Titre: Title:	Classification non supervisée de cyberattaques par inspection de paquets
Auteur: Author:	Victor Aurora
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Aurora, V. (2022). Classification non supervisée de cyberattaques par inspection de paquets [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10711/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10711/
Directeurs de recherche: Advisors:	Frédéric Cuppens, & Nora Boulahia Cuppens
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Classification non supervisée de cyberattaques par inspection de paquets

VICTOR AURORA
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Décembre 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Classification non supervisée de cyberattaques par inspection de paquets

présenté par **Victor AURORA**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Martine BELLAÏCHE, présidente
Frédéric CUPPENS, membre et directeur de recherche
Nora BOULAHIA CUPPENS, membre et codirectrice de recherche
Omar ABDUL WAHAB, membre

DÉDICACE

À ma famille et mes amis qui m'ont soutenu depuis le début...

REMERCIEMENTS

Merci à mes directeurs de recherche, Frédéric Cuppens et Nora Boulahia-Cuppens, de m'avoir accompagné tout au long de ma maîtrise pour ce projet.

Merci à Christopher Neal d'avoir également encadré mes travaux, notamment pour l'écriture d'un article de recherche.

Merci à Thales Digital Solutions, sans qui ce projet n'aurait pas pu voir le jour. Plus particulièrement, merci à Alexandre Proulx, qui a été comme un maître de stage, à Junior Lopez Yepez, Dominique Perron, Sébastien Moisan, Jonathan Ouellet et Eric Chabot de m'avoir apporté toute l'aide technique et toutes les réponses à mes nombreuses questions durant ces 2 années.

RÉSUMÉ

Lorsqu'un système informatique subit une cyberattaque, il peut être important pour l'organisation attaquée d'avoir des informations sur l'attaquant et sur l'attaque en cours. En effet, en ayant une bonne connaissance du cyberattaquant et de la cyberattaque que ce dernier souhaite réaliser, il est possible d'effectuer une analyse de risques pertinente et de construire une réaction défensive adaptée à l'attaque en cours. Cette classification se doit d'être automatique, en raison du nombre important de cybermenaces auquel un système informatique peut être confronté ; il serait déraisonnable d'essayer de classifier manuellement des centaines, des milliers et parfois même des millions de potentielles menaces qu'un système informatique peut subir quotidiennement.

Il s'agit donc de réaliser une classification automatique de cyberattaquants et de cyberattaque. Le sujet de nos travaux présentés dans ce mémoire porte plus précisément sur la classification de cyberattaques. La classification de cyberattaques proposée sera la base d'un projet plus ample, qui réutilisera ce travail afin de construire une classification de cyberattaquants.

Pour réaliser cette classification, il est compliqué de trouver des données représentatives de l'ensemble de la population cybercriminelle, car les entreprises ne divulguent pas publiquement les données des cybermenaces qu'elles subissent quotidiennement. Cependant, nous avons eu la chance de réaliser nos travaux de recherche en coopération avec l'entreprise Thales Digital Solutions, qui a mis à disposition une base de données non labellisées très pertinente pour notre objectif.

Ce travail de recherche emploie alors des techniques d'apprentissage non supervisées afin de construire un modèle utilisable sur des données non labellisées. L'objectif est alors d'isoler des ensembles de données représentant différents types d'attaques dans des partitions séparées. Idéalement, on cherche donc à isoler un type d'attaque dans une partition. Ainsi, lorsqu'on a accès à de nouvelles données en temps réel, on peut leur attribuer une partition, ce qui revient à les classifier. Ceci facilite alors le travail des analystes qui ont pour but d'analyser des quantités importantes de trafic malveillant.

Le modèle qui a été développé dans nos travaux de recherche a la particularité de ne s'appliquer qu'à un protocole à la fois, ce qui permet d'accéder au contenu des paquets (inspection approfondie), plutôt que de se baser uniquement sur les métadonnées de ces paquets. Dans nos travaux, le modèle a été appliqué aux protocoles Hypertext Transfer Protocol (HTTP) et Secure SHell (SSH).

Afin de valider ce modèle, il s'agit de l'appliquer dans un premier temps sur des jeux de données labellisées publics. Il sera alors possible d'utiliser des métriques pour évaluer la précision du modèle, pour l'appliquer par la suite aux données non supervisées fournies par notre partenaire industriel.

Ainsi, trois jeux de données labellisées différents ont été utilisés pour valider notre modèle. La validation s'est faite uniquement sur des attaques web. Avec un des jeux de données les plus utilisés dans la littérature, notre modèle a réussi à parfaitement classifier chacune des attaques. Bien que les scores permettant d'évaluer la performance de notre modèle soient moins élevés pour les deux autres jeux de données, les résultats obtenus sont tout de même très pertinents une fois interprétés. Le travail sur les données de notre partenaire industriel a révélé qu'une très grande majorité des données sont relatives à du trafic automatisé. Ce type de trafic ne comporte pas réellement de cyberattaques complexes. Néanmoins, notre classification apporte une certaine compréhension de ces données.

Cette classification peut ainsi servir de base pour d'autres travaux. Il est par exemple possible d'utiliser des techniques de corrélation sur notre classification afin de détecter des scénarios de cyberattaques complexes. On pourrait aussi utiliser cette classification afin d'augmenter la précision de certaines classifications de cyberattaquants existantes. Mais afin d'utiliser notre modèle en temps réel, il faudrait l'optimiser pour améliorer sa vitesse d'exécution.

ABSTRACT

Computer networks, supporting the economic and political activities of society, are under threat from malicious actors conducting a wide range of cyberattacks. At the forefront of defending systems from threats, it is necessary to detect malicious events and differentiate anomalous activities from legitimate network traffic. Traditionally, this process is performed using hard-coded detection rules to identify violations to genuine traffic and employs the use of security analysts to manually triage suspicious events. These defensive activities can establish a baseline defensive posture for organizations, however, they come with limitations. The scarcity of security analysts is widely spread across organizations and their time is increasingly being stretched thin as the quantity of attacks they are faced are constantly mounting. Recent advancements in Machine Learning (ML) methods offer promising approaches for automating attack classification tasks and streamlining the work of security personnel.

Supervised learning methods are a paradigm of ML where a model learns to classify samples by first training on labelled data to establish decision boundaries. However, these methods are dependent on a massive amount of labelled data to train upon, a constraint that is generally not readily available in cybersecurity domains, and they are limited to classifying events that they have been trained upon, restricting their ability to generalize to unseen events.

In response to the shortcomings of supervised learning methods, we propose to advance the study of unsupervised learning methods for attack classification by presenting a novel unsupervised learning approach. Under the unsupervised learning paradigm, there is the opportunity to learn patterns in unlabelled data and ultimately classify attacks that have never been encountered during the training of an ML model.

Furthermore, we propose a new approach by building a classification model specific to the Hypertext Transfer Protocol (HTTP) and Secure SHell (SSH). Rather than limiting our work to numerical networking attributes, which are common to all the protocols, we focus on more specific attributes by choosing to work protocol by protocol. Although, in this work, we test our method with the HTTP and SSH protocols, the approach has the potential to be extended to any communication protocol.

Our model uses a new approach combining a pre-processing method, a recent dimension reduction algorithm, and an efficient clustering technique to classify malicious data. While the final purpose of this algorithm is to be applied on real unlabelled traffic data, we evaluate the performances of our model by using labelled data, in order to validate our model. We

test our model on 3 different labelled datasets, and we achieve to perfectly classify a very popular dataset in cyberattack classification.

Finally, we have the chance to access to data provided by Thales Digital Solutions. We apply our model in these data, which were not labelled. We propose a detailed analysis of each cluster, for both HTTP and SSH data.

As a future work, we could apply our model on several protocols and combine the results. Further, we could use correlation algorithms in order to recognize each step of a multi-step attack in our classified data. Thus, these algorithms could describe a multi-step attack as a sequence of clusters computed with our model. An optimization work has to be done to make real time classification possible with our model.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

SCDM	Système de Classification de Données Malveillantes
IT	Information Technology
OT	Operational Technology
NVU	Nombre de Valeurs Uniques
IoT	Internet of Things
ML	Machine Learning
ACP	Analyse en Composantes Principales
ACM	Analyse en Correspondances Multiples
AFDM	Analyse Factorielle de Données Mixtes
t-NSE	t-Distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection

CHAPITRE 1 INTRODUCTION

Les cybermenaces connaissent une augmentation constante et deviennent une préoccupation de premier ordre pour de nombreux acteurs. La pandémie de la COVID-19 a aggravé les choses : le télétravail n'a fait qu'augmenter les risques de cyberattaques, particulièrement dans les entreprises qui n'avaient pas les infrastructures nécessaires pour rendre le télétravail possible de façon sécuritaire. D'ailleurs, le Centre Canadien pour la Cybersécurité explique dans son bulletin sur les cybermenaces [1] que depuis mars 2020, un quart des petites entreprises canadiennes ont déclaré avec subi un cyberincident malveillant. Le chiffre réel est probablement plus élevé. Ce même bulletin indique aussi une utilisation croissante des rançongiciels, qui sont des cyberattaques ayant pour but de priver un système informatique de ses données et de ne les restaurer que lors de paiement d'une rançon. En effet, le montant des rançons payées dans le cas de rançongiciels est plus important dans les 6 premiers mois de 2021 (590 M$ US) que pour l'ensemble de l'année 2020 (416 M$ US). Plus généralement, le rapport annuel de Cybersecurity Ventures [2] estime que le risque cyber dans sa totalité à 6 000 milliards dollars US. La croissance et la prépondérance du cyberrisque ont entraîné une évolution dans les méthodes de détection et de classification de cyberattaques.

1.1 Définitions et concepts de base

1.1.1 Les différents systèmes informatiques

Avant d'exposer les différentes défenses existantes pour lutter contre le cyberrisque, il s'agit de définir les systèmes informatiques à défendre. Un système informatique est un ensemble de machines pouvant traiter et échanger des données. Il existe deux catégories de systèmes informatiques : les systèmes de technologie de l'information (**IT**) et les systèmes opérationnels (**OT**). Les systèmes opérationnels, à la différence des systèmes de technologie de l'information, ont pour objectif d'assister une entreprise dans ses tâches quotidiennes. Il est donc très important de protéger les systèmes OT, puisqu'une indisponibilité d'un système opérationnel peut paralyser la production d'une entreprise.

1.1.2 Présentation des outils permettant la détection de cyberattaques et leurs limites

Les systèmes de détection d'intrusion

Afin de lutter contre le cyberrisque, de nombreuses entreprises se dotent d'un **IDS** (Intrusion Detection System ou « système de détection d'intrusion » en français). Ces technologies permettent de lever des **alertes** lorsqu'une cyberattaque est identifiée.

Cependant, les IDS ne sont pas les seules technologies permettant de détecter des données malveillantes. Des pare-feux ou des antivirus peuvent également être utilisés. La pluralité des technologies permettant la détection de cyberattaques et le nombre important d'alertes qui peuvent être générées constituent alors une problématique importante. En effet, analyser manuellement les sorties de chacune de ces technologies constituerait une perte de temps considérable.

Les systèmes de gestion des événements et des informations de sécurité et leurs limites

Afin de faciliter l'analyse des sorties des différents points de collecte d'évènements (appelés **capteurs**), il convient de centraliser la gestion de la détection de données malveillantes. C'est l'objectif d'un **SIEM** (Security Information and Event Management, ou « gestion des événements et des informations de sécurité » en français), qui collecte, classifie et corrèle ces données.

En analysant les sorties d'un SIEM, des analystes en cybersécurité peuvent alors prendre des décisions de sécurité au sein de leur **SOC** (Security Operations Center, ou « Centre des Opérations de Sécurité » en français) en fonction de la gravité de la situation. Cependant, même en centralisant les nombreuses alertes (ou autres informations relatives à la sécurité) provenant de nombreuses sources, le volume de données à traiter reste trop important. Fireye explique dans son rapport [3] que la majorité des SOC reçoit plus de 10 000 alertes par jours. Ce même rapport donne des chiffres alarmants : seuls 29% des alertes concernant un logiciel malveillant font l'objet d'une investigation et 40% sont des faux positifs.

En clair, les SIEM actuels continuent de produire des faux positifs de façon trop importante, ce qui empêche les vrais positifs d'être traités correctement. De même, la notion de faux positif en elle-même est ambiguë : doit-on traiter un néophyte utilisant un outil de reconnaissance automatique comme un vrai, ou un faux positif ? En théorie, c'est un vrai positif. Pourtant, il ne faudra pas le traiter de la même façon qu'un vrai positif issu d'une organisation criminelle, puisque le risque n'est pas le même. Aussi, il faudrait être capable de reconnaître les alertes

liées au passage de botnets. Un **botnet** est un réseau d'appareils connectés qui ont été compromis et qui sont contrôlables par une personne ou un groupe de personnes, pour des fins malveillantes. Ces derniers constituent une menace assez faible, puisqu'ils ne font que tester des vulnérabilités aveuglement.

Il est donc essentiel de connaître l'attaquant qui souhaite compromettre notre réseau, ou plus généralement, qui a généré une alerte. Cela permet de réaliser un tri dans les alertes, d'écarter les faux positifs et les risques très faibles, afin de se concentrer sur les réelles menaces. Afin de classifier correctement un attaquant, il faut être capable de connaître l'attaque qu'il est en train de réaliser. Pour ce faire, il faut être en mesure de classifier les alertes levées par l'attaquant. Le sujet de ce mémoire porte précisément sur ce point, à savoir l'élaboration d'un Système de Classification de Données Malveillantes (**SCDM**), afin de proposer une classification d'attaques. Cette dernière sera utilisée par la suite pour une classification d'attaquants, qui sera réalisée par un de mes camarades.

En classifiant et en corrélant les différentes données malveillantes, il est possible de reconstruire la cyberattaque qui est en cours, ce qui peut s'avérer très utile. Ceci permet en effet de choisir pertinemment les mesures défensives à adopter. Par exemple, si la cyberattaque est jugée comme étant extrêmement critique, il est possible de décider de mettre hors ligne une partie, voire l'ensemble du réseau. Aussi, avec une classification pertinente de l'attaque, il sera plus aisé de prédire les prochains mouvements de l'attaquant et de réagir en conséquence. Par exemple, la menace peut être classifiée comme étant interne ou externe à l'entreprise ; la façon de réagir à l'attaque ne sera alors pas la même en fonction de cette classification. Enfin, la classification d'attaques permet aussi de faciliter l'attribution et l'investigation de l'attaque. En ayant une idée du type d'attaque auquel on est confronté, il est plus aisé de classifier l'attaquant et donc d'attribuer l'attaque à un cyberattaquant, ou au moins une catégorie de cyberattaquant. Les équipes d'investigation numérique voient alors leur travail largement facilité. La corrélation et la classification de trafic malveillant dans le but d'extraire des scénarios de cyberattaques peuvent donc procurer de nombreux avantages défensifs à un système informatique subissant une cyberattaque.

Aussi, bien que certains SIEM prétendent être capables de corréler et classifier des données issues d'actions malveillantes, la réalité est en fait plus complexe. Par exemple, les SIEM peuvent proposer des règles de corrélation, qui sont difficiles à produire, qui ne s'adaptent pas au cours du temps et dont l'efficacité est limitée. Ces règles ne sont pas en mesure de reconnaître et de classifier une cyberattaque lorsqu'elle devient trop complexe (notamment lorsqu'elle comporte plusieurs étapes). En effet, ces règles sont souvent bien trop simples, comme « S'il y a plus de X alertes relatives à une tentative d'authentification par Y temps,

alors une attaque de type force brute est en cours ». Aussi, elles ne peuvent pas reconnaître de nouvelles attaques qui n'ont pas encore été observées dans le passé, et qui n'ont donc pas de règles associées.

Seuls certains SIEM plus élaborés (et bien plus chers) proposent un SCDM. Évidemment, ces SCDM sont absolument opaques et ne proposent pas de réelles analyses de leurs performances. De plus, ils ne peuvent pas encore détecter des scénarios d'attaques complexes, étalés sur le temps, qui se décomposent en différentes étapes, avec des mouvements latéraux.

1.1.3 Les différents types de données utilisables pour construire un SCDM

Utilisation d'alertes pour réaliser des classifications de cyberattaques

De nombreux travaux de recherche ont été menés afin de développer un SCDM. Ceux-ci sont présentés en détail dans la revue de littérature de la Section 2. Ces travaux effectuent généralement une classification qui se base directement sur des alertes, et non pas sur les données brutes. L'avantage est de pouvoir utiliser le format de données standard IDMEF [4] ; bien que des données concernent différents capteurs et différents protocoles, il est tout de même possible de traiter les alertes toutes en même temps, car elles ont un format standard. Ainsi, il est courant de collecter des données au format PCAP [5] correspondant à l'ensemble du trafic réseau, puis d'utiliser un IDS comme Snort [6] ou Suricata [7] afin d'extraire les alertes relatives à ces données. Pour obtenir des données pertinentes, on ne peut pas capturer n'importe quel trafic réseau. Généralement, les auteurs des jeux de données réalisent des attaques sur leurs propres systèmes afin d'obtenir du trafic malveillant. Ceci présente aussi l'avantage de pouvoir labelliser les données, puisque les auteurs connaissent les dates de chacune des attaques. Cependant, cette méthode de collecte de données présente aussi un désavantage : les données obtenues ne sont pas toujours semblables à de réelles cyberattaques, puisque ce sont des cyberattaques qui ont été choisies par des auteurs, qui n'ont pas les mêmes méthodes ni les mêmes moyens que la population cybercriminelle.

Définition et particularités d'un pot de miel

Une autre méthode de collecte de données malveillantes consiste à mettre en place un pot de miel (« Honeypot » en anglais). Un pot de miel est un leurre qui simule un réseau informatique dans le but d'y attirer des acteurs malveillants et d'analyser leur comportement. Les données sont alors enregistrées dans des journaux (« logs » en anglais). Ces journaux sont par la suite utilisés afin de classifier et corréler des données issues du trafic malveillant. Ces journaux ont l'avantage de comporter uniquement du trafic que l'on peut considérer comme

étant malveillant, puisqu'un utilisateur lambda n'a pas de raison de se retrouver sur un pot de miel (seuls certains outils utilisés par des cyberattaquants peuvent mener à un pot de miel). Cependant, ces données ne sont pas enregistrées dans un format standard, puisque le choix des attributs à enregistrer est propre à chaque pot de miel. Bien qu'il serait possible d'enregistrer tout le trafic réseau puis d'utiliser des IDS pour générer des alertes, cela représenterait de grandes quantités de données à gérer, ce qui est donc compliqué à mettre en place. Cette méthode de collecte de données présente l'avantage d'être plus représentative de la population cybercriminelle, puisque ce sont des vrais cyberattaquants qui interagissent avec le pot de miel. Mais, en plus de la difficulté d'obtenir des données au format standard, cette méthode ne permet pas d'avoir des données labellisées, puisqu'on ne sait pas quelles ont été les cyberattaques effectuées.

Les degrés d'interaction d'un pot de miel

Aussi, les pots de miel peuvent être classés en fonction de leur degré d'interaction. Un pot de miel à faible interaction est caractérisé par sa simplicité : la marge de manœuvre de l'attaquant est extrêmement faible, il n'y a pas ou peu de services, et ces derniers sont sécurisés. À l'inverse, un pot de miel à forte interaction permet à un attaquant d'interagir avec différents services. Parfois, ces derniers sont vulnérables et constituent alors un leurre plus élaboré pour les attaquants, qui pensent attaquer de réels services vulnérables. Bien que les données récoltées par un pot de miel à forte interaction soient plus intéressantes, il est compliqué de mettre en place un tel pot de miel, puisqu'il y a un risque que l'attaquant prenne le contrôle du pot de miel et étende son attaque au reste du système informatique. Lorsque l'on met en place un pot de miel à forte interaction, il faut donc trouver un compromis entre la sécurité de l'architecture retenue et le nombre et le degré de vulnérabilité des services exposés.

Le pot de miel utilisé dans le cadre de nos travaux

Les données fournies sont issues d'un pot de miel qui a agi comme un leurre pour les attaquants depuis le 7 novembre 2019. Ces derniers ont interagi avec le pot de miel comme ils auraient interagi avec le système informatique initialement ciblé. Il est alors possible d'établir une classification des menaces, qu'elles soient internes ou externes, grâce à un SCDM. De telles données sont très rares, puisque les entreprises ne partagent pas publiquement les cyberattaques qu'elles subissent.

Le pot de miel mis en place collecte des données relatives à différents protocoles. Le degré d'interaction diffère en fonction des protocoles. Par exemple, pour **HTTP** (Hypertext Trans-

fer Protocol : protocole utilisé pour naviguer sur le web), le pot de miel ne se limite qu'à une simple page de connexion (identifiant/mot de passe), avec aucune possibilité de se connecter. L'interaction est donc faible. Au contraire, pour SSH, un agent d'apprentissage par renforcement adapte les interactions possibles entre le pot de miel et l'attaquant. Ceci simule donc la réaction défensive d'un système informatique qui se fait attaquer. L'attaquant a plus de chance de penser qu'il est sur un système réel et non pas un pot de miel. L'attaquant va alors adopter un comportement plus proche de celui qu'il aurait dans le cas d'une cyberattaque d'un « vrai » système informatique, ce qui rend le jeu de données plus pertinent.

1.2 Problématique

L'objectif de cette section est de mettre en évidence certains points qui permettent l'élaboration d'une problématique pertinente.

1.2.1 Hétérogénéité des données d'un pot de miel

Les données d'un pot de miel sont enregistrées selon un certain format ; au lieu d'enregistrer l'intégralité du paquet, on ne journalise que les attributs jugés pertinents. D'un pot de miel à un autre, les attributs journalisés peuvent alors varier. En outre, d'un protocole à l'autre, différents attributs sont choisis pour journaliser les données (on n'utilise pas les mêmes attributs pour journaliser un paquet HTTP qu'un paquet SSH). Il faut ajouter à cela le fait qu'au sein d'un même protocole, les attributs peuvent également être différents. Par exemple, si un utilisateur utilise une requête HTTP de type POST, qui permet d'envoyer des données à un serveur, alors l'attribut « payload », qui contient les données à envoyer, sera journalisé. Dans le cas d'une requête GET, qui permet de demander des données au serveur, l'attribut « payload » ne sera pas présent. Pourtant, les deux requêtes sont relatives au même protocole. Il existe donc une triple hétérogénéité dans les données d'un pot de miel : une qui dépend du pot de miel en lui-même (deux pots de miel différents ne journalisent pas les mêmes attributs), une « interprotocole » (deux protocoles différents ne journalisent pas les mêmes attributs) et une « intraprotocole » (différents attributs journalisés pour un même protocole).

1.2.2 Conversion des données d'un pot de miel et conséquences

Il faut également noter l'existence d'une perte d'information lorsque l'on journalise nos données en ne retenant que certains attributs. En effet, il n'est pas possible de reconstruire l'intégralité d'un paquet à partir de données journalisées (toujours en ne retenant que certains attributs). On peut, à partir d'un paquet, journaliser des données, mais on ne peut pas,

à partir d'un journal, reconstruire un paquet. Il est alors impossible d'utiliser un système de détection d'intrusion tel que Snort [6] ou Suricata [7] sur nos données, puisque ces derniers n'acceptent que des paquets complets (format PCAP).

Comme les données journalisées par le pot de miel ne sont pas au format PCAP et qu'il est impossible de transformer les données fournies dans le format PCAP, on ne peut pas bénéficier de l'avantage d'utiliser des données ayant un format standard. Ce format aurait permis de traiter plusieurs protocoles simultanément. Il s'agit donc de fonctionner protocole par protocole. Cela permet de construire un algorithme sur mesure, plus adapté, pour chaque protocole. Ce point sera détaillé dans la Section 3.

1.2.3 Des données non labellisées

Il est important de souligner le fait que les données issues d'un pot de miel sont non labellisées. On ne sait donc absolument pas quelles sont les attaques qui composent nos données, notre contribution étant justement de répondre à cette question. Le fait que ces données soient non labellisées complique l'évaluation de notre modèle. C'est pourquoi il est pertinent d'appliquer notre modèle en utilisant d'autres données publiques et labellisées qui sont utilisées par la communauté scientifique. L'objectif est de pouvoir évaluer nos travaux et le comparer à d'autres travaux existants. Les labels des données ne devront pas être utilisés lors de la classification, mais uniquement pour l'évaluation de celle-ci, afin de ne pas fausser les résultats. Notre modèle doit donc accepter différents formats en entrée.

1.2.4 Caractéristiques des cyberattaques

Aussi, notre modèle devra être capable de détecter des cyberattaques complexes et non uniformes. Une attaque complexe est une attaque avec un scénario, comprenant plusieurs phases. Des attaques sont dites non uniformes lorsqu'elles s'étalent de façon hétérogène dans le temps. Par exemple, une attaque peut se dérouler sur plusieurs mois avec peu d'alertes, ou sur quelques minutes avec de nombreuses alertes, ce qui complique leur détection. Ces attaques peuvent aussi contenir des vulnérabilités de type 0-day ; ces vulnérabilités ont la particularité de ne pas avoir été dévoilées au grand public et de ne pas avoir encore de correctif.

1.2.5 Énoncé de la problématique

Les éléments cités précédemment montrent qu'il est pertinent d'élaborer un SCDM grâce à un modèle de classification non supervisé. Ce modèle doit être compatible avec les données

fournies par notre partenaire industriel, ainsi qu'avec les autres formats de données existants. Ainsi, il sera possible d'utiliser des jeux de données labellisées accessibles publiquement, qui ont un format différent des données qui nous sont fournies.

Notre question de recherche principale est donc la suivante : comment classifier des cyberattaques complexes et non uniformes, incluant des attaques de type 0-day, en utilisant des données ayant différents formats ? Comment évaluer les performances d'un modèle répondant à cette problématique ?

1.3 Objectifs

Afin de répondre à notre question de recherche principale, il s'agit de :

- Créer un modèle de classification d'attaque, permettant de détecter différentes attaques au sein de données malveillantes
- Rendre le modèle adaptable à différents formats de données
- Utiliser des jeux de données afin de mesurer les performances de notre modèle
- Appliquer notre modèle à des données non labellisées ayant un format non standard et interpréter les résultats
- Définir et utiliser des métriques permettant d'évaluer nos résultats sur des données non labellisées

1.4 Plan du mémoire

Après avoir introduit le sujet, les concepts de base et la problématique dans ce chapitre, il convient d'établir, dans le Chapitre 2, une revue de littérature. La section 2.1 est un état de l'art des différents jeux de données malveillantes labellisées qui nous permettent de mesurer les performances de notre modèle. Les différentes méthodes de réduction de dimension et de classification non supervisées sont exposées à la Section 2.2. Une comparaison des méthodes évoquée dans la revue de littérature est également proposée à la Section 2.3. Cette comparaison permet de justifier les choix qui ont été faits pour construire le modèle utilisé dans nos travaux.

Dans le Chapitre 3, la méthodologie de la solution proposée dans ce mémoire sera présentée. Elle évoquera les différentes méthodes explorées à la Section 3.1. En comprenant pourquoi certaines pistes n'ont pas abouti, il est alors plus aisé de comprendre comment et pourquoi notre modèle a été conçu. La Section 3.2 propose alors une présentation globale du modèle, suivie d'une présentation détaillée de chacun de ses modules à la Section 3.3. La méthode pour appliquer notre modèle à d'autres jeux de données est expliquée à la Section 3.4.

Par la suite, il s'agira d'évaluer les performances du modèle retenu au Chapitre 4. La méthodologie de la validation est présentée à la Section 4.1. Chacun de trois jeux de données labellisées est présenté, puis utilisé dans les Sections 4.2, 4.3 et 4.4, avant de faire l'objet d'une discussion croisée à la Section 4.5.

Puis, le Chapitre 5 présente les données fournies à la Section 5.1 et expose puis analyse les résultats obtenus sur les données relatives au protocole HTTP et SSH, respectivement aux Sections 5.2 et 5.3.

Enfin, une conclusion au Chapitre 6 offre une synthèse des travaux à la Section 6.1, met en évidence les limitations de la solution proposée à la Section 6.2, et donne des améliorations futures pour notre modèle à la Section 6.3

CHAPITRE 2 REVUE DE LITTÉRATURE

L'objectif de cette partie est d'élaborer un état de l'art des techniques permettant de reconnaître des cyberattaques dans des données malveillantes. La problématique à la Section 1.2.5 met en évidence la pertinence d'avoir un modèle capable de traiter des données non labellisées, c'est pourquoi cette revue de littérature se concentrera davantage sur les méthodes de classification non supervisées. Aussi, les jeux de données labellisées et couramment employés par la communauté scientifique doivent être utilisés afin d'évaluer notre méthode de classification. L'état de l'art s'organise donc de la façon suivante : une première section présente les jeux de données labellisées permettant de classifier des attaques malveillantes. Dans cette première section, il s'agit aussi de présenter les différentes techniques, supervisées ou non, qui ont été utilisées sur ces jeux de données par le passé. Cette première partie permet de sélectionner les jeux de données qui sont pertinents dans notre cas, ainsi que les différentes méthodes auxquelles il faudra comparer notre modèle afin de l'évaluer. La Section 2.2 détaille les différentes méthodes de classification non supervisées permettant de réaliser une classification d'attaques. Elle s'articule aussi en différentes sous-sections : il convient d'évoquer de prime abord les différentes méthodes de réduction de dimension et de représentation des données, puis les différentes méthodes de classification non supervisées et enfin d'étudier comment ces méthodes ont été utilisées dans le passé pour classifier des données malveillantes. Cette revue de littérature a donc pour but de mettre en avant certains algorithmes pertinents à notre cas d'usage. Il s'agira finalement de comparer ces différents algorithmes afin de justifier les choix ayant mené à l'élaboration de notre modèle.

2.1 Jeux de données de cyberattaques

La première sous-section confronte les études comparatives réalisées sur des jeux de données contenant du trafic malveillant et permet d'aboutir à la sélection de 3 jeux de données pertinents pour notre cas. Les 3 sous-sections suivantes présentent différents travaux ayant été menés sur chacun de ces jeux de données.

