Vers l'automatisation de la détection d'anomalies réseaux

Christophe Maudoux

HESAM UNIVERSITÉ

le cnam

ÉCOLE DOCTORALE Sciences des Métiers de l'Ingénieur
Centre d'études et de recherche en informatique et communications

THÈSE

présentée par : **Christophe MAUDOUX**
soutenue le : **19 juin 2024**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**
Discipline : **Informatique**
Spécialité : **Informatique**

Vers l'automatisation de la détection d'anomalies réseaux

THÈSE dirigée par :
Mme. BOUMERDASSI Selma MdC HDR, Cnam Paris

Jury composé de :

M^me^ Anne FLADENMULLER	PU, LIP6, Sorbonne Université	Présidente
M. André-Luc BEYLOT	PU, IRIT, Toulouse INP	Rapporteur
M. Sidi-Mohammed SENOUCI	PU, ISAT, Université Bourgogne FC	Rapporteur
M. Hacène FOUCHAL	PU, Université de Reims CA	Examinateur
M. Yacine GHAMRI-DOUDANE	PU, L3i, Université de La Rochelle	Examinateur
M^me^ Véronique VÈQUE	PU, L2S, Université de Paris-Saclay	Examinatrice
M. Sébastien HAMEL	Colonel, Chargé de missions, ANFSI	Invité

THÈSE

Je dédie ce travail à mon épouse *Nadia* qui m'accompagne depuis maintenant 24 ans dont 12 années d'études ainsi qu'à mon fils *Charles*. Que de chemin parcouru depuis mon inscription en licence ! J'adresse une pensée chaleureuse à mes parents et plus particulièrement à ma maman qui a toujours était très fière de son fils, mais qui, malheureusement, n'a pas toujours su lui rendre tout l'amour qu'elle aurait mérité...

Remerciements

Je tiens tout d'abord à adresser mes remerciements au Général de brigade (2s) Xavier Guimard qui m'a fait confiance en 2016 et sans qui je n'en serais pas là aujourd'hui.

J'adresse mes plus chaleureux remerciements à ma directrice de thèse, M^{me} Selma Boumerdassi, qui a cru en moi, m'a toujours guidé, conseillé et accompagné depuis le diplôme d'Ingénieur en 2018 jusqu'au bout de ce parcours doctoral. *Merci Selma !!!*

Merci aux professeurs André-Luc Beylot et Sidi-Mohammed Senouci pour le temps et l'énergie consacrés à rapporter cette thèse. Leurs remarques pertinentes ont permis d'améliorer la qualité de ce manuscrit et de sa présentation.

Je remercie le Général Vincent Bérériat, Chef de la Direction de la Sécurité & de l'Architecture au sein de l'Agence du Numérique des Forces de Sécurité Intérieure (ANFSI) qui m'a toujours soutenu dans ce parcours doctorant et dans mes différentes démarches ainsi que le Colonel Daniel Orvöen, Conseiller Technique pour la spécialité Systèmes d'Information & de Communication (SIC) qui m'a appuyé pour intégrer le parcours doctorant au sein de la Gendarmerie nationale.

Je remercie également le Colonel Sébastien Hamel, Chargé de Missions & de Projets au sein de l'ANFSI, qui a également été membre de mon jury pour l'obtention de mon diplôme d'Ingénieur et dont la bonne humeur m'accompagne encore aujourd'hui pour mon jury de thèse ;-).

Enfin, je remercie la Gendarmerie nationale et le Centre de Recherche de l'École des Officiers de la Gendarmerie Nationale (CREOGN) qui m'ont octroyé du temps pour réaliser mes travaux de recherche, rédiger ce document et accepté de financer ce parcours doctorant.

Une pensée également pour mes collègues qui m'ont épaulé pendant toutes ces années, en particulier au Chef d'Escadron Jérémie Fabre, Chef de la Section Gestion des Identités & des Accès, au Major Antoine Rosier, à messieurs Thierry Ramon, Adam Sores et à madame Églantine Mallinger.

