'objets connectés. Enfin, la cybersécurité nécessaire dans ces dispositifs est abordée sous l'angle des propriétés, des menaces, des vulnérabilités, des attaques mais également des contre-mesures possibles.

L'étude des intrusions réseau nécessite une compréhension approfondie des systèmes de communications, décrits selon le modèle OSI et le modèle TCP/IP. Les différents protocoles rencontrés dans les principaux corpus numériques de détection d'intrusions sont présentés ainsi que les mécanismes de sécurisation existants.

L'état de l'art sur la détection d'intrusions réseau est couvert dans le chapitre 2. Après une présentation des cyberattaques, les différentes techniques de détection d'intrusions ont été évoquées. Pour se positionner dans notre contexte, nous avons limité la suite de nos travaux sur les IDS basés sur la surveillance réseau et utilisant une analyse des conversations plutôt que des paquets réseau. Une revue des corpus numériques de détection d'intrusions est proposée avec une analyse plus détaillée pour les plus récents d'entre eux.

La troisième partie de l'état de l'art, dans le chapitre 3, porte sur l'apprentissage automatique, sur ses différentes méthodes d'apprentissage et les types de tâches pouvant être adressés. Des algorithmes de différentes familles, qu'ils soient classiques ou bien basés sur les réseaux de neurones, sont expliqués. Nous avons également présenté le processus d'entraînement de ces derniers et les challenges associés.

Après cette présentation de l'existant, nous abordons chacune des hypothèses dans des chapitres dédiés couvrant les aspects méthodologiques et les résultats expérimentaux obtenus. Dans le chapitre 4, nous avons sélectionné selon une liste de critères deux corpus numériques, CIC-IDS2017 et CSE-CIC-IDS-2018. Nous avons ensuite proposé et testé une méthode sur le premier corpus permettant de comparer différents algorithmes de machine learning dans le but de positionner les performances d'un réseau de neurones vis-à-vis des algorithmes classiques. Ces derniers utilisent une implémentation de référence. Les résultats obtenus ont été comparés aux résultats déjà publiés sur ce corpus.

La même méthode est appliquée sur le deuxième corpus pour étudier la capacité de généralisation de cette méthode. Une discussion adresse particulièrement les cyberattaques les plus difficilement détectables.

Le chapitre 5 propose une méthode de fiabilisation d'un corpus numérique de détection d'intrusions réseau dans le but d'atteindre une sobriété dans les exploitations des ressources nécessaires au sein des objets connectés. L'étude menée sur CIC-IDS2017 démontre la présence de plusieurs problèmes majeurs tels que des erreurs de calcul, des problèmes de détection de protocoles, une mauvaise gestion des conversations vis-à-vis de la machine d'état TCP et des problèmes de labellisation. Nous avons proposé et mis à disposition en accès libre un extracteur de caractéristiques de conversations ainsi qu'un outil de labellisation afin d'obtenir la version fiabilisée du corpus. Nous avons ensuite exposé une méthodologie d'expérimentation permettant la comparaison des deux versions du corpus numérique, suivie des résultats obtenus. Les difficultés de comparaisons sont discutées, sans pour autant remettre en cause nos conclusions. Enfin, nous avons montré que les problèmes détectés sur CIC-IDS2017 sont également présents sur d'autres corpus numériques pouvant donc tirer avantage de nos outils.

Pour finir, le chapitre 6 présente une approche globale d'un détecteur d'intrusions réseau en le décomposant en sous-systèmes et identifie un goulet d'étranglement au niveau de l'extraction des caractéristiques. Cette limitation, commune au corpus d'origine et au corpus fiabilisé, est mise en évidence expérimentalement avec un système embarqué correspondant à la fois au dispositif éducatif du mur d'écrans et au système télématique connectant les voitures à des serveurs externes. Notre proposition permet de créer des identifiants de conversation autorisant l'utilisation d'algorithme de recherche plus rapide et ainsi, supprimer le goulet d'étranglement. Une méthodologie expérimentale est utilisée pour tester l'effet de notre proposition.

Résultats obtenus

Le chapitre 4 montre qu'en suivant une méthode appropriée, un réseau de neurones MLP permet de classifier les détections d'intrusions du corpus numérique CIC-IDS2017 avec des résultats similaires à ceux d'algorithmes classiques de machine learning utilisant des librairies de référence. Par son application sur un deuxième corpus numérique, la capacité de généralisation de la méthode a été démontrée. Nous avons montré qu'un MLP de complexité réduite permet de détecter les intrusions, ouvrant la voie à l'utilisation d'accélérateur matériel pour les réseaux de neurones de taille modeste afin de libérer des capacités de traitements des CPU, validant ainsi notre première hypothèse.

La comparaison de performances avec les publications antérieures sont souvent difficiles. En effet, dans de nombreux cas, les descriptions des paramètres des algorithmes et de la préparation de données sont incomplètes. Dans certains cas, des métriques, telles que le taux de faux positifs, sont manquantes ou encore des attaques, généralement les plus difficiles à détecter, sont purement écartées. Ces difficultés sont en adéquation avec une étude reportant des résultats trop souvent non reproductibles et inhabituellement élevés (Leevy & Khoshgoftaar, 2020). Nos travaux apportent une description complète, un jeu de données mieux équilibré et un ensemble de métriques permettant aux futurs travaux de se positionner par rapport à nos résultats.

Le chapitre 5 démontre que la méthodologie de fiabilisation d'un corpus de détection d'intrusions est efficace. Son application sur le corpus numérique CIC-IDS2017 a révélé qu'il contient des défauts majeurs. Les deux contributions proposées ont conduit à la mise à disposition d'un outil d'extraction de caractéristiques de conversations appelée LycoSTand ainsi que d'un corpus fiabilisé LYCOS-IDS2017. Les résultats d'expérimentations démontrent que le système de détection d'intrusions avec le corpus numérique fiabilisé obtient de meilleures performances de détection tout en réduisant de manière significative le modèle de détection, validant ainsi notre deuxième hypothèse.

Enfin, notre proposition permettant une recherche efficace parmi les conversations en cours a été validée dans le chapitre 6, de façon expérimentale sur la plateforme télématique et le corpus non fiabilisé. Cette validation atteste la suppression du goulet d'étranglement précédemment identifié. Notre proposition a été déposée auprès de l'Institut National de la Propriété Industrielle (INPI) comme brevet sous le numéro 2207253.

L'expérimentation combinant les apports présentés dans les chapitres 5 et 6 est démontré sur un microcontrôleur du domaine automobile avec un ensemble logiciel construit autour d'un système d'exploitation temps réel.

Ce chapitre démontre la possibilité de supporter un système complet de détection d'intrusions réseau sur des objets connectés, conformément à notre troisième hypothèse.

L'ensemble de nos résultats prouve que des algorithmes d'apprentissage automatique, tels qu'un réseau de neurones simple, entraînés en mode supervisé avec un corpus fiabilisé, permet d'obtenir un système de détection d'intrusions réseau applicable aux objets connectés, répondant ainsi à la problématique adressée dans cette thèse.

Les apports de cette thèse permettent d'envisager des solutions visant la sécurisation des dispositifs numériques pour l'éducation, tel que le mur d'écrans développé au sein du Centre de Recherche en Éducation de Nantes (CREN). Plus important encore, au-delà des dispositifs, ce sont les utilisateurs qui se retrouvent protégés des attaques de cybersécurité.

Pour STMicroelectronics, les apports prouvent qu'il est possible de mettre en œuvre des réseaux de neurones sur des accélérateurs matériels embarqués dans les microcontrôleurs, et ainsi, délester les processeurs de tâches dédiées à la détection d'intrusions. Ceci est applicable dans les microcontrôleurs des voitures connectées, mais aussi, de façon plus générale, sur l'ensemble des MCU utilisés pour réaliser des objets connectés.

Les perspectives décrites dans la section suivante sont des pistes de recherche à étudier, de potentielles collaborations à venir entre le CREN et STMicroelectronics afin de continuer à faire progresser la cybersécurité des objets connectés.

Perspectives

Au cours de nos travaux, les expérimentations ont permis de vérifier nos hypothèses et de répondre à notre problématique. Pour autant, même si nous avons démontré qu'il est possible de détecter des intrusions réseaux sur des objets connectés, nous avons rencontré

certaines limites et identifié des améliorations possibles. Nous proposons donc d'aborder les perfectionnements envisageables. Des pistes restent à explorer au-delà de l'apprentissage supervisé, mais également pour combler les limites des systèmes de détection d'intrusions réseaux basées sur les conversations.

Perfectionnements envisageables

Dépendance à la configuration réseau L'étude des caractéristiques, mentionnée dans le chapitre 5, montre que la durée des conversations est l'une des informations les plus importantes. En changeant la configuration réseau, par exemple en ajoutant des routeurs entre la machine lançant l'attaque et la machine victime, il est facile d'imaginer que les valeurs de cette caractéristique évolueront et impacteront la capacité de détection de certaines attaques. Dans un environnement suffisamment contrôlé, ce point ne pose pas de problème. Toutefois, une étude des performances de détection sans utiliser cette caractéristique pourrait permettre une meilleure résilience aux changements de configuration réseau.

Corpus numériques et apprentissage supervisé Pour tout domaine adressé avec des algorithmes de machine learning, la quantité et la qualité des données sont des éléments essentiels. Les corpus numériques de détection d'intrusions que nous avons étudiés ne sont pas exhaustifs. D'une part, ils ne contiennent pas toutes les attaques connues et celles présentes ne sont pas toujours en quantité suffisante. D'autre part, les descriptions des attaques, des outils utilisés et des configurations logicielles sont souvent incomplètes et empêchent la reproduction d'attaques similaires.