2.1.1 Sélection des jeux de données malveillantes

Pour réaliser une classification d'attaques, différents jeux de données malveillantes ont été utilisés. Celui qui est probablement le plus ancien et qui a été le plus utilisé est DARPA99 [8]. Ce jeu de données regroupe 200 attaques de 58 types différents, toutes labellisées avec un

taux de faux positifs assez faible (moins de 10 par jour). Cependant, ce jeu de données date de 1999 et devient de plus en plus obsolète au fil des années.

Plusieurs articles se sont alors basés sur ce jeu de données pour utiliser des méthodes de classification supervisées. C'est notamment le cas de [9] qui utilise de réseaux de graphes convolutionels (Graphs Convolutional Networks) dans le but de classifier les attaques présentes dans DARPA99. Les auteurs comparent leurs résultats avec d'autres algorithmes d'apprentissage supervisés (Naive Bayes, Support Vector Machine, Logistic Regression, Decision Tree, KNN, Random Forest et MLP) et obtiennent des résultats bien meilleurs sur toutes les métriques : precision, recall, f1-score, exactitude. L'ensemble de ces métriques dépassent les 92%, en se basant sur 6 355 alertes différentes après prétraitement des données, constituant 7 attaques différentes. Il est donc intéressant de se baser, en premier lieu, sur un article comparant des méthodes de machine learning très populaires sur un jeu de données très populaire. Néanmoins, la comparaison semble se baser sur les résultats obtenus sur les données d'entraînements, alors que le modèle souffre de surapprentissage d'après les auteurs, ce qui peut expliquer les bonnes performances du modèle exposées dans l'article. Aussi, il est important de stipuler que le modèle n'est pas en mesure de détecter d'autres attaques que celles sur lesquelles il a été entraîné. Ceci est un problème récurrent lors de l'utilisation de méthodes d'apprentissage supervisé.

Cet exemple est assez révélateur ; pour qu'un modèle puisse être utilisé en pratique dans des conditions réelles, il faut donc des jeux de données plus récents. En effet, comme les algorithmes de classification supervisés ne peuvent reconnaître que les attaques sur lesquelles ils sont entraînés, il est donc capital de pouvoir entraîner les différents modèles sur des jeux de données récents, afin de pouvoir détecter des attaques qui sont d'actualité. C'est dans cette démarche que s'inscrivent de nombreux jeux de données bien plus d'actualité. D'ailleurs, Ring et al. [10] comparent pas moins de 34 jeux de données malveillantes. C'est probablement, à ce jour, la comparaison de jeux de données malveillants la plus complète et la plus récente (2019) qui a été menée. Cette étude comparative ne recommande pas l'utilisation de DARPA99, notamment en raison de son obsolescence, mais donne la préférence à des jeux de données plus récents, comme CIC-IDS2017 [11], CIDDS-001 [12], UGR'16 [13] et UNSW-NB15 [14]. De plus, ils précisent que chaque étude est unique et peut être plus adaptée à certains jeux de données, c'est pourquoi ils mentionnent le fait que leur recommandation est générique et subjective. Mais ils donnent toutes les caractéristiques (type d'attaques, protocole, format, etc.) des 34 jeux de données qu'ils comparent, ce qui a été très utile pour choisir un jeu de données approprié.

D'ailleurs, dans notre cas, une contrainte importante est à mentionner : les jeux de données

sélectionnés *doivent* être disponibles au format brut (capture réseau PCAP). En effet, le modèle proposé (voir section 3) ne se base pas sur les attributs couramment extraits dans les jeux de données. C'est pourquoi nous ne pouvons pas utiliser les jeux de données qui ne proposent pas un accès public à leurs données brutes. Dans les quatre modèles suggérés par Ring et al., seuls CIC-IDS2017 et UNSW-NB15 offrent un accès public à leurs captures réseau. Ces deux jeux de données sont suffisamment récents et contiennent des attaques variées et pertinentes à notre cas d'étude. Ghurab et al. [15] donnent une description encore plus précise pour 8 jeux de données, dont CIC-IDS2017 et UNSW-NB15. Ils décrivent plus précisément les attaques de chaque jeu de données et leurs caractéristiques. Nous pouvons identifier qu'aucun de ces jeux de données n'a des attributs catégoriels. Seules des caractéristiques numériques génériques sont extraites des données brutes et utilisées par les auteurs. C'est d'ailleurs le cas de tous les jeux de données rencontrés dans la littérature. Pour avoir accès à des caractéristiques catégorielles spécifiques, nous devons extraire ces données des jeux de données, car nous ne voulons pas seulement nous limiter à utiliser les caractéristiques numériques extraites existantes. Le modèle proposé sera également capable de se baser sur des caractéristiques numériques ; le fait de pouvoir traiter également des caractéristiques catégorielles apporte de nouvelles informations à notre modèle, ce qui devrait le rendre plus précis. En effet, certaines attaques sont difficilement détectables en utilisant un modèle qui se base uniquement sur des caractéristiques numériques. Par exemple, un modèle ne supportant que des caractéristiques numériques devrait plus difficilement différencier une attaque de type Cross-site scripting (injection pouvant permettre l'exécution de code) d'une injection SQL (injection permettant d'interagir avec une base de données). Pour justifier ceci, prenons l'exemple de deux paquets parfaitement identiques, excepté pour la charge utile qui permet de réaliser l'une des deux injections. Les caractéristiques catégorielles (comme la charge utile) vont très bien représenter ce changement, contrairement aux attributs numériques, qui ne pourraient détecter qu'une variation de la taille de la charge utile, par exemple.

Yavanoglu et Aydos [16] réalisent une autre étude avec 19 articles qui utilisent différentes techniques de classification d'attaques avec différents jeux de données. Ils ne recommandent pas un ensemble de données ou une méthode spécifique, mais ils décrivent chaque ensemble de données avec précision. On remarque d'ailleurs avec cette étude que le jeu de données le plus utilisé est celui de DARPA99 et ses dérivés, à savoir des modifications de DARPA99 pour créer d'autres jeux de données, tels que KDD CUP-99 [17] [18], NSL-KDD [19] et PU-IDS [20]. Les auteurs évoquent aussi un autre jeu de données pertinent à notre cas d'étude, qui est CTU-13 [21] et dont les données brutes sont disponibles publiquement.

Les 3 études comparatives ont donc permis la sélection de 3 jeux de données pertinents à notre étude : CIC-IDS2017, CTU-13 et UNSW-NB15. Cette sélection, en plus des recommandations

des différents auteurs et de la contrainte d'avoir les données brutes disponibles publiquement, s'est réalisée selon différents critères. Parmi ces critères, on peut citer notamment le type des attaques (on est intéressé principalement par des attaques web), la complexité des attaques, la popularité du jeu de données (afin de pouvoir le comparer à différentes études), la taille du jeu de donnée, la facilité d'utilisation et la date à laquelle ces données ont été collectées. Bien que les 3 jeux de données n'excellent pas dans chacun de ces critères, ils restent les plus pertinents à notre cas d'usage. De nombreux jeux de données n'ont pas pu être intégrés à cette étude uniquement parce que leur capture réseau n'était pas disponible publiquement.

Après avoir choisi les jeux de données sur lesquels nous allons baser notre comparaison, on peut maintenant présenter les différents travaux réalisés sur ces jeux de données, après les avoir décrits sommairement. La présentation détaillée du contenu de ces jeux de données se trouve dans le Chapitre 4.

2.1.2 CIC-IDS2017

Le jeu de données CIC-IDS2017 [11] a été développé à l'université du New Brunswick au Canada. Ce jeu de données représente le réseau informatique d'une petite entreprise où plusieurs employés effectuent des tâches professionnelles. Le trafic malveillant est généré en menant plusieurs attaques telles que des attaques par déni de service (DoS), par force brute, ainsi que des scans de ports. CIC-IDS2017 est un des ensembles de données malveillantes récents les plus courants et les plus utilisés par les chercheurs en cybersécurité (1 398 citations à l'heure de la rédaction de ce mémoire).

Panigrahi et Borah [22] proposent une analyse complète de cet ensemble de données. Ils concluent que CIC-IDS2017 est compliqué à utiliser car il s'agit d'un jeu de données épars et volumineux. De plus, il y a un nombre important de valeurs manquantes et de déséquilibres dans CIC-IDS2017. Néanmoins, il s'agit d'un ensemble de données complet avec différentes attaques, y compris des attaques web, ce qui est très pertinent dans notre cas. De nombreux chercheurs ont essayé d'appliquer différents modèles pour classifier les données de CIC-IDS2017. Ils ont dû faire face au déséquilibre de l'ensemble de données. Par exemple, Pelletier et Abualkibash [23] proposent de relabelliser et de réallouer les données afin de donner aux classes minoritaires une quantité plus proportionnelle de données. Ceci permet l'utilisation d'un modèle prédictif basé sur un réseau neuronal artificiel et des algorithmes d'apprentissage automatique. Ils utilisent un algorithme de forêt aléatoire (Random Forest) et la bibliothèque Caret [24] pour obtenir une précision de 96,24%, après 68,35 heures de calcul. Pour déterminer quelles sont les caractéristiques pertinentes à utiliser, ils utilisent le test d'importance de Boruta [25]. Mais ils limitent le test aux caractéristiques numériques

extraites par défaut de l'ensemble de données.

Arif Yulianto et al. [26] utilisent la technique de suréchantillonnage de minorité synthétique (Synthetic Minority Oversampling Technique) pour gérer le déséquilibre de l'ensemble de données. Ensuite, ils utilisent l'Analyse en Composantes Principales [27] et la sélection de caractéristiques d'ensemble (Ensemble Feature Selection) pour améliorer les performances d'un système de détection d'intrusion basé sur AdaBoost [28]. Grâce à cette technique, ils parviennent à atteindre une exactitude, une précision, un rappel et un score F1 de 81,83%, 81,83%, 100% et 90,01%, respectivement.

Zhang et al. [29] ont décidé d'utiliser gcForest [30] et CNN [31], qui sont adaptés aux données déséquilibrées. Leur meilleur modèle de détection (DCF-IDS) peut détecter avec précision une attaque par force brute (99,88% du taux de détection) et une attaque XSS (99,87%), mais a plus de difficultés à détecter une injection SQL (77,78% de taux de détection).

Faker et Dogdu [32] ont construit un modèle en 3 phases : tout d'abord, ils proposent un prétraitement détaillé, puis un classement et une sélection des caractéristiques avec K-Means. Enfin, ils essayent différents réseaux neuronaux profonds et techniques d'ensemble pour classifier les données de CIC-IDS2017. Les meilleurs résultats sont obtenus avec les réseaux de neurones profonds ; ils atteignent une précision de 99,57% avec leurs 75 caractéristiques numériques sélectionnées, avec un temps de prédiction de 0,70 ns.

2.1.3 UNSW-NB15

Bien qu'il soit un peu moins récent que CIC-IDS2017, UNSW-NB15 [14] est également un des plus utilisés avec 1 598 citations à l'heure de la rédaction de ce mémoire. Les auteurs justifient la nécessité de la création de jeux de données malveillantes récents en soulignant l'obsolescence de jeux de données comme KDD98, KDDCUP99 et NSLKDD. D'ailleurs, les auteurs du jeu de données proposent une comparaison [33] de leur jeu de données à KDDCUP99. Pour créer ce jeu de données, l'outil IXIA perfectstorm [34] est utilisé afin de générer du trafic bénin et des attaques. Les attaques générées sont réparties dans 9 catégories différentes. Dishan Jing et Hai-Bao Chen [35] utilisent des Machines à Support de Vecteurs (Support Vector Machines) afin de classifier ces 9 attaques différentes. Leur particularité est d'utiliser une fonction non linéaire logarithmique de mise à l'échelle (scaling) plutôt que la fonction traditionnelle et linéaire de normalisation. Ceci leur permet d'atteindre une exactitude de 75.77% sur les données de test pour la classification multi-classes. On peut remarquer que ces résultats ne sont pas très élevés comparés aux résultats habituels que proposent les Machines à Support de Vecteurs, sur les données de CIC-IDS2017 par exemple. Cela traduit la complexité des données de UNSW-NB15 ; chaque catégorie d'attaque regroupe de nombreuses attaques

différentes. Nawir et al. [36] ont utilisé un nouvel algorithme nommé Online Average One Dependence Estimator (AODE), qui se base sur Naive Bayes (Bayes Naif). Cet algorithme a l'avantage de convenir aux jeux de données volumineux et qui contiennent plusieurs classes. Ceci permet aux auteurs d'atteindre 83.47% d'exactitude sur la classification multi-classes. Leur algorithme a été conçu pour répondre également à une problématique de performance ; il serait capable de pouvoir traiter de grands flux de données en direct. En effet, l'entièreté du jeu de données de UNSW-NB15 a été classifiée en seulement 1.27 seconde. Aleesa et al. [37] sont probablement ceux qui ont obtenu les meilleurs résultats sur UNSW-NB15. En comparant les résultats obtenus avec des Réseaux de Neurones Artificiels, des Réseaux de Neurones Profonds et des Réseaux de Neurones Récurrents, on remarque que la meilleure exactitude a été obtenue pour les Réseaux de Neurones Profonds. En effet, les auteurs ont obtenu 99.59% d'exactitude ! Mais ce chiffre est à contraster avec le fait qu'ils n'aient considéré initialement qu'une partie des données (un peu plus d'un quart des données ont été écartées).

2.1.4 CTU-13

Le jeu de données CTU-13 [21] est moins courant que le jeu de données CIC-IDS2017 (536 citations à la date d'écriture de ce mémoire), probablement en raison de sa complexité. En effet, plutôt que de contenir de simples attaques répétitives comme le jeu de données CIC-IDS2017, CTU-13 contient l'exécution complète de logiciels malveillants (malwares) complexes, provenant de 13 botnets différents, dans un environnement réaliste. Comme CIC-IDS2017, CTU-13 est déséquilibré, car chaque attaque de chaque botnet a une taille très différente et génère une quantité de données différente. C'est pourquoi Velasco-Mata et al. [38] proposent une étude sur leur jeu de données Quasi-Balanced CTU-13 (QB-CTU13) avec une sélection de seulement 8 botnets parmi 13, et Extended QB-CTU13 qui ajoute 3 botnets pour atteindre 11 botnets parmi 13. Ils utilisent les arbres de décision, les forêts aléatoires et K-Means pour classifier les données de QB-CTU13 et de QB-CTU13 Extended. En utilisant l'Importance de Gini (Gini Importance) et le Gain d'Information (Information Gain) pour la sélection des caractéristiques, leurs meilleures performances ont été obtenues avec l'arbre de décision avec un ensemble de cinq caractéristiques choisies, pour lequel ils obtiennent un score F1 de 85%.

Les techniques d'apprentissage profond sont généralement très performantes pour la classification, et CTU-13 ne fait pas exception : Ahmed et al. [39] parviennent à atteindre une précision de 99,6% (même si cet article n'inclut pas d'autres métriques pour une comparaison plus large) avec la technique d'apprentissage profond des réseaux de neurones artificiels à rétro-propagation, qui donne de meilleurs résultats que la machine à vecteurs de support (Support Vector Machines), Naive Bayes et les arbres de décision. Chowdhury et al. [40] ont

réalisé un travail différent, mais intéressant sur CTU-13. Ils ont utilisé une méthode de partitionnement par carte auto-organisatrice afin d'isoler chacun des botnets dans des clusters de petite taille pour détecter aisément le trafic de botnet et d'isoler le trafic bénin dans un cluster de grande taille.

2.2 Représentation de données catégorielles et méthodes de classification non supervisées

Dans cette section, il convient d'étudier quelles sont les différentes techniques de réduction de dimension ou de conversion de type dans un premier temps, puis d'exposer les méthodes existantes de classification non supervisées. Enfin, il s'agit de montrer les travaux ayant utilisé ces méthodes pour réaliser des classifications de cyberattaques. Le terme « représentation de données catégorielles » sera employé afin de désigner la « réduction de dimension et/ou conversion de type de donnée ».

2.2.1 Conversion des données catégorielles en données numériques et réduction de dimension

Les méthodes de classification non supervisées peuvent s'utiliser de différentes façons selon les cas d'utilisation. Il est commun d'utiliser ces méthodes directement sur un jeu de données, après des pré-traitements qui n'impactent pas le nombre ou le type d'attributs utilisés. Généralement, ce sont alors des attributs numériques qui sont utilisés, car la presque totalité des méthodes de classification non supervisées opère sur des données numériques. Cependant, si l'on souhaite utiliser des attributs catégoriels, ou même utiliser des attributs catégoriels *et* numériques, il faut alors convertir le type des données que l'on utilise. Aussi, il peut s'avérer intéressant de réduire le nombre d'attributs utilisé lorsque celui-ci est important. La conversion de type de données et la réduction de dimension correspondent à une double problématique qui peut parfois être résumée à une seule : comment représenter des données, qu'elles soient catégorielles ou catégorielles et numériques, dans un espace numérique ? Certaines méthodes, en représentant des données catégorielles ou catégorielles et numériques, réduisent également le nombre de dimensions utilisées. Ceci s'avère particulièrement intéressant puisqu'en représentant nos données avec seulement 3 dimensions, on peut alors obtenir une visualisation graphique de celles-ci, ce qui facilite l'interprétation de notre partitionnement. Les algorithmes de partitionnement qui utilisent directement des données catégorielles ont une de ces méthodes intégrées. Il est donc intéressant d'étudier les différentes méthodes qui permettent de convertir des données catégorielles (du texte) en données numériques (des nombres). Globalement, les techniques d'analyse en composantes principales [27]

permettent cette réduction de dimension, en plus de la conversion. Les auto-encodeurs [41], eux, permettent la conversion, mais pas la réduction de dimension. Certains auto-encodeurs augmentent même considérablement le nombre de dimensions et ne sont donc pas intéressants dans notre cas où l'on souhaite rendre possible une visualisation de nos données.

Le tableau 2.1 montre les différentes méthodes de représentation de données catégorielles, numériques, ou catégorielles et numériques, qui ont été recensées par notre état de l'art. Ce tableau précise également pour chaque méthode si elle permet une réduction de dimension. Les différentes abréviations utilisées sont Analyse Factorielle de Données Mixtes (AFDM), Analyse en Correspondances Multiples (ACM), Analyse en Composantes Principales (ACP), t-Distributed Stochastic Neighbor Embedding (t-SNE) et Uniform Manifold Approximation and Projection (UMAP).

Algorithme	**Format supporté en entrée**	**Réduction de dimension**
`AFDM`	Numérique et catégoriel	Oui
`ACM`	Catégoriel	Oui
`ACP`	Numérique	Oui
`t-SNE`	Numérique	Oui
`UMAP`	Numérique	Oui
`Auto-encodeurs`	Catégoriel	Non, parfois augmentation

Tableau 2.1 Méthodes de représentation de données catégorielles, numériques, ou catégorielles et numériques

Les techniques de réduction de dimension ne supportant que des données de types numériques en entrées peuvent être utilisées en combinaison avec des auto-encodeurs. Les performances et les comparaisons de ces différentes méthodes sont données dans la section 2.3.

2.2.2 Méthodes de classification non supervisées

2.2.3 Classification de données malveillante par méthodes non supervisées

Nous observons que presque toutes les techniques utilisées sur les jeux de données CIC-IDS2017, CTU-13 et UNSW-NB15 étaient des techniques supervisées. Cependant, notre objectif est de développer un modèle capable de traiter des données non labellisées. Nous allons maintenant discuter de certaines méthodes non supervisées et de leur prétraitement adaptés qui ont été utilisés sur d'autres jeux de données et qui pourraient être pertinents pour nos travaux. Un point de départ intéressant est l'étude de Wu et al. [42] sur les méthodes de détection de cyberattaques basées sur des techniques d'apprentissage profond. Ils ont divisé les techniques non supervisées en trois catégories : les méthodes basées sur les auto-encodeurs,

les méthodes basées sur les réseaux de croyance profonds (Deep Belief Networks) et les méthodes basées sur les réseaux adversariaux génératifs. Mais il existe aussi d'autres méthodes qui n'utilisent pas de techniques d'apprentissage profond et qui ne sont donc pas évoquées dans cet article. Par exemple, Takyi et al. [43] ont écrit une revue des techniques de partitionnement pour la classification de trafic. Même s'il ne s'agit pas de classification d'attaques, cela reste pertinent pour ce travail, car les deux utilisent des données réseau. Ainsi, l'article [43] compare différentes techniques en décrivant les caractéristiques utilisées, ainsi que les objectifs, les limites et les résultats de chaque méthode. Nous pouvons remarquer que K-Means est l'algorithme le plus utilisé. De plus, aucune de ces techniques n'utilise des caractéristiques catégorielles ou des caractéristiques spécifiques à un protocole. On peut également observer que les techniques de partitionnement hiérarchique sont généralement trop complexes et trop lentes pour les grands ensembles de données.

Un autre exemple intéressant qui n'est pas cité dans ces deux études est le travail de Zanero et Savaresi [44]. Ils ont construit un modèle en deux étapes pour effectuer de la détection de cyberattaque sur le jeu de données DARPA99 [8]. Bien que la détection reste différente de la classification, il est possible d'adapter les méthodes de détection en méthodes de classification. Les auteurs ont utilisé, dans une première étape, un algorithme non supervisé pour regrouper les charges utiles des paquets afin d'obtenir une taille exploitable. Dans la deuxième étape, l'Algorithme de Direction Principale (Principal Direction Algorithm), K-Means et l'algorithme de carte auto-organisatrice (S.O.M) [45] sont utilisés de manière comparative pour obtenir les meilleurs résultats. Ils montrent ensuite qu'avec l'algorithme S.O.M., ils ont pu aisément détecter le trafic malveillant du trafic bénin. Dans un travail futur, ils pourraient utiliser le même modèle pour effectuer la classification des mêmes données, et pas seulement de la détection binaire.

Meira et al. [46] comparent différentes techniques de regroupement pour effectuer de la détection d'anomalies liées aux cyberattaques. Ils ont utilisé différentes techniques de prétraitement (Z-score, et Equal frequency + minmax) avec des algorithmes de partitionnement tels qu'Isolation forest, K-means, 1-Plus Proches Voisins, Scaled Convex Hull et les Machines à Support de Vecteurs. Leurs meilleurs résultats ont été obtenus avec Zscore et le partitionnement par 1-Plus Proches Voisins. Ce modèle a obtenu respectivement, pour le score F1, le rappel et la précision, 83%, 94% et 75% pour le jeu de données NSL-KDD et 60%, 83% et 46% pour le jeu de données ISCX. Là encore, les auteurs ont décidé d'utiliser des caractéristiques numériques qui ne sont pas spécifiques à un protocole.

Finalement, cet état de l'art montre que les algorithmes les plus adaptés à notre cas sont K-Means, K-Prototype, K-Mode, HDBSCAN, Amazon Dense Clus, l'Algorithme de Cluste-

ring Gower (ACG) et la Classification Ascendante Hiérarchique (CAH). Les trois premiers algorithmes se ressemblent, mais diffèrent dans le format des données en entrée. En effet, les données à partitionner peuvent se composer d'attributs uniquement numériques (exemple : « durée de la session SSH en secondes » : « 130 »), ou uniquement catégoriels (exemple : « message » : « connexion refusée »), ou avoir certains attributs numériques, et d'autres catégoriels. Le Tableau 2.2 récapitule quels algorithmes supportent quels formats.

Algorithme	**Format supporté**
K-Means	Numérique
CAH	Numérique
HDBSCAN	Numérique
K-Prototype	Numérique et catégoriel
Amazon Dense Clus	Numérique et catégoriel
K-Mode	Catégoriel
Gower	Catégoriel et numérique

Tableau 2.2 Méthodes de classification non supervisées, numériques, ou catégorielles et numériques

L'ensemble de ces techniques a donc été testé. Les résultats issus des différentes comparaisons entre ces méthodes sont donnés à la Section 2.3. Aussi, il est pertinent d'essayer les algorithmes ne supportant que des données au format numérique sur des données qui sont originellement catégorielles, mais qu'on a converties à l'aide de différentes techniques, afin de ne pas être limité à un algorithme seulement lorsque nous avons des données au format mixte.

2.3 Justification de la méthode retenue

L'objectif de cette partie est de comparer les différentes méthodes évoquées dans l'état de l'art (Section 2.2) et de justifier le choix qui a été fait pour notre modèle. Dans un premier temps, il s'agit de comparer les méthodes de représentation de données catégorielles (ou catégorielles et numériques) en données numériques, puis, de comparer les méthodes de partitionnement.

2.3.1 Comparaison des méthodes de représentation de données catégorielles et de réduction de dimension

Énumération des méthodes potentiellement utilisables

Les algorithmes t-SNE et UMAP (voir Tableau 2.1) auraient pu être utilisés au lieu de l'ACM et l'AFDM. Mais ces algorithmes ne supportent que des données numériques en entrées et

ne sont donc utiles que dans l'objectif d'effectuer une réduction de dimension. Ainsi, bien que ces algorithmes auraient pu être plus pertinents que ACM et AFDM pour la partie réduction de dimension, ils ne pouvaient être sélectionnés uniquement en les combinant avec un algorithme qui permettait de convertir les données catégorielles en données numériques. Cette problématique est traitée par les auto-encodeurs, présentés dans l'état de l'art. Il existe différents auto-encodeurs qui ont chacun leurs spécificités [41]. Tandis que certains sont adaptés à des données labellisées, d'autres peuvent aussi être utilisés pour de l'apprentissage non supervisé. Ici, seuls les auto-encodeurs compatibles avec des données non labellisées sont pertinents, puisqu'on souhaite élaborer une classification non supervisée de nos données non labellisées. Il existe principalement 10 encodeurs répondant à ce critère, tous implémentés par la librairie scikit-learn [47] :

- Backward Difference Contrast [48]
- BaseN [49]
- Binary [50]
- Count [51]
- Hashing [52]
- Helmert Contrast [48]
- Ordinal [48]
- One-Hot [48]
- Polynomial Contrast [48]
- Sum Contrast [48]

Certaines de ces méthodes ne sont pas du tout adaptées à notre cas puisqu'elles causent une augmentation du nombre de dimensions bien trop importante en fonction du nombre de valeurs uniques présentes dans notre jeu de données. Cette augmentation de dimension (même après pré-traitement qui limite le nombre de valeurs uniques) ne peut pas être supportée, dans notre cas où l'on fait face à des jeux de données qui peuvent être volumineux. D'ailleurs, ces méthodes causent un arrêt du fonctionnement du noyau lors de l'exécution en raison d'un manque de mémoire vive (sur une machine de 32GB de RAM), avec seulement une petite partie des données. Seules les méthodes BaseN, Binary, Count, Ordinal et Hashing n'engendrent pas une importante augmentation de dimension et proposent des performances acceptables. Les auto-encodeurs peuvent permettre de représenter des données catégorielles en données numériques, mais engendrent une augmentation du nombre de dimensions. C'est pourquoi il est pertinent de les utiliser avant d'appliquer les algorithmes de réduction de dimensions qui ne fonctionnent que sur des données numériques. Ces couples (auto-encodeur, algorithme de réduction de dimensions) sont alors comparés aux algorithmes de représentation de données catégorielles en données numériques tels que AMC ou AFDM.

Expérimentations sur la vitesse d'exécution

Afin de décider lequel de ces encodeurs utiliser avec un algorithme de réduction de dimension, il convient d'étudier une contrainte importante dans notre cas, qui est celle de la vitesse d'exécution. En effet, notre algorithme doit pouvoir s'exécuter sur des volumes très importants de données pour pouvoir fonctionner en temps réel. C'est pourquoi un test de portant sur la vitesse d'exécution a été mené. Un échantillon de 123 538 données SSH (mélange de données catégorielles et numériques) a été choisi pour effectuer ce test. 10 couples (auto-encodeur, algorithme de réduction de dimension) ont été comparés. Ces 10 couples correspondent aux 5 auto-encodeurs cités précédemment ayant des performances acceptables et les deux algorithmes de réduction de dimension qui ne supportent que des données numériques en entrées, t-SNE et UMAP.

Ainsi, le couple (auto-encodeur / algorithme de réduction de dimension) le plus performant parmi les 10 couples existants s'est révélé être (Hashing, UMAP), avec un temps d'exécution de 5 minutes et 7 secondes. La très grande majorité du temps d'exécution est liée aux auto-encodeurs et non aux algorithmes de réduction de dimension. AFDM, sur ces mêmes données, s'est exécuté en 4.1 secondes. En termes de vitesse d'exécution, AFDM (et AMC) sont donc les seuls algorithmes pertinents pour notre cas d'usage.

Expérimentations sur la qualité de la représentation

Aussi, il est évident que la qualité de la représentation est un paramètre essentiel et que l'on ne pourrait pas résumer le choix de notre algorithme de représentation à la vitesse d'exécution. Néanmoins, il est bien plus compliqué de mesurer la qualité de la représentation des données catégorielles (ou catégorielles et numériques) en données numériques. Pour ce faire, voici la méthodologie proposée ici qui permet la comparaison de deux méthodes de représentation (aucune méthodologie de comparaison n'a été trouvée dans l'état de l'art) :

1. Choisir un algorithme de partitionnement (par exemple K-Means)
2. Exécuter le module de choix du nombre de clusters de notre modèle
3. Pour le nombre de clusters idéal proposé pour chacune des méthodes, comparer le score de Davies-Bouldin et le score Silhouette
4. La méthode avec le score de Davies-Bouldin le plus proche de 0 et le score Silhouette le plus proche de 1 est jugée comme étant la meilleure

En comparant les mêmes 10 couples à AFDM, seul le couple (Count, t-SNE) a eu de meilleurs résultats que l'AFDM, mais avec un temps d'exécution bien plus long. De plus, les résultats de AFDM étaient tout à fait satisfaisants au vu du temps d'exécution proposé, puisqu'on

obtient un score Silhouette de 0,96. On rappelle que le score Silhouette est borné entre -1 (pire classification) et 1 (meilleure classification). Le score de Davies-Bouldin obtenu est de 0.3 mais est plus difficilement interprétable puisqu'il n'est pas borné.