Résumé

Nous vivons dans un monde hyperconnecté. À présent, la majorité des objets qui nous entourent échangent des données soit entre-eux soit avec un serveur. Ces échanges produisent alors de l'activité réseau. C'est l'étude de cette activité réseau qui nous intéresse ici et sur laquelle porte ce document. En effet, tous les messages et donc le trafic réseau généré par ces équipements est voulu et par conséquent légitime. Il est de ce fait parfaitement formaté et connu. Parallèlement à ce trafic qui peut être qualifié de "normal", il peut exister du trafic qui ne respecte pas les critères attendus. Ces échanges non conformes aux attendus peuvent être catégorisés comme étant du trafic "anormal". Ce trafic illégitime peut être dû à plusieurs causes tant internes qu'externes. Tout d'abord, pour des raisons bassement mercantiles, la plus part de ces équipements connectés (téléphones, montres, serrures, caméras,...) est peu, mal, voire pas protégée du tout. De ce fait, ils sont devenus les cibles privilégiées des cybercriminels. Une fois compromis, ces matériels communiquant constituent des réseaux capables de lancer des attaques coordonnées : des *botnets*. Le trafic induit par ces attaques ou les communications de synchronisation internes à ces botnets génèrent alors du trafic illégitime qu'il faut pouvoir détecter. Notre première contribution a pour objectif de mettre en lumière ces échanges internes, spécifiques aux botnets. Du trafic anormal peut également être généré lorsque surviennent des événements externes non prévus ou extra-ordinaires tels des incidents ou des changements de comportement des utilisateurs. Ces événements peuvent impacter les caractéristiques des flux de trafic échangés comme leur volume, leurs sources, destinations ou encore les paramètres réseaux qui les caractérisent. La détection de ces variations de l'activité réseau ou de la fluctuation de ces caractéristiques est l'objet de nos contributions suivantes. Il s'agit d'un framework puis d'une méthodologie qui en découle permettant d'automatiser la détection de ces anomalies réseaux et éventuellement de lever des alertes.

Mots-clés : cybersécurité, apprentissage automatique, détection d'anomalies, réseaux, botnets, ADN, signatures numériques, distance de Damerau-Levenstein, corrélation d'événements

Abstract

We live in a hyperconnected world. Currently, the majority of the objects surrounding us exchange data either among themselves or with a server. These exchanges consequently generate network activity. It is the study of this network activity that interests us here and forms the focus of this thesis. Indeed, all messages and thus the network traffic generated by these devices are intentional and therefore legitimate. Consequently, it is perfectly formatted and known. Alongside this traffic, which can be termed "normal," there may exist traffic that does not adhere to expected criteria. These non-conforming exchanges can be categorized as "abnormal" traffic. This illegitimate traffic can be due to several internal and external causes. Firstly, for purely commercial reasons, most of these connected devices (phones, watches, locks, cameras, etc.) are poorly, inadequately, or not protected at all. Consequently, they have become prime targets for cybercriminals. Once compromised, these communicating devices form networks capable of launching coordinated attacks: botnets. The traffic induced by these attacks or the internal synchronization communications within these botnets then generates illegitimate traffic that needs to be detected. Our first contribution aims to highlight these internal exchanges, specific to botnets. Abnormal traffic can also be generated when unforeseen or extraordinary external events occur, such as incidents or changes in user behavior. These events can impact the characteristics of the exchanged traffic flows, such as their volume, sources, destinations, or the network parameters that characterize them. Detecting these variations in network activity or the fluctuation of these characteristics is the focus of our subsequent contributions. This involves a framework and resulting methodology that automates the detection of these network anomalies and potentially raises alerts.