Pour illustrer les conséquences de ce dernier point, nous prenons l'exemple d'une attaque en force brute en SSH. La sécurisation de ce type de connexion peut utiliser des algorithmes asymétriques comme le RSA avec une longueur de clef de 2 048 bits comme c'est le cas dans le corpus numérique CIC-IDS2017. RSA est de moins en moins utilisé, au profit des courbes elliptiques (ECC) utilisant des clefs nettement plus petites pour le même niveau de sécurité. La différence de taille de clefs est due à l'existence d'attaques non génériques telles que la factorisation, affectant le problème du logarithme discret, base du RSA. Ainsi, pour un niveau de 112 et 256 bits de sécurité, l'algorithme RSA nécessite respectivement des clefs de 2 048 et 15 360 bits tandis qu'elles sont respectivement de 224 et 512 bits avec des courbes elliptiques (Barker, 2020, p. 54-55). Le nombre de paquets ainsi que leurs tailles varieront selon l'algorithme choisi, impactant la capacité de détection de l'attaque.

De la même manière, une attaque DoS de type Slowloris envoie des requêtes HTTP partielles à intervalles réguliers. Le corpus numérique CIC-IDS2017 a été créé pour un envoi périodique toutes les 150 s. Une configuration différente de l'attaque avec une période de 60 s modifiera les caractéristiques comme le nombre de paquets par seconde et le débit. Un tel changement risque de mettre le détecteur d'intrusions en difficulté.

Cette limitation n'est pas surprenante. En effet, un modèle de machine learning en apprentissage supervisé apprend une classification basée sur les données présentes dans le jeu de données d'entraînement. Pour une mise en application d'un tel modèle, un corpus numérique devrait être adapté selon les attaques et les configurations contre lesquelles l'objet connecté doit être protégé. D'un point de vue général, un corpus contenant une plus grande variété de configurations et avec des quantités d'attaques identiques pour chacune d'elle serait d'un grand intérêt pour la recherche sur la détection d'intrusions.

Au-delà de l'apprentissage supervisé

Tant qu'un hackeur trouve un bénéfice à lancer des cyberattaques, la recherche pour contourner les mécanismes de protection existera. La création de nouvelles attaques échappant à la détection vient mettre en péril un système de détection basé sur l'apprentissage supervisé. Dans un tel contexte, ce type d'apprentissage nécessite un entraînement régulier prenant en compte toute nouvelle attaque.

L'étude des IDS utilisant un apprentissage semi-supervisé ou non-supervisé seul ou bien en complément d'un apprentissage supervisé semble un champ intéressant à explorer. Pour illustrer cette idée, nous pourrions imaginer un IDS en mode supervisé détectant un ensemble d'attaques et toutes les conversations considérées normales pourraient être injectées dans un IDS semi-supervisé qui aurait appris la distribution des caractéristiques du trafic normal. Toute anomalie par rapport à cette distribution pourrait être considérée comme une attaque ou *a minima* reportée comme attaque potentielle pouvant nécessiter une mise à jour du système en mode supervisé.

Limites des IDS basés sur les conversations

Dans le chapitre 4, nous avons levé un inconvénient des systèmes de détection d'intrusions basées sur les conversations. L'analyse des méta-données caractérisant les conversations ne permettent pas de détecter certaines attaques. L'injection SQL en est un parfait exemple : la différence entre une requête mal formée et une requête légitime ne génère pas de différence de comportement observable. De la même manière, lors d'un téléchargement d'un fichier, les caractéristiques des conversations ne varient pas selon la présence ou non d'un exploit au sein du fichier permettant par exemple d'ouvrir une porte dérobée. Des solutions alternatives existent pour ces problèmes, telles que la vérification du résultat de hachage du fichier reçu ou la mise en place d'un meilleur contrôle des commandes SQL.

Au-delà des deux exemples cités, il existe probablement d'autres attaques difficilement détectables, voire impossible à détecter par un IDS basé sur les conversations. Une étude détaillée de ces cas permettrait de considérer des moyens supplémentaires de protection.

Mot de la fin

De manière générale, la sécurité ne peut pas reposer sur un unique outil de protection. La mise en place de protections à différents niveaux est une nécessité absolue. Dans les châteaux-forts du moyen-age, il fallait d'abord passer les douves, les remparts puis prendre le donjon avant de s'emparer du trésor. De la même manière, la cybersécurité requiert une accumulation de protections assurant une résilience dans l'hypothèse où la première fortification ne jouerait pas complètement son rôle.

Le système de détection d'intrusions étudié dans cette thèse est un élément de protection adapté aux objets connectés. Pour autant, il est nécessaire de le compléter par d'autres mesures afin de mieux sécuriser les objets connectés. Ce sujet mérite des études et des contributions complémentaires afin d'assurer une protection plus forte et limiter les impacts potentiels des attaques venant de hackeurs malintentionnés.

Références

Barker, E., (2020), Recommendation for Key Management - NIST SP 800-57 Part 1 Rev. 5, https://doi.org/10.6028/NIST.SP.800-57pt1r5

Leevy, J. L. & Khoshgoftaar, T. M., (2020), A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data, *Journal of Big Data*, *7 1*, 104, https://doi.org/10.1186/s40537-020-00382-x

BIBLIOGRAPHIE

Agilent Technologies, (2007), The journal of Internet Test Methodologies, Récupérée 15 juillet 2022, à partir de https://web.archive.org/web/20130127153505/http://www.ixiacom.com/pdfs/test_plans/agilent_journal_of_internet_test_methodologies.pdf

Akram, A. & Sawalha, L., (2019), A Study of Performance and Power Consumption Differences Among Different ISAs, *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 628-632, https://doi.org/10.1109/DSD.2019.00098

Albawi, S., Mohammed, T. A. & Al-Zawi, S., (2017), Understanding of a convolutional neural network, *2017 International Conference on Engineering and Technology (ICET)*, 1-6, https://doi.org/10.1109/ICEngTechnol.2017.8308186

Altman, N. S., (1992), An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression, *The American Statistician*, *46 3*, 175-185, https://doi.org/10.1080/00031305.1992.10475879

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K. & Zhou, Y., (2017), Understanding the Mirai Botnet, *26th USENIX Security Symposium (USENIX Security 17)*, 1093-1110, https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

Arkko, J. & Bradner, S., (2008), *IANA Allocation Guidelines for the Protocol Field* (BCP N° 37), RFC Editor, http://www.rfc-editor.org/rfc/rfc5237.txt

Arkko, J. & Pignataro, C., (2009), *IANA Allocation Guidelines for the Address Resolution Protocol (ARP)* (RFC N° 5494), RFC Editor, http://www.rfc-editor.org/rfc/rfc5494.txt

Bandur, V., Selim, G., Pantelic, V. & Lawford, M., (2021), Making the Case for Centralized Automotive E/E Architectures, *IEEE Transactions on Vehicular Technology*, *70 2*, 1230-1245, https://doi.org/10.1109/TVT.2021.3054934

Banerjee, U., Vashishtha, A. & Saxena, M., (2010), Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection [Published By Foundation of Computer Science], *International Journal of Computer Applications*, *6 7*, 1-5, https://doi.org/doi.org/10.5120/1092-1427

Banko, M. & Brill, E., (2001), Scaling to very very large corpora for natural language disambiguation, *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, 26-33.

Barkalov, A., Titarenko, L. & Mazurkiewicz, M., (2019), *Foundations of embedded systems*, Springer International Publishing.

Barker, E., (2020), Recommendation for Key Management - NIST SP 800-57 Part 1 Rev. 5, https://doi.org/10.6028/NIST.SP.800-57pt1r5

Belshe, M., Peon, R. & Thomson, M., (2015), *Hypertext Transfer Protocol Version 2 (HTTP/2)* (RFC N° 7540), RFC Editor, http://www.rfc-editor.org/rfc/rfc7540.txt

Bergstra, J. & Bengio, Y., (2012), Random Search for Hyper-parameter Optimization, *The Journal of Machine Learning Research*, *13*, 281-305.

Berners-Lee, T., Fielding, R. T. & Nielsen, H. F., (1996), *Hypertext Transfer Protocol – HTTP/1.0* (RFC N° 1945), RFC Editor, http://www.rfc-editor.org/rfc/rfc1945.txt

Bishop, C. M., (2006), *Pattern Recognition and Machine Learning* (M. Jordan, J. Kleinberg & B. Schölkopf, Éd. ; T. 4), Springer, https://doi.org/10.1117/1.2819119

Bishop, M., (2022), *HTTP/3* (RFC N° 9114), RFC Editor, http://www.rfc-editor.org/rfc/rfc9114.txt

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N. & Hartog, F. T. H. d., (2022), ToN_IoT : The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets, *IEEE Internet of Things Journal*, *9 1*, 485-496, https://doi.org/10.1109/JIOT.2021.3085194

Bošnjak, L., Sreš, J. & Brumen, B., (2018), Brute-force and dictionary attack on hashed real-world passwords, *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1161-1166, https://doi.org/10.23919/MIPRO.2018.8400211

Bottou, L., Weston, J. & Bakir, G., (2004), Breaking SVM Complexity with Cross-Training, In L. Saul, Y. Weiss & L. Bottou (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/2004/file/dbbf603ff0e99629dda5d75b6f75f966-Paper.pdf

Braden, R., (1989), *Requirements for Internet Hosts - Communication Layers* (STD N° 3), RFC Editor, http://www.rfc-editor.org/rfc/rfc1122.txt

Breiman, L., (2001), Random forests, *Machine learning*, *45 1*, 5-32.

Breiman, L., (2017), *Classification and Regression Trees*, CRC Press.