Il n'a donc pas été compliqué de sélectionner un algorithme de représentation de données, puisque AFDM propose une très bonne qualité de représentation (deuxième meilleure performance sur 10 méthodes et 0.96 de score silhouette) avec des temps d'exécution records. Aussi, cet algorithme est plus simple d'utilisation et se décline de différentes façons en fonction de la nature des entrées, comme le montre le Tableau 2.1.

2.3.2 Comparaison des algorithmes de partitionnement

Énumération des méthodes potentiellement utilisables

En ce qui concerne le choix de l'algorithme de partitionnement, les algorithmes retenus par la revue de littérature sont exposés au Tableau 2.2

L'idée d'appliquer des algorithmes de partitionnement qui acceptent des données catégorielles en entrée a été écartée. En effet, ces algorithmes sont généralement des adaptations d'algorithmes de partitionnement existants, pour lesquels on intègre une méthode de représentation des données. Par exemple, Amazon Dense Clus correspond en fait à l'utilisation en chaîne, et dans des conditions particulières, de UMAP et de HDBSCAN. Lorsque ce n'est pas le cas (comme avec K-Prototype ou K-Mode), il est impossible d'avoir une représentation numérique de nos données, ce qui empêche l'utilisation de métriques pour évaluer la qualité de notre partitionnement et donc pour choisir le nombre de clusters idéal. C'est aussi le cas pour la distance de Gower qui est une distance qui peut être utilisée dans des algorithmes de partitionnement afin d'utiliser directement des données catégorielles (ou catégorielles et numériques). CAH n'a pas pu être utilisé en raison de ses performances : le noyau a cessé de fonctionner alors que seul un échantillon de nos données a été utilisé.

Ainsi, les deux algorithmes de partitionnement pertinents pour notre cas d'usage sont K-Means et HDBSCAN. Contrairement aux méthodes de représentation de données catégorielles, il a été plus compliqué de départager ces deux algorithmes, c'est pourquoi le détail de notre comparaison est présenté ici.

Fonctionnement pratique

Premièrement, il faut noter que les deux algorithmes fonctionnent différemment. Il s'agit donc d'étudier leur fonctionnement pratique dans un premier temps. K-Means fonctionne

comme la grande majorité des algorithmes de partitionnement : un ensemble de points et un nombre de clusters désiré sont donnés en entrée, afin d'obtenir en sortie un numéro de partition correspondant à une classification pour chacun des points. Pour HDBSCAN, le fonctionnement diffère. Au lieu de donner un nombre de clusters souhaité avec des données en entrée, on donne le nombre de points maximum pouvant appartenir à un seul cluster (c'est-à-dire la taille maximale d'une partition). L'algorithme décide lui-même du nombre de clusters à définir. Aussi, la sortie d'HDBSCAN diffère, puisque tous les points ne sont pas classifiés. En effet, HDBSCAN crée une classe regroupant toutes les valeurs aberrantes. L'algorithme va donc décider qu'un ensemble de points n'est pas suffisamment pertinent pour obtenir un numéro de cluster comme les autres points. Ceci peut être vu comme un avantage ou un désavantage dépendamment des cas d'étude. En effet, parfois, on cherche à détecter les valeurs aberrantes d'un jeu de données et classifier les autres. Dans ce cas, HDBSCAN se révèle être un avantage. Dans d'autres cas, on souhaite que *tous* les points soient classifiés, car on a besoin d'un résultat en sortie. Pour notre étude, nos jeux de données correspondent à des données malveillantes uniquement, il est donc important que chaque entrée soit assignée à une classe. Néanmoins, il peut être intéressant d'étudier et de comprendre pourquoi certains points n'ont pas été classifiés, c'est pourquoi on décide de ne pas écarter directement HDBSCAN. En effet, dans le cas où HDBSCAN présente de meilleures performances que K-Means, on pourrait alors trouver un moyen de gérer et d'étudier la classe des valeurs aberrantes. Il faut également noter qu'il est plus compliqué pour HDBSCAN d'ingérer de nouvelles données (comparé à K-Means) et que HDBSCAN est souvent plus efficace sur des jeux de données qui ne sont pas très volumineux. K-Means convient donc plus à notre cas d'usage d'un point de vue pratique, puisqu'on souhaite obtenir une classe pour chaque donnée, on veut pouvoir ingérer aisément de nouvelles données et on veut un algorithme qui fonctionne sur des jeux de données volumineux.

Comparaison des performances

Comparons maintenant les performances de chacun des algorithmes. Nos tests se basent sur deux jeux de données. Le premier est un échantillon de 50,000 données SSH extraites des données de notre partenaire industriel. Le second est le jeu de données UNSW-NB15 présenté dans la section 2.1.3, comprenant 3 832 entrées. Il s'agit d'appliquer AFDM sur chacun de ces jeux de données afin d'avoir une représentation numérique en 3 dimensions. Puis, on utilise K-Means avec différente valeur de nombre de clusters et HDBSCAN avec différentes valeurs de taille maximale de cluster. On peut alors comparer les scores de Davies-Bouldin et Silhouette. Le score WCSS n'est pas employé ici puisqu'il ne traduit pas la qualité d'un partitionnement, il est surtout utile pour appliquer la méthode du coude afin de trouver le

nombre de clusters idéal.

Pour K-Means, on teste un partitionnement pour des valeurs de nombre de clusters allant de 2 à 19 clusters. Pour HDBSCAN, on crée une liste avec différentes valeurs de nombre maximal de clusters. Cette liste a été modifiée au fil des exécutions. Pour deux valeurs a et b de la liste, on teste davantage de valeurs dans un intervalle [a,b] si a et b ont obtenu les meilleurs scores de Davies-Bouldin et Silhouette. À l'inverse, on écarte les valeurs qui ne donnent pas de bons scores. Les comparaisons en termes de vitesse d'exécution se font sur SSH puisque c'est le jeu de données avec le plus de données. Il est donc important dans nos tests qu'il y ait autant d'exécution de chaque algorithme pour SSH afin de pouvoir comparer les vitesses d'exécution. Voici les valeurs de taille maximale de cluster qui ont été retenues pour SSH (Tableaux 2.3 et 2.4) et UNSW-NB15 (Tableaux 2.5 et 2.6). On donne aussi le nombre de clusters que suggère l'algorithme pour chacune de ces valeurs.

Taille maximale du cluster	5	10	50	100	250	400	500	600	700	800
Nombre de clusters	509	226	57	43	25	20	17	17	15	15

Tableau 2.3 Tailles maximales des clusters imposées et nombre de clusters associés pour SSH (partie 1 sur 2).

Taille maximale du cluster	900	1000	1250	1500	1750	2000	3000	4000
Nombre de clusters	14	14	11	11	11	11	10	8

Tableau 2.4 Tailles maximales des clusters imposées et nombre de clusters associés pour SSH (partie 2 sur 2).

Taille maximale du cluster	2	3	4	5	6	8	10	25	50	100
Nombre de clusters	313	234	191	164	141	102	82	32	18	10

Tableau 2.5 Tailles maximales des clusters imposées et nombre de clusters associés pour UNSW-NB15 (partie 1 sur 2).

Taille maximale du cluster	200	300	400	500	750	1000
Nombre de clusters	8	5	4	4	3	3

Tableau 2.6 Tailles maximales des clusters imposées et nombre de clusters associés pour UNSW-NB15 (partie 2 sur 2).

De ces premiers résultats, on peut déjà remarquer que le nombre de clusters imposé par l'algorithme est parfois trop grand pour être exploitable. En effet, il est compliqué de pouvoir

interpréter plus d'une vingtaine de clusters. Pour SSH, une taille maximale de 400 points par cluster serait le minimum, puisqu'avec moins de points, on obtient plus de 20 clusters. Pour UNSW-NB15, le nombre maximal de clusters est atteint pour une taille maximale de cluster de 50.

Les résultats pour SSH sont présentés dans les Figures 2.1 et 2.2. Une échelle logarithmique a été adoptée pour SSH pour une meilleure lisibilité des résultats. Les résultats obtenus pour UNSW-NB15 sont présentés dans les Figures 2.3 et 2.4. Rappel important : le score silhouette est compris entre -1 et 1. Plus on s'approche de 1, meilleure est la classification. À l'inverse, pour le score de Davies-Bouldin, compris entre 0 et $+\infty$, un score proche de 0 indique une meilleure classification.

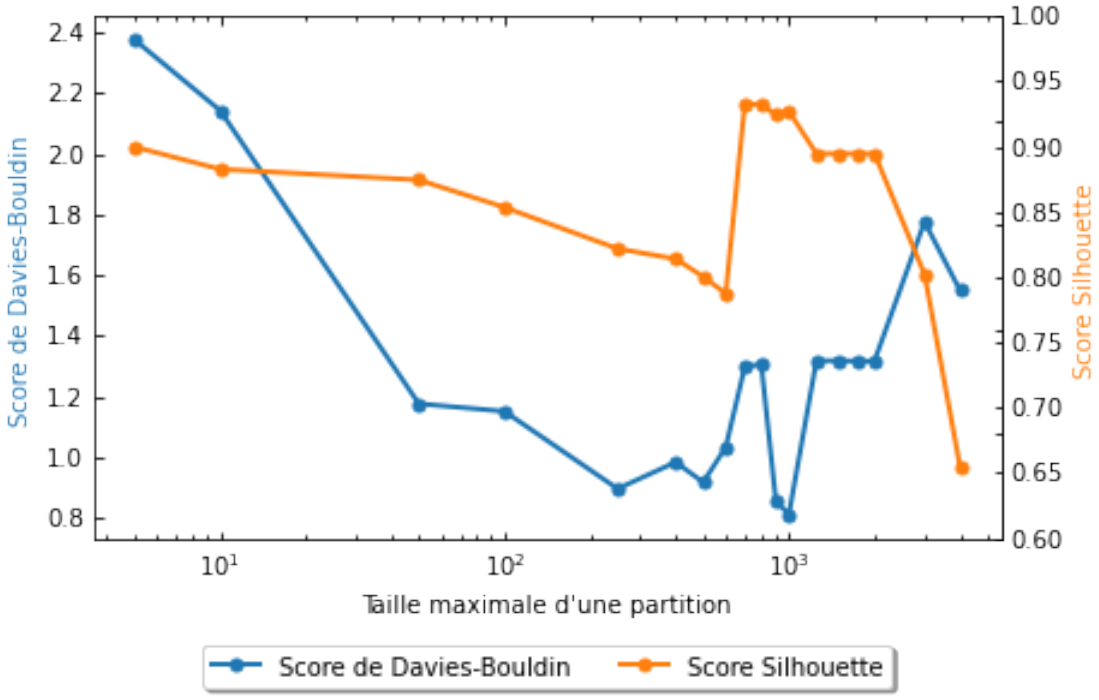


FIGURE 2.1 Résultats de HDBSCAN sur le jeu de données SSH.

Pour SSH, on remarque qu'avec HDBSCAN, le minimum du score de Davies-Bouldin est atteint pour une taille de maximale de 1000 points. Pour cette valeur, on obtient 99.44% du maximum du score Silhouette. La meilleure exécution de HDBSCAN a donc eu lieu pour une taille maximum de cluster de 1000 (ici aucune autre valeur ne pourrait être envisagée).

Avec K-Means, le score de Davies-Bouldin minimum est obtenu pour 13 clusters et le score Silhouette maximum est obtenu avec 14 clusters. À 13 clusters, on obtient 99.74% du score Silhouette maximum et à 14 clusters, on triple le score minimum de Davies-Bouldin. C'est

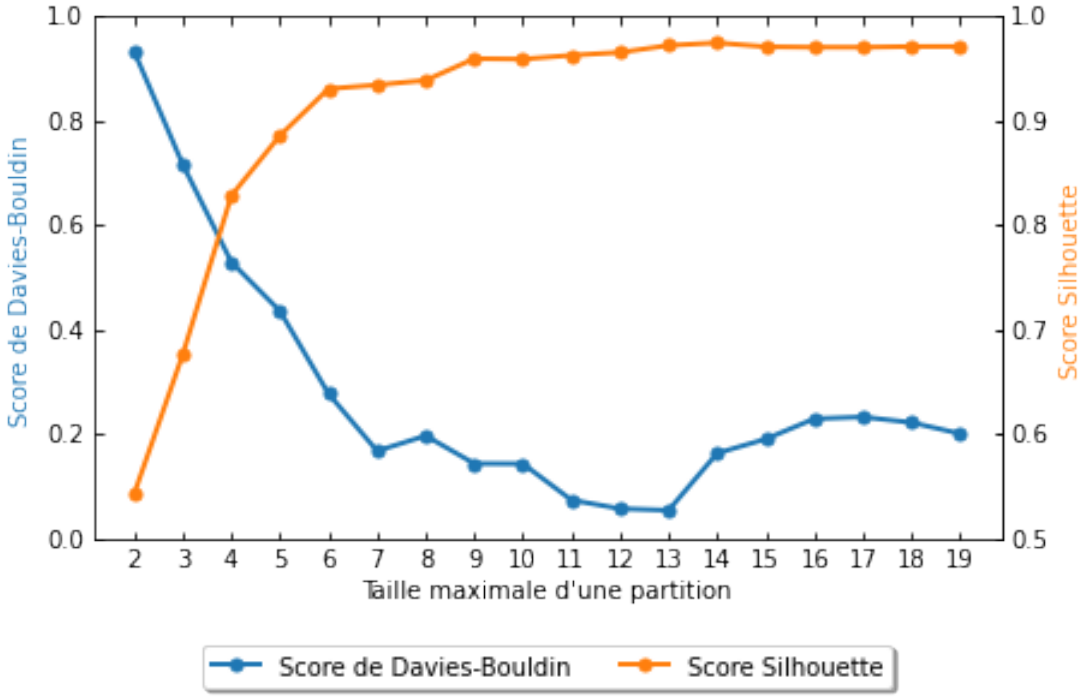


FIGURE 2.2 Résultats de K-Means sur le jeu de données SSH.

pourquoi la meilleure exécution pour K-Means a eu lieu pour 13 clusters.

On peut aussi remarquer que, pour la meilleure exécution de HDBSCAN (taille de cluster de 1000), HDBSCAN propose 14 clusters, ce qui est très proche de la meilleure exécution de K-Means (13 clusters). Ce résultat est plutôt rassurant, car les deux algorithmes convergent vers un nombre de clusters proche.

On peut alors comparer, pour chacune des deux méthodes, les scores Silhouette maximums, les scores de Davies-Bouldin minimums, ainsi que les scores obtenus pour la meilleure exécution de chaque algorithme. La dernière colonne de la Table 2.7 représente le ratio des scores obtenus $\frac{HDBSCAN_{score}}{K-Means_{score}}$. Dans le cas du score silhouette, si ce ratio est inférieur à 100%, cela signifie que K-Means est plus performant que HDBSCAN. Pour Davies-Bouldin, si ce ratio est inférieur à 100%, cela signifie que HDBSCAN est plus performant que K-Means.

Sur UNSW-NB15, pour HDBSCAN, on remarque que la meilleure exécution est la première (lorsque le score Silhouette est maximum et le score Davies-Bouldin minimum), avec une taille minimale de clusters de 2. Cependant, on obtient alors 313 clusters, ce qui est bien trop élevé. On ne devrait donc pas accepter cette valeur, bien qu'elle soit la meilleure. Cependant, il va être possible d'observer que même en acceptant cette valeur, HDBSCAN reste bien en dessous des performances de K-Means.

	HDBSCAN	K-Means	HDBSCAN/K-Means
Meilleur Silhouette	0.93271	0.97434	95.73%
Meilleur Davies-Bouldin	0.81075	0.05389	1505%
Silhouette meilleure exécution	0.92714	0.97180	95.40%
Davies-Bouldin meilleure exécution	0.81075	0.05388	1504%

TABLEAU 2.7 Meilleurs scores généraux et scores de la meilleure exécution pour chaque algorithme sur SSH

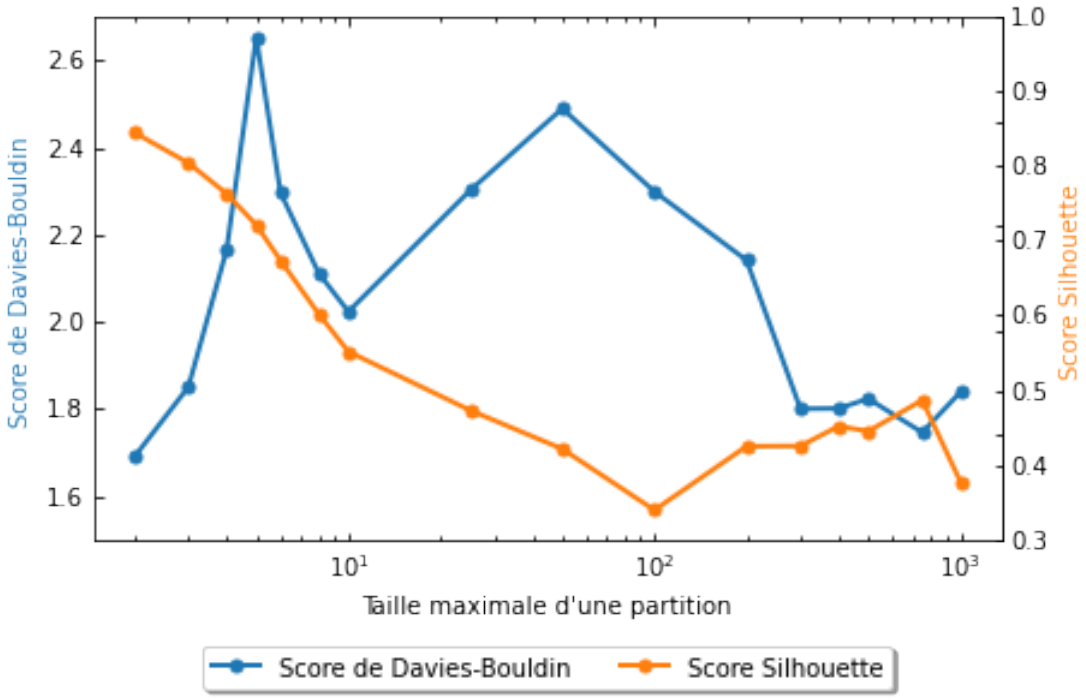


FIGURE 2.3 Résultats de HDBSCAN sur le jeu de données UNSW-NB15.

Pour K-Means, le meilleur score de Davies-Bouldin est obtenu pour 4 clusters et le meilleur score Silhouette pour 3 clusters. A 4 clusters, on obtient 99.87% du score Silhouette maximum et à 3 clusters, on obtient 106.73 du score de Davies-Bouldin minimum (on le dépasse donc de 6.73%). La meilleure exécution est donc obtenue avec 4 clusters. Les résultats sont donnés au Tableau 2.8

En ce qui concerne les temps d'exécution sur SSH (pour UNSW-NB15, ils ne sont que de quelques secondes comme le jeu de données est plus petit), K-Means a effectué les 18 exécutions en 9 minutes et 17 secondes, contre 17 minutes et 56 secondes pour les 18 exécutions de HDBSCAN. K-Means s'est donc exécuté presque deux (1.93) fois plus rapidement que HDBSCAN. Globalement, on remarque que K-Means obtient de bien meilleures performances sur

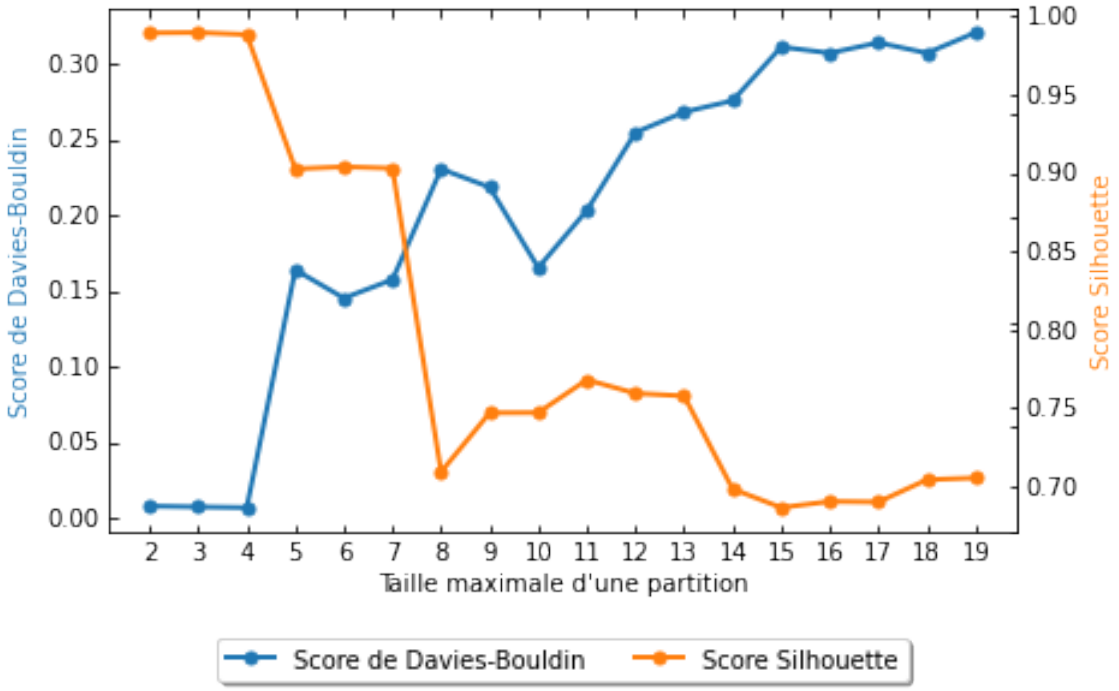


FIGURE 2.4 Résultats de K-Means sur le jeu de données UNSW-NB15.

	HDBSCAN	K-Means	HDBSCAN/K-Means
Meilleur Silhouette	0.84502	0.98983	85.37%
Meilleur Davies-Bouldin	1.68979	0.00684	24 704%
Silhouette meilleure exécution	0.84502	0.98853	85.48%
Davies-Bouldin meilleure exécution	1.68979	0.00684	24 704%

TABLEAU 2.8 Meilleurs scores généraux et scores de la meilleure exécution pour chaque algorithme sur UNSW-NB15

tous les points. En effet, sur les deux jeux de données, K-Means obtient un meilleur score Silhouette, un meilleur score de Davies-Bouldin, ainsi qu'un meilleur score Silhouette et de Davies-Bouldin pour la meilleure exécution, même en prenant la meilleure exécution de HDBSCAN qui n'aurait pas dû être comptabilisée. Enfin, il a été vu que le fonctionnement de K-Means convenait plus à notre cas d'usage que celui d'HDBSCAN, et que K-Means est presque deux fois plus rapide que HDBSCAN dans notre cas. C'est donc pour ces raisons que l'on choisit K-Means comme algorithme de partitionnement.

CHAPITRE 3 MÉTHODOLOGIE

L'objectif de ce chapitre est de présenter la méthodologie qui a été adoptée pour nos travaux. Dans un premier temps, il s'agit d'exposer les premières pistes qui ont été explorées en expliquant pourquoi elles ne se sont pas avérées pertinentes. Par la suite, le modèle qui a finalement été construit sera présenté de façon globale. Une explication plus précise de chaque module composant le modèle suivra cette présentation. Enfin, une dernière section aura pour objectif de justifier les différents choix qui ont été faits pour aboutir à ce modèle.

3.1 Méthodes explorées

Avant d'aboutir au modèle qui a finalement été retenu, différentes pistes ont été essayées. Il est intéressant de savoir pourquoi ces différentes pistes n'ont pas abouti pour mieux comprendre le modèle qui a finalement été construit. Afin de structurer cette section, il est important de souligner le fait que les données, ainsi que leur format, n'ont pas pu être communiquées dès le début de nos travaux. Plusieurs méthodes ont donc été testées sur des données composées d'alertes, avant d'obtenir les données de notre partenaire industriel. Ces méthodes sont exposées dans une première partie. Dans une deuxième partie, il s'agira de mettre en évidence l'impossibilité d'utiliser ces méthodes sur des données issues d'un pot de miel. Enfin, la présentation et la comparaison des différentes méthodes adaptées au format des données d'un pot de miel ont été données à la Section 2.3

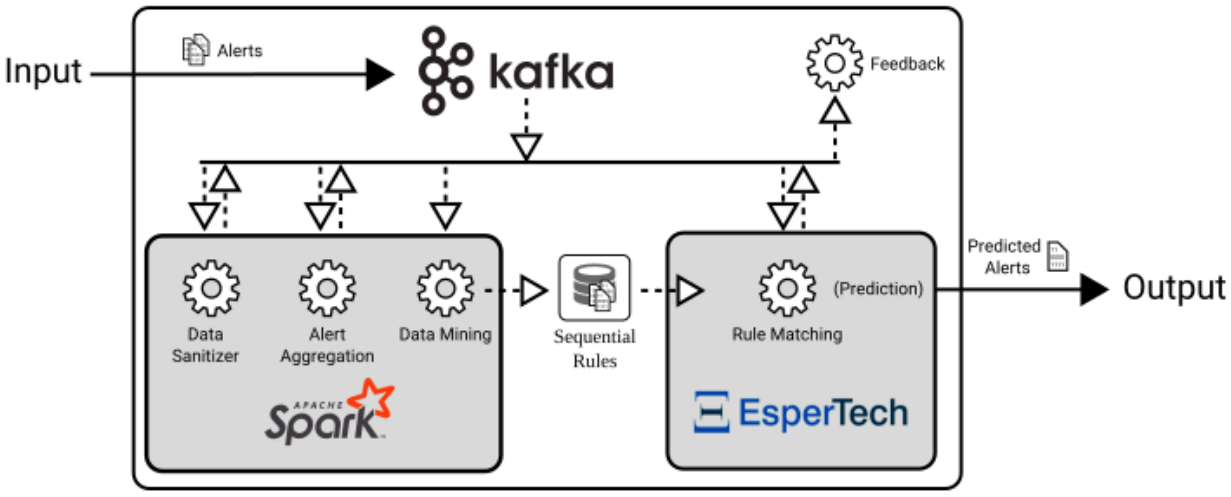
3.1.1 Méthodes de classification et de corrélation d'attaques sur des données composées d'alertes

Dans un premier temps, plusieurs solutions existantes ont été testées sur des données disponibles publiquement. Ces données correspondent à des captures du trafic réseau durant une compétition de cybersécurité, la National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) [53].

La première approche a été de tester un SIEM publiquement accessible et proposant un système de corrélation d'alertes. Prelude SIEM [54] a alors été choisi. Ce SIEM est utilisé avec l'IDS Suricata [7]. Les données ont donc été rejouées dans Suricata grâce à la capture réseau. En sortie, Suricata produit des alertes, qui alimentent Prelude SIEM. Le SIEM pourra alors mettre en œuvre son module de corrélation d'alertes [55]. Après l'utilisation du module de corrélation, le résultat n'a pas été jugé comme étant pertinent. Une inspection plus poussée

du module de corrélation explique ce résultat : le module de corrélation est un ensemble de 11 règles assez simples. Ces règles, codées en python, se limitent à traduire des actions telles que « Si plus de X tentatives de login en Y secondes ont été recensées, alors une tentative d'attaque par force brute a eu lieu » ou « Si plus de X évènements identiques sont recensés contre la même machine en Y secondes, alors une tentative de scan réseau a eu lieu ». Cet ensemble simpliste de règles ne permet donc pas de répondre à notre problématique, puisqu'il est compliqué d'extraire des scénarios de cyberattaques avec cette approche. Aussi, il est fort probable de passer à côté de certaines attaques, qui peuvent aisément contourner ces règles, qui sont accessibles publiquement. On peut tout de même noter la possibilité d'ajouter ses propres règles qui s'avère intéressante. Mais même avec cette possibilité, il est impossible de répondre à notre problématique, puisqu'il faudrait connaître l'ensemble des cyberattaques possibles et écrire des règles pour chacune. Même si cette hypothèse irréaliste pouvait être remplie, on ne pourrait pas détecter de nouvelles attaques. Globalement, cette approche montre que les SIEM existants (en tout cas, accessibles gratuitement) sont très limités en ce qui concerne la classification et corrélation d'alertes. Ils permettent néanmoins de réduire grandement le nombre d'alertes qu'un analyste aurait à traiter.

Une deuxième approche envisagée a été l'utilisation du framework AIDA [56]. L'architecture de ce framework est résumée sur la Figure 3.1.1. En entrée, on fournit des données composées d'alertes. Ces alertes sont traitées puis agrégées par un module, avant d'être stockées. Un module de data mining observe les enchaînements d'alertes les plus fréquents et les considère comme des attaques (ou des étapes d'attaques).



Ainsi, il est possible de faire de la prédiction d'attaques. Par exemple, en représentant une alerte par une lettre, si une séquence récurrente d'alertes est la séquence « A, B, C, D » et que

la séquence d'alertes « A, B, C » a été détectée, la prochaine alerte sera probablement « D ». Il est donc possible de réagir en conséquence en adaptant sa stratégie défensive, puisque l'on connaît le prochain mouvement de l'attaquant. Ce système remplit la majorité de nos critères, puisqu'il permet de détecter de nouvelles cyberattaques. En effet, le module de minage des séquences d'alertes les plus courantes est lancé de façon récurrente, ce qui permet de mettre à jour régulièrement la base de données de séquences d'alertes. Cependant, AIDA a certaines limites. Ce framework n'est pas capable de détecter des attaques qui s'étendent sur la durée. Si les alertes d'une attaque ne sont pas sur une courte période de temps, elles ne seront pas considérées comme une séquence et ne seront donc pas détectées. Aussi, le système pourrait souffrir d'une attaque par inondation (flooding) : en générant des séquences d'alertes n'ayant aucun sens, l'attaquant pourrait cacher la vraie séquence d'alertes traduisant son attaque, comme AIDA ne considère que les séquences d'alertes les plus fréquentes.

3.1.2 Élargissement du format des données acceptées en entrée

Après avoir reçu les données de notre partenaire industriel, nous avons préféré nous orienter vers des algorithmes compatibles à la fois avec des données issues d'alertes, ainsi qu'avec des données issues d'un pot de miel. Mais les données d'un pot de miel ne correspondent ni à une capture réseau (format PCAP), ni à des alertes. Or, la majorité des articles de recherches et des solutions existantes se basent sur ces formats pour de la classification et de la corrélation d'attaque.

Une première approche serait de convertir ces données dans un format standard, afin de pouvoir appliquer et adapter des méthodes existantes. De nombreuses tentatives ont été menées pour reconstruire les paquets au format PCAP. Ces tentatives se sont avérées infructueuses, puisqu'il est en fait impossible de convertir ces données au format PCAP. En effet, lorsqu'un paquet est reçu par le pot de miel, seulement certains attributs sont journalisés. Or, le format PCAP présente le paquet tel qu'il circule sur le réseau, avec tous ses attributs. On a donc une perte de données irréversible, ce qui rend la conversion impossible. Or, les IDS n'acceptent que des formats comme le format PCAP (qui représentent l'intégralité du paquet). Il n'est donc pas possible d'utiliser un IDS pour générer des alertes avec des données issues d'un pot de miel, ce qui limite l'utilisation de certaines des méthodes existantes dans l'état de l'art. Il est, cependant, toujours possible de s'en inspirer afin de construire un modèle fonctionnant sur des données issues d'un pot de miel `et` sur des données issues d'alertes.

L'approche retenue consiste à construire un modèle fonctionnant sur les données d'un pot de miel. Pour utiliser des données qui sont issues d'autres sources de données, il s'agit d'utiliser la capture réseau de ces données et d'en extraire les attributs pertinents pour notre modèle.

3.2 Présentation globale

Dans cette section, il s'agit de présenter le modèle finalement retenu, de façon macroscopique. Les différents modules composant le modèle seront présentés, mais ils ne seront détaillés que dans la section suivante.

Bien que le modèle pour HTTP et le modèle pour SSH se ressemblent, trois différences existent tout de même :

— Les attributs ne sont pas les mêmes (voir la Section 5.1).
— Les algorithmes de réduction de dimension et de conversion utilisés ne sont pas les mêmes, puisque le format des données n'est pas le même (on peut utiliser AMC ou AFDM)
— Le prétraitement des données diffère (valeurs des hyperparamètres différents), puisque le contenu des données est différent pour chaque protocole.

Ce qui est présenté dans cette section est commun aux deux protocoles étudiés et pourrait également être étendu à d'autres protocoles.

Le fonctionnement général du modèle ainsi que l'évolution d'un paquet à travers les différentes étapes du modèle sont représentés à la Figure 3.1.

Notre modèle prend en entrée des données ayant des attributs qui sont catégoriels et/ou numériques. En sortie de ce modèle, on obtient des clusters définissant des attaques. Le premier module de notre modèle est celui de prétraitement des données. Une fois les données prétraitées, le module de classification peut être utilisé. Ce module se divise en trois sous-modules. Un sous-module permet de représenter les données en trois dimensions (numériques) à l'aide d'un algorithme de réduction de dimension et de conversion de type de données. Comme on a plus de trois attributs numériques, on peut alors utiliser le second sous-module de classification, qui nous permet d'obtenir le nombre idéal de clusters pour nos données en utilisant trois métriques (présentées à la Section 3.3.5). Puis, le troisième sous-module, K-Means, est utilisé dans le but d'assigner un cluster à chaque entrée. Un module d'évaluation et d'interprétation, dont l'exécution est facultative, permet de faciliter l'interprétation des résultats obtenus et de les évaluer dans le cas où l'on utilise un jeu de données labellisé.

Les attributs utilisés pour chacun des protocoles sont indépendants de la date, de l'adresse IP source et du numéro de session. Bien qu'une dépendance temporelle pourrait permettre l'obtention de meilleurs résultats, elle empêcherait aussi la détection d'attaques qui seraient étalées dans le temps. En effet, en utilisant la date comme attribut, les données qui se rapprochent dans le temps seront également rapprochées dans notre représentation. Réciproquement, des données éloignées dans le temps seront éloignées dans notre représentation. Ainsi,

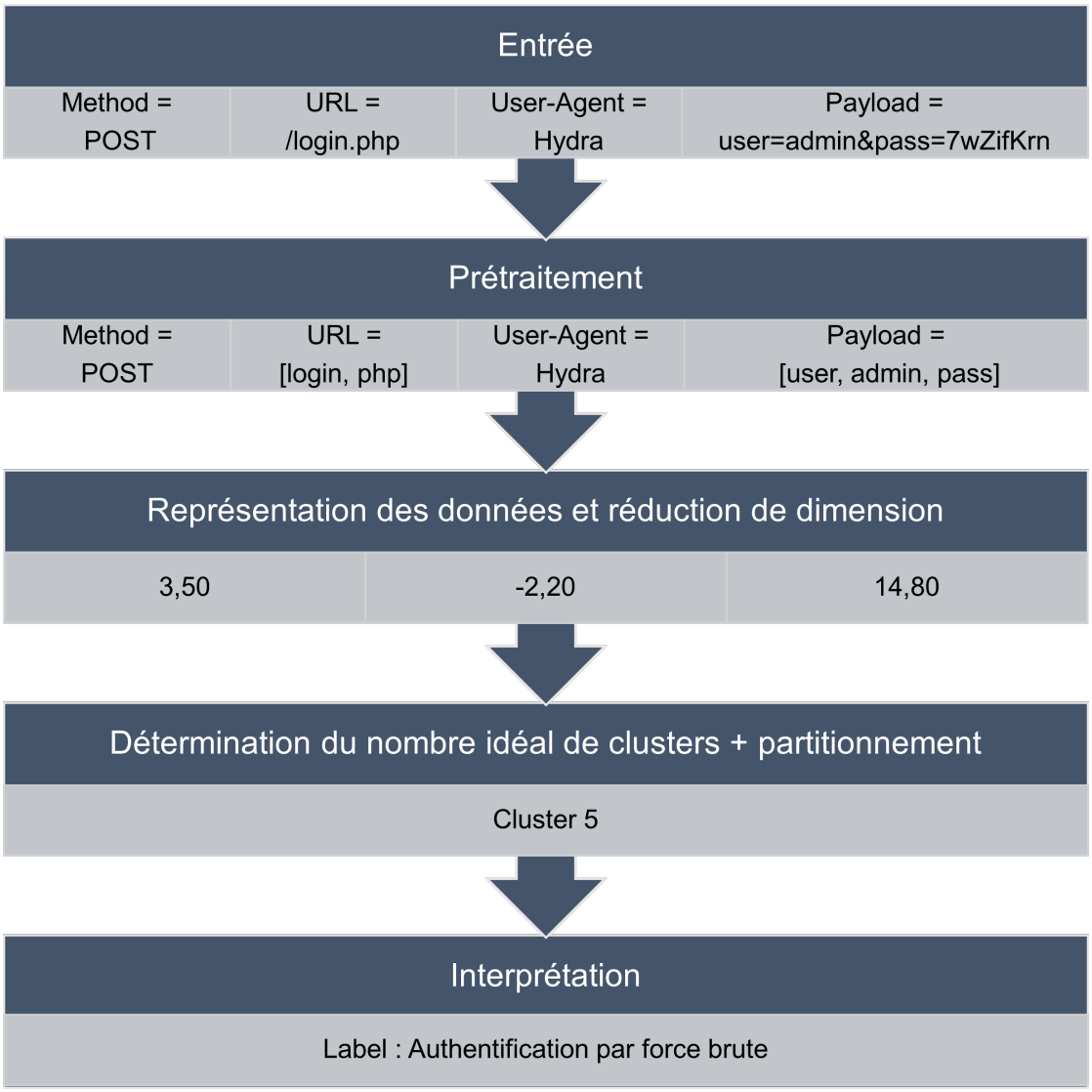


FIGURE 3.1 Structure du modèle et évolution d'un paquet à travers le modèle.

des données éloignées dans le temps, mais qui représente la même attaque, pourraient être dans des clusters différents si on choisissait un attribut relatif à la date dans notre classification. En écartant la date des attributs sélectionnés, on a alors autant de chance de détecter une attaque qui se fait sur quelques heures que sur quelques mois. Le fait d'ignorer aussi l'adresse IP et le numéro de session permet, pour un travail futur, l'application de techniques de corrélation. En effet, les attributs comme l'adresse IP et le numéro de session permettent d'établir des liens entre les différentes données. Ils permettent, sous certaines conditions, de pouvoir associer plusieurs données à un même attaquant.

Bien que notre modèle fonctionne sur des journaux qui ne recensent que certains attributs

(format des données du pot de miel de notre partenaire industriel), il est tout de même possible de l'utiliser sur d'autres formats de données. En effet, la seule condition est d'avoir accès à certains attributs des paquets envoyés. Le modèle construit est donc plutôt polyvalent. Même si le modèle doit être adapté pour chaque protocole, certaines similarités entre les protocoles pourraient faciliter l'adaptation de notre modèle. Par exemple, l'attribut `payload` pour HTTP pourrait être traité d'une façon assez similaire à l'attribut `message` dans SSH. Par extension, tout attribut ressemblant à une charge utile dans n'importe quel autre protocole pourrait être traité de la même façon.

Il faut également noter que des cyberattaques réelles peuvent contenir des nombres de données tout à fait différents et il est primordial de pouvoir détecter une attaque qui ne concerne que quelques dizaines de paquets parmi plusieurs milliers. En théorie, notre modèle supporte très bien les jeux de données déséquilibrés par construction. En effet, aucun paramètre ne dépend du nombre de données totales et K-Means et AFDM ou AMC sont connus pour supporter les jeux de données déséquilibrés.

3.3 Présentation détaillée du modèle

Il s'agit ici de présenter en détail le fonctionnement de chacun des modules du modèle. En plus du choix initial des attributs, notre modèle est composé de trois modules :

- Module de prétraitement
- Module de classification (comprenant trois sous-modules : représentation des données, définition du nombre de clusters et K-Means)
- Module d'évaluation et d'interprétation

3.3.1 Choix des attributs

Avant d'exécuter le module de prétraitement, il s'agit de choisir correctement les attributs sur lesquels s'applique notre modèle. Ces attributs doivent être pertinents ; ils doivent être utiles à la classification des données, sans comporter trop de valeurs manquantes. En effet, les algorithmes tels que l'ACM et l'AFDM supportent très mal les attributs peu représentés. Ceci s'explique par le fait que ces algorithmes calculent des distances entre les points. Ainsi, les classes peu représentées vont créer de plus en plus de distance entre les points, puisqu'elles apportent peu de similarité entre ceux-ci. Ces grandes distances vont alors absorber les faibles différences qu'il y a entre les classes correctement représentées, ce qui limitera les performances de notre algorithme et donc de notre classification. Notre algorithme se base sur l'inspection de paquet, il est donc possible de choisir des attributs catégoriels propres à

un protocole. Pour que ce modèle fonctionne de façon idéale, il faut suivre cette règle : les attributs sélectionnés doivent pouvoir représenter la plus grande partie caractéristique d'un paquet, tout en limitant le nombre d'attributs faiblement représentés et sans ajouter d'informations superflues. Il y a donc parfois un compromis à trouver : bien que certains attributs pourraient améliorer la représentation d'un paquet, ils ne sont pas retenus, car trop peu représentés. Aussi, les attributs peu caractéristiques d'un paquet ne doivent pas être retenus puisqu'ils apportent principalement du bruit et étouffent les autres attributs qui apportent plus d'informations. Afin d'illustrer ceci, il convient de donner un exemple avec HTTP. Dans le cas d'HTTP, les attributs qui ont été retenus sont les suivants : `method`, `url`, `user-agent` et `payload`. Lorsqu'on observe un paquet HTTP, on remarque que ces attributs représentent la quasi-totalité d'un paquet et que seul un des attributs (payload) est peu représenté. Les seules informations manquantes pour décrire un paquet seraient certainement les en-têtes (headers). Mais ces en-têtes ne sont pas toujours présents dans les paquets et diffèrent généralement d'un paquet à l'autre. Aussi, ils apportent bien moins d'informations que les quatre attributs précédents. En effet, il est difficile de caractériser une attaque grâce aux en-têtes, ceux-ci apportent bien souvent des informations complémentaires qui ressemblent davantage à des métadonnées. Enfin, d'autres attributs, qui peuvent être considérés comme des métadonnées, auraient pu être considérés. On peut notamment penser aux attributs retenus par les travaux présentés dans l'état de l'art à la Section 2, comme la longueur d'un paquet, le port destination (qui, pour HTTP, est très souvent le même et qui n'est pas vraiment caractéristique d'une attaque), le temps d'arrivée inter-paquet, le numéro de la séquence TCP, etc. Ces attributs peuvent être très nombreux (85 pour CIC-IDS2017 et 49 pour UNSW-NB15) mais ne sont pas suffisamment caractéristiques d'un paquet (et encore moins d'une attaque) pour être retenus pour l'instant. Pour travailler avec de tels attributs, il s'agirait de mettre en place un système automatisé de sélection d'attributs, comme le test d'importance de Boruta [25], afin de ne pas sélectionner des attributs qui nuiraient à la contribution des attributs caractéristiques initialement sélectionnés. Il faudrait probablement tout de même limiter la contribution de certains de ces autres attributs, même si on ne choisit que les plus pertinents. Ceci présente une amélioration future de notre modèle. Dans nos travaux, l'objectif est donc de choisir un ensemble d'attributs qui vont permettre de décrire correctement un paquet, sans informations superflues qui absorberaient la contribution des attributs décrivant un paquet.

3.3.2 Module de prétraitement

Les données à prétraiter

Lors de son exécution, ce module va dans un premier temps réunir toutes les données qui sont présentes dans différents fichiers en une seule trame de données. Pour ce faire, on utilise Pandas [57], qui est une librairie permettant notamment la manipulation de données. Les attributs numériques ne reçoivent aucun prétraitement dans cette phase. En effet, la normalisation des données se fait automatiquement lors de l'exécution du module de représentation des données.

Utilisation de l'ACM et l'AFDM sur des attributs catégoriels

En ce qui concerne les attributs catégoriels (texte), une distinction est faite en fonction du nombre de valeurs uniques de l'attribut. L'ACM et l'AFDM donnent de meilleurs résultats lorsque les valeurs uniques pour chaque attribut ne sont pas nombreuses. En effet, ces algorithmes se basent sur la différence des valeurs des attributs des entrées : si deux entrées n'ont aucun attribut catégoriel identique, leurs coordonnées numériques seront éloignées. À l'inverse, deux entrées ayant les mêmes valeurs pour chacun des attributs catégoriels seront très proches (les données numériques permettront de les éloigner plus ou moins dans le cas d'AFDM). Ainsi, s'il y a trop de valeurs différentes possibles pour les attributs catégoriels, il sera très rare que deux entrées aient la même valeur. Dans le cas extrême où chaque entrée a une valeur différente pour un attribut catégoriel, alors cet attribut n'apporte aucune information puisqu'aucune entrée n'aura la même valeur pour cet attribut. Aussi, l'utilisation d'attributs catégoriels vient avec un autre inconvénient : deux valeurs qui se ressemblent considérablement seront considérées comme étant totalement différentes par l'ACM et l'AFDM. Par exemple, les charges utiles « h=die(@md5(Apri1)) » [1] et « m=die(@md5(Apri1)) » seront considérées comme étant totalement différentes par l'Analyse en Correspondance Multiples, alors que l'attaque tentée est la même pour ces deux charges utiles.

Suppression des mots à faible occurrence d'apparition

Il faut donc limiter le nombre de valeurs uniques et considérer deux valeurs représentant la même attaque comme étant identique. C'est justement le rôle du module de prétraitement des données. Pour expliquer ce module, nous allons reprendre l'exemple précédent d'une charge utile HTTP. Premièrement, il s'agit d'extraire les mots-clés de la charge utile (tokenization). Tous les signes de ponctuation sont considérés comme étant des séparateurs. Ainsi, pour

1. Payload utilisé pour exploiter la vulnérabilité CVE-2019-16759

la charge utile « h=die(@md5(Apri1)) », on obtiendra « h, die, @md5, Apr1 ». Puis, un dictionnaire est construit avec tous les mots présents dans les charges utiles, tout en comptant la présence de chaque mot. On peut ainsi éliminer tous les mots qui ne sont pas récurrents. La récurrence minimale est un hyperparamètre du modèle que l'on peut nommer $récu_{min}$. Avec une récurrence minimale de 0, on garde tous les mots. Avec une récurrence minimale de 1, on supprime tous les mots qui ne sont apparus qu'une fois. Ceci permet de supprimer une grande quantité de bruit dans les charges utiles ; les mots qui n'apparaissent qu'une fois ne sont pas les plus pertinents, puisque leur participation à l'élaboration de la classification est limitée (on cherche surtout les similarités entre les charges utiles, plutôt que leurs différences). À cette étape, on a extrait les mots-clés de la charge utile.

Similarité entre deux valeurs

Par la suite, il s'agit de rendre égales deux charges utiles très similaires. Cette étape va donc considérer les deux charges utiles « h, die, @md5, Apr1 » et « m, die, @md5, Apr1 » comme étant égales. Pour ce faire, il s'agit d'évaluer la similarité entre chacune des charges utiles à l'aide de la libraire Difflib [58]. Cette libraire permet d'évaluer la similarité entre deux chaines de caractères en retournant une valeur entre 0 (totalement différentes) et 1 (parfaitement égales). On fixe alors un seuil de similarité à partir duquel les deux charges utiles sont considérées comme étant identiques. Ce seuil de similarité est un nouvel hyperparamètre de ce modèle. Notons cet hyperparamètre $taux\_similarité_{min}$. Il s'agira donc d'ajuster $taux\_similarité_{min}$ afin d'obtenir les meilleurs résultats possibles. Cette façon de traiter les attributs catégoriels a été choisie puisqu'elle permet de détecter certains motifs caractéristiques dans les attributs. Dans l'exemple avec « h, die, @md5, Apr1 », on remarque que la charge utile a la structure suivante : une variable représentée par une lettre, suivie du signe égal (qui a joué le rôle de séparateur), le mot clé « die », puis « md5 » et un autre mot-clé pouvant varier. Cette structure va être facilement repérée par difflib qui va pouvoir reconnaître des charges utiles similaires à celle-ci. Cette étape est particulièrement demandeuse en temps de calcul. En effet, il faut comparer chaque charge utile à toutes les autres. Afin d'optimiser notre algorithme, une liste des charges utiles qui ont déjà été testées a été mise en place, afin de gagner du temps.

Il est donc possible, grâce à ce module, de réduire le nombre de valeurs uniques pour chaque attribut, en ajustant le seuil de similarité à partir duquel deux valeurs pour un attribut sont considérées comme étant identiques. Les attributs catégoriels pour lesquels il n'y a initialement que très peu de valeurs uniques ne sont pas concernés par ce module. C'est par exemple le cas de l'attribut « method » pour HTTP, qui ne prend que quelques valeurs

différentes (9 en théorie). Il est dans ce cas inutile d'essayer de comparer des valeurs de l'attribut « method » afin de détecter des similarités, puisqu'il n'y en aurait aucune (chaque méthode est unique et différente des autres).

Le module de prétraitement a donc pour but de trouver des similarités entre les valeurs prises par chacun des attributs concernés. On observe alors une diminution du nombre de valeurs uniques pour chacun des attributs. Lors de l'exécution de la phase de prétraitement, l'algorithme montre quelles sont les valeurs qui ont été considérées comme étant égales, avec leur taux de similarité. On peut ainsi inspecter cette sortie afin de déterminer au mieux $taux\_similarité_{min}$. La méthodologie pour déterminer $taux\_similarité_{min}$ est la suivante : on commence avec une valeur suffisamment basse (comme 0.3). En sortie, on pourra voir quels sont les valeurs qui ont été considérées comme étant égales alors qu'elle ne devait pas l'être. Il faut alors regarder le taux de similarité de ces sorties et relancer l'algorithme avec un $taux\_similarité_{min}$ plus élevé. On peut alors répéter cette opération jusqu'à ce que l'on considère que les valeurs qui sont jugées égales devraient effectivement être jugées comme étant égales. À l'inverse, on peut se rendre compte dans le fichier qui regroupe toutes les valeurs les plus communes pour chaque attribut pour chaque cluster (présenté à la Section 3.3.6) que certaines valeurs sont présentes en double et ne sont pas correctement regroupées. Dans ce cas, on peut alors diminuer ce taux de similarité afin de regrouper de telles valeurs.

Évolution du module de prétraitement

Il serait compliqué d'automatiser cette étape, puisque seule une analyse manuelle permet de définir si l'égalisation de deux valeurs différentes par l'algorithme est pertinente ou non. Néanmoins, il serait possible, dans un premier temps, de semi-automatiser cette étape. Pour ce faire, il s'agirait d'ajuster automatiquement les hyperparamètres en demandant à l'utilisateur exécutant le modèle si certaines valeurs devraient être considérées comme étant égales ou non. Avec l'exemple précédent, le modèle demanderait à l'utilisateur si « h, die, @md5, Apr1 » et « m, die, @md5, Apr1 » représentent la même charge utile. Si l'utilisateur répond « Oui » et que le modèle n'allait pas les considérer comme étant égales avec les hyperparamètres actuels, le modèle diminue le taux de similarité minimum et continue ses calculs. Si l'utilisateur répond « Non » et que le modèle allait les considérer comme étant égales avec les hyperparamètres actuels, le modèle augmente le taux de similarité minimum et continue ses calculs. Au bout d'un certain nombre de réponses identiques entre l'utilisateur et le modèle, on estime qu'on a trouvé les bons hyperparamètres. Une fois ce travail réalisé manuellement plusieurs fois, on pourrait essayer d'utiliser un modèle de machine learning pour automatiser totalement cette tâche.

Système de sauvegarde

Cette étape est sans aucun doute la plus longue de notre modèle. C'est pourquoi un système de sauvegarde a été mis en place manuellement. On peut alors exécuter l'algorithme en plusieurs fois et pas seulement en une seule fois, ce qui pourrait demander plusieurs jours de calculs (sans nous mettre à l'abri d'une éventuelle erreur). De cette façon, l'ingestion de nouvelles données par notre modèle est grandement facilitée. La sauvegarde permet de restaurer une session de calcul, en y ajoutant de nouvelles données. Les nouvelles données seront alors comparées aux données initiales afin d'évaluer leurs similarités (il aurait pu être problématique que de nouvelles données ne soient comparées qu'entre elles-mêmes et pas avec les anciennes données).

Avec cette étape, on obtient un jeu de données avec peu de valeurs uniques, ce qui nous permet d'appliquer de façon pertinente des algorithmes de représentation de données, puis, de classification.

3.3.3 Module de classification

Le module de classification se décompose en trois sous-modules :

- un module de conversion des données catégorielles en données numériques et de réduction de dimension (appelé représentation numérique des données)
- un module de détermination du nombre idéal de clusters
- un module de clustering (K-Means)

3.3.4 Module de représentation numérique des données

Ce module prend en entrée le jeu de données prétraité (voir section 3.3.2). L'objectif de ce module est d'avoir une représentation numérique en trois dimensions de notre jeu de données. Pour cela, on utilise ACM si l'on a seulement des données numériques et AFDM si l'on a un mélange de données numériques et catégorielles. Dans notre cas, nous n'avons pas eu à traiter uniquement des données numériques. Mais on aurait pu utiliser une ACP si le cas se présentait.

ACP, ACM et AFDM font partis de la même famille d'algorithmes de réduction de dimension. Présentons leur fonctionnement. Si l'on a P variables numériques $\{p_i, i = 1...P\}$ et Q variables catégorielles $\{q_i, i = 1...Q\}$ et qu'on note, pour une variable quantitative z, $r(z, p)$ le coefficient de corrélation entre z et p et $\eta(z, q)$ le rapport de corrélation entre z et q, alors on a :

— Pour l'ACP : on cherche une fonction qui attribue à chaque entrée une valeur et qui maximise $\sum_{p=1...P} r^2(z,p)$
— Pour l'ACM : on cherche une fonction qui attribue à chaque entrée une valeur et qui maximise $\sum_{q=1...Q} \eta^2(z,q)$
— Pour l'AFDM : on cherche une fonction qui attribue à chaque entrée une valeur et qui maximise $\sum_{p=1...P} r^2(z,p) + \sum_{q=1...Q} \eta^2(z,q)$ (la contribution de chaque variable à ce critère est bornée par 1 pour que chaque type de variable ait le même rôle)

3.3.5 Module de détermination du nombre idéal de clusters

L'objectif de ce module est de connaître le nombre de clusters à adopter afin d'effectuer le partitionnement. Pour ce faire, notre modèle se base sur trois métriques : la Somme des Carrés à l'Intérieur d'un Groupe (Within-Cluster Sum of Squares Score, abrégé ici WCSS), le score de Davies-Bouldin et le score Silhouette. Chacune de ces métriques fournit une indication du nombre idéal de clusters pour un ensemble de données. En comparant les résultats des différentes métriques, un nombre idéal de clusters pour nos données émerge. WCSS est lié à une méthode, la méthode du coude (elbow method). Cette méthode suggère un nombre de clusters pour un modèle de clustering et est détaillée ci-après. Le score Silhouette doit être aussi proche que possible de 1 et le score Davies-Bouldin doit être aussi proche que possible de 0. En comparant le résultat de la méthode Elbow et des deux autres scores pour différents nombres de clusters, le choix du nombre de clusters converge vers un unique nombre.

Score WCSS et méthode du coude

La méthode du coude permet de déterminer le nombre idéal de clusters pour un partitionnement. Pour utiliser la méthode du coude, le score WCSS doit être calculé pour différents nombres de clusters. Le score WCSS est la somme des distances au carré entre chaque point et le centroïde d'un cluster. Voici la formule du score WCSS :

$$WCSS = \sum_{C_k in C} (\sum_{d_i in C_k})(distance(d_i, C_k)^2)$$

Où C est l'ensemble des centroïdes des clusters et d_i un point de données dans un cluster C_k

Dans notre cas, nous calculons le WCSS de 2 à 20 clusters. Ensuite, nous traçons le graphique de l'évolution du score WCSS en fonction du nombre de clusters. Le nombre idéal de clusters est situé au point de coude. Le point de coude est le point où les rendements décroissants ne valent plus le coût d'un cluster supplémentaire. Cela signifie que l'ajout d'un cluster supplémentaire n'aide plus à représenter les données. Graphiquement, on obtient une visualisation

en forme de coude, dans laquelle le nombre optimal de clusters est le point représentant la pointe du coude, c'est-à-dire le point correspondant au nombre de clusters à partir duquel la variance ne diminue plus significativement.

Score de Davies-Bouldin

Ce score est défini comme la mesure de similarité moyenne de chaque cluster avec son cluster le plus similaire. La similarité est le rapport entre les distances intra-clusters et les distances inter-clusters. Le score minimum est de zéro, et des valeurs plus faibles indiquent un meilleur clustering. La formule pour le score de Davies-Bouldin est la suivante :

$$DB = \frac{1}{n}\sum_{k=1}^{n}\max_{k'\neq k}(\frac{\delta_k + \delta_{k'}}{d(\mu_k, \mu_{k'})})$$

Où $\mu_k = \frac{1}{|C_k|}\sum_{i\in C_k} d_i$ et $\delta_k = \frac{1}{|C_k|}\sum_{i\in C_k} d(d_i, \mu_k)$ (d est la distance euclidienne)

Score Silhouette

Pour chaque point, le coefficient de Silhouette est la différence entre la distance moyenne aux points du même groupe (cohésion) et la distance moyenne aux points des autres groupes voisins (séparation). Si cette différence est négative, le point est en moyenne plus proche du groupe voisin que du sien ; il est donc mal classé. À l'inverse, si cette différence est positive, le point est en moyenne plus proche de son groupe que du groupe voisin ; il est donc bien classé. Le score Silhouette est la moyenne du coefficient de silhouette de tous les points. Le score (et le coefficient) de Silhouette varie entre 1 (meilleur classement) et -1 (pire classement). Voici la formule du score Silhouette :

$$S_{sil} = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{|C_k|}\sum_{iinC_k} s_{sil}(i)$$

où le coefficient de silhouette est : $s_{sil}(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$. et a(i) est la distance moyenne du point à son groupe : $a(i) = \frac{1}{|C_k|-1}\sum_{j\in C_k, j\neq i} d(d_i, d_j)$ et b(i) la distance moyenne du point à son groupe voisin : $b(i) = \min_{k'\neq k}\frac{1}{|C'_k|}\sum_{i'\in C_{k'}} d(d_i, d_{i'})$

K-Means

K-Means [59] est un algorithme de clustering très populaire. Avec K-Means, on considère la distance d'un point à la moyenne des points de son cluster. Ainsi, la fonction à minimiser est la somme des carrés de ces distances. Ainsi, K-Means est un algorithme permettant de

résoudre un problème d'optimisation où l'on veut trouver : $\arg\min_C \sum_{k=1}^{n} \sum_{\mathbf{d}_i \in C_k} \|\mathbf{d}_i - \boldsymbol{\mu}_k\|^2$ où les μ_k sont les barycentres des points dans chaque C_k.

En sortie du module de classification, on a donc réussi à assigner un cluster à chacune de nos entrées.

3.3.6 Module d'interprétation et d'évaluation

L'objectif de ce module est d'interpréter les résultats des modules précédents et de les évaluer dans le cas où le jeu de données utilisé est labellisé.