Brown, C., Cowperthwaite, A., Hijazi, A. & Somayaji, A., (2009), Analysis of the 1999 DARPA/Lincoln Laboratory IDS evaluation data with NetADHICT, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-7, https://doi.org/10.1109/CISDA.2009.5356522

Bul'ajoul, W., James, A. & Pannu, M., (2015), Improving network intrusion detection system performance through quality of service configuration and parallel technology [Special Issue on Optimisation, Security, Privacy and Trust in E-business Systems], *Journal of Computer and System Sciences*, *81 6*, 981-999, https://doi.org/10.1016/j.jcss.2014.12.012

Cain, B., Deering, S., Kouvelas, I., Fenner, B. & Thyagarajan, A., (2002), *Internet Group Management Protocol, Version 3* (RFC N° 3376), RFC Editor, http ://www.rfc-editor.org/rfc/rfc3376.txt

Canadian Institute for Cybersecurity, (2017a), Applications - CICFlowMeter (formerly ISCXFlowMeter), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/research/applications.html

Canadian Institute for Cybersecurity, (2017b), Intrusion Detection Evaluation Dataset (CICIDS2017), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2017.html

Canadian Institute for Cybersecurity, (2018), CSE-CIC-IDS2018 on AWS, A collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), Récupérée 24 juillet 2022, à partir de https://www.unb.ca/cic/datasets/ids-2018.html

Canadian Institute for Cybersecurity, (2020), CicFlowMeter source code, Récupérée 24 juillet 2022, à partir de https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter

Carlier, F. & Renault, V., (2016), Iot-a, embedded agents for smart internet of things : Application on a display wall., *In 2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA), IEEE Computer Society*, 80-83.

Cauchy, A. et al., (1847), Méthode générale pour la résolution des systemes d'équations simultanées, *Comp. Rend. Sci. Paris*, *25 1847*, 536-538.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P., (2002), SMOTE : Synthetic Minority over-Sampling Technique, *J. Artif. Intell. Res.*, *16*, 321-357.

Cheshire, S., (2008), *IPv4 Address Conflict Detection* (RFC N° 5227), RFC Editor, http://www.rfc-editor.org/rfc/rfc5227.txt

Chicco, D., Tötsch, N. & Jurman, G., (2021), The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation, *BioData Mining*, *14 1*, 13, https://doi.org/10.1186/s13040-021-00244-z

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y., (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv :1406.1078*.

Claise, B., Trammell, B. & Aitken, P., (2013), *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* (STD N° 77), RFC Editor, http://www.rfc-editor.org/rfc/rfc7011.txt

Comer, D. E., (2000), *Internetworking with TCP/IP Vol.1 : Principles, Protocols, and Architecture* (4e éd.), Pearson.

Conte, G., Tommesani, S. & Zanichelli, F., (2000), The long and winding road to high-performance image processing with MMX/SSE, *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, 302-310, https://doi.org/10.1109/CAMP.2000.875989

Coppola, M., Kornaros, G., Andrade, L. & Rousseau, F., (2021), Automation for industry 4.0 by using secure lorawan edge gateways, *Multi-Processor System-on-Chip*, *2*, 49-66.

Cortes, C., Jackel, L. D. & Chiang, W.-P., (1994), Limits on Learning Machine Accuracy Imposed by Data Quality, In G. Tesauro, D. Touretzky & T. Leen (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/1994/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf

Cortes, C. & Vapnik, V., (1995), Support-vector networks, *Machine learning*, *20 3*, 273-297.

Cotton, M., Eggert, L., Touch, J., Westerlund, M. & Cheshire, S., (2011), *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry* (BCP N° 165), RFC Editor, http://www.rfc-editor.org/rfc/rfc6335.txt

Courbariaux, M., Bengio, Y. & David, J.-P., (2015), BinaryConnect : Training Deep Neural Networks with binary weights during propagations, In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2015/file/3e15cc11f979ed25912dff5b0669f2cd-Paper.pdf

Cramer, J., (2002), *The Origins of Logistic Regression* (Tinbergen Institute Discussion Papers N° 02-119/4), Tinbergen Institute, https://doi.org/10.2139/ssrn.360300

Dang, L. M., Piran, M. J., Han, D., Min, K. & Moon, H., (2019), A Survey on Internet of Things and Cloud Computing for Healthcare, *Electronics*, *8 7*, https://doi.org/10.3390/electronics8070768

Debar, H., Dacier, M. & Wespi, A., (1999), Towards a taxonomy of intrusion-detection systems, *Computer Networks*, *31 8*, 805-822, https://doi.org/10.1016/S1389-1286(98)00017-6

Debouk, R. et al., (2018), Overview of the 2nd Edition of ISO 26262 : Functional safety-road vehicles, *General Motors Company, Warren, MI, USA*.

Deering, S. & Hinden, R., (2017), *Internet Protocol, Version 6 (IPv6) Specification* (STD N° 86), RFC Editor, http://www.rfc-editor.org/rfc/rfc8200.txt

Deering, S., (1989), *Host extensions for IP multicasting* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc1112.txt

Derbyshire, R., Green, B., Prince, D., Mauthe, A. & Hutchison, D., (2018), An Analysis of Cyber Security Attack Taxonomies, *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 153-161, https://doi.org/10.1109/EuroSPW.2018.00028

Desoli, G., Chawla, N., Boesch, T., Singh, S.-p., Guidetti, E., De Ambroggi, F., Majo, T., Zambotti, P., Ayodhyawasi, M., Singh, H. & Aggarwal, N., (2017), 14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems, *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 238-239, https://doi.org/10.1109/ISSCC.2017.7870349

Devarakonda, A., Sharma, N., Saha, P. & Ramya, S., (2022), Network intrusion detection : a comparative study of four classifiers using the NSL-KDD and KDD'99 datasets, *Journal of Physics : Conference Series*, *2161 1*, 012043, https://doi.org/10.1088/1742-6596/2161/1/012043

Dietzfelbinger, M., (1996), Universal hashing and k-wise independent random variables via integer arithmetic without primes, In C. Puech & R. Reischuk (Éd.), *STACS 96* (p. 567-580), Springer Berlin Heidelberg.

Ding, X., ding guiguang, g., Zhou, X., Guo, Y., Han, J. & Liu, J., (2019), Global Sparse Momentum SGD for Pruning Very Deep Neural Networks, In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2019/file/f34185c4ca5d58e781d4f14173d41e5d-Paper.pdf

Dockhorn, T., Yu, Y., Sari, E., Zolnouri, M. & Partovi Nia, V., (2021), Demystifying and Generalizing BinaryConnect, In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang & J. W. Vaughan (Éd.), *Advances in Neural Information Processing Systems* (p. 13202-13216), Curran Associates, Inc., https://proceedings.neurips.cc/paper/2021/file/6e0cf80a83327822a972bcde3c1d9740-Paper.pdf

Dozat, T., (2016), Incorporating nesterov momentum into adam.

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I. & Ghorbani., A. A., Characterization of Encrypted and VPN Traffic using Time-related Features, in : *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, INSTICC, SciTePress, 2016, 407-414, isbn : 978-989-758-167-0, https://doi.org/10.5220/0005740704070414.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. & Vapnik, V., (1996), Support Vector Regression Machines, In M. Mozer, M. Jordan & T. Petsche (Éd.), *Advances in Neural Information Processing Systems*, MIT Press, https://proceedings.neurips.cc/paper/1996/file/d38901788c533e8286cb6400b40b386d-Paper.pdf

Duchi, J., Hazan, E. & Singer, Y., (2011), Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, *12 61*, 2121-2159, http://jmlr.org/papers/v12/duchi11a.html

Duda, R. O., Hart, P. E. & Stork, D. G., (2001), *Pattern Classification* (2e éd.), Wiley.

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M. & Halderman, J. A., (2014), The Matter of Heartbleed, *Proceedings of the 2014 Conference on Internet Measurement Conference*, 475-488, https://doi.org/10.1145/2663716.2663755

Eastlake, D. & Ableу, J., (2013), *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters* (BCP No 141), RFC Editor, http://www.rfc-editor.org/rfc/rfc7042.txt

Ejaz, W. & Anpalagan, A., (2019), Internet of Things for Smart Cities : Overview and Key Challenges. *Internet of Things for Smart Cities : Technologies, Big Data and*

Security (p. 1-15), Springer International Publishing, https://doi.org/10.1007/978-3-319-95037-2_1

Elnawawy, M., Farhan, A., Nabulsi, A. A., Al-Ali, A. & Sagahyroon, A., (2019), Role of FPGA in Internet of Things Applications, *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 1-6, https://doi.org/10.1109/ISSPIT47144.2019.9001747

Eslahi, M., Salleh, R. & Anuar, N. B., (2012), Bots and botnets : An overview of characteristics, detection and challenges, *IEEE International Conference on Control System, Computing and Engineering*, 349-354, https://doi.org/10.1109/ICCSCE.2012.6487169

European Parliament, (2015), Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 April 2015 Concerning Type-Approval Requirements for the Deployment of the eCall in-Vehicle System Based on the 112 Service and Amending Directive 2007/46/EC, *Official Journal of the European Union*.

Felt, A. P., Barnes, R., King, A., Palmer, C., Bentzel, C. & Tabriz, P., (2017), Measuring HTTPS Adoption on the Web, *Proceedings of the 26th USENIX Conference on Security Symposium*, 1323-1338.

Fenner, W. C., (1997), *Internet Group Management Protocol, Version 2* (RFC N° 2236), RFC Editor, http://www.rfc-editor.org/rfc/rfc2236.txt

Ferrag, M. A. & Maglaras, L., (2019), DeliveryCoin : An IDS and Blockchain-Based Delivery Framework for Drone-Delivered Services, *Computers*, *83*, https://doi.org/10.3390/computers8030058

Ferrag, M. A., Maglaras, L., Moschoyiannis, S. & Janicke, H., (2020), Deep learning for cyber security intrusion detection : Approaches, datasets, and comparative study, *Journal of Information Security and Applications*, *50*, 102419, https://doi.org/10.1016/j.jisa.2019.102419

Finlayson, R., Mann, T., Mogul, J. & Theimer, M., (1984), *A Reverse Address Resolution Protocol* (STD N° 38), RFC Editor, http://www.rfc-editor.org/rfc/rfc903.txt

Fisher, R. A., (1936), The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, *72*, 179-188, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Fonseca, J., Vieira, M. & Madeira, H., (2009), Vulnerability attack injection for web applications, *IEEE/IFIP International Conference on Dependable Systems Networks*, 93-102, https://doi.org/10.1109/DSN.2009.5270349

Friha, O., Ferrag, M. A., Shu, L., Maglaras, L. & Wang, X., (2021), Internet of Things for the Future of Smart Agriculture : A Comprehensive Survey of Emerging Technologies, *IEEE/CAA Journal of Automatica Sinica*, *8*, 718-752.

Fukushima, K. & Miyake, S., (1982), Neocognitron : A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition, In S.-i. Amari & M. A. Arbib (Éd.), *Competition and Cooperation in Neural Nets* (p. 267-285), Springer Berlin Heidelberg.

Gamage, S. & Samarabandu, J., (2020), Deep learning methods in network intrusion detection : A survey and an objective comparison, *Journal of Network and Computer Applications*, *169*, 102767, https://doi.org/10.1016/j.jnca.2020.102767

Garcia, S., Parmisano, A. & Erquiaga, M. J., (2020), IoT-23 : A labeled dataset with malicious and benign IoT network traffic, https://doi.org/10.5281/zenodo.4743746

Geron, A., (2019), *Hands-on machine learning with scikit-learn, keras, and TensorFlow : Concepts, tools, and techniques to build intelligent systems* (2e éd.), O'Reilly Media.

Gers, F. A., Schmidhuber, J. & Cummins, F., (2000), Learning to Forget : Continual Prediction with LSTM, *Neural Computation*, *12 10*, 2451-2471, https://doi.org/10.1162/089976600300015015

Geurts, P., Ernst, D. & Wehenkel, L., (2006), Extremely randomized trees, *Machine learning*, *63 1*, 3-42.

Glorot, X. & Bengio, Y., (2010), Understanding the difficulty of training deep feedforward neural networks, In Y. W. Teh & M. Titterington (Éd.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (p. 249-256), PMLR, https://proceedings.mlr.press/v9/glorot10a.html

Gohari, M., Hashemi, S. & Abdi, L., (2021), Android Malware Detection and Classification Based on Network Traffic Using Deep Learning, *2021 7th International Conference on Web Research (ICWR)*, 71-77, https://doi.org/10.1109/ICWR51868.2021.9443025

Gont, F. & Bellovin, S., (2012), *Defending against Sequence Number Attacks* (RFC N° 6528), RFC Editor, http://www.rfc-editor.org/rfc/rfc6528.txt

Gont, F. & Yourtchenko, A., (2011), *On the Implementation of the TCP Urgent Mechanism* (RFC N° 6093), RFC Editor, http://www.rfc-editor.org/rfc/rfc6093.txt

Goodfellow, I., Bengio, Y. & Courville, A., (2016), *Deep Learning*, MIT Press.

Goto, K. & Geijn, R. A. v. d., (2008), Anatomy of High-Performance Matrix Multiplication, *ACM Trans. Math. Softw.*, *34 3*, https://doi.org/10.1145/1356052.1356053

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J., (2017), LSTM : A Search Space Odyssey, *IEEE Transactions on Neural Networks and Learning Systems*, *28 10*, 2222-2232, https://doi.org/10.1109/TNNLS.2016.2582924

Gregorutti, B., Michel, B. & Saint-Pierre, P., (2017), Correlation and variable importance in random forests, *Statistics and Computing*, *27 3*, 659-678, https://doi.org/10.1007/s11222-016-9646-1

Guan, F., Peng, L., Perneel, L. & Timmerman, M., (2016), Open source FreeRTOS as a case study in real-time operating system evolution, *Journal of Systems and Software*, *118*, 19-35, https://doi.org/10.1016/j.jss.2016.04.063

He, K., Zhang, X., Ren, S. & Sun, J., (2015), Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Henzinger, T. A. & Sifakis, J., (2007), The Discipline of Embedded Systems Design, *Computer*, *40 10*, 32-40, https://doi.org/10.1109/MC.2007.364

Herculano-Houzel, S., (2020), Chapter 33 - Remarkable, But Not Special : What Human Brains Are Made of, In J. H. Kaas (Éd.), *Evolutionary Neuroscience (Second Edition)* (Second Edition, p. 803-813), Academic Press, https://doi.org/10.1016/B978-0-12-820584-6.00033-7

Herculano-Houzel, S., (2009), The human brain in numbers : a linearly scaled-up primate brain, *Frontiers in Human Neuroscience*, *3*, https://doi.org/10.3389/neuro.09.031.2009

Hochreiter, S. & Schmidhuber, J., (1997), Long Short-Term Memory, *Neural Computation*, *9 8*, 1735-1780, https://doi.org/10.1162/neco.1997.9.8.1735

Hornik, K., (1991), Approximation capabilities of multilayer feedforward networks, *Neural Networks*, *4 2*, 251-257, https://doi.org/10.1016/0893-6080(91)90009-T

Hssina, B., Merbouha, A., Ezzikouri, H. & Erritali, M., (2014), A comparative study of decision tree ID3 and C4.5, *International Journal of Advanced Computer Science and Applications*, *4 2*, 13-19.

Hua, Y., (2020), An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM, *Information Communication Technologies Conference (ICTC)*, 125-130, https://doi.org/10.1109/ICTC49638.2020.9123302

Hussain, J. & Hnamte, V., (2021), Deep Learning Based Intrusion Detection System : Software Defined Network, *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 1-6, https://doi.org/10.1109/ASIANCON51346.2021.9544913

IANA, (2021), Protocol Numbers, Récupérée 7 août 2022, à partir de https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

IEEE Standard for Ethernet, (2018), *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, 1-5600, https://doi.org/10.1109/IEEESTD.2018.8457469

IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Redline, (2021), *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline*, 1-7524.

IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery, (2016), *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, 1-146, https://doi.org/10.1109/IEEESTD.2016.7433915

IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security, (2018), *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, 1-239, https://doi.org/10.1109/IEEESTD.2018.8585421

IoT Attacks Skyrocket, Doubling in 6 Months, (2021), Threatpost. Récupérée 6 septembre 2021, à partir de https://threatpost.com/iot-attacks-doubling/169224/

ISO, (2009), ISO/IEC 15408 : Information technology — Security techniques — Evaluation criteria for IT security, Récupérée 3 août 2022, à partir de https://www.iso.org/standard/50341.html

ISO, (2021), ISO/SAE 21434 : Road vehicles — Cybersecurity engineering, Récupérée 3 août 2022, à partir de https://www.iso.org/standard/70918.html

ITU, (2021), Measuring digital development, Facts and figures, Récupérée 26 juillet 2022, à partir de https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2021.pdf

IXIA, (2016), PerfectStorm, Industry's highest performing application and security test platform, Récupérée 24 juillet 2022, à partir de https://www.keysight.com/fr/en/products/network-test/network-test-hardware/perfectstorm.html

Jabbar, W. A., Wei, C. W., Azmi, N. A. A. M. & Haironnazli, N. A., (2021), An IoT Raspberry Pi-based parking management system for smart campus, *Internet of Things*, *14*, 100387, https://doi.org/10.1016/j.iot.2021.100387

Jiang, J., Yu, Q., Yu, M., Li, G., Chen, J., Liu, K., Liu, C. & Huang, W., (2018), ALDD : A Hybrid Traffic-User Behavior Detection Method for Application Layer DDoS, *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 1565-1569, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00225

Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Tso, F. P., Pezaros, D., Mullins, R. D., Yoneki, E., Cox, S. J. & Singer, J., (2018), Commodity single board computer clusters and their applications, *Future Generation Computer Systems*, *89*, 201-212, https://doi.org/10.1016/j.future.2018.06.048

Kabalci, Y., Kabalci, E., Padmanaban, S., Holm-Nielsen, J. B. & Blaabjerg, F., (2019), Internet of Things Applications as Energy Internet in Smart Grids and Smart Environments, *Electronics*, *8 9*, https://doi.org/10.3390/electronics8090972

Kali, (2017), Kali Linux, The Most Advanced Penetration Testing Distribution, Récupérée 24 juillet 2022, à partir de https://www.kali.org/

Kang, M. & Jameson, N. J., (2018), Machine Learning : Fundamentals. *Prognostics and Health Management of Electronics* (p. 85-109), https://doi.org/10.1002/9781119515326.ch4

Kang, S., Park, H. & Park, J.-I., (2019), CNN-Based Ternary Classification for Image Steganalysis, *Electronics*, *8 11*, https://doi.org/10.3390/electronics8111225

Karatas, G., Demir, O. & Sahingoz, O. K., (2020), Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset, *IEEE Access*, *8*, 32150-32162, https://doi.org/10.1109/ACCESS.2020.2973219

Kent, S., (2005a), *IP Authentication Header* (RFC N° 4302) [http://www.rfc-editor.org/rfc/rfc4302.txt], RFC Editor, RFC Editor, http://www.rfc-editor.org/rfc/rfc4302.txt

Kent, S., (2005b), *IP Encapsulating Security Payload (ESP)* (RFC N° 4303) [http://www.rfc-editor.org/rfc/rfc4303.txt], RFC Editor, RFC Editor, http://www.rfc-editor.org/rfc/rfc4303.txt

Kent, S. & Seo, K., (2005), *Security Architecture for the Internet Protocol* (RFC N° 4301), RFC Editor, http://www.rfc-editor.org/rfc/rfc4301.txt

Khan, F. A., Gumaei, A., Derhab, A. & Hussain, A., (2019), TSDL : A TwoStage Deep Learning Model for Efficient Network Intrusion Detection, *IEEE Access*, 1-1, https://doi.org/10.1109/ACCESS.2019.2899721

Kim, J., Shin, Y. & Choi, E., (2019), An Intrusion Detection Model based on a Convolutional Neural Network, *Journal of Multimedia Information System*, *6 4*, 165-172, https://doi.org/10.33851/JMIS.2019.6.4.165

Kingma, D. P. & Ba, J., (2015), Adam : A Method for Stochastic Optimization, *3rd International Conference for Learning Representations*.