En ce qui concerne l'interprétation, le module propose une visualisation du partitionnement en trois dimensions. En plus de cela, un fichier est créé pour chaque attribut et l'algorithme regroupe les valeurs les plus courantes dans chaque fichier pour chaque cluster. Voici un exemple afin d'illustrer ce module : pour HTTP, en choisissant les attributs `method`, `user-agent`, `url` et `payload`, un fichier sera créé pour chacun de ces attributs. Le fichier *payload* indiquera, pour chaque cluster, quelles sont les valeurs les plus courantes. Il est alors plus aisé, en croisant le contenu de chacun de ces fichiers, de comprendre pourquoi les données présentes dans chacun des clusters ont été regroupées ensemble. Dans le cas où les fichiers sont trop volumineux, ce qui peut nuire à leur lisibilité, on peut choisir de n'afficher que les entrées uniquement à partir d'une certaine récurrence. On peut par exemple décider de ne pas afficher les payloads qui n'apparaissent qu'une fois. En effet, généralement, les quelques entrées les plus communes peuvent suffire à donner un sens à un cluster.

Dans le cas où les données sont labellisées, on peut aussi utiliser le module d'évaluation. Ce module permet d'obtenir des métriques qui peuvent être utiles pour comparer notre modèle à d'autres modèles de classification. Pour ce faire, il faut de prime abord définir quelques notions. Dans un algorithme de classification binaire (l'ensemble des classes existantes est $\{0, 1\}$), on définit :

- les vrais positifs (VP) : valeurs classifiées positives et qui sont effectivement positives
- les faux positifs (FP) : valeurs classifiées positives alors qu'elles étaient en fait négatives
- les faux négatifs (FN) : valeurs classifiées négatives alors qu'elles étaient en fait positives
- les vrais négatifs (VN) : valeurs classifiées négatives et qui sont effectivement négatives

Avec ces définitions, on peut définir les métriques couramment utilisées dans des problèmes de classification :

- la précision : sur l'ensemble des prédictions positives, combien étaient vraiment positives ? La précision est donnée par : $\frac{VP}{VP+FP}$
- le rappel (ou « recall » en Anglais) : sur l'ensemble des valeurs qui étaient vraiment

positives, combien ont été classifiées comme étant positives ? Le rappel est donné par : $\frac{VP}{VP+FN}$

— Le score F1 : métrique permettant de prendre en compte à la fois la précision et le rappel. Le score F1 est donné par : $\frac{2\times Precision\times Rappel}{Precision+Rappel} = \frac{VP}{VP+\frac{1}{2}(FP+FN)}$

Dans le cas d'une classification multi-classes qui comporte donc plus que deux classes, les métriques précédentes doivent être redéfinies. Une première approche est de calculer le nombre de VP, FP et FN pour chaque classe puis de calculer le score F1 pour chacune des classes. Cette approche se nomme « un-contre-tous » (One-Versus-Rest) : pour chaque classe, on calcule un score F1 en considérant un problème de classification binaire. Par la suite, on peut alors calculer un score F1 moyen grâce à chacun de ces scores. Cette approche se nomme le macro-moyennage (macro-averaging). On remarque que si une classe n'est que très peu représentée (peu d'entrées), la contribution de son score F1 au score F1 final sera la même qu'une classe très représentée (avec de nombreuses entrées). C'est pourquoi cette façon de calculer le score F1 ne convient pas aux jeux de données déséquilibrés. Dans notre cas, on a rarement des jeux de données équilibrés en cybersécurité (voir le Chapitre 2 sur l'état de l'art). Deux approches existent pour traiter les jeux de données déséquilibrés. On peut pondérer la moyenne du score F1 avec le support (c'est-à-dire le poids qu'occupe une classe par rapport à l'ensemble du jeu de données). Le moyennage pondéré (weighted-averaging) n'est pas la seule méthode existante dans la littérature. En effet, il existe également le micro-moyennage (micro-average). Cette approche consiste à appliquer la formule du score F1 en ayant calculé le nombre cumulé de VP, FP et de FN : $\frac{VP}{VP+\frac{1}{2}(FP+FN)}$. On peut alors montrer que le score F1 sera égal à la précision et au rappel dans ce cas.

Le module d'évaluation donne donc accès à l'ensemble de ces métriques (score F1, précision et rappel par macro-moyennage, moyennage pondéré et micro-moyennage). Ceci permet une comparaison avec des méthodes de classification supervisées étudiées dans l'état de l'art (Section 2).

3.4 Application de notre modèle à d'autres formats de données

Afin d'utiliser notre modèle, un certain format est attendu. En effet, notre modèle effectue de l'inspection de paquet et doit donc avoir accès à de nombreuses informations contenues dans ces paquets. Comme expliqué dans la Section 5.1, les pots de miel journalisent ces informations au format NDJSON (Newline Delimited JavaScript Object Notation) [60] et sont donc directement utilisables. L'algorithme a d'ailleurs été conçu pour répondre à la particularité de ce format. L'objectif de cette section est de montrer que l'on peut utiliser aussi notre modèle sur des jeux de données standards. La seule contrainte est d'avoir accès

aux données désirées au format PCAP (capture réseau). Ce format est réputé pour être standard et est souvent proposé par les auteurs des jeux de données. Cependant, des jeux de données ne proposent que les valeurs pour certains attributs présélectionnés. Ces attributs sont numériques et communs à tous les protocoles, ce qui ne peut pas convenir à notre modèle. Seuls ces jeux de données ne pourront pas être utilisés par notre modèle. La capture réseau au format PCAP peut s'effectuer avec des logiciels comme Wireshark [61]. Avec ce logiciel, elle s'obtient très aisément (il suffit de lancer une capture sur l'interface désirée). Une fois que l'on collecte les données au format PCAP, on peut les traiter au fur et à mesure en décidant d'une taille maximale à partir de laquelle on termine d'enregistrer les données dans un fichier avant de passer au fichier suivant. Une fois le fichier PCAP représentant notre jeu de données (qu'il ait été récupéré manuellement ou qu'il provienne d'un jeu de données existant), il y a néanmoins quelques étapes à effectuer afin d'obtenir un fichier utilisable par notre modèle. Pour qu'un fichier soit compatible avec notre modèle, il doit pouvoir être lu par la librairie Pandas [57]. Pandas accepte en entrée des formats de données classiques, comme le format Comma-Separated Values (CSV), où chaque donnée occupe une ligne, et les valeurs des attributs pour une même donnée sont séparées par des virgules.

Dans notre cas, les jeux de données sélectionnés (voir Section 2.1.1) que l'on souhaite utiliser au format PCAP sont labellisés. La problématique est donc la suivante : comment récupérer les labels au format CSV ? Généralement, les attaques sont datées. Les auteurs des jeux de données indiquent donc dans un fichier les dates de début et fin de chacune des attaques. Un script bash a été développé afin de :

- Isoler dans des fichiers séparés chaque heure de début, heure de fin, label et adresses IP source et destination
- Appliquer un filtre pour ne garder que le protocole et les adresses IP sources et destination qui sont concernées par les attaques (obtenues à l'étape 1). On élimine alors tout le trafic bénin et autres bruits.
- Créer un répertoire pour chacun des labels présents dans le jeu de données
- Extraire la portion PCAP qui correspond à chaque attaque (grâce aux fichiers créés à l'étape 1) et la placer dans le répertoire adéquat (une attaque avec le label XSS est donc placée dans le répertoire XSS)
- Fusionner les paquets PCAP présents dans chacun des répertoires afin de n'avoir qu'un fichier PCAP par type d'attaque
- Exporter chacun des fichiers PCAP dans les différents répertoires au format CSV

La deuxième étape permet d'éviter des temps de calcul trop longs par la suite. Elle est réalisée avec tshark [62], qui permet de filtrer des paquets au format PCAP en ligne de commande. La dernière étape (extraction) est réalisée avec editcap [63] qui permet notamment d'extraire

des paquets en fonction de leur date. C'est à cette étape que l'on peut isoler chacune des attaques en les plaçant dans les répertoires adéquats grâce à leur datation. L'étape de fusion des paquets s'effectue avec mergecap [64]. La dernière étape est parfois la plus problématique, car le module d'exportation au format CSV de tshark et wireshark ne produisent pas toujours les bons résultats. L'exportation au format JSON, plus complet, peut alors être employée. Un code python a été développé afin de convertir ces données JSON au format CSV. Le format JSON comporte un volume important de données et ce code permet de sélectionner seulement celles qui sont pertinentes pour notre cas d'usage. Ainsi, le code python va parcourir chacun des répertoires, importer le fichier CSV correspondant à un type d'attaque, y ajouter le label correspondant pour chacune des entrées et l'enregistrer à nouveau au format CSV. Les différents fichiers CSV sont par la suite concaténés pour n'obtenir plus qu'un seul jeu de données avec ses labels, qui peut alors être utilisé par le module de prétraitement (les labels sont mis de côté jusqu'à l'exécution du module d'évaluation). Il faut noter que les adresses IP qui sont utilisées pour appliquer un filtre à la deuxième étape ne servent qu'à écarter le trafic qui ne comporte pas d'attaques. Elles ne sont pas retenues dans le fichier CSV final.

CHAPITRE 4 VALIDATION EXPÉRIMENTALE

L'objectif de ce chapitre est de justifier la méthode de validation retenue dans un premier temps, puis de présenter les données, les résultats et leur analyse pour chacun des trois jeux de données retenus (CIC-IDS2017, UNSW-NB15 et CTU-13). Une discussion générale sur la phase de validation et les résultats obtenus sera également proposée pour conclure ce chapitre.

4.1 Méthodologie de la validation

Il convient de prime abord d'expliquer en quoi consiste la validation de notre modèle. Notre modèle fonctionne sur des données non supervisées ; il s'agit de l'utiliser sur des données labellisées, en les traitant comme des données non labellisées, afin de procéder à l'évaluation du modèle. En effet, lorsque les données vont être réparties dans différents clusters, il est possible d'étudier le partitionnement obtenu en fonction des labels. Il est pertinent d'utiliser, comme premier facteur de validation, les métriques traditionnelles de classification (precision, exactitude, rappel et f1-score), afin de mesurer la qualité de notre classification. Bien qu'on ne s'attende pas à obtenir des scores semblables aux méthodes de classification supervisées, l'étude de la répartition des labels dans notre partitionnement va permettre également de définir si la classification de notre modèle est pertinente. Dans le cas où l'on ne parviendrait absolument pas à donner du sens à la classification obtenue (comme le ferait, dans un cas extrême, un classificateur aléatoire attribuant aléatoirement un cluster à chaque donnée), le modèle aurait été invalidé. Aussi, un dernier facteur de validation sera le nombre idéal de clusters proposé par notre modèle : celui-ci doit se rapprocher le plus possible du nombre d'attaques contenues dans le jeu de données. Les scores obtenus, l'interprétation de la répartition des labels dans les clusters et le nombre idéal de clusters proposé par notre modèle constituent donc nos trois facteurs de validation.

La justification du choix des jeux de données sélectionnés a été faite à la Section 2.1.1. Pour valider notre modèle qui fonctionne protocole par protocole, il convient de sélectionner un sous-ensemble de chacun des jeux de données. Ici, il a été choisi d'utiliser uniquement les données web (relatives au protocole HTTP) pour valider notre modèle. Pourtant, les données de notre partenaire industriel contiennent d'autres protocoles et on applique d'ailleurs notre modèle au protocole SSH. Il aurait donc été pertinent de valider également notre modèle sur des données SSH. Néanmoins, il est actuellement, à notre connaissance, impossible d'obtenir des jeux de données labellisées relatives au protocole SSH compatible avec notre modèle (c'est-à-dire où la charge utile des paquets est disponible). En effet, le protocole SSH est un

protocole chiffré. Les données pertinentes à notre classification contenues dans chaque paquet ne sont donc pas directement exploitables. Le chiffrement des données étant asymétrique, il faudrait donc avoir accès aux clés de déchiffrement pour déchiffrer les données relatives à SSH et accéder au contenu des paquets. Cependant, ce secret partagé est seulement calculé en local par le client et par le serveur ; le code d'un serveur SSH devrait donc être modifié afin d'avoir accès au secret partagé. Comme cette fonctionnalité est compliquée à mettre en place et ne serait pertinente que dans très peu de cas, elle n'est pas répandue sur les serveurs SSH. Ceci a pour conséquence qu'aucun outil d'analyse de trafic réseau, à notre connaissance actuelle, ne propose de déchiffrer du trafic SSH lorsque l'on dispose des clés de chiffrement. Pour Wireshark, qui est l'analyseur de paquet le plus populaire, le travail pour supporter cette fonctionnalité est, à l'heure d'écriture de ce mémoire, en cours [65]. Un patch proposant cette fonctionnalité devrait bientôt voir le jour, mais il est encore trop tôt pour que les auteurs de jeux de données puissent l'utiliser. C'est pourquoi il n'a pas été possible de valider notre modèle sur des données malveillantes relatives au protocole SSH.

Pour la validation de notre modèle, l'hypothèse suivante est donc émise : la validation de notre modèle sur le protocole HTTP permet également de valider l'utilisation de notre modèle sur le protocole SSH. Il convient alors de justifier pourquoi cette hypothèse a été jugée raisonnable. La partie la plus importante de notre modèle est le prétraitement des données avec l'analyse de similarité. Le protocole SSH a un attribut représentant la charge utile du paquet, de façon très similaire à HTTP. Cet attribut est d'ailleurs bien plus présent pour SSH que HTTP (puisque pour HTTP, il n'est présent que pour les requêtes de type POST). Pour SSH, l'attribut relatif à la charge utile est celui qui contient la plus grande partie des informations relatives à un paquet SSH. Notre modèle se basant sur l'inspection de charge utile de paquet, il est donc tout à fait raisonnable de l'utiliser également sur des paquets SSH. De façon générale, tout protocole dont la charge utile peut faire l'objet d'une analyse de similarité peut être utilisé par notre modèle. Des protocoles comme DNP3 qui se basent sur des échanges de code devant être interprétés par la suite devraient faire l'objet d'une validation supplémentaire afin d'être utilisés par notre modèle.

4.2 CIC-IDS2017

4.2.1 Présentation des données sélectionnées

CIC-IDS2017 a été présenté dans sa globalité dans la Section 2.1.2. Dans cette section, il avait notamment été évoqué que CIC-IDS2017 est un des jeux de données de cyberattaques les plus utilisés actuellement. Aussi, ce jeu de données est très déséquilibré : certaines attaques sont

représentées par bien plus de données que d'autres. Ce jeu de données contient différentes attaques qui ciblent différents protocoles. CIC-IDS2017 contient 2 830 540 entrées et 14 attaques différentes. Le protocole contenant le plus d'attaques est HTTP et c'est pourquoi nous allons nous concentrer sur ce protocole. Quatre attaques concernent HTTP :

— Des attaques par force brute : tentatives de connexion en testant différentes combinaisons d'identifiants et de mots de passe. Ces attaques seront abrégées par la suite `Bruteforce`

— Des attaques de scripting intersite (Cross-Site Scripting) : injection de contenu dans une page web entraînant une action de celle-ci. Ces attaques seront abrégées par la suite `XSS`

— Des injections SQL : tentative d'exploitation d'une base de données SQL par injection de requêtes SQL malveillantes. Ces attaques seront abrégées par la suite `SQLi`

— Des attaques de botnet : logiciel malveillant essayant d'exploiter des vulnérabilités de façon automatique. Ces attaques seront abrégées par la suite `Botnet`

Parmi les 2 830 540 entrées de CIC-IDS2017, 2 359 087 correspondent à du trafic bénin. Sur les 471 453 entrées malveillantes restantes, 130 855 concernent les quatre attaques relatives au protocole HTTP. La répartition des 130 855 entrées entre les différentes attaques se trouve à la Table 4.1

Label	**Nombre d'entrées**
`Botnet`	$118,765$
`XSS`	$5,505$
`Bruteforce`	$6,548$
`SQLi`	37

Tableau 4.1 Nombre d'entrées pour chaque attaque pour CICIDS-2017

Bien que l'on sache déjà que le jeu de données est globalement déséquilibré comme indiqué dans la section 2.1.2, on peut remarquer que ce déséquilibre est aussi valable pour le protocole HTTP. En effet, les attaques de type `Botnet` représentent 90.76% du jeu de données, alors que les attaques de type `SQLi` représentent moins de 0.03% du jeu de données. Il est très pertinent de tester notre algorithme sur un jeu de données déséquilibré, car cette situation est représentative d'une situation réelle.

Nous utilisons donc un sous-semble de CIC-IDS2017, mais par abus de langage, le sous-ensemble créé qui ne concerne que les données HTTP de CIC-IDS2017 sera tout de même nommé CIC-IDS2017. Pour créer ce sous-ensemble, nous avons suivi la méthode permettant d'extraire des données d'une capture réseau au format PCAP et de les rendre compatibles avec notre modèle. Cette méthode est décrite à la Section 3.4.

4.2.2 Résultats

Le choix des attributs concernant HTTP a été justifié dans la Section 3.3.1 qui concerne la méthodologie (`Payload`, `URL`, `User-Agent` et `Method`). Dans cette même section, il a été expliqué que le nombre de valeurs uniques pour chaque attribut ne doit pas être trop élevé pour que notre modèle fonctionne correctement. Pour CIC-IDS2017, le nombre de valeurs uniques par attribut avant et après le prétraitement est donné à la Table 4.5. On peut remarquer que le prétraitement n'est effectué que sur les attributs `Payload` et `URL`. En effet, il n'y a que trois valeurs uniques différentes pour l'attribut `User-Agent`, ce qui est assez rare dans un jeu de données. Il n'y a donc pas de raison de chercher à uniformiser l'information procurée par cet attribut en cherchant des liens sémantiques entre les trois valeurs différentes. Pour `Payload` et `URL`, on remarque que le nombre de valeurs uniques a considérablement chuté grâce au prétraitement, ce qui est l'objectif recherché. En ce qui concerne `Method`, le même raisonnement que pour `User-Agent` s'applique. Mais ceci est moins surprenant pour l'attribut `Method`, puisqu'il ne peut par définition pas prendre plus que 9 valeurs différentes (bien que dans certains cas assez rares, on puisse avoir des méthodes qui n'existent pas qui sont employées par des attaquants).

Attribut	**NVU avant prétraitement**	**NVU après prétraitement**
`Payload`	$7,086$	27
`URL`	$5,548$	39
`User-Agent`	3	3
`Method`	2	2

Tableau 4.2 Nombre de valeurs uniques (NVU) avant et après le prétraitement des données pour CIC-IDS2017

Les hyperparamètres du modèle (voir Section 3.3.1 pour plus de détails) sont $récu_{min}$ pour la fréquence d'apparition minimale d'un mot à respecter pour apparaître dans les données prétraitées et $taux\_similarité_{min}$ le taux de similarité minimum à partir duquel on considère deux valeurs comme étant égales. Ces hyperparamètres ont été fixés aux valeurs suivantes : $récu_{min} = 1$ et $taux\_similarité_{min} = 0.5$. Cela signifie qu'un mot doit apparaître deux fois ou plus pour ne pas être supprimé d'un `Payload` ou d'une `URL` et que deux `Payload` ou deux `URL` sont considérés comme identiques par notre modèle s'ils sont similaires à plus de 50%.

Avec ces valeurs, on calcule les différentes métriques présentées à la section 3.3.5 pour trouver le nombre idéal de clusters. Les résultats sont illustrés à la Figure 4.1.

Ici, le choix de nombre de clusters est plutôt trivial. En effet, la méthode du coude présente clairement un coude à quatre clusters, le score de Davies-Bouldin est minimal pour quatre

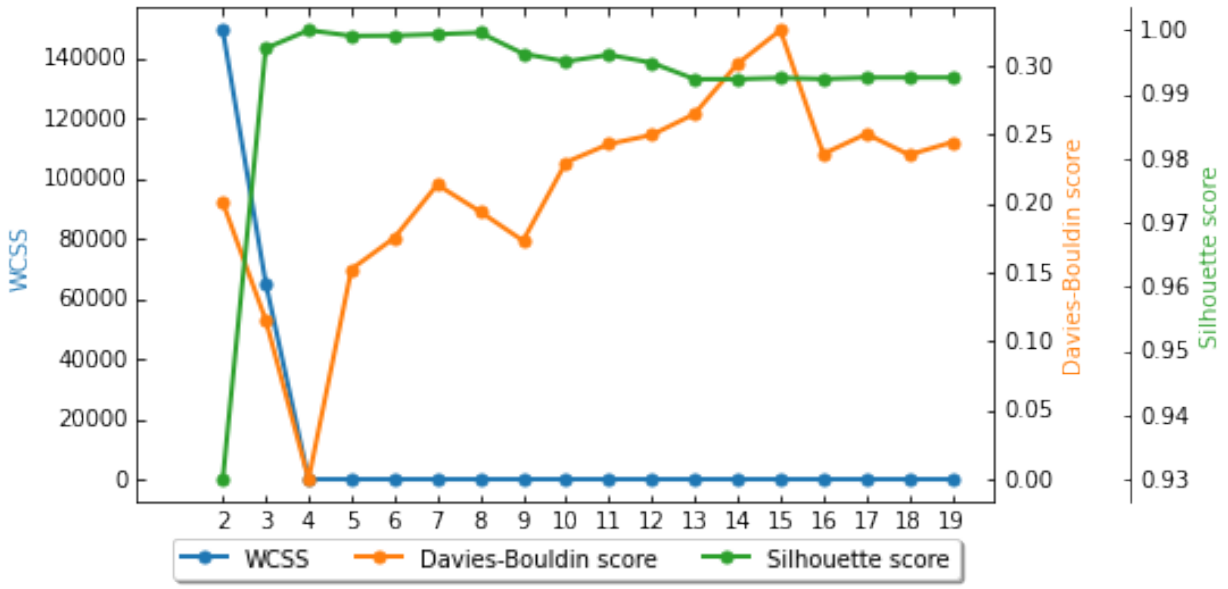


FIGURE 4.1 Résultats des trois différents scores pour CIC-IDS2017.

clusters et le score Silhouette est maximal pour quatre clusters. On choisit donc d'effectuer notre partitionnement avec quatre clusters. Ce résultat est très encourageant puisqu'il y a quatre attaques différentes, on peut donc espérer que chacune des attaques a été isolée dans un cluster différent.

On effectue alors notre partitionnement avec quatre clusters. Une fois ce partitionnement effectué, on peut observer le nombre de chaque label obtenu dans chaque cluster. Aussi, on assigne à chaque cluster un label en utilisant le module d'interprétation. Chacune des quatre attaques étant suffisamment simple et caractéristique, l'inspection manuelle n'est pas laborieuse. Par exemple, dans le fichier regroupant chaque `URL` par cluster, on observe que toutes les `URL` contiennent au moins les valeurs « document » et « cookie ». On reconnaît le « document.cookie » utilisé par les attaquants pour voler des cookies d'authentification via une faille XSS. L'assignation d'un label à chaque cluster est réalisée avant l'obtention du nombre de chaque label dans chaque cluster afin de ne pas être biaisé dans notre interprétation. En effet, on pourrait être tenté de simplement assigner le label qui est le plus présent dans chaque cluster. Mais notre objectif ici est de valider nos résultats, on ne peut donc pas utiliser les labels pour effectuer notre classification. Ces derniers doivent être uniquement utilisés dans le but d'obtenir des scores permettant d'évaluer notre classification une fois celle-ci terminée. La présence de chaque label dans chaque cluster ainsi que le label assigné à chaque cluster sont montrés à la Figure 4.3.

On peut alors remarquer qu'on a parfaitement réussi à isoler chacune des attaques dans des

Label/Cluster	1	2	3	4
Botnet	118,765	0	0	0
XSS	0	5,505	0	0
Bruteforce	0	0	6,548	0
SQLi	0	0	0	37
Label assigné	Botnet	XSS	Bruteforce	SQLi

Tableau 4.3 Nombre de labels dans chaque cluster pour les données de CIC-IDS2017

clusters différents. On obtient donc un F1-score de 100%, peu importe la façon dont on le calcule.

4.2.3 Discussion

On peut conclure que la classification s'est parfaitement déroulée pour les données de CIC-IDS2017, avec un F1-Score de 100%. Plusieurs éléments peuvent justifier ce score. Premièrement, les attaques de CIC-IDS2017 sont assez simples et n'ont qu'une seule étape. Par exemple, pour les attaques `XSS` chaque entrée correspond à une tentative de XSS, pour les `SQLi`, chaque entrée correspond à une tentative d'injection SQL, etc. Notre algorithme fonctionne très bien sur ce type de donnée, puisque chacun des attributs va être une parfaite caractéristique de chacune des attaques. Aussi, les attaques contenues dans CIC-IDS2017 ont été effectuées sur DVWA [66], une application vulnérable par définition dédiée à l'entraînement. Ainsi, chaque attaque s'effectue sur des URL caractéristiques : les `SQLi` sur *« DVWA/vulnerabilities/sqli... »*, les `XSS` sur *« DVWA/vulnerabilities/xss... »*, etc. Ceci donne une légère indication à notre modèle (quelques caractères sur plusieurs dizaines de caractères) et rend chaque URL un peu plus caractéristique d'une attaque. Aussi, seules les attaques `Botnet` et `Bruteforce` possèdent des attributs de type `Payload`. Elles seront donc aisément distinguées des deux autres attaques qui n'en n'ont pas, non seulement grâce à la présence ou non de cet attribut, mais aussi grâce à l'attribut `Method` qui prend la valeur *POST* si l'attribut `Payload` est présent et *GET* sinon. Il est par la suite assez aisé de distinguer une attaque de type `Botnet` d'une attaque de type `Bruteforce`, car le contenu de leur `Payload` est très caractéristique. En effet, les tentatives de connexion par force brute contiennent toutes un « username = » et un « password = », contrairement au *Payload* des attaques de botnet.

4.3 UNSW-NB15

4.3.1 Présentation des données

De même que pour CIC-IDS2017, UNSW-NB15 a été présenté dans sa globalité dans l'état de l'art à la Section 2.1.3. La même méthode a aussi été utilisée pour extraire uniquement les données concernant le protocole HTTP et pour labelliser ces données. Ce jeu de données contient 2 540 044 entrées, dont 2 218 761 qui correspondent à du trafic bénin. 321 283 correspondent alors à du trafic malveillant. Parmi ces données, seulement 3 832 correspondent à des attaques qui concernent HTTP uniquement. D'autres attaques peuvent utiliser HTTP, mais ne sont pas principalement constituées de données relatives au protocole HTTP. Ce jeu de données est aussi très déséquilibré, comme le montre la répartition des treize attaques à la Figure 4.4. Ces attaques sont bien plus complexes que les attaques contenues dans CIC-IDS2017. Une présentation détaillée de ces attaques a été faite par Moustafa et al. [14].

Label (+ abréviation)	**Nombre d'entrées**
Domino Web Server Database Access (`Dom`)	1,857
Fuzzer (`Fuz`)	955
PHP5 php_register_variable_ex Buffer Overflow (`BO`)	404
Cold Fusion File Access (`CFFA`)	275
Microsoft FrontPage(`MFP`)	106
HTTP Shell Access(`SA`)	77
UNIX HTTP Server File Access (`USFA`)	50
Apache System User Directory Access (`ASUDA`)	38
Apache File Access (`AFA`)	24
Sensitive File Access (`SFA`)	19
Microsoft IIS(`IIS`)	13
BGP Update (`BGP`)	11
Apache 2.2.14 mod_isapi Dangling Pointer DoS (`AmDP`)	3

Tableau 4.4 Nombre d'entrées pour chaque attaque pour UNSW-NB15

De la même façon qu'avec CICIDS-2017, UNSW-NB15 désigne le sous-ensemble de UNSW-NB15 avec lequel nous avons choisi de travailler et nous avons obtenu ces résultats avec la méthode exposée à la Section 3

4.3.2 Résultats

La Table 4.5 montre les différents nombres de valeurs uniques (NVU) obtenues pour chacun des attributs avant et après le prétraitement. Cette fois, les hyperparamètres utilisés sont les suivants : $récu_{min} = 3$ et $taux\_similarité_{min} = 0.5$.

Attribut	NVU avant prétraitement	NVU après prétraitement
`Payload`	84	59
`URL`	2,120	152
`User-Agent`	196	34
`Method`	4	4

TABLEAU 4.5 Nombre de valeurs uniques (NVU) avant et après le prétraitement des données pour CIC-IDS2017

Comme expliqué pour CIC-IDS2017, on n'applique pas de prétraitement à l'attribut `Method`.

Avec ces valeurs, on calcule les différentes métriques présentées à la Section 3.3.5 pour trouver le nombre idéal de clusters. Les résultats sont illustrés à la Figure 4.2

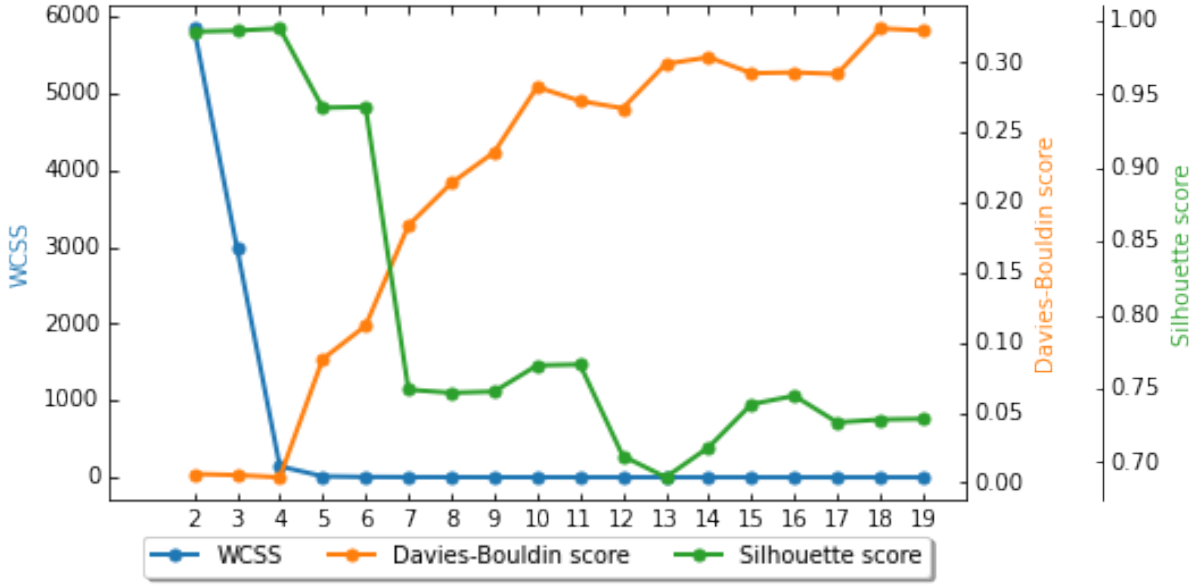


FIGURE 4.2 Résultats des trois différents scores pour UNSW-NB15.