Kjaerland, M., (2006), A taxonomy and comparison of computer security incidents from the commercial and government sectors, *Computers & Security*, *25 7*, 522-538, https://doi.org/10.1016/j.cose.2006.08.004

Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S., (2017), Self-Normalizing Neural Networks, *Advances in Neural Information Processing Systems*, 971-980.

Knuth, D. E., (1998), *The Art of Computer Programming, Volume 3 : (2nd Ed.) Sorting and Searching*, Addison Wesley Longman Publishing Co., Inc.

Kong, X., Wang, C., Sun, S. & Gao, F., (2012), A Method of Digital to Shaft-Angle Converting Using ARM Series MCU & CPLD, *2012 Fifth International Symposium on Computational Intelligence and Design*, *2*, 262-265, https://doi.org/10.1109/ISCID.2012.217

Koroniotis, N., Moustafa, N., Sitnikova, E. & Turnbull, B., (2019), Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics : Bot-IoT dataset, *Future Generation Computer Systems*, *100*, 779-796, https://doi.org/10.1016/j.future.2019.05.041

Krishna, R. R., Priyadarshini, A., Jha, A. V., Appasani, B., Srinivasulu, A. & Bizon, N., (2021), State-of-the-Art Review on IoT Threats and Attacks : Taxonomy, Challenges and Solutions, *Sustainability*, *13 16*, https://doi.org/10.3390/su13169463

Kruegel, C., Valeur, F. & Vigna, G., (2004), *Intrusion detection and correlation : challenges and solutions* (T. 14), Springer Science & Business Media, https://doi.org/https://doi.org/10.1007/b101493

Kumar, A. C., (2009), Analysis of unsupervised dimensionality reduction techniques, *Computer science and information systems*, *6 2*, 217-227.

Kumar, S., Dalal, S. & Dixit, V., (2014), The OSI model : Overview on the seven layers of computer networks, *International Journal of Computer Science and Information Technology Research*, *2 3*, 461-466.

Kurachi, R., Takada, H., Tanabe, M., Anzai, J., Takei, K., Iinuma, T., Maeda, M. & Matsushima, H., (2018), Improving secure coding rules for automotive software by using a vulnerability database, *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 1-8, https://doi.org/10.1109/ICVES.2018.8519496

Lai, J., Wu, J., Chen, S., Wu, C. & Yang, C., (2008), Designing a Taxonomy of Web Attacks, *International Conference on Convergence and Hybrid Information Technology (ICHIT)*, 278-282, https://doi.org/10.1109/ICHIT.2008.280

Lashkari, A. H., Kadir, A. F. A., Taheri, L. & Ghorbani, A. A., (2018), Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification, *2018 International Carnahan Conference on Security Technology (ICCST)*, 1-7, https://doi.org/10.1109/CCST.2018.8585560

Lashkari, A. H., Gil, G. D., Mamun, M. S. I. & Ghorbani, A. A., (2017), Characterization of Tor Traffic using Time based Features, *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1 : ICISSP,* 253-262, https://doi.org/10.5220/0006105602530262

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P., (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, *86 11*, 2278-2324, https://doi.org/10.1109/5.726791

Lee, E. A., (2007), *Computing Foundations and Practice for Cyber-Physical Systems : A Preliminary Report* (rapp. tech. UCB/EECS-2007-72), EECS Department, University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-72.html

Leevy, J. L. & Khoshgoftaar, T. M., (2020), A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data, *Journal of Big Data*, *7 1*, 104, https://doi.org/10.1186/s40537-020-00382-x

Lehtinen, S. & Lonvick, C., (2006), *The Secure Shell (SSH) Protocol Assigned Numbers* (RFC N° 4250), RFC Editor, http://www.rfc-editor.org/rfc/rfc4250.txt

Li, F., Jiang, Q., Zhang, S., Wei, M. & Song, R., (2019), Robot skill acquisition in assembly process using deep reinforcement learning [Deep Learning for Intelligent Sensing, Decision-Making and Control], *Neurocomputing*, *345*, 92-102, https://doi.org/10.1016/j.neucom.2019.01.087

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H. & Zhao, W., (2017), A Survey on Internet of Things : Architecture, Enabling Technologies, Security and Privacy, and Applications, *IEEE Internet of Things Journal*, *4 5*, 1125-1142, https://doi.org/10.1109/JIOT.2017.2683200

Lin, W., Lin, H., Wang, P., Wu, B. & Tsai, J., (2018), Using Convolutional Neural Networks to Network Intrusion Detection for Cyber Threats, *IEEE International Conference on Applied System Invention (ICASI)*, 1107-1110, https://doi.org/10.1109/ICASI.2018.8394474

Madakam, S., Lake, V., Lake, V., Lake, V. et al., (2015), Internet of Things (IoT) : A literature review, *Journal of Computer and Communications*, *3 05*, 164.

Marikyan, D., Papagiannidis, S. & Alamanos, E., (2019), A systematic review of the smart home literature : A user perspective, *Technological Forecasting and Social Change*, *138*, 139-154, https://doi.org/10.1016/j.techfore.2018.08.015

Marwedel, P., (2021), *Embedded System Design : Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things* (4th), Springer Publishing Company, Incorporated, https://doi.org/10.1007/978-3-030-60910-8

Matthews, B., (1975), Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme, *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405 2*, 442-451, https://doi.org/10.1016/0005-2795(75)90109-9

McGregor, G., (1992), *The PPP Internet Protocol Control Protocol (IPCP)* (RFC N° 1332), RFC Editor, http://www.rfc-editor.org/rfc/rfc1332.txt

McHugh, J., (2000), Testing Intrusion Detection Systems : A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory, *ACM Trans. Inf. Syst. Secur.*, *3 4*, 262-294, https://doi.org/10.1145/382912.382923

Mehta, D. P. & Sahni, S., (2004), *Handbook Of Data Structures And Applications (Chapman & Hall/Crc Computer and Information Science Series.)*, Chapman & Hall/CRC.

Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A. & Payne, B. D., (2015), Evaluating Computer Intrusion Detection Systems : A Survey of Common Practices, *ACM Comput. Surv.*, *48 1*, https://doi.org/10.1145/2808691

Miller, C., (2019), Lessons learned from hacking a car, *IEEE Design & Test*, *36 6*, 7-9, https://doi.org/10.1109/MDAT.2018.2863106

Miller, P. E., Pawar, S., Vaccaro, B., McCullough, M., Rao, P., Ghosh, R., Warier, P., Desai, N. R. & Ahmad, T., (2019), Predictive Abilities of Machine Learning Techniques May Be Limited by Dataset Characteristics : Insights From the UNOS Database, *Journal of Cardiac Failure*, *25 6*, 479-483, https://doi.org/https://doi.org/10.1016/j.cardfail.2019.01.018

Ming Gao, Kenong Zhang & Jiahua Lu, (2006), Efficient packet matching for gigabit network intrusion detection using TCAMs, *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, *2*, 6 pp.-254, https://doi.org/10.1109/AINA.2006.164

Mitchell, T., (1997), *Machine Learning*, McGraw-Hill Professional.

Moustafa, N. & Slay, J., (2015), UNSW-NB15 : A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set), *Military Communications and Information Systems Conference (MilCIS)*, 1-6, https://doi.org/10.1109/MilCIS.2015.7348942

Moustafa, N. & Slay, J., (2016), The evaluation of Network Anomaly Detection Systems : Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, *Information Security Journal : A Global Perspective*, *25 1-3*, 18-31, https://doi.org/10.1080/19393555.2015.1125974

Murphy, K. P., (2012), *Machine Learning : A Probabilistic Perspective*, MIT Press.

Murtagh, F. & Contreras, P., (2017), Algorithms for hierarchical clustering : an overview, II, *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, *7 6*, e1219.

Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K. & Xiao, Y., (2015), Network forensics analysis using Wireshark, *International Journal of Security and Networks*, *10 2*, 91-106, https://doi.org/10.1504/IJSN.2015.070421

Nesterov, Y., (1983), A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2), *Doklady an ussr*, *269*, 543-547.

NXP, (2021), i.MX RT1170 CROSSOVER MCUs Ushering in the GHZ ERA, Récupérée 8 janvier 2021, à partir de https://www.nxp.com/docs/en/fact-sheet/i.MX-RT1170-FS.pdf

Omran, M. G., Engelbrecht, A. P. & Salman, A., (2007), An overview of clustering methods, *Intelligent Data Analysis*, *11 6*, 583-605.

OWASP, (2017), OWASP top ten 2017, Récupérée 24 juillet 2022, à partir de https://owasp.org/www-project-top-ten/OWASP%5C_Top%5C_Ten%5C_2017/

Palattella, M. R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L. A., Boggia, G. & Dohler, M., (2013), Standardized Protocol Stack for the Internet of (Important) Things, *IEEE Communications Surveys Tutorials*, *15 3*, 1389-1406, https://doi.org/10.1109/SURV.2012.111412.00158

Patterson, D., (2018), 50 Years of computer architecture : From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set, *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 27-31, https://doi.org/10.1109/ISSCC.2018.8310168

Paullada, A., Raji, I. D., Bender, E. M., Denton, E. & Hanna, A., (2021), Data and its (dis)contents : A survey of dataset development and use in machine learning research, *Patterns*, *2 11*, 100336, https://doi.org/https://doi.org/10.1016/j.patter.2021.100336

Paxson, V., (1999), Bro : a system for detecting network intruders in real-time, *Computer Networks*, *31 23*, 2435-2463, https://doi.org/10.1016/S1389-1286(99)00112-7

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al., (2011), Scikit-learn : Machine learning in Python, *the Journal of machine Learning research*, *12*, 2825-2830.

Perwej, D., Qamar Abbas, S., Pratap Dixit, J., Akhtar, D. N. & Kumar Jaiswal, A., (2021), A Systematic Literature Review on the Cyber Security, *International Journal of scientific research and management*, *9 12*, 669-710, https://doi.org/10.18535/ijsrm/v9i12.ec04

Picard, S., Chapdelaine, C., Cappi, C., Gardes, L., Jenn, E., Lefevre, B. & Soumarmon, T., (2020), Ensuring Dataset Quality for Machine Learning Certification, *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 275-282, https://doi.org/10.1109/ISSREW51248.2020.00085

Platt, J., (1998), *Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines* (rapp. tech. MSR-TR-98-14), Microsoft, https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/

Plummer, D. C., (1982), *Ethernet Address Resolution Protocol : Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware* (STD N° 37), RFC Editor, http://www.rfc-editor.org/rfc/rfc826.txt

Polyak, B. T., (1964), Some methods of speeding up the convergence of iteration methods, *Ussr computational mathematics and mathematical physics*, *4 5*, 1-17.

Postel, J., (1980), *User Datagram Protocol* (STD N° 6), RFC Editor, http://www.rfc-editor.org/rfc/rfc768.txt

Postel, J., (1981a), *Internet Control Message Protocol* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc792.txt

Postel, J. & Reynolds, J., (1985), *File Transfer Protocol* (STD N° 9), RFC Editor, http://www.rfc-editor.org/rfc/rfc959.txt

Postel, J., (1981b), *Internet Protocol* (STD N° 5), RFC Editor, http://www.rfc-editor.org/rfc/rfc791.txt

Postel, J., (1981c), *Transmission Control Protocol* (STD N° 7), RFC Editor, http://www.rfc-editor.org/rfc/rfc793.txt

Prasad, M. D., V, P. B. & Amarnath, C., (2019), Machine Learning DDoS Detection Using Stochastic Gradient Boosting, *International Journal of Computer Sciences and Engineering*, *7*, 157-166, https://doi.org/10.26438/ijcse/v7i4.157166

Pugh, W., (1990a), *A Skip List Cookbook* (rapp. tech.), USA, University of Maryland at College Park.

Pugh, W., (1990b), Skip Lists : A Probabilistic Alternative to Balanced Trees, *Commun. ACM*, *33 6*, 668-676, https://doi.org/10.1145/78973.78977

Quinlan, J. R., (1986), Induction of decision trees, *Machine learning*, *1 1*, 81-106.

Quinlan, J. R., (1993), *C4.5 : Programs for Machine Learning*, Morgan Kaufmann.

Raileanu, L. E. & Stoffel, K., (2004), Theoretical comparison between the gini index and information gain criteria, *Annals of Mathematics and Artificial Intelligence*, *41 1*, 77-93, https://doi.org/10.1023/B:AMAI.0000018580.96245.c6

Ramakrishnan, K., Floyd, S. & Black, D., (2001), *The Addition of Explicit Congestion Notification (ECN) to IP* (RFC N° 3168), RFC Editor, http://www.rfc-editor.org/rfc/rfc3168.txt

Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A., (2016), XNOR-Net : ImageNet Classification Using Binary Convolutional Neural Networks, In B. Leibe, J. Matas, N. Sebe & M. Welling (Éd.), *Computer Vision – ECCV 2016* (p. 525-542), Springer International Publishing.

Rayes, A. & Salam, S., (2017), The Internet in IoT—OSI, TCP/IP, IPv4, IPv6 and Internet Routing. *Internet of Things From Hype to Reality : The Road to Digitization* (p. 35-56), Springer International Publishing, https://doi.org/10.1007/978-3-319-44860-2_2

Redfern, A. J., Zhu, L. & Newquist, M. K., (2021), BCNN : A Binary CNN With All Matrix Ops Quantized to 1 Bit Precision, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4604-4612.

Renault, V., Carlier, F. & Bourdon, P., (2014), TIFAIFAI : Conception de Nouveaux Espaces d'Interactions pour Apprendre [(Accessed Mar 14, 2022)], *Distances et médiations des savoirs*, https://doi.org/10.4000/dms.588

Rescorla, E., (2018), *The Transport Layer Security (TLS) Protocol Version 1.3* (RFC N° 8446), RFC Editor, http://www.rfc-editor.org/rfc/rfc8446.txt

Rizvi, S., Kurtz, A., Pfeffer, J. & Rizvi, M., (2018), Securing the Internet of Things (IoT) : A Security Taxonomy for IoT, *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering*, 163-168, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00034

Roesch, M., (1999), Snort - Lightweight Intrusion Detection for Networks, *Proceedings of the 13th USENIX Conference on System Administration*, 229-238, https://doi.org/10.5555/1039834.1039864

Rokach, L. & Maimon, O., (2005), Top-down induction of decision trees classifiers - a survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *35 4*, 476-487, https://doi.org/10.1109/TSMCC.2004.843247

Romkey, J., (2017), Toast of the IoT : The 1990 Interop Internet Toaster, *IEEE Consumer Electronics Magazine*, *6 1*, 116-119, https://doi.org/10.1109/MCE.2016.2614740

Rosenblatt, F., (1958), The Perceptron : A Probabilistic Model for Information Storage and Organization in The Brain, *Psychological Review*, 65-386.

Samuel, A. L., (1959), Some Studies in Machine Learning Using the Game of Checkers, *IBM Journal of Research and Development*, *3 3*, 210-229, https://doi.org/10.1147/rd.33.0210

Satam, P., Satam, S. & Hariri, S., (2018), Bluetooth Intrusion Detection System (BIDS), *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, 1-7, https://doi.org/10.1109/AICCSA.2018.8612809

Segars, S., (1997), ARM7TDMI power consumption, *IEEE Micro*, *17 4*, 12-19, https://doi.org/10.1109/40.612178

Serpanos, D., (2019), There Is No Safety Without Security and Dependability, *Computer*, *52 6*, 78-81, https://doi.org/10.1109/MC.2019.2903360

Shaffer, C., (2011), *Data Structures and Algorithm Analysis*, Dover Publications.

Shah, D. K., B., C. R., Sen, D. & Goyal, S., (2015), Vehicle parking system implementation using CPLD, *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, 1-5, https://doi.org/10.1109/ICCICT.2015.7045668

Sharafaldin, I., Lashkari, A. H., Hakak, S. & Ghorbani, A. A., (2019), Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,

International Carnahan Conference on Security Technology (ICCST), 1-8, https://doi.org/10.1109/CCST.2019.8888419

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., (2018), Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, *1*, 108-116, https://doi.org/10.5220/0006639801080116

Sharma, N., Shamkuwar, M. & Singh, I., (2019), The History, Present and Future with IoT, In V. E. Balas, V. K. Solanki, R. Kumar & M. Khari (Éd.), *Internet of Things and Big Data Analytics for Smart Generation* (p. 27-51), Springer International Publishing, https://doi.org/10.1007/978-3-030-04203-5_3

Sharma, N. V. & Yadav, N. S., (2021), An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers, *Microprocessors and Microsystems*, *85*, 104293, https://doi.org/10.1016/j.micpro.2021.104293

Shiravi, A., Shiravi, H., Tavallaee, M. & Ghorbani, A. A., (2012), Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection, *Comput. Secur.*, *31 3*, 357-374, https://doi.org/10.1016/j.cose.2011.12.012

Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A. & Mishra, A. K., (2018), Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools, *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 318-322, https://doi.org/10.1109/ICACCI.2018.8554590

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al., (2017), Mastering the game of go without human knowledge, *nature*, *550 7676*, 354-359.

Simmons, C. B., Ellis, C., Shiva, S., Dasgupta, D. & Wu, Q., (2009), AVOIDIT : A Cyber Attack Taxonomy, *CTIT technical reports series*.

Sinha, P., Boukhtouta, A., Belarde, V. H. & Debbabi, M., (2010), Insights from the analysis of the Mariposa botnet, *Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS)*, 1-9, https://doi.org/10.1109/CRISIS.2010.5764915

Smith, S., (2020), Neon Coprocessor. *Programming with 64-Bit ARM Assembly Language : Single Board Computer Development for Raspberry Pi and Mobile Devices* (p. 291-306), Apress, https://doi.org/10.1007/978-1-4842-5881-1_13

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R., (2014), Dropout : a simple way to prevent neural networks from overfitting, *The journal of machine learning research*, *15 1*, 1929-1958.

Stewart, R., (2007), *Stream Control Transmission Protocol* (RFC N° 4960), RFC Editor, http://www.rfc-editor.org/rfc/rfc4960.txt

STMicroelectronics, (2020), STM32MP1 Series Microprocessors, Récupérée 3 août 2022, à partir de https://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/flyer/group0/1f/4c/20/0a/43/94/4f/70/flstm32mp/files/flstm32mp.pdf/jcr:content/translations/en.flstm32mp.pdf

STMicroelectronics, (2022), Data Brief - Stellar SR6 G7 line, Récupérée 26 juillet 2022, à partir de https://www.st.com/resource/en/data_brief/sr6g7c4.pdf

Suresh, P., Daniel, J. V., Parthasarathy, V. & Aswathy, R. H., (2014), A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment, *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, 1-8, https://doi.org/10.1109/ICSEMR.2014.7043637

Sutton, R. S. & Barto, A. G., (2018), *Reinforcement Learning : An Introduction* (2e éd.), Bradford Books.