Cette fois encore, le nombre idéal de clusters est explicite. En effet, le score silhouette est extrêmement proche de son maximum à quatre clusters, le score de Davies-Bouldin est minimum à quatre clusters et on observe un coude qui se forme distinctement à quatre clusters pour le score WCSS. Cependant, contrairement à CIC-IDS2017, quatre clusters n'est pas le nombre idéal de clusters attendu. En effet, on aurait idéalement souhaité obtenir treize clusters puisqu'il y a treize clusters différents. Évidemment, on ne peut pas utiliser treize comme nombre idéal de clusters, puisque notre modèle est un modèle non supervisé et que nous ne sommes pas censés utiliser ce chiffre pour notre classification. D'ailleurs, les résultats présentés ci-après ne sont pas meilleurs lorsqu'on impose treize clusters à notre algorithme.

De la même façon qu'avec CIC-IDS2017, la présence de chaque label dans chaque cluster ainsi que le label assigné après inspection manuelle des clusters sont donnés à la Figure 4.10.

Label/Cluster	**1**	**2**	**3**	**4**
Dom	1,558	0	0	299
Fuz	0	836	0	119
BO	0	88	316	0
CFFA	189	0	0	86
MFP	0	0	0	106
SA	0	0	0	77
USFA	50	0	0	0
ASUDA	0	0	0	38
AFA	24	0	0	0
SFA	19	0	0	0
IIS	0	13	0	0
BGP	0	0	0	11
AmDP	0	0	0	3
Label assigné	**Dom**	**Fuz**	**BO**	**SA**

TABLEAU 4.6 Nombre de labels dans chaque cluster pour les données de UNSW-NB15

La librairie Scikit-Learn [47] est utilisée pour calculer les métriques classiques dans le cas d'une classification multiclasses. Les différences entre les métriques utilisées sont expliquées à la Section 3.3.5. Les métriques permettant d'évaluer les résultats obtenus sont données à la Figure 4.7.

	Précision	**Rappel**	**Score F1**
`Micro-moyennage`	0.73	0.73	0.73
`Macro-moyennage`	0.20	0.25	0.20
`Moyennage pondéré`	0.74	0.73	0.73

TABLEAU 4.7 Métriques obtenues pour UNSW-NB15

4.3.3 Discussion

On remarque que chaque attaque n'a pas été correctement isolée dans un cluster (ce qui n'était de toute façon pas possible avec seulement quatre clusters). Les scores obtenus par micro-moyennage et moyennage pondéré sont presque égaux, alors qu'ils sont calculés d'une façon différente. Ces deux scores sont bien plus élevés qu'avec le macro-moyennage. Ceci s'explique par le fait que la majorité des attaques `Dom`, `Fuz` et `BO`, qui sont les trois techniques avec le

plus d'entrées, ont correctement été isolées dans des clusters séparés. Ainsi, comme le micro-moyennage et le moyennage pondéré prennent en compte le nombre d'entrées dans chaque cluster, les scores obtenus sont plus élevés qu'avec le macro-moyennage qui ne fait que la moyenne de chaque score F1 obtenu. Or, dans notre cas, pour tous les autres labels que `Dom`, `Fuz`, `BO` et `SA`, on a un score F1 de 0, ce qui fait considérablement chuter la moyenne du score F1 final.

Enfin, bien que chaque attaque ne soit pas isolée dans des clusters différents, on peut tout de même remarquer certaines tendances au sein de chaque cluster. En effet, le cluster 1 regroupe, en plus des nombreux `Domino Web Server Database Access`, majoritairement des attaques qui concernent l'accès à des fichiers : `Cold Fusion File Access`, `UNIX HTTP Server File Access`, `Apache File Access` et `Sensitive File Access`. De plus, mis à part `Cold Fusion File Access`, ces attaques ont été totalement isolées dans ce même cluster 1 (elles ne sont présentes dans d'autres clusters). Ce premier cluster peut donc correspondre au cluster des tentatives d'accès malveillants. Le cluster 2 regroupe presque toutes les attaques de type `Fuzzer` et quelques attaques de type `Buffer Overflow`. Le fait que ces deux types d'attaques soient mélangés au sein d'un cluster reste cohérent. En effet, en inspectant le contenu des paquets qui sont regroupés ensemble, on remarque que les payloads de ces deux attaques sont très similaires ; ce sont de grandes séquences de caractères (principalement des « A ») qui sont utilisées pour des tentatives d'attaques par débordement de tampon, mais aussi par les Fuzzers. Le reste des attaques de type `BufferOverflow` ont correctement été isolées dans le cluster 3. Le cluster 4 regroupe quant à lui plusieurs types d'entrées qui n'ont pas été placées dans les clusters précédents. Ces entrées n'ont pas de caractéristiques suffisamment originales pour être séparées dans des clusters, ce qui explique leur regroupement.

Finalement, notre modèle a regroupé de façon globalement pertinente les entrées du jeu de données UNSW-NB15. Bien que certaines attaques qui ne se démarquent pas par des caractéristiques particulières n'aient pas pu être isolées dans des clusters différents, on obtient tout de même un F1 score (en micro-moyennage ou en moyennage pondéré) de 73%, ce qui reste tout à fait acceptable pour une classification non supervisée.

4.4 CTU-13

4.4.1 Présentation des données

De même que pour les deux jeux de données étudiés précédemment, CTU-13 a été présenté dans sa globalité dans l'état de l'art au Chapitre 2. Il ne contient que du trafic de botnet (en plus du trafic bénin). Treize scénarios différents ont été capturés, avec différentes exécutions

de malwares sur différents protocoles. Dans notre étude de cas, nous nous concentrons sur les botnets utilisant HTTP. Certains botnets utilisent différents protocoles pour réaliser leur attaque, y compris le protocole HTTP. Seuls trois botnets réalisent des attaques purement HTTP. Nous ne sélectionnons un botnet que lorsque la quantité de données HTTP est jugée suffisante pour appliquer notre modèle. Il suffit donc qu'un botnet utilise suffisamment le protocole HTTP lors de son attaque pour qu'il soit retenu dans notre étude, même s'il utilise d'autres protocoles. Par conséquent, nous nous basons sur six attaques de botnet différentes : Neris (`Ne`), Neris-2 (`Ne2`), Qvod (`Qv`), Sogou (`So`), Fast-flux (`FF`), Fast-flux-2 (`FF2`). Les noms de botnet suivis d'un -2 indiquent que le même botnet a réalisé différents scénarios d'attaque. Le tableau 4.8 montre comment les données sont réparties entre les différents scénarios d'attaque. Il convient de préciser à nouveau le fait que CTU-13 est un jeu de données totalement différent de CIC-IDS2017 et de UNSW-NB15. En effet, ce jeu de données contient des attaques bien plus complexes qui comprennent plusieurs étapes.

Label	**Nombre d'entrées**
`FF2`	2,484
`Ne2`	1,675
`Ne`	1,288
`Qv`	1,077
`FF`	110
`So`	84

Tableau 4.8 Nombre d'entrées pour chaque attaque pour CTU-13

4.4.2 Résultats

De même que pour les deux jeux de données précédents, la Table 4.9 montre le nombre de valeurs uniques (NVU) avant et après prétraitement. Ici, les hyperparamètres utilisés sont les suivants : $récu_{min} = 1$ et $taux\_similarité_{min} = 0.5$.

Attribut	**NVU avant prétraitement**	**NVU après prétraitement**
`Payload`	56	12
`URL`	2579	168
`User-Agent`	72	14
`Method`	2	2

Tableau 4.9 Nombre de valeurs uniques (NVU) avant et après le prétraitement des données pour CTU-13

Avec ces données, nous obtenons les scores visibles à la Figure 4.3.

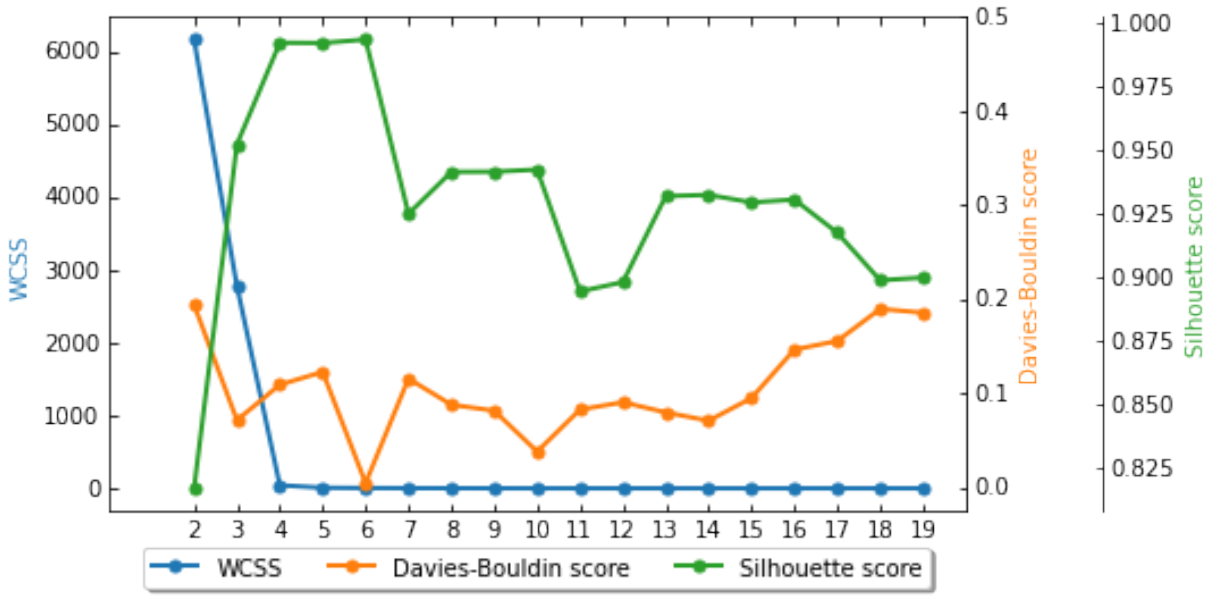


FIGURE 4.3 Résultats des trois différents scores pour CTU-13.

Dans ce cas, le choix du nombre de clusters est moins évident. La méthode du coude (score WCSS) indique clairement que le nombre idéal de clusters est quatre. Néanmoins, un minimum très prononcé est visible pour le score de Davies-Bouldin à six clusters. Le score silhouette est presque égal pour quatre et six clusters, mais tout de même légèrement supérieur pour six clusters. De façon générale, dans ce genre de situation, il est plutôt conseillé de prendre le nombre de clusters le plus élevé. Il sera alors possible par la suite de remarquer si certains clusters auraient pu être regroupés ensemble (ce qui s'avère être plus compliqué à réaliser dans l'autre sens).

Label/Cluster	**1**	**2**	**3**	**4**	**5**	**6**
FF2	2,463	10	0	1	10	0
Ne2	1,661	0	0	2	0	12
Ne	928	0	0	360	0	0
Qv	40	0	1,037	0	0	0
FF	76	22	0	0	12	0
So	84	0	0	0	0	0
Label assigné	**FF2**	**FF**	**Qv**	**Ne**	**FF**	**Ne2**

TABLEAU 4.10 Nombre de labels dans chaque cluster pour les données de CTU-13

De même que pour UNSW-NB15, on obtient avec ces valeurs les métriques présentées dans le Tableau 4.11.

	Précision	**Rappel**	**Score F1**
`Micro-moyennage`	0.58	0.58	0.58
`Macro-moyennage`	0.59	0.35	0.34
`Moyennage pondéré`	0.78	0.58	0.48

Tableau 4.11 Métriques obtenues pour CTU-13

4.4.3 Discussion

Encore une fois, chaque attaque n'a pas correctement été isolée dans des clusters, bien que cette fois, le nombre idéal de clusters suggéré par notre modèle corresponde au nombre d'attaques différentes présentes dans le jeu de données. On remarque aussi dans le Tableau 4.11 que les scores obtenus sont plutôt faibles. Ceci s'explique par plusieurs facteurs. Premièrement, il est clair que CTU-13 est le jeu de données qui contient les attaques les plus complexes parmi les trois jeux de données étudiés. Ces attaques ont la particularité d'être multi-étapes. Bien que ce soit également le cas d'UNSW-NB15, les attaques de CTU-13 diffèrent tout de même, car elles sont réalisées par des botnets qui sont des réelles cybermenaces, alors que dans UNSW-NB15, les attaques sont simulées par un algorithme et sont donc plus simples que dans CTU-13. Il faut également souligner que les attaques contenues dans CTU-13 sont multi-protocoles ; on analyse donc uniquement une partie de l'attaque, qui est celle qui concerne HTTP. Aussi, il est certain que chacun de ces botnets ont des similarités et partagent des phases communes dans leurs attaques. En pratique, il n'a pas été possible de mettre en évidence ces phases partagées dans nos clusters, car le nombre de données et la complexité des attaques sont trop importants pour une analyse manuelle. D'ailleurs, le fonctionnement détaillé de chacun des botnets n'est pas connu et il aurait été très laborieux d'analyser chacun de ces botnets pour notre étude. Néanmoins, on peut remarquer certaines tendances qui se dégagent au sein de notre partitionnement.

Par exemple, on remarque que bien que le premier cluster contienne des entrées pour chaque botnet, il isole particulièrement trois d'entre eux. En effet, il contient toutes les attaques relatives à Sogou, presque toutes (99.15%) les attaques relatives à Fast-Flux-2 et presque toutes les attaques relatives à Neris-2 (98.69%). Il est donc très probable que le comportement de ces botnets soit très similaire, et il est intéressant que notre modèle ait réussi à les isoler presque totalement dans un cluster, bien qu'ils soient mélangés entre eux. Les autres botnets sont bien moins présents dans ce cluster. Le cluster 3 donne aussi des résultats encourageants, puisqu'il parvient à isoler 96.29% des entrées relatives à Qvod, sans avoir aucune autre entrée. En plus du fait que les attaques contenues dans CTU-13 soient multi-étapes, issues de vrais botnets et multi-protocoles, il a été relevé dans notre analyse que certains de ces botnets

utilisent très certainement des User-Agent aléatoires. En effet, cette pratique est commune pour un cyberattaquant, qui va choisir aléatoirement un user-Agent parmi une liste de User-Agents existants réellement, afin de passer pour un utilisateur légitime (puisqu'un User-Agent complètement aléatoire, qui n'existerait pas, serait suspect). En analysant les valeurs prises par les valeurs de User-Agent grâce à notre module d'interprétation, on remarque que certains User-Agent sont ceux d'une Playstation ou encore d'Itunes. Il est donc clair que ces User-Agents ont été obtenus en étant tirés aléatoirement dans une liste de User-Agents existants, puisqu'il ne semble pas raisonnable qu'un attaquant puisse effectuer des attaques depuis une Playstation. Ceci fausse donc un des attributs de notre modèle, ce qui représente une limite à notre méthode.

On remarque, grâce à cette expérimentation, qu'il serait pertinent d'utiliser des techniques de corrélation afin de détecter des scénarios d'attaque ayant plusieurs étapes qui sont classifiées dans plusieurs clusters. Ce point sera discuté et développé dans la Section 6.3, qui traite des améliorations futures de notre modèle.

4.5 Discussion croisée et validation

Finalement, on a obtenu 100% de précision pour CIC-IDS2017, 73% pour UNSW-NB15 (pour le micro-moyennage et le moyennage pondéré, le macro-moyennage n'est pas pertinent puisqu'il ne prend pas en compte la taille des clusters), et 58% et 78% pour CTU-13 (respectivement pour le micro-moyennage et le moyennage pondéré). Ces scores sont très encourageants pour de la classification non supervisée. De plus, lorsque ces derniers sont bas, il est possible de justifier la composition des clusters et de donner tout de même un sens à chaque cluster, ce qui est également positif pour notre modèle. Enfin, le nombre idéal de clusters est égal au nombre d'attaques contenues pour CIC-IDS2017 et CTU-13 (bien que pour CTU-13, il semble que cela puisse être une coïncidence), mais pas pour UNSW-NB15. Les performances de notre modèle sont donc tout à fait correctes, puisque nos trois critères de validation (scores, interprétabilité des clusters et nombre idéal de clusters). Le modèle peut donc maintenant être utilisé sur les données de notre partenaire industriel.

CHAPITRE 5 APPLICATION DU MODÈLE AUX DONNÉES DE NOTRE PARTENAIRE INDUSTRIEL

L'objectif de ce chapitre est d'utiliser le modèle précédemment présenté à la Section 3 et validé à la Section 4 sur les données de notre partenaire industriel. Il s'agit de prime abord de présenter l'ensemble de ce jeu de données. Par la suite, il convient d'étudier plus en détail les données pour chacun des protocoles sélectionnés (HTTP et SSH), avant de présenter les résultats ainsi qu'une discussion autour de ceux-ci.

5.1 Présentation des données et choix des protocoles utilisés

Pour des raisons de confidentialité, cette section ne peut pas rentrer dans certains détails, qui sont donc volontairement omis. Les données fournies concernent de nombreux protocoles et il a fallu retenir parmi ceux-ci les plus intéressants pour notre approche. Initialement, il était prévu d'utiliser un protocole des Technologies de l'Information (I.T) et un protocole sur les Technologies Opérationnelles (O.T). Mais plusieurs facteurs nous ont finalement poussés à choisir deux protocoles I.T que sont HTTP et SSH.

Présentation de l'agent d'apprentissage par renforcement

Premièrement, le pot de miel dispose, pour certains protocoles, d'une technologie d'apprentissage par renforcement qui permet d'augmenter les interactions entre l'attaquant et le pot de mot de miel et d'améliorer l'aspect réaliste et la crédibilité de ce dernier. Cette technologie permet d'offrir des données bien plus riches, puisque plus l'interaction entre un pot de miel et un attaquant est forte, plus les données sont intéressantes à classifier. On préfère en effet classifier les interactions d'un humain exerçant une cyberattaque complexe que celle d'un robot (botnet) qui ne s'adapte pas au système qu'il attaque et qui n'effectue qu'une séquence de requêtes prédéfinies.

Cependant, l'existence de cette technologie ainsi que la liste des protocoles qui en disposaient ont été communiquées après un premier choix des protocoles retenus pour appliquer notre modèle. Une fois ces informations communiquées, nous avons préféré nous orienter vers SSH, disposant de la technologie d'apprentissage par renforcement, plutôt que vers un protocole O.T sans cette technologie initialement choisi.

Nombre de données et interprétabilité

Aussi, le nombre de données disponibles a été un facteur déterminant dans notre choix. En effet, certains protocoles auraient pu être un choix intéressant, comme MODBUS qui est un protocole O.T et qui dispose de la technologie d'apprentissage par renforcement. Mais l'index relatif à MODBUS ne contenait pas suffisamment de données pour être utilisé. L'interprétabilité du protocole a aussi été un facteur de sélection : DNP3 est un protocole O.T (qui ne dispose pas de la technologie d'apprentissage par renforcement) et qui aurait pu être sélectionné à la place d'HTTP. Mais DNP3 est un protocole qui échange des codes dont l'interprétation est libre et dépend de chaque utilisation. Il n'a donc pas été possible de trouver l'interprétation de ces codes, ce qui aurait empêché l'interprétation des résultats.

À l'inverse, HTTP est probablement le protocole le plus aisé à interpréter, puisque c'est un protocole très utilisé. SSH est un vecteur d'attaque légèrement moins courant, mais le pot de miel contient beaucoup de données relatives à ce protocole, pour lequel la technologie d'apprentissage par renforcement a été implémentée. C'est pourquoi nous avons choisi HTTP et SSH. Il est d'ailleurs intéressant de comparer les résultats de ces deux protocoles afin de déterminer dans quelle mesure l'agent d'apprentissage par renforcement influence la qualité et la nature des données journalisées.

Les données HTTP sont au nombre de 179 768, ce qui n'a posé aucun problème pour notre algorithme. À l'inverse, les données SSH, déchiffrées par le pot de miel, sont au nombre de 34 529 257, ce qui s'est révélé être problématique. Néanmoins, après une analyse approfondie des données, il est apparu que beaucoup de données ont été dupliquées. Aussi, certaines données contiennent uniquement des connexions sans aucune donnée. Ces connexions représentent uniquement un scan de port afin de déterminer si la connexion est acceptée et si l'hôte est en fonction (up). Après avoir appliqué un algorithme permettant de détecter et supprimer les doublons, et de supprimer les valeurs ne comportant aucune information (elles étaient trop nombreuses pour les conserver), les données SSH sont au nombre de 1 369 692. Cependant, ce nombre reste trop important pour avoir été analysé complètement par notre modèle. Il a donc été décidé de traiter un échantillon de plus petite taille (100 000 entrées), sélectionnées au hasard dans les 1 369 692 données. Voici pourquoi ce choix a été fait :

- Il est plus aisé, pour interpréter les résultats, d'utiliser un échantillon
- Notre modèle nécessite, pour la première fois, plusieurs exécutions afin de fixer les hyperparamètres
- L'exécution de notre modèle sur 100 000 entrées prend environ 2.5 jours (cela dépend des hyperparamètres)
- Des études [67] [59] montrent qu'il est possible d'obtenir des précisions très proches

(lors de l'assignation de nouveaux labels) en utilisant un sous-ensemble de données avec K-Means, comparé à lorsque l'on utilise l'ensemble complet de données

5.2 Application du modèle aux données relatives au protocole HTTP

5.2.1 Application du modèle

Pour rappel, les quatre attributs choisis pour HTTP sont `payload`, `URL`, `user-agent` et `method`.

La Table 5.1 montre le nombre de valeurs uniques (NVU) avant et après prétraitement. Ici, les hyperparamètres utilisés sont les suivants : $récu_{min} = 1$ et $taux\_similarité_{min} = 0.5$.

Attribut	NVU avant prétraitement	NVU après prétraitement
`Payload`	66671	139
`URL`	37279	1519
`User-Agent`	1284	169
`Method`	52	52

Tableau 5.1 Nombre de valeurs uniques (NVU) avant et après le prétraitement pour les données HTTP

Il est intéressant de commenter les valeurs de la Table 5.1. On remarque que le nombre de valeurs uniques pour le payload est initialement presque deux fois supérieur au nombre de valeurs uniques pour l'URL, alors qu'après le prétraitement, il est plus de dix fois inférieur. Cela signifie que notre algorithme a trouvé plus de similarités entre les payloads qu'entre les URLs. Ceci est tout à fait cohérent, puisque les payloads contiennent généralement des valeurs très similaires. Par exemple, une tentative d'attaque par force brute va générer des payloads qui sont presque identiques. Aussi, cela se justifie par la présence d'une attaque qui a été testée un nombre important de fois, avec des payloads identiques, mais des URLs différentes (voir la Section 5.2.2).

Il faut également noter que le prétraitement n'a pas été appliqué à l'attribut method. En effet, celui-ci ne contient que très peu de valeurs uniques. Cependant, bien qu'il soit faible, le NVU de l'attribut method est tout même surprenant puisqu'il n'existe normalement que 9 méthodes différentes pour HTTP, alors qu'ici, nous en avons 52. Certains frameworks (structures logicielles) proposent d'autres verbes pour HTTP. Nessus [68] propose d'ailleurs un scan [69] des extensions de méthodes HTTP les plus populaires, qui correspondent parfaitement aux méthodes que l'on a dans nos données. Il est donc clair qu'un attaquant a utilisé ce scan, ce qui a produit un NVU anormalement élevé pour l'attribut method.

Nous obtenons alors les métriques présentées à la Figure 5.1.

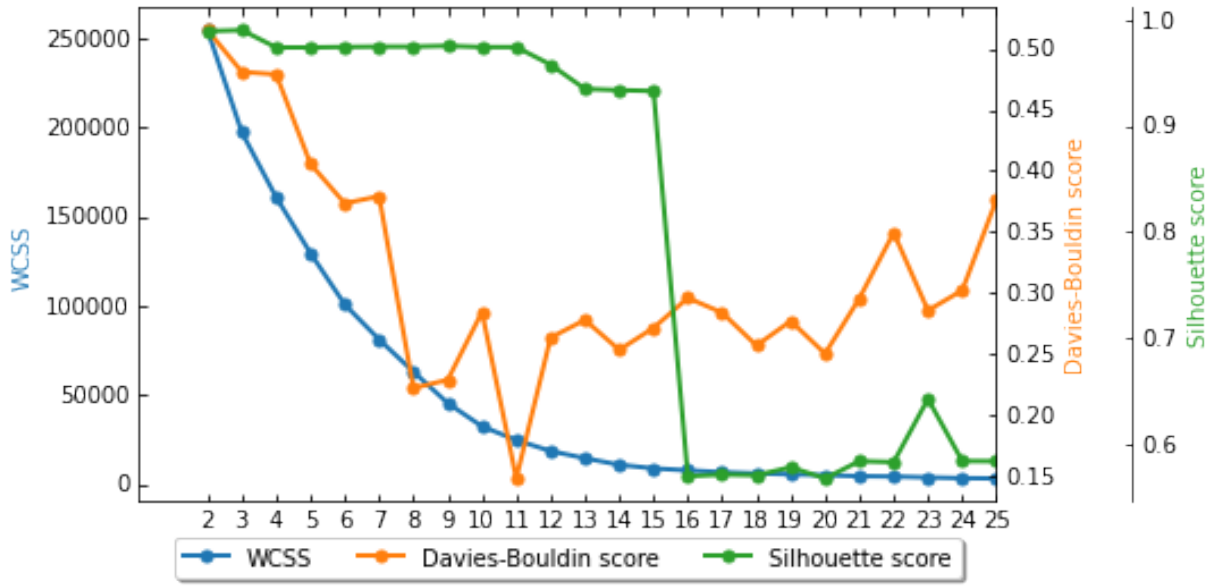


FIGURE 5.1 Résultats des trois différents scores pour les données HTTP.

Ici, on remarque que le score Silhouette est élevé pour des nombres de clusters allant de 2 à 11 (inclus), puis il diminue légèrement, avant de diminuer considérablement à partir de 16 clusters. Le score Silhouette nous informe donc que le nombre idéal de clusters se situe entre 2 et 11. Le score de Davies-Bouldin nous donne plus d'informations, car il présente clairement un minimum à 11 clusters. Le score WCSS ne présente pas particulièrement un coude, il est donc compliqué à utiliser dans ce cas. Finalement, il est pertinent de choisir 11 comme nombre idéal de clusters, puisque c'est à ce nombre que le score de Davies-Bouldin atteint son minimum et que le score Silhouette est très proche de son maximum.

On rappelle que la seule métrique qui est directement interprétable est le score Silhouette, puisque ce dernier est borné. Ainsi, pour un point donné, un score de -1 représente la pire classification et un score de 1 la meilleure. Un score de 0 indique que le point est parfaitement entre les différents clusters. Ici, le score silhouette est de 0.9748, ce qui est encourageant pour la qualité de notre classification.

5.2.2 Analyse du partitionnement

Utilisation du module d'interprétation

Pour cette analyse, le module d'interprétation présenté à la Section 3 est utilisé. Premièrement, il est intéressant d'observer la répartition du nombre de données par clusters, qui est

donnée à la Table 5.2.

Numéro du cluster	Nombre de données
0	178619
1	1
2	96
3	14
4	1
5	6
6	439
7	215
8	276
9	21
10	80

Tableau 5.2 Répartition des données HTTP dans les différents clusters

On remarque que le cluster 0 contient la majorité des données (99.36%). Il est maintenant intéressant de comprendre pourquoi beaucoup de données ont été regroupées au sein du même cluster.

Après avoir appliqué notre modèle, il est possible d'observer la visualisation en trois dimensions du résultat dans les Figures 5.2 et 5.3.

Avec la visualisation 3D, on remarque que les clusters 1, 4, 5 et, dans une moindre mesure, 3, se distinguent particulièrement d'une plus grosse concentration de points représentée à la Figure 5.3. Aussi, il s'agit des clusters qui contiennent le plus petit volume de données : respectivement 1, 1, 6 et 14.

Le module d'interprétation du modèle, en plus d'une visualisation 3D, permet de regrouper les valeurs des attributs les plus fréquentes de chaque cluster, ce qui facilite l'interprétation.