Taheri, L., Kadir, A. F. A. & Lashkari, A. H., (2019), Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls, *International Carnahan Conference on Security Technology (ICCST)*, 1-8, https://doi.org/10.1109/CCST.2019.8888430

Tang, H., (2019), A new method of bottleneck analysis for manufacturing systems, *Manufacturing Letters*, *19*, 21-24, https://doi.org/10.1016/j.mfglet.2019.01.003

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R. & Ghogho, M., (2016), Deep learning approach for Network Intrusion Detection in Software Defined Networking, *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 258-263, https://doi.org/10.1109/WINCOM.2016.7777224

Tartaglione, E., Lepsøy, S., Fiandrotti, A. & Francini, G., (2018), Learning sparse neural networks via sensitivity-driven regularization, In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett (Éd.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2018/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A., (2009), A Detailed Analysis of the KDD CUP 99 Data Set, *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6, https://doi.org/10.1109/CISDA.2009.5356528

Trammell, B. & Boschi, E., (2008), *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)* (STD N° 5103), RFC Editor, http://www.rfc-editor.org/rfc/rfc5103.txt

Ullah, I. & Mahmoud, Q. H., (2019), A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks, *16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 1-6, https://doi.org/10.1109/CCNC.2019.8651782

Ullah, I. & Mahmoud, Q. H., (2020a), A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks, In C. Goutte & X. Zhu (Éd.), *Advances in Artificial Intelligence* (p. 508-520), Springer International Publishing.

Ullah, I. & Mahmoud, Q. H., (2020b), A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks, In C. Goutte & X. Zhu (Éd.), *Advances in Artificial Intelligence* (p. 508-520), Springer International Publishing, https://doi.org/10.1007/978-3-030-47358-7_52

Ustebay, S., Turgut, Z. & Aydin, M. A., (2018), Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier, *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 71-76, https://doi.org/10.1109/IBIGDELFT.2018.8625318

Van, H. P., (2021), *State of IoT - summer report*. Récupérée 26 août 2022, à partir de https://iot-analytics.com/product/state-of-iot-summer-2021/

van Veen, F., (2016), The neural network zoo, https://www.asimovinstitute.org/neural-network-zoo/

Verma, A. & Ranga, V., (2019a), ELNIDS : Ensemble Learning based Network Intrusion Detection System for RPL based Internet of Things, *2019 4th International Conference on Internet of Things : Smart Innovation and Usages (IoT-SIU)*, 1-6, https://doi.org/10.1109/IoT-SIU.2019.8777504

Verma, A. & Ranga, V., (2019b), Evaluation of Network Intrusion Detection Systems for RPL Based 6LoWPAN Networks in IoT, *Wireless Personal Communications*, *108 3*, 1571-1594, https://doi.org/10.1007/s11277-019-06485-w

V.Harika Naidu, M. V., Shreya Valluri, (2018), Exposure Of Sql Injection In Packet Stream, *International Journal of Engineering and Computer Science*, *5 11*.

Vignau, B., Khoury, R., Hallé, S. & Hamou-Lhadj, A., (2021), The evolution of IoT Malwares, from 2008 to 2019 : Survey, taxonomy, process simulator and perspectives, *Journal of Systems Architecture*, *116*, 102143, https://doi.org/10.1016/j.sysarc.2021.102143

Vinayakumar, R., Soman, K. P. & Poornachandran, P., (2017), Applying Convolutional Neural Network for Network Intrusion Detection, *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1222-1228, https://doi.org/10.1109/ICACCI.2017.8126009

Völske, M., Bevendorff, J., Kiesel, J., Stein, B., Fröbe, M., Hagen, M. & Potthast, M., (2021), Web Archive Analytics, In R. H. Reussner, A. Koziolek & R. Heinrich (Éd.), *INFORMATIK 2020* (p. 61-72), Gesellschaft für Informatik, Bonn, https://doi.org/10.18420/inf2020_05

Welford, B., (1962), *Technometrics*, *4 3*, 419-420.

Wikipedia, (2017), internet Mix - Wikipedia, Récupérée 15 juillet 2022, à partir de https://en.wikipedia.org/wiki/Internet_Mix

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J. & Alexander, R., (2012), *RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks* (RFC N° 6550), RFC Editor, http://www.rfc-editor.org/rfc/rfc6550.txt

Wolpert, D. H., (1996), The Lack of A Priori Distinctions Between Learning Algorithms, *Neural Computation*, *8 7*, 1341-1390, https://doi.org/10.1162/neco.1996.8.7.1341

Wright, G. R. & Stevens, W. R., (1995), *TCP/IP Illustrated, Volume 2 (paperback) : The Implementation*, Addison-Wesley Professional.

Xu, H., Yu, W., Griffith, D. & Golmie, N., (2018), A Survey on Industrial Internet of Things : A Cyber-Physical Systems Perspective, *IEEE Access*, *6*, 78238-78259, https://doi.org/10.1109/ACCESS.2018.2884906

Xu, J., Wang, B., Li, J., Hu, C. & Pan, J., (2017), Deep learning application based on embedded GPU, *2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS)*, 1-4, https://doi.org/10.1109/EIIS.2017.8298723

Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y. & Sabrina, F., (2021), Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset, *IEEE Access*, *9*, 140136-140146, https://doi.org/10.1109/ACCESS.2021.3116612

Yadav, T. & Rao, A. M., (2015), Technical Aspects of Cyber Kill Chain, In J. H. Abawajy, S. Mukherjea, S. M. Thampi & A. Ruiz-Martínez (Éd.), *Security in Computing and Communications* (p. 438-452), Springer International Publishing.

Ylonen, T. & Lonvick, C., (2006a), *The Secure Shell (SSH) Authentication Protocol* (RFC N° 4252), RFC Editor, http://www.rfc-editor.org/rfc/rfc4252.txt

Ylonen, T. & Lonvick, C., (2006b), *The Secure Shell (SSH) Connection Protocol* (RFC N° 4254), RFC Editor, http://www.rfc-editor.org/rfc/rfc4254.txt

Ylonen, T. & Lonvick, C., (2006c), *The Secure Shell (SSH) Protocol Architecture* (RFC N° 4251), RFC Editor, http://www.rfc-editor.org/rfc/rfc4251.txt

Ylonen, T. & Lonvick, C., (2006d), *The Secure Shell (SSH) Transport Layer Protocol* (RFC N° 4253), RFC Editor, http://www.rfc-editor.org/rfc/rfc4253.txt

Zhou & Chellappa, (1988), Computation of optical flow using a neural network, *IEEE 1988 International Conference on Neural Networks*, 71-78 vol.2, https://doi.org/10.1109/ICNN.1988.23914

Zhu, F., Gong, R., Yu, F., Liu, X., Wang, Y., Li, Z., Yang, X. & Yan, J., (2020), Towards Unified INT8 Training for Convolutional Neural Network, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhu, K. & Zhang, T., (2021), Deep reinforcement learning based mobile robot navigation : A review, *Tsinghua Science and Technology*, *26 5*, 674-691, https://doi.org/10.26599/TST.2021.9010012

Zhu, X. & Goldberg, A., (2009), *Introduction to semi-supervised learning*, Springer.

PUBLICATIONS

Revues scientifiques avec comité de lecture

Rosay, A., Riou, K., Carlier, F. & Leroux, P., (2022), Multi-layer perceptron for network intrusion detection, *Annals of Telecommunications*, *77*5, 371-394, https://doi.org/10.1007/s12243-021-00852-0

Conférences internationales avec comité de lecture

Carlier., F., Fresse., V., Jamont., J., Pallardy., L. & Rosay., A., Pooling of Heterogeneous Computing Resources : A Novel Approach based on Multi-Edge-Agent Concept, in : *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1 : ICAART,* INSTICC, SciTePress, 2020, 185-192, isbn : 978-989-758-395-7, https://doi.org/10.5220/0008954301850192.

Carlier, F., Fresse, V., Jamont, J.-P., Rosay, A. & Pallardy, L., (2020b), A Multi-Edge-Agent System approach for sharing heterogeneous computing resources, *2020 IEEE 92nd Vehicular Technology Conference*, https://doi.org/10.1109/VTC2020-Fall49728.2020.9348722

Rosay, A., Carlier, F., Cheval, E. & Leroux, P., (2021), From CIC-IDS2017 to LYCOS-IDS2017 : A Corrected Dataset for Better Performance, *IEEE/WIC/ACM International Conference on Web Intelligence*, 570-575, https://doi.org/10.1145/3486622.3493973

Rosay, A., Carlier, F. & Leroux, P., (2020a), Feed-forward neural network for Network Intrusion Detection, *2020 IEEE 91st Vehicular Technology Conference*, https://doi.org/10.1109/VTC2020-Spring48590.2020.9129472

Rosay, A., Carlier, F. & Leroux, P., (2020c), MLP4NIDS : An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset, In S. Boumerdassi, É. Renault & P. Mühlethaler (Éd.), *Machine Learning for Networking* (p. 240-254), Springer International Publishing, https://doi.org/10.1007/978-3-030-45778-5_16

Rosay, A., Cheval, E., Carlier, F. & Leroux, P., Network Intrusion Detection : A Comprehensive Analysis of CIC-IDS2017 [Best student paper award], in : *Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP,* Best student paper award, INSTICC, SciTePress, 2022, 25-36, isbn : 978-989-758-553-1, https://doi.org/10.5220/0010774000003120.

Conférences nationales avec comité de lecture

Rosay, A., Carlier, F. & Leroux, P., (2020b), MLP4NIDS : Application pratique d'un réseau de neurones pour la détection d'intrusions réseau dans les voitures connectées, *Applications Pratiques de l'Intelligence Artificielle (APIA 2020)*, 27-34, http://pfia2020.fr/wp-content/uploads/2020/08/Actes_CH_PFIA2020_V3.pdf

Brevets

Rosay, A., Carlier, F. & Leroux, P., (2022, juillet 22), *Network anomalies detection method* (pareqfr N° 2207253).

Dépôt open source

Rosay, A. & Carlier, F., (2022), Intrusion Detection Evaluation Dataset (LYCOS-IDS2017), Récupérée 26 juillet 2022, à partir de https://lycos-ids.univ-lemans.fr/

DISTINCTION

INSTICC
Institute for Systems and Technologies of Information, Control and Communication

Best Student Paper Award Certificate

for the paper entitled:

Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017

authored by:

Arnaud Rosay, Eloïse Cheval, Florent Carlier and Pascal Leroux

received at the

8th International Conference on Information Systems Security and Privacy (ICISSP)

held from February 9 - 11, 2022

On behalf of the Organizing Committee,

Steven Furnell
ICISSP Conference Chair

e-mail: secretariat@insticc.org website: http://www.insticc.org

Abréviations, sigles et acronymes

AE autoencoder, auto-encodeur. 91, 95

ARP Address Resolution Protocol, protocole de résolution d'adresse. 40, 142

CDP Cisco Discovery Protocol, protocole propriétaire de rôle similaire à LLDP. 40, 142

CFM Fichier CSV de caractéristiques extraites par CICFlowMeter sur les PCAP de CIC-IDS2017. 140, 141

CIC Canadian Institute for Cybersecurity, institut canadien pour la cybersécurité de l'Université du Nouveau-Brunswick, Canada. 64, 68, 123

CNN Convolutional Neural Network, réseau de neurones convolutif. 8, 91, 93–95, 98, 113, 114, 130, 161

CPLD Complex Programmable Logic Device, composant à logique programmable complexe. 29

CPS Cyber-Physical System, système cyber-physique. 28

CPU Central Processing Unit, unité centrale de traitement. 30, 170, 185, 191

CSV Comma-Separated Values, format texte représentant des données tabulaires sous forme de valeurs séparées par des virgules. 63, 64, 68, 123, 157

CV Coefficient de variation ou écart-type relatif. 155

CVE Common Vulnerabilities and Exposures, liste publique de failles de sécurité informatique. 64

DDoS Distributed Denial of Service, déni de service distribué. 55, 56, 123, 164, 167

DNS Domain Name System, service informatique distribué utilisé pour traduire les noms de domaine internet en adresse IP. 14

DoS Denial of Service, déni de service. 55, 56, 122, 129, 164, 167

DSP Digital Signal Processor, processeur de traitement du signal. 30

DT Decision Tree, arbre de décision. 87, 112, 127, 129, 130, 150, 153, 154

ECC Elliptic Curve Cryptography, cryptographie à courbe elliptique. 193

FGPA Field Programmable Gate Arrays, réseaux de portes programmables in situ. 29

FTP File Transfer Protocol, protocole de transfert de fichier. 46, 63, 130, 152, 167, 184

GPU Graphics Processing Unit, unité de traitements graphiques. 30, 33

GRU Gated Recurrent Unit, unité récurrente à porte. 93

HSM Hardware Security Module, module matériel de sécurité. 32, 35

HTTP HyperText Transfer Protocol, protocole de transfert hypertexte. 46, 63, 193

HTTPS HyperText Transfer Protocol Secure, protocole de transfert hypertexte sécurisé. 47, 63, 107

IA Intelligence Artificielle. 77

ICMP Internet Control Message Protocol, protocole utilisé pour véhiculer des messages de contrôle et d'erreur. 40, 42, 142

IDS Intrusion Detection System, système de détection d'intrusions. 18, 58, 59, 61, 130, 131, 165, 183–186, 190, 194

IGMP Internet Group Management Protocol, protocole qui permet à des routeurs IP de déterminer de façon dynamique les groupes multicast. 40, 42, 43, 142

IMIX Internet Mix, profils de trafic réseau sur Internet. 169

IoT Internet of Things, internet des objets. 13, 27, 28

IP Internet Protocol, protocole internet. 40, 42, 60, 68, 106, 107, 111, 122, 123, 131, 142, 144, 150, 158, 189, 190

ISCX Fichier CSV de caractéristiques de CIC-IDS2017. 139–141, 143, 144, 150

k-NN k-Nearest Neighbors, k plus proches voisins. 86, 87, 98, 112, 120, 129, 150, 153, 154

LDA Linear Discriminant Analysis, analyse discriminante linéaire. 83–85, 98, 112, 120, 150, 153

LLDP Link Layer Discovery Protocol, protocole de découverte des topologies réseau. 40, 142, 221

LPWAN Low Power Wide Area Network, réseau à faible consommation d'énergie. 16, 61

LSTM Long Short Term Memory, mémoire à court et long terme. 93

MAC Medium Acces Control, contrôle d'accès au médium. 40

MCU Microcontroller, microcontrôleur. 8, 31, 32, 35, 183, 184, 192

ML Machine Learning, apprentissage automatique. 18, 77

MLP Multi-Layer Perceptron, perceptron multi-couches. 10, 91, 94, 95, 98, 113, 115, 116, 118–122, 125, 127, 129, 132, 150, 151, 153, 154, 164, 168, 191

PAN Personal Area Network, réseau personnel. 61

PCAP Packet CAPture, format d'enregistrement des trames réseau compatible avec les outils Wireshark, TCPDUMP. 63, 64, 67, 68, 123, 138–140, 142, 144, 145, 148, 157

PPP Point-to-Point Protocol, protocole point à point. 40

QDA Quadratic Discriminant Analysis, analyse discriminante quadratique. 84, 85, 98, 112, 120, 127

RARP Reverse Address Resolution Protocol, protocole de résolution d'adresse inverse. 40

RF Random Forest, forêt aléatoire. 88, 112, 127, 129, 150–154

RFC Request for Comments, Requête de commentaires. 36

RFE Recursive Feature Elimination, élimination récursive des caractéristiques. 151, 152

RNN Recurrent Neural Network, réseau de neurones récurrents. 91–93, 95, 98, 113, 114, 130

SCTP Stream Control Transmission Protocol, protocole de transmission de contrôle de flux. 43, 46

SoC System on chip, système sur puce. 32, 33, 35, 165

SQL Structured Query Language, Langage de requêtes structurées. 56, 109, 122, 124, 131, 194

SSH Secure SHell, protocole de communication sécurisé. 17, 46, 47, 63, 130, 152, 167, 184, 193

SVM Support Vector Machine, machine à vecteurs de support. 85–87, 98, 112, 120, 130

TCP Transmission Control Protocol, protocole de contrôle de tranmission. 43, 44, 46, 47, 60, 61, 140–145, 149, 150, 153, 155, 157, 164, 168

TF TensorFlow, outil d'apprentissage automatique développé par Google. 116, 167

TF-Lite TensorFlow-Lite, version allégée de TensorFlow. 167, 170, 184

TLS Transport Layer Security, sécurité de la couche de transport. 47

UDP User Datagram Protocol, protocole de datagramme utilisateur. 43, 44, 46, 142, 145, 168

XSS Cross-site scripting, script intersite. 56

COLLEGE DOCTORAL PAYS DE LA LOIRE
EDUCATION, COGNITION, LANGAGES, INTERACTIONS, SANTE

Le Mans Université

Titre : Détection d'intrusions dans les objets connectés par des techniques d'apprentissage automatique : étude dans les domaines de l'éducation et des voitures connectées

Mot clés : Détection d'intrusions réseau, objets connectés, apprentissage automatique, réseaux de neurones

Résumé : Alors que les objets connectés deviennent omniprésents, la détection des intrusions dans les réseaux devient plus critique que jamais. Nous proposons une étude sur la détection d'intrusions réseau à base d'apprentissage automatique dans le domaine de l'automobile et de l'éducation, deux champs *a priori* très différents mais ayant en commun la notion d'objet connecté. Les voitures connectées comme les dispositifs numériques pour l'éducation tels qu'un mur d'écrans sont des objets connectés, des victimes potentielles pouvant soit mettre danger la vie des passagers, soit afficher du contenu inapproprié. Face à ces menaces, nous proposons l'étude de détection d'intrusions réseau basée sur les conversations et amenons trois contributions. Tout d'abord, nous montrons que les réseaux de neurones obtiennent des performances de détection similaires à celles des algorithmes classiques. Nous proposons ensuite une méthodologie de fiabilisation de corpus numériques de détection d'intrusions permettant de simplifier le modèle de détection sans diminution de performance. Enfin, une étude du système complet de détection dans les objets connectés met en évidence l'existence d'un goulet d'étranglement. Grâce à une proposition sur la recherche efficace de conversations en cours, nous le supprimons et démontrons ainsi que la détection d'intrusion sur les objets connectés est réalisable, même quand ils sont contraints en ressources.

Title: Network intrusion detection in IoT devices using machine learning techniques: a study in education and connected cars

Keywords: Network intrusion detection, IoT devices, machine learning, neural networks

Abstract: As connected objects become ubiquitous, network intrusion detection becomes more critical than ever. We propose a study on network intrusion detection based on machine learning in the automotive and education domains, two very different fields but having in common the notion of connected object. Both connected cars and digital devices for education, such as a screen wall, are connected objects, potential victims that can either endanger the lives of passengers or display inappropriate content. To address these threats, we propose the study of flow-based network intrusion detection and make three contributions. First, we show that neural networks achieve detection performances similar to those of classical algorithms. Second, we propose a methodology for the reliability of network intrusion detection dataset that allows the simplification of the detection model without any decrease in performance. Finally, a study of the complete detection system in connected objects highlights the existence of a bottleneck. Thanks to a proposal on the efficient search of ongoing flows, we remove this bottleneck and thus demonstrate that intrusion detection on connected objects is feasible, even when they are resource constrained.