Nature des données

Lors de l'interprétation des résultats, une discussion avec notre partenaire industriel a soulevé un point important : le pot de miel pour les données HTTP n'est constitué que d'un portail de connexion qui invite à saisir un identifiant et un mot de passe, sans possibilité réelle de se connecter. Les interactions avec ce portail sont donc extrêmement limitées, voire nulles. Après analyse des résultats, on peut constater que l'intégralité du trafic est issue d'outils automatisés. Il ne sera donc pas possible de classifier des cyberattaques dans ces données, puisqu'il n'y en a pas. Seuls des passages de botnets ou d'outils de test de vulnérabilités qui envoient les mêmes requêtes indépendamment du système attaqué sont présents dans ces

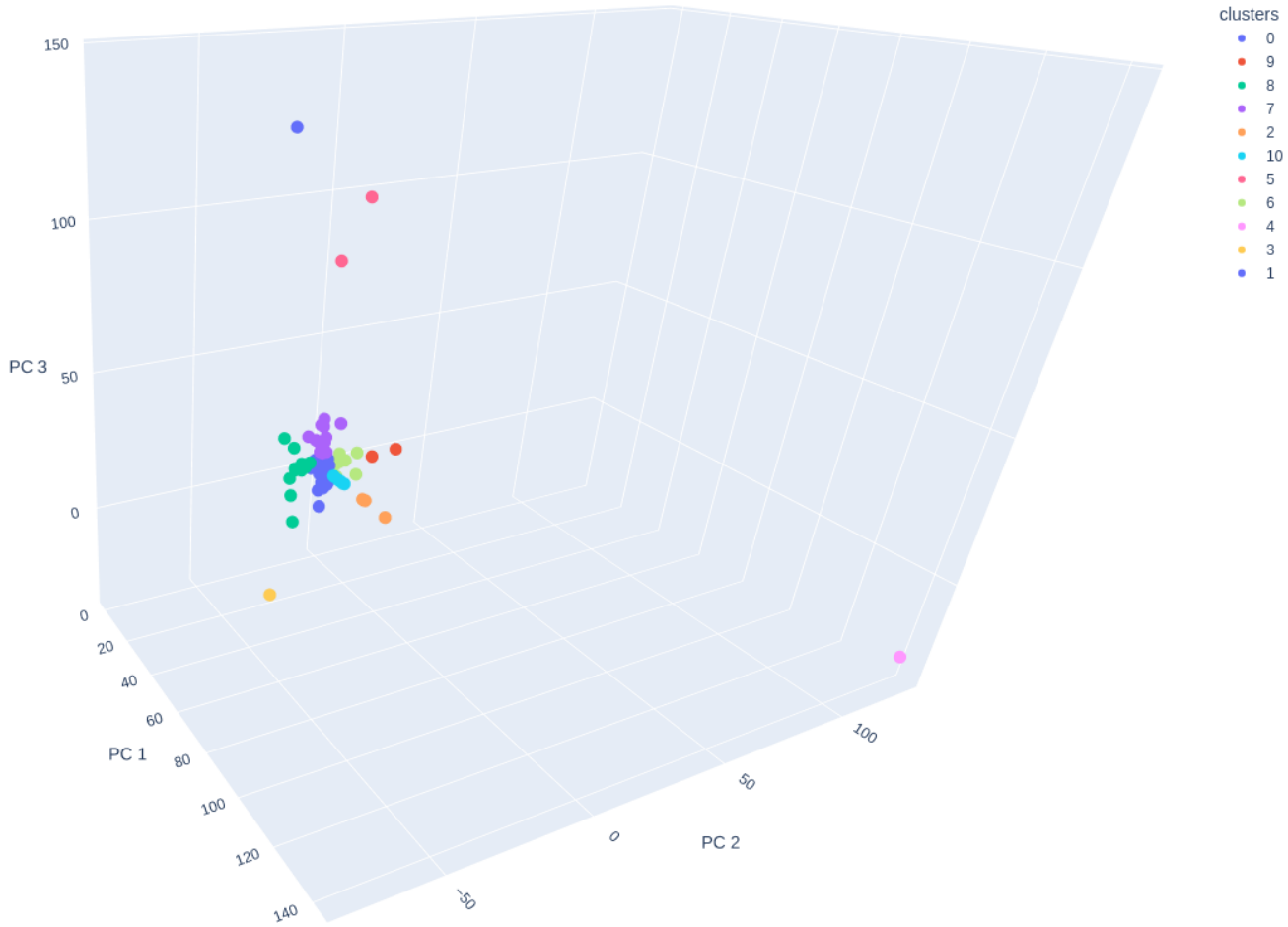


FIGURE 5.2 Visualisation en trois dimensions du partitionnement pour les données HTTP.

données. Ces scans sont utilisés pour préparer une attaque, mais ne constituent pas une réelle cyberattaque à eux seuls.

Interprétation des clusters

Il reste tout de même intéressant d'analyser les résultats, pour comprendre pourquoi certaines données ont été isolées du cluster principal (cluster 0 contenant 99.36% des données) et pourquoi d'autres ont été séparées. Dans le cluster principal (cluster 0), un tiers des données n'ont ni payload, ni URL et correspondent probablement à des tests afin de savoir si l'hôte est disponible et accepte une connexion sur le port 80. Sinon, pour les deux autres tiers, beaucoup

FIGURE 5.3 Zoom de la visualisation en trois dimensions du partitionnement pour les données HTTP.

de vulnérabilités sont testées, et certaines, de nombreuses fois. Par exemple, le payload le plus populaire est constitué des mots clés « admin », « die », « md5 », « S3pt3mb3r », revient 20 383 fois, et est présent uniquement dans ce cluster. Ceci correspond à une attaque sur le logiciel propriétaire vBulletin [70], qui souffrait de la vulnérabilité CVE-2019-16759 [71], qui concernait, le jour de sa découverte, plus de 100 000 sites web. Cette vulnérabilité permet, pour un utilisateur non authentifié, de prendre le contrôle du serveur assez rapidement, c'est pourquoi elle a beaucoup été utilisée par des outils de test automatisé. Il ne serait pas pertinent d'exposer chacune des vulnérabilités testées dans ce cluster, puisqu'il y en a des dizaines. L'exemple précédent permet de montrer qu'une même vulnérabilité peut

avoir beaucoup d'entrées dans un même cluster. Aussi, cet exemple a été choisi car c'est un bon exemple de ce que les attaquants cherchent à tester. Afin d'avoir une description plus macroscopique du cluster principal, il est intéressant d'analyser les user-agents. En effet, ceux-ci peuvent parfois directement donner l'identité du logiciel ou du botnet qui est utilisé. Dans d'autres cas, des user-agents aléatoires sont choisis, c'est pourquoi on ne peut pas uniquement se baser sur les user-agents pour interpréter nos clusters. Néanmoins, voici les différents user-agents et leurs catégories pour le cluster 0 :

- Des exploreurs qui testent aveuglément des vulnérabilités : ZmEu [72] ou l9explore [73]
- Des scanners de ports : CensysInspect [74], ProjectDiscovery [75], Masscan [76], Nessus, MercuryBoard [77], zGrab [78]
- Des scripts « fait maison » : Curl [79], Python Request [80], OKhttp [81], PyCurl [82], UrlLib [83], GoHttp [84], jndi/ldap [85], AIOhttp [86]
- Des robots d'indexation utilisés par des moteurs de recherche : Google Bot, Baidu Spider

Les clusters qui semblent très éloignés des autres (1, 3, 4 et 5) ont un point commun : chacun de ces clusters ont des user-agents uniques et distincts des autres clusters. Pour expliquer le placement des clusters 1 et 5, qui sont plutôt proches, il faut rapidement évoquer un point concernant notre modèle. Lorsque la valeur d'un attribut dépasse une certaine taille définie (ici, 100 000), notre modèle lui assigne alors un UUID [87] fixé. Ceci permet d'éviter à notre algorithme de traiter des données trop importantes, qui auraient pour but d'atteindre la disponibilité du système attaqué (un user-agent de plusieurs giga octets ne peut avoir que cet objectif). Le fait de choisir un UUID permet de caractériser cette tentative, puisque cette valeur ne se retrouvera pas ailleurs dans les données. Afin de fixer le seuil, les valeurs des attributs ont été classées par taille décroissante. On observe alors une diminution considérable de la taille après quelques valeurs. La dernière valeur avant cette forte diminution représente notre seuil. Cette étape peut être automatisée assez aisément, en calculant les dérivées des différents points afin de détecter cette forte variation. Ainsi, toutes les tentatives de saturation avec de longues valeurs d'attributs auront le même UUID. Dans les données HTTP, il n'y a eu que 7 valeurs concernées, qui ont été réparties dans les clusters 1 et 5. Ces clusters, bien qu'ils soient proches, sont tout de même différents, car ici, seuls les user-agents se sont vu attribuer un UUID, les données diffèrent donc tout de même par leurs url, payload et method.

Le cluster 4 a pour user-agent « ApiTool ». Cet outil teste un exploit très particulier qui porte sur l'IoT, plus précisément sur des caméras souffrant d'une vulnérabilité. Les valeurs des différents attributs n'ont rien en commun avec les autres données, ce qui explique pourquoi le cluster 4 est très éloigné des autres clusters.

Le cluster 3, qui est moins éloigné du cluster 0 que les clusters 1, 4 et 5 a pour user-agent « XTC Botnet » et cette valeur n'est présente que dans ce cluster. Peu d'informations fiables sont disponibles sur ce botnet, seul un article de blog [88] nous donne quelques informations. Il est en tout cas intéressant qu'un botnet ait été isolé dans un cluster.

Les clusters 2, 6, 7, 8, 9 et 10 sont plus proches les uns des autres et sont centrés autour du cluster principal (0). Ces clusters contiennent aussi des vulnérabilités testées aléatoirement. Si elles sont isolées dans ces clusters, c'est parce que les valeurs des attributs utilisées par ces vulnérabilités diffèrent de celles du cluster 0. Elles ont donc peu de valeurs communes avec le cluster 0, mais ne sont pas non plus suffisamment différentes pour en être éloignées (comme pour les clusters 1, 3, 4 et 5). Chacun de ces clusters isole une vulnérabilité différente, ce qui est encourageant. Voici, pour chacun de ces clusters, la vulnérabilité qui a été testée d'après notre interprétation, ainsi qu'une justification en fonction de la valeur des attributs. On préfère justifier avec l'URL qui est toujours présente et généralement plus caractéristique d'une vulnérabilité.

Cluster 2 :

Vulnérabilité exploitée : CVE-2018-10561 [89].

Justification : URL : « GponForm », « diag », « Form », « images »

Cluster 6 :

Vulnérabilité exploitée : Pas de CVE associée, mais exploitation des routeurs Netlink 1.0.11 [90].

Justification : URL : « boaform », « admin », « formLogin », « username », « user », « psd », « user »

Cluster 7 :

Vulnérabilité exploitée : CWE-912 [91] : tentative d'exécution d'une invite de commande à distance.

Justification : URL : « shell », « cd », « tmp », « rm », « rf », « wget », « ip_masqué_pour_confidentialité », « jaws », « sh », « tmp », « jaws ».

Ici, l'attaquant essaye de télécharger une porte dérobée, se déplace dans le répertoire « tmp », lui donne la permission d'être exécuté et l'exécute. Cela ne marche que si un shell est installé sur le serveur web, on a donc probablement affaire à un botnet. D'ailleurs, le user-agent qui est ici « Hello World » a déjà été recensé comme étant utilisé par des Botnets, notamment Mirai [92].

Cluster 8 :

Vulnérabilité exploitée : Identique au cluster 6 : Pas de CVE associée, mais exploitation des routeurs Netlink 1.0.11. Les clusters 6 et 8 diffèrent finalement simplement par leurs user-agents, qui ont probablement été choisis aléatoirement, ce qui a induit en erreur notre modèle.

Justification : URL : (« boaform », « admin », « formLogin », « username », « user », « psd », « user »)

Cluster 9 :

Vulnérabilité exploitée : CVE-2017-9248 [93].

Justification : URL : « DesktopModules », « Admin », « RadEditorProvider », « DialogHandler », « aspx ». Le scan utilisé [94] est disponible sur github (l'URL utilisé est identique).

Cluster 10 :

Vulnérabilité exploitée : CVE-2020-28036 [95] Justification : URL : « xmlrpc », « php », « rsd »

5.2.3 Discussion

Finalement, il est clair que le fait que les données contiennent uniquement du trafic issu d'outils automatisés a compliqué la classification. En effet, il est compliqué de donner un sens à ces données qui correspondent à des vulnérabilités testées aveuglément. Néanmoins, des points intéressants ont été dégagés. Les données trop volumineuses ont été isolées dans deux clusters différents. 7 clusters réussissent à isoler des vulnérabilités, dont un qui est très éloigné des autres clusters car la vulnérabilité exploitée est très caractéristique et n'a aucun attribut en commun avec les autres. Un cluster principal regroupe néanmoins la majorité des données, avec un tiers de ce cluster qui ne correspond qu'à des tests afin de déterminer si l'hôte accepte des connexions. Il serait très pertinent de mêler ces données à des attaques réalisées par des humains, afin de voir si les comportements issus d'un humain sont isolés de ceux issus d'un robot. Aussi, une limite de notre modèle a été atteinte : le fait que les user-agents puissent être aléatoirement définis peut induire notre modèle en erreur.

5.3 Application du modèle aux données relatives au protocole SSH

5.3.1 Application du modèle

Les quatre attributs choisis pour SSH sont `eventid`, `dst_port`, `duration` et `message`. Comme ils n'ont pas encore été présentés dans ce mémoire, voici un descriptif de ces attributs.

— Eventid : attribut permettant de catégoriser la nature de la connexion SSH. On peut savoir si l'entrée représente une tentative d'authentification, le téléchargement d'un fichier, ou encore l'exécution d'une commande
— Dst_port : représente le port de destination. En effet, il a été identifié que beaucoup de ports différents peuvent être utilisés pour SSH.
— Duration : durée de la connexion.
— Message : charge utile.

On peut noter que pour SSH, on a un attribut numérique qui est « duration ». Nous allons donc utiliser AFDM plutôt qu'ACM.

La Table 5.3 montre le nombre de valeurs uniques (NVU) avant et après prétraitement. Ici, les hyperparamètres utilisés sont les suivants : $récu_{min} = 2$ et $taux\_similarité_{min} = 0.5$.

Attribut	**NVU avant prétraitement**	**NVU après prétraitement**
`eventid`	18	18
`dst_port`	24	24
`message`	15981	1505

Tableau 5.3 Nombre de valeurs uniques (NVU) avant et après le prétraitement pour les données SSH.

L'attribut « duration » n'est pas concerné par le NVU puisque c'est une donnée numérique. On remarque que, pour les mêmes raisons évoquées à la Section 5.2.2, « eventid » et « dst_port » ne sont pas concernés par le prétraitement (trop peu de valeurs uniques initialement). Seul l'attribut « message » a donc subi un prétraitement.

Nous obtenons alors les métriques présentées à la Figure 5.4. Ici, le choix est plutôt évident, puisque le score Silhouette présente un maximum là où le score de Davies-Bouldin présente un minimum, à savoir pour 7 clusters. Seul la méthode du coude pourrait contredire ce résultat, car on observe une cassure à 5 clusters. Mais cette méthode est moins fiable puisqu'elle se base sur une interprétation graphique, il est donc préférable de se fier aux deux autres métriques. Pour les mêmes raisons qu'à la Section 5.2.2, le score Silhouette permet d'évaluer la qualité du partitionnement. Ici, pour 7 clusters, celui-ci vaut 0.9560, ce qui est très encourageant.

5.3.2 Analyse du partitionnement

Utilisation du module d'interprétation

Pour cette analyse, le module d'interprétation présenté à la Section 3 est utilisé. Premièrement, il est intéressant d'observer la répartition du nombre de données par clusters, qui est

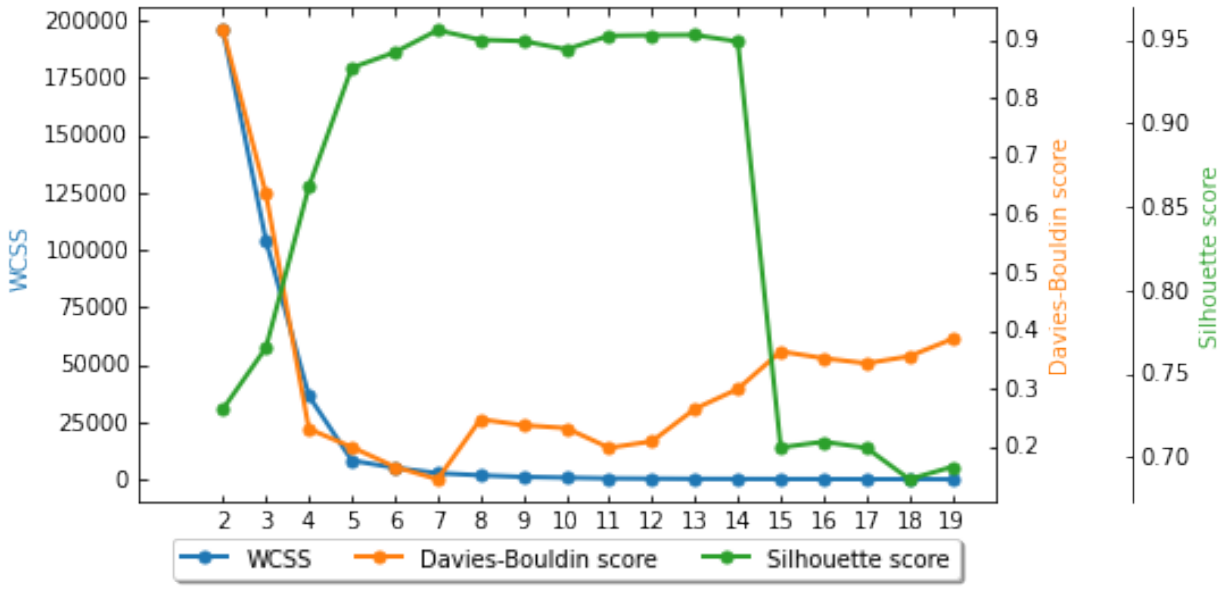


FIGURE 5.4 Résultats des trois différents scores pour les données SSH.

donnée à la Table 5.4.

Numéro du cluster	Nombre de données
0	67700
1	7241
2	5227
3	1169
4	9457
5	7117
6	2089

TABLEAU 5.4 Répartition des données SSH dans les différents clusters

Premièrement, on peut remarquer que les données sont mieux réparties qu'avec HTTP : le plus gros cluster représente 67.7% du jeu de données (contre 99.36% pour HTTP) et le plus petit cluster comporte 1 169 données (contre 1 pour HTTP). Après avoir appliqué notre modèle, il est possible d'observer la visualisation en trois dimensions du résultat dans la Figure 5.5.

On remarque que la représentation en trois dimensions est très différente comparée à HTTP. Ceci est dû au fait que SSH possède un attribut numérique (« duration ») qui va « étaler » les différents points selon certains axes. Il est important de noter que l'attribut représentant le port de destination n'est pas considéré comme un nombre mais comme une donnée

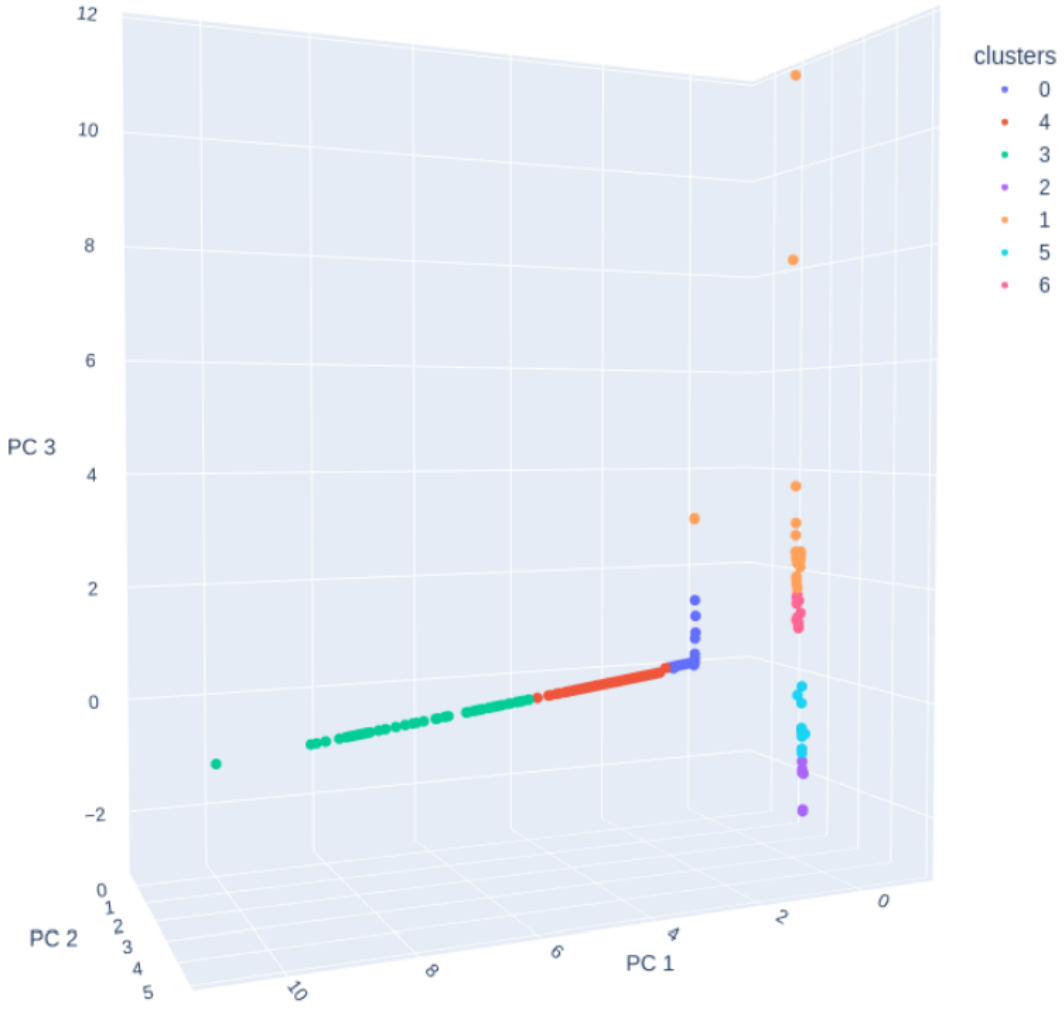


FIGURE 5.5 Visualisation en trois dimensions du partitionnement pour les données SSH.

catégorielle. En effet, ce n'est pas parce que les numéros de deux ports sont proches que leur signification est proche (exemple : 80 et 8080 désignent un port utilisé par le web, pourtant 80 est plus proche de 22 qui est utilisé pour SSH).

Il faut également noter l'apparition d'un cluster principal, le cluster 0. La formation de ce cluster est une conséquence de notre prétraitement. En effet, les données appartenant à ce cluster ont des caractéristiques très similaires (détaillées dans le paragraphe suivant) : les « eventid » sont souvent les mêmes, les « duration » sont très proches, les ports sont identiques et les messages assez similaires puisque débutant par « CMD », ce qui représente une commande. Notre prétraitement et notre module de représentation de données vont donc regrouper ces valeurs, ce qui entraîne la formation du cluster 0. Il s'agit maintenant d'analyser la composition de chaque cluster.

Interprétation des clusters

Cluster 0 :

Ce cluster est probablement le plus important. Il contient toutes les entrées contenant une commande SSH (mais pas uniquement). Ces dernières sont précédées par CMD, il est donc aisé de les identifier. Les commandes sont variées et ont différents buts. Elles ont été correctement regroupées entre elles, puisqu'il n'y a pas deux commandes qui se ressemblent et qui auraient dû être considérées comme étant égales par notre algorithme de prétraitement. Certaines commandes correspondent à une tentative d'authentification : 4 814 connexions ont été un succès (« eventid »=« cowrie.login.success ») et 4 076 connexions ont été un échec (« eventid »=« cowrie.login.failed »). Lorsque l'agent d'apprentissage par renforcement du pot de miel intervient, le journal du pot de miel reçoit une entrée supplémentaire : ceci est traduit par 6 950 « eventid » ayant la valeur « api_bird.action » et 6 950 « message » composés des mots-clés « RL », « Action » et « allow », qui sont présents uniquement dans le cluster 0. On a donc bien eu des interactions de l'agent d'apprentissage par renforcement, mais il est compliqué de connaître ces réactions. On peut tout de même observer les commandes des attaquants ayant réussi à se connecter. La commande la plus courante correspond à « uname -a » (2 433 entrées) qui permet d'obtenir des informations sur la machine à laquelle nous sommes connectés en SSH, comme la version de son OS, son nom, le type de distribution, etc. Beaucoup de commandes ont pour but d'identifier le processeur, c'est notamment le cas des commandes contenant les mots-clés « cat », « proc », « cpuinfo », « grep », « name », « head », « n », « 1 », « awk », « print », « 4 », « 5 », « 6 », « 7 », « 8 », « 9 » (2 431 entrées) ainsi que la commande contenant les mots-clés « lscpu », « grep », « Model » (2237 entrées). Il est possible qu'en obtenant des informations sur la distribution et le processeur, un attaquant comprenne qu'il soit sur un pot de miel. Une commande qui est aussi très récurrente comporte les mots-clés « root », « chpasswd » et « bash » (1 338 entrées), qui sont une tentative de changement du mot de passe root, qui permet aussi de savoir si l'on dispose des privilèges du superutilisateur. 582 commandes ont également pour objectif de rajouter une clé RSA afin de faciliter une reconnexion ultérieure. Les autres commandes constituent principalement des tentatives de téléversement (upload) d'une porte dérobée. Ceci peut se faire de différentes façons, c'est pourquoi ces commandes ne sont pas considérées comme étant similaires par l'algorithme de prétraitement. Ces commandes sont caractérisées par des séquences de mots-clés enchaînant des changements de répertoire et suppression de leur contenu (« cd » et « rm »), des téléchargements à des adresses IP (« wget » ou « curl » + IP), ainsi que des tentatives de changement des permissions pour le fichier téléchargé (« chmod » + « 777 » ou une autre valeur). Il y a de grandes chances pour que ces tentatives

soient issues d'un réel cyberattaquant et non d'un robot, au vu du contenu des commandes. On peut aussi remarquer que le logiciel malveillant dota3 [96] a essayé d'être téléchargé puis décompressé pour être par la suite utilisé, comme l'indique le message constitué des mots-clés « tar », « dota3 » et « gz ». On peut remarquer que 14 fichiers ont été téléversés avec succès (« eventid »=« cowrie.upload.success »), ce qui représente des potentielles portes dérobées. Néanmoins, il est probable que ceci soit le résultat d'une action de l'agent d'apprentissage par renforcement, qui décide ou non d'autoriser le téléversement de fichier.

Cluster 1 et 6 :

L'ensemble des données contenues dans ces clusters suit le même schéma. Toutes ont pour « eventid » « cowrie.direct-tcpip.data » et une « duration » de 0 seconde. Pour le port, la plupart des ports utilisés sont des ports utilisés pour le web (443 occupe presque toutes les valeurs). Les messages ont tous la même structure : ils commencent par les mots-clés « discarded », « direct », « tcp », « forward », « request », « to », puis une URL, puis par un enchaînement de données hexadécimales différentes à chaque fois. En fait, il existe dans openSSH [97] une directive permettant d'activer ou non le TCP forwarding (transfert TCP). Ainsi, un serveur SSH pourrait être utilisé comme un proxy afin de rediriger le trafic vers la victime. Heureusement, le pot de miel désactive cette fonctionnalité (c'est pourquoi la durée de la session SSH est nulle à chaque fois), mais l'attaquant peut tout de même croire que ses requêtes sont redirigées. Ce trafic est donc très probablement issu d'outils automatisés.

Le cluster 6 constitue la version non chiffrée du cluster 1. En effet, la seule différence avec le cluster 1 est l'utilisation du port 80 (relatif à HTTP) à la place de 443 (relatif à HTTPS). Ceci s'illustre d'ailleurs par des tailles de cluster très similaires entre le cluster 1 et 6 (respectivement 7 241 et 7 117). Il y a seulement quelques messages qui n'ont pas été transmis en clair. Ce cluster est donc utile pour accéder au contenu des requêtes du cluster 1. On imagine que l'attaquant a utilisé un outil automatisé permettant d'essayer de transférer des requêtes HTTP en utilisant SSH et en essayant les ports 80 et 443.

Cluster 2 et 5 :

Les clusters 2 et 5 ont quelques points en commun avec le cluster 1. Les durées de connexion sont également nulles, mais l'eventid vaut maintenant « cowrie.direct-tcpip.request ». Maintenant, au lieu de transférer des données en hexadécimal, l'attaquant demande simplement au serveur de se connecter à l'adresse qu'il souhaite. Les messages sont tous sous la forme des mots-clés suivants : « direct », « tcp », « connection », « request », « to », puis une URL, puis « 80 », « from », « IP ». Les ports utilisés correspondent presque intégralement au port 80. Ici, on n'essaie pas de transférer une requête, mais plutôt de demander directement au serveur de se connecter à une URL en utilisant le port 80.

Le cluster 5 est exactement identique en remplaçant le numéro de port 80 par 443. Il correspond à la version chiffrée du cluster 2.

Cluster 3 :

Le cluster 3 correspond aux messages de fin de session SSH. En effet, l'eventid vaut « cowrie.session.closed ». La particularité de ce cluster est le fait que toutes les « duration » valent approximativement 180 secondes. Ce cluster correspond donc aux sessions qui ont expiré (la durée de vie maximale d'une session serait donc fixée à 180 secondes). Les messages confirment cette hypothèse puisqu'ils sont sous la forme « Connection », « lost », « after », puis un nombre, « seconds ».

Cluster 4 :

Ce cluster est presque identique au cluster 3, mais à une différence près : les durées de session sont toutes inférieures à 180 secondes. Il correspond donc à toutes les sessions qui se sont terminées avant la fin de la durée de vie maximale de la session SSH.

5.3.3 Discussion

On remarque que les clusters 3 et 4 se situent sur le même axe et sont différenciés surtout par la durée de la session. De même pour les clusters 1 et 6, et 2 et 5, qui se différencient par leurs différents messages et ports (c'est pourquoi la répartition est moins linéaire qu'entre les clusters 3 et 4 qui se différencient avec des données numériques).

Globalement, les résultats de la classification pour les données SSH sont très satisfaisants. En effet, on parvient à isoler différents comportements d'attaquants dans des clusters séparés. Avec le module d'interprétation, il a également été possible de donner un sens à chacun des clusters formés, ce qui constituait les deux objectifs principaux de cette classification. Les commandes ont été correctement regroupées par notre algorithme de prétraitement, ce qui est encourageant. Néanmoins, notre algorithme reste en deçà des performances attendues en termes de vitesse d'exécution, ce qui nous a obligés à utiliser un sous-ensemble du jeu de données. Aussi, il aurait été intéressant que notre modèle sépare le cluster 0 en plusieurs clusters, en fonction des commandes. Il est possible, dans un travail futur, d'effectuer une seconde phase de partitionnement uniquement dans le cluster 0.

CHAPITRE 6 CONCLUSION

L'objectif de ce chapitre est de proposer une conclusion des travaux effectués. Pour ce faire, une synthèse des travaux effectués sera donnée, avant de mettre en avant les limitations de la solution élaborée. Enfin, des pistes d'améliorations futures seront proposées.

6.1 Synthèse des travaux

L'objectif de cette recherche était de construire un modèle de classification de données malveillantes répondant aux exigences suivantes :

1. Compatibilité avec le format non standard des pots de miel
2. Compatibilité avec les formats standards d'autres jeux de données
3. Classification pertinente, mesurable et interprétable des données
4. Classification permettant de détecter des attaques non uniformes dans le temps
5. Classification permettant de détecter des vulnérabilités de type 0-day
6. Classification permettant de détecter des scénarios complexes de cyberattaques
7. Classification dans des temps de calcul raisonnables, afin d'être compatible avec des jeux de données volumineux
8. Validation sur d'autres jeux de données

Les objectifs 1, 4, 5 et 8 ont été atteints par le modèle proposé. Les objectifs 2 et 3 ont été atteints, mais pas totalement. Les objectifs 6 et 7 n'ont pas été directement atteints, mais notre modèle a été développé afin qu'il puisse être complété pour remplir ces deux objectifs (voir la Section 6.3).

En effet, notre modèle fonctionne avec le format non standard des données de pots de miel (il a d'ailleurs été développé sur-mesure pour de telles données). Il est possible d'utiliser notre modèle sur des formats de données standards, mais il faut que la capture réseau soit disponible, ou que les attributs des différents protocoles soient journalisés (c'est pour cela que l'objectif n'est pas complètement atteint, il existe certains jeux de données standards ne remplissant pas ces conditions).

En ce qui concerne l'objectif 3, il a été jugé comme étant partiellement atteint car notre classification ne permet pas de proposer directement un label pour chacun des clusters, il faut une analyse afin de donner du sens à chaque cluster. Néanmoins, il est possible de

considérer l'appartenance à un cluster comme une information utilisable dans un système de classification d'attaquants, ce qui est l'objectif initial.

Le modèle proposé ne prend pas en compte des informations qui dépendent du temps, ce qui lui permet de détecter des attaques non uniformes temporellement. Comme notre approche est non supervisée, il est possible de détecter des attaques de type 0-day. Pour détecter des scénarios complexes de cyberattaques, un module de corrélation doit être développé afin d'établir un lien entre les données d'un même attaquant, c'est pourquoi l'objectif 6 n'est pas rempli.

L'objectif 7 n'est pas totalement complété, car les temps de calcul sont encore trop importants pour la phase de prétraitement. Néanmoins, il existe plusieurs façons d'optimiser cette phase qui n'est pour l'instant qu'à l'état de prototype (voir la Section 6.3). Enfin, notre modèle a correctement été validé sur trois jeux de données différents, dont un où les données ont été parfaitement classifiées (100% de précision).

6.2 Limitations de la solution proposée

Bien que la majorité des objectifs aient été remplis, notre algorithme a tout de même quelques limitations. Comme exposé ci-dessus, la vitesse d'exécution de notre algorithme n'est pas satisfaisante. Aussi, notre algorithme ne peut pas détecter des scénarios complexes de cyberattaques. Mais ces points ne sont pas les seules limitations de notre algorithme.

Premièrement, notre algorithme de prétraitement présente un défaut : les valeurs des attributs sont comparées deux à deux pour éventuellement être égalisées. Par exemple, si l'on a les charges utiles A, B, et C, on compare A et B, si la similarité dépasse un certain seuil, on fixe B = A. Imaginons que A et C ne dépassent pas le seuil de similarité. On passe alors à B, qui vaut maintenant A, et qui ne sera donc pas comparé à C. Pourtant, peut-être que la valeur initiale de B aurait pu être suffisamment similaire C. En fait, une limite de notre algorithme est le fait que de nombreuses valeurs vont être fixées à la première valeur rencontrée par l'algorithme ayant une similarité suffisante, sans pour autant que celle-ci soit la meilleure.

Aussi, notre algorithme peut connaître des difficultés lorsqu'il s'applique sur des données chiffrées ou encodées différemment. Néanmoins, si ce ne sont pas toutes les données qui sont chiffrées ou encodées différemment, notre modèle est capable d'isoler ces données dans un même cluster, comme on l'a vu avec les données SSH fournies. Ceci s'explique par le fait que les données chiffrées ou encodées différemment ont généralement le même format entre elles. Par exemple, un encodage en hexadécimal comportera un nombre important de « 0x », donc les données en hexadécimal ont donc plus de chance de se retrouver dans le même cluster.

Pourtant, ce n'est pas parce que deux entrées sont en hexadécimal qu'elles ont le même sens et qu'elles doivent se retrouver dans le même cluster.

Une autre limite de notre modèle est sa capacité à s'adapter lors de l'ingestion de nouvelles données. En effet, lorsque notre modèle doit ingérer de nouvelles données, il existe un module de sauvegarde permettant de faire comme si l'on reprenait le prétraitement, en y ajoutant ces nouvelles données. Si l'on a les données initiales A, B, et C et que l'algorithme les a transformées en A, A et C, alors si on rajoute D et E, il faudra seulement les comparer à A et C. Bien que ce système fonctionne correctement, il faut tout de même, lors de l'ingestion de données : exécuter le module de prétraitement pour les nouvelles données, exécuter le module de représentation des données, puis le module de détermination du nombre idéal de clusters. La méthode d'ingestion de données est donc plutôt laborieuse. Le nombre idéal de clusters doit d'ailleurs être déterminé manuellement grâce à la lecture des métriques, ce qui constitue une limitation. Néanmoins, lorsqu'on souhaite seulement ingérer une faible quantité de données, on peut émettre l'hypothèse que le nombre de clusters ne va pas changer, et mettre à jour celui-ci uniquement lorsque le volume de données a augmenté d'un certain pourcentage. Les vulnérabilités de type 0-day ne seront détectées que lorsque l'algorithme de détermination du nombre idéal de clusters sera exécuté. Il faut donc veiller à actualiser le nombre idéal de clusters suffisamment régulièrement.

Enfin, notre modèle a la particularité de fonctionner protocole par protocole. Bien que ceci nous offre des avantages (comme la possibilité d'utiliser la charge utile des paquets), cette spécificité a aussi ses inconvénients. En effet, il n'est actuellement pas encore possible de classifier une cyberattaque qui s'étend sur plusieurs protocoles avec notre modèle.

6.3 Travaux futurs

6.3.1 Optimisation

Plusieurs améliorations peuvent être apportées au modèle que nous avons défini. Premièrement, il est clair que l'algorithme de prétraitement peut être largement optimisé. Pour ce faire, il faut notamment travailler sur la parallélisation des calculs, qui pourrait considérablement améliorer la vitesse d'exécution. Ceci permettrait à notre algorithme d'être utilisé en temps réel. D'ailleurs, d'autres améliorations peuvent être envisagées pour rendre possible l'utilisation de notre modèle en direct. On pourrait par exemple améliorer le module de détermination du nombre idéal de clusters en l'automatisant. Une première idée d'implémentation serait la suivante : on détermine le score Silhouette maximum et le score de Davies-Bouldin minimum, puis, pour chaque valeur de cluster, on calcule le rapport entre le score Silhouette

et le score Silhouette maximum (de même pour Davies-Bouldin). On additionne ces deux rapports et le nombre idéal de clusters est celui pour lequel la somme de ces deux rapports est la plus grande. Par exemple, si on obtient 98% du score Silhouette maximum et 95% du score de Davies-Bouldin minimum pour une valeur X de nombre de clusters et respectivement 100% et 90% pour une valeur Y, alors X sera le nombre idéal de clusters retenu ($98 + 95 > 100 + 90$).

6.3.2 Choix des attributs

Certaines améliorations peuvent également être réalisées sur le choix des attributs. Actuellement, on ne se limite qu'à choisir les attributs les plus caractéristiques et représentatifs d'un paquet, sans laisser la place à d'autres attributs tels que des métadonnées, qui sont utilisés par les travaux de classification de cyberattaques présentés dans l'état de l'art. Néanmoins, l'utilisation de ces attributs pourrait améliorer les performances de notre modèle. Il faudrait alors mettre en place une sélection automatique de ces attributs, puisqu'ils sont souvent nombreux (85 pour CIC-IDS2017 par exemple). Il faudrait peut-être également limiter la contribution de certains de ces attributs, pour ne pas absorber la contribution des attributs initiaux, qui devraient être plus caractéristiques et qui devraient contenir plus d'informations que les métadonnées.

6.3.3 Assignation des labels aux clusters

Il faudrait aussi créer un module permettant d'utiliser les résultats de différentes classifications de notre modèle sur différents protocoles afin de proposer une seule et unique classification, mélangeant tous les protocoles. Ceci permettrait alors une interprétation qui se baserait sur plusieurs protocoles. On pourrait également chercher à automatiser l'assignation d'un label pour un cluster en travaillant avec une base de signature. L'objectif pourrait alors être la mise en relation d'un cluster avec un numéro de CVE, mais cela suppose que chaque cluster contient exactement une CVE, ce qui est une hypothèse très forte. On pourrait également utiliser des bases de signatures pour interpréter automatiquement les clusters, en utilisant par exemple un IDS. Pour cela, il faut alors disposer du format PCAP des données que l'on souhaite analyser. Ainsi, lorsque l'on utilise les données d'un pot de miel, celui-ci doit alors enregistrer les données au format PCAP, en parallèle de la journalisation des attributs.

6.3.4 Corrélation

Un autre module pourrait utiliser cette classification, ou même directement être appliqué sur une classification n'utilisant qu'un seul protocole, afin d'établir une corrélation entre les données. Comme le modèle n'utilise pas les données relatives à l'attaquant, comme son identifiant de session ou son adresse IP, il est alors possible d'utiliser ces informations afin de définir une cyberattaque comme étant un enchaînement de numéros de cluster pour une même IP ou une même session. Par exemple, si un attaquant (défini par une session ou une IP) a des entrées dans le cluster 2 et 5 de la classification HTTP et 1, 3 et 9 de la classification SSH, alors on peut dire que la séquence $[(2,5)_{HTTP}, (1,3,9)_{SSH}]$ (l'ordre dépend des timestamp), est définie comme une cyberattaque, ou au moins comme une partie d'une cyberattaque. Ainsi, on pourrait alors faire de la prédiction d'attaque : si un attaquant est en train d'effectuer la séquence $[(2,5)_{HTTP}, (1,3)_{SSH}]$, il y a de grandes chances pour que sa prochaine action appartienne au cluster 9. Un attaquant pourrait alors changer d'adresse IP ou ouvrir une nouvelle session afin de contourner notre modèle. C'est pourquoi on estime qu'une séquence de numéro de clusters représente une partie d'une cyberattaque, et non pas une cyberattaque au complet. Par exemple, si un attaquant génère la séquence $[(1,3,9,4,5,6)_{SSH}]$, et que deux autres attaquants ont généré les séquences $[(1,3,9)_{SSH}]$ et $[(4,5,6)_{SSH}]$, le modèle devra être capable de comprendre que ces deux dernières séquences représentent en fait la même cyberattaque, puisqu'il aura déjà rencontré une séquence constituée de la concaténation de ces deux séquences. Il est alors possible de détecter des changements d'adresse IP ou de session.

RÉFÉRENCES

[1] C. C. pour la Cybersécurité, "Bulletin sur les cybermenaces : La menace des rançongiciels en 2021," 2022. [En ligne]. Disponible : https://cyber.gc.ca/sites/default/files/cyber/2021-12/Cyber-ransomware-update-threat-bulletin_fra.pdf

[2] S. Morgan, "CYBERCRIME FACTS AND STATISTICS," janv. 2021. [En ligne]. Disponible : https://cybersecurityventures.com/wp-content/uploads/2021/01/Cyberwarfare-2021-Report.pdf

[3] Fireye, "Outils SIEM — Trop d'alertes tue l'alerte," 2019. [En ligne]. Disponible : https://www.fireeye.com/content/dam/fireeye-www/regional/fr_FR/products/pdfs/wp-siem-that-cried-wolf.pdf

[4] H. Debar, D. Curry et B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," *RFC4765*, 2007.

[5] E. G. Harris et M. Richardson, "PCAP Capture File Format," déc. 2020. [En ligne]. Disponible : https://tools.ietf.org/id/draft-gharris-opsawg-pcap-00.html

[6] SNORT, "Snort - Network Intrusion Detection & Prevention System," 2022. [En ligne]. Disponible : https://www.snort.org/

[7] T. O. I. S. Foundation, "Suricata," 2022. [En ligne]. Disponible : https://suricata.io/

[8] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba et K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, vol. 34, nº. 4, p. 579–595, oct. 2000. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1389128600001390

[9] Q. Cheng, C. Wu et S. Zhou, "Discovering Attack Scenarios via Intrusion Alert Correlation Using Graph Convolutional Networks," *IEEE Communications Letters*, vol. 25, nº. 5, p. 1564–1567, mai 2021, conference Name : IEEE Communications Letters.

[10] M. Ring, S. Wunderlich, D. Scheuring, D. Landes et A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, p. 147–167, sept. 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S016740481930118X

[11] I. Sharafaldin, A. Habibi Lashkari et A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization :," dans *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 2018, p. 108–116. [En ligne]. Disponible : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116

[12] M. Ring, S. Wunderlich, D. Grüdl, D. Landes et A. Hotho, “Flow-based benchmark data sets for intrusion detection,” *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, juin 2017.

[13] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro et R. Therón, “UGR’16 : A new dataset for the evaluation of cyclostationarity-based network IDSs,” *Computers & Security*, vol. 73, p. 411–424, mai 2018.

[14] N. Moustafa et J. Slay, “UNSW-NB15 : a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015.

[15] M. Ghurab, G. Gaphari, F. Alshami, R. Alshamy et S. Othman, “A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System,” *Asian Journal of Research in Computer Science*, vol. 7, n°. 4, avr. 2021.

[16] O. Yavanoglu et M. Aydos, “A review on cyber security datasets for machine learning algorithms,” dans *2017 IEEE International Conference on Big Data (Big Data)*, déc. 2017, p. 2186–2193.

[17] “KDD Cup 1999 Data,” 1999. [En ligne]. Disponible : http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[18] A. A. Olusola, A. S. Oladele et D. O. Abosede, “Analysis of KDD ’99 Intrusion Detection Dataset for Selection of Relevance Features,” *Proceedings of the World Congress on Engineering and Computer Science 2010*, vol. 1, 2010.

[19] M. Tavallaee, E. Bagheri, W. Lu et A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” dans *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, juill. 2009, p. 1–6, iSSN : 2329-6275.

[20] R. Singh, H. Kumar et R. K. Singla, “A Reference Dataset for Network Traffic Activity Based Intrusion Detection System,” *International journal of computers communications & control*, vol. 10, n°. 3, p. 390–402, avr. 2015, number : 3. [En ligne]. Disponible : https://www.univagora.ro/jour/index.php/ijccc/article/view/1924

[21] S. Garcia, M. Grill, J. Stiborekn et A. Zunino, “An empirical comparison of botnet detection methods,” *Computers & Security*, vol. 45, p. 100–123, sept. 2014. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0167404814000923

[22] R. Panigrahi et S. Borah, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” *International Journal of Engineering & Technology*, vol. 7, p. 479–482, janv. 2018.

[23] Z. Pelletier et M. Abualkibash, “Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing

Language R," *International Research Journal of Advanced Engineering and Science*, vol. 5, n°. 2, p. 5, mai 2020.

[24] M. Kuhn, *The caret Package*, 2019. [En ligne]. Disponible : https://topepo.github.io/caret/

[25] M. Kursa et W. Rudnicki, "Feature Selection with Boruta Package," *Journal of Statistical Software*, vol. 36, p. 1–13, sept. 2010.

[26] A. Yulianto, P. Sukarno et N. A. Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset," *Journal of Physics : Conference Series*, vol. 1192, p. 012018, mars 2019, publisher : IOP Publishing. [En ligne]. Disponible : https://doi.org/10.1088/1742-6596/1192/1/012018

[27] H. Abdi et L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, n°. 4, p. 433–459, 2010, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.101. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101

[28] R. E. Schapire, "Explaining AdaBoost," dans *Empirical Inference : Festschrift in Honor of Vladimir N. Vapnik*, B. Schölkopf, Z. Luo et V. Vovk, édit. Berlin, Heidelberg : Springer, 2013, p. 37–52. [En ligne]. Disponible : https://doi.org/10.1007/978-3-642-41136-6_5

[29] X. Zhang, J. Chen, Y. Zhou, L. Han et J. Lin, "A Multiple-Layer Representation Learning Model for Network-Based Attack Detection," *IEEE Access*, vol. 7, p. 91 992–92 008, 2019, conference Name : IEEE Access.

[30] J. Feng, "gcForest," oct. 2022, original-date : 2017-06-01T06 :23 :57Z. [En ligne]. Disponible : https://github.com/kingfengji/gcForest

[31] K. O'Shea et R. Nash, "An Introduction to Convolutional Neural Networks," déc. 2015, arXiv :1511.08458 [cs]. [En ligne]. Disponible : http://arxiv.org/abs/1511.08458

[32] O. Faker et E. Dogdu, "Intrusion Detection Using Big Data and Deep Learning Techniques," dans *Proceedings of the 2019 ACM Southeast Conference*, ser. ACM SE '19. New York, NY, USA : Association for Computing Machinery, avr. 2019, p. 86–93. [En ligne]. Disponible : https://doi.org/10.1145/3299815.3314439

[33] N. Moustafa et J. Slay, "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*, vol. 25, n°. 1-3, p. 18–31, avr. 2016, publisher : Taylor & Francis _eprint : https ://doi.org/10.1080/19393555.2015.1125974. [En ligne]. Disponible : https://doi.org/10.1080/19393555.2015.1125974

[34] "PerfectStorm | Keysight," oct. 2022. [En ligne]. Disponible : https://www.keysight.com/us/en/products/network-test/network-test-hardware/perfectstorm.html

[35] D. Jing et H.-B. Chen, "SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset," dans *2019 IEEE 13th International Conference on ASIC (ASICON)*, oct. 2019, p. 1–4, iSSN : 2162-755X.

[36] A. Roy et K. J. Singh, "Multi-classification of UNSW-NB15 Dataset for Network Anomaly Detection System," dans *Proceedings of International Conference on Communication and Computational Technologies*, S. D. Purohit, D. Singh Jat, R. C. Poonia, S. Kumar et S. Hiranwal, édit. Springer Singapore, 2021, p. 429–451. [En ligne]. Disponible : http://link.springer.com/10.1007/978-981-15-5077-5_40

[37] A. Aleesa, M. Thanoun, A. Mohammed et N. Sahar, "DEEP-INTRUSION DETECTION SYSTEM WITH ENHANCED UNSW-NB15 DATASET BASED ON DEEP LEARNING TECHNIQUES," *Journal of Engineering Science and Technology*, vol. 16, p. 711–727, févr. 2021.

[38] J. Velasco-Mata, V. González-Castro, E. F. Fernández et E. Alegre, "Efficient Detection of Botnet Traffic by Features Selection and Decision Trees," *IEEE Access*, vol. 9, p. 120 567–120 579, 2021.

[39] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq et H. Patel, "Deep learning-based classification model for botnet attack detection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, n°. 7, p. 3457–3466, juill. 2022. [En ligne]. Disponible : https://doi.org/10.1007/s12652-020-01848-9

[40] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman et L. Bian, "Botnet detection using graph-based feature clustering," *Journal of Big Data*, vol. 4, n°. 1, p. 14, déc. 2017. [En ligne]. Disponible : http://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0074-7

[41] "Categorical Encoding Methods," oct. 2022, original-date : 2015-11-29T19 :32 :37Z. [En ligne]. Disponible : https://github.com/scikit-learn-contrib/category_encoders

[42] Y. Wu, D. Wei et J. Feng, "Network Attacks Detection Methods Based on Deep Learning Techniques : A Survey," *Security and Communication Networks*, vol. 2020, p. e8872923, août 2020, publisher : Hindawi. [En ligne]. Disponible : https://www.hindawi.com/journals/scn/2020/8872923/

[43] K. Takyi, A. Bagga et P. Goopta, "Clustering Techniques for Traffic Classification : A Comprehensive Review," dans *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, août 2018, p. 224–230.

[44] S. Zanero et S. M. Savaresi, “Unsupervised learning techniques for an intrusion detection system,” dans *Proceedings of the 2004 ACM symposium on Applied computing - SAC ’04*. Nicosia, Cyprus : ACM Press, 2004, p. 412. [En ligne]. Disponible : http://portal.acm.org/citation.cfm?doid=967900.967988

[45] “SOM Toolbox,” déc. 2012. [En ligne]. Disponible : http://www.cis.hut.fi/projects/somtoolbox/

[46] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos et G. Marreiros, “Performance evaluation of unsupervised techniques in cyber-attack anomaly detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, n°. 11, p. 4477–4489, nov. 2020. [En ligne]. Disponible : http://link.springer.com/10.1007/s12652-019-01417-9

[47] “scikit-learn : machine learning in Python — scikit-learn 1.0.2 documentation,” 2022. [En ligne]. Disponible : https://scikit-learn.org/stable/

[48] “R Library Contrast Coding Systems for categorical variables,” 2011. [En ligne]. Disponible : https://stats.oarc.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/

[49] “BaseN Encoding and Grid Search in category_encoders – PyBloggers,” déc. 2016. [En ligne]. Disponible : https://www.pybloggers.com/2016/12/basen-encoding-and-grid-search-in-category_encoders/,https://www.pybloggers.com/2016/12/basen-encoding-and-grid-search-in-category_encoders/

[50] W. McGinnis, “Beyond One-Hot : an exploration of categorical variables,” 2015. [En ligne]. Disponible : https://www.kdnuggets.com/beyond-one-hot-an-exploration-of-categorical-variables.html

[51] N. Singh, “Overview of Encoding Methodologies,” déc. 2018. [En ligne]. Disponible : https://www.datacamp.com/tutorial/encoding-methodologies

[52] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford et A. Smola, “Feature Hashing for Large Scale Multitask Learning,” févr. 2010, arXiv :0902.2206 [cs]. [En ligne]. Disponible : http://arxiv.org/abs/0902.2206

[53] “PCAP files from the US National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC),” 2012. [En ligne]. Disponible : https://www.netresec.com/?page=MACCDC

[54] K. Zaraska, *Prelude IDS : current state and development perspectives*, mai 2003.

[55] “prelude-correlator,” sept. 2020. [En ligne]. Disponible : https://github.com/Prelude-SIEM/prelude-correlator

[56] M. Husák et J. Kašpar, “AIDA Framework : Real-Time Correlation and Prediction of Intrusion Detection Alerts,” dans *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES ’19. New York, NY, USA : Association for Computing Machinery, août 2019, p. 1–8. [En ligne]. Disponible : https://doi.org/10.1145/3339252.3340513

[57] “pandas.DataFrame — pandas 1.5.1 documentation,” 2022. [En ligne]. Disponible : https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[58] “difflib — Helpers for computing deltas — Python 3.10.6 documentation,” 2022. [En ligne]. Disponible : https://docs.python.org/3/library/difflib.html

[59] K. P. Sinaga et M.-S. Yang, “Unsupervised K-Means Clustering Algorithm,” *IEEE Access*, vol. 8, p. 80 716–80 727, 2020.

[60] “ndjson,” 2022. [En ligne]. Disponible : http://ndjson.org/

[61] G. Combs, “Wireshark · Go Deep.” 2022. [En ligne]. Disponible : https://www.wireshark.org/

[62] ——, “tshark,” 2022. [En ligne]. Disponible : https://www.wireshark.org/docs/man-pages/tshark.html

[63] ——, “editcap,” 2022. [En ligne]. Disponible : https://www.wireshark.org/docs/man-pages/editcap.html

[64] ——, “mergecap,” 2022. [En ligne]. Disponible : https://www.wireshark.org/docs/man-pages/mergecap.html

[65] “Add SSH decryption support (#16054) · Issues · Wireshark Foundation / wireshark · GitLab,” 2022. [En ligne]. Disponible : https://gitlab.com/wireshark/wireshark/-/issues/16054

[66] R. Wood, “DAMN VULNERABLE WEB APPLICATION,” oct. 2022, original-date : 2013-05-01T13 :03 :10Z. [En ligne]. Disponible : https://github.com/digininja/DVWA

[67] J. Bejarano, K. Bose, T. Brannan, A. Thomas, K. Adragni, N. Neerchal et G. Ostrouchov, “Sampling Within k-Means Algorithm to Cluster Large Datasets,” Rapport technique ORNL/TM-2011/394, 1025410, août 2011. [En ligne]. Disponible : http://www.osti.gov/servlets/purl/1025410/

[68] “Nessus® | Tenable®,” 2022. [En ligne]. Disponible : https://fr.tenable.com/products/nessus

[69] Tenable, “HTTP Methods Allowed (per directory),” déc. 2009. [En ligne]. Disponible : https://vulners.com/nessus/WEB_DIRECTORY_OPTIONS.NASL/

[70] “vBulletin, The World’s Leading Community Software,” 2020. [En ligne]. Disponible : https://www.vbulletin.com/

[71] M. Corporation, “CVE - CVE-2019-16759,” 2019. [En ligne]. Disponible : https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16759

[72] ZmEu, “Zmeubot - module for ZNC (v0.1),” mars 2021, original-date : 2016-01-22T12 :00 :27Z. [En ligne]. Disponible : https://github.com/happyhater/zmeubot-znc

[73] “l9explore,” oct. 2022, original-date : 2020-12-15T00 :39 :15Z. [En ligne]. Disponible : https://github.com/LeakIX/l9explore

[74] “Censys | Industry-Leading Cloud and Internet Asset Discovery Solutions,” 2022. [En ligne]. Disponible : https://censys.io/

[75] “Projectdiscovery.io,” 2022. [En ligne]. Disponible : https://projectdiscovery.io/#/

[76] R. D. Graham, “MASSCAN : Mass IP port scanner,” nov. 2022, original-date : 2013-07-28T05 :35 :33Z. [En ligne]. Disponible : https://github.com/robertdavidgraham/masscan

[77] “Mercury Security Access Control Hardware & Solutions,” 2022. [En ligne]. Disponible : https://mercury-security.com/portal/

[78] “ZGrab 2.0,” nov. 2022, original-date : 2016-08-19T23 :22 :02Z. [En ligne]. Disponible : https://github.com/zmap/zgrab2

[79] “curl,” 2022. [En ligne]. Disponible : https://curl.se/

[80] “requests · PyPI,” 2022. [En ligne]. Disponible : https://pypi.org/project/requests/

[81] Square, “OkHttp,” 2019. [En ligne]. Disponible : https://square.github.io/okhttp/

[82] “PycURL Home Page,” 2022. [En ligne]. Disponible : http://pycurl.io/

[83] “urllib — URL handling modules — Python 3.11.0 documentation.” [En ligne]. Disponible : https://docs.python.org/3/library/urllib.html

[84] N. B. Azhar, “gohttp,” nov. 2022, original-date : 2017-11-08T15 :28 :32Z. [En ligne]. Disponible : https://github.com/nahid/gohttp

[85] Oracle, “JNDI/LDAP Service Provider.” [En ligne]. Disponible : https://docs.oracle.com/javase/8/docs/technotes/guides/jndi/jndi-ldap.html

[86] N. Kim et A. Svetlov, “Aiohttp 3.8.3 documentation,” 2022. [En ligne]. Disponible : https://docs.aiohttp.org/en/stable/

[87] TransparenTech, “Online UUID Generator Tool,” 2022. [En ligne]. Disponible : https://www.uuidgenerator.net/

[88] D. Smith, “Who’s Viktor ? Tracking down the XTC/Polaris Botnets. | Radware Blog,” mai 2020. [En ligne]. Disponible : https://blog.radware.com/security/botnets/2020/05/whos-viktor-tracking-down-the-xtc-polaris-botnets/

[89] MITRE, “NVD - cve-2018-10561,” 2018. [En ligne]. Disponible : https://nvd.nist.gov/vuln/detail/cve-2018-10561

[90] shellord, “Netlink GPON Router 1.0.11 - Remote Code Execution,” mars 2020. [En ligne]. Disponible : https://www.exploit-db.com/exploits/48225

[91] “CWE - CWE-912 : Hidden Functionality (4.9),” 2012. [En ligne]. Disponible : https://cwe.mitre.org/data/definitions/912.html

[92] R. Nigam, “New Mirai Variant Adds 8 New Exploits, Targets Additional IoT Devices,” juin 2019. [En ligne]. Disponible : https://unit42.paloaltonetworks.com/new-mirai-variant-adds-8-new-exploits-targets-additional-iot-devices/

[93] MITRE, “NVD - CVE-2017-9248,” 2017. [En ligne]. Disponible : https://nvd.nist.gov/vuln/detail/CVE-2017-9248

[94] rintod, “exploit-kita,” juin 2021, original-date : 2019-11-01T09 :37 :34Z. [En ligne]. Disponible : https://github.com/rintod/exploit-kita/blob/191a8b07b11e4d5f0eccf1afcf5a7b4c955647a7/Telerix.Scan.py

[95] “CVE-2020-28036 : wp-includes/class-wp-xmlrpc-server.php in WordPress before 5.5.2 allows attackers to gain privileges by using XML-RPC to,” 2020. [En ligne]. Disponible : https://www.cvedetails.com/cve/CVE-2020-28036/

[96] J. Lands, “Dota3 : Is your Internet of Things device moonlighting ?” févr. 2020. [En ligne]. Disponible : https://blogs.juniper.net/en-us/threat-research/dota3-is-your-internet-of-things-device-moonlighting

[97] O. Foundation, “OpenSSH,” 2022. [En ligne]. Disponible : https://www.openssh.com/