Keywords: Cybersecuriry, Machine Learning, Anomalies Detection, Networks, Botnets, DNA, Digital Signatures, Damerau-Levenstein Distance, Events Correlation

Table des matières

Liste des tableaux

Table des figures

Publications

Conférences internationales

— C. Maudoux et S. Boumerdassi, « DiNATrAX : a Network Anomalies Detection Framework », in *2024 IEEE International Conference on Communications (ICC)*, jui. 2024

— C. Maudoux et S. Boumerdassi, « Unsupervised Anomaly Knowledge Flow : A Digital Signatures Extraction Approach », in *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, oct. 2023, p. 1-6. DOI : 10.1109/WINCOM59760.2023.10323022. adresse : https://ieeexplore.ieee.org/abstract/document/10323022 (visité le 24/12/2023)

— C. Maudoux et S. Boumerdassi, « Network Anomalies Detection by Unsupervised Activity Deviations Extraction », in *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, sept. 2022, p. 1-5. DOI : 10.1109/GIIS56506.2022.9937022. adresse : https://ieeexplore.ieee.org/document/9937022 (visité le 24/12/2023)

— C. Maudoux et S. Boumerdassi, « LemonLDAP : :NG A Full AAA Free Open Source WebSSO Solution », in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, nov. 2022, p. 277-281. DOI : 10.1109/CloudNet55617.2022.9978777. adresse : https://ieeexplore.ieee.org/abstract/document/9978777

— C. Maudoux, S. Boumerdassi, A. Barcello et E. Renault, « Combined Forest : A New Supervised Approach for a Machine-Learning-based Botnets Detection », in *2021 IEEE Global Communications Conference (GLOBECOM)*, déc. 2021, p. 01-06. doi : 10.1109/GLOBECOM46510.2021.9685261

— C. Maudoux et S. Boumerdassi, « Smart and Sustainable Agriculture », in *Smart and Sustainable Agriculture*, S. Boumerdassi, M. Ghogho et É. Renault, éd., sér. Communications in Computer and Information Science, Cham : Springer International Publishing, 2021, p. 103-121, isbn : 978-3-030-88259-4. doi : 10.1007/978-3-030-88259-4_8

Poster

— C. Maudoux et S. Boumerdassi, *LemonLDAP : :NG - A Full AAA Free Open Source WebSSO Solution*, nov. 2022. doi : 10.1109/CloudNet55617.2022.9978777. adresse : https://hal.science/hal-03949890 (visité le 20/01/2024)

Logiciel

— X. Guimard, C. Oudot, C. Maudoux et M. Besson, « LemonLDAP : :NG », déc. 2010. adresse : https://hal.inria.fr/hal-03776592

Chapitre 1

Introduction

1.1 Contexte & motivations

Le début des années 2000 a marqué l'émergence du Web 2.0 avec notamment l'arrivée des réseaux sociaux et la collaboration en ligne. Nous avons alors vu naître un monde connecté. Au cours de la décennie suivante, l'explosion de l'Internet des Objets (IoT) a révolutionné notre univers pour le faire évoluer d'un *monde connecté* à un *univers hyperconnecté*. Ceci a ouvert la voie à un large éventail d'applications notamment dans des domaines tels que la domotique, la santé, l'industrie, les transports et bien d'autres encore. Ainsi, de nombreux appareils et équipements sont désormais dotés de capacités de communication, d'échange et d'interaction.

Tout ceci nous a apporté son lot d'avantages et de fonctionnalités devenus "indispensables" mais également son lot de nouvelles *menaces*. En effet, derrière cette révolution de l'IoT, les grands gagnants sont bien-sûr les consommateurs mais surtout les industriels et les fabricants d'équipements certes, mus par l'envie de nous apporter de nouveaux services, mais surtout par l'aspect mercantile lié à ce nouveau marché. Pour s'assurer des revenus constants et confortables, deux solutions s'offraient à eux. Soit nous vendre de plus en plus chers les équipements connectés ainsi que tous les services gravitant autour (stockage ou analyse des données générées, prédictions,...), soit miser sur l'obsolescence programmée afin de produire et vendre toujours plus de ces objets connectés.

Dans cette course en avant effrénée à l'innovation, la sécurité a bien souvent été laissée de côté pour gagner en temps de conception et en facilité de déploiement. À présent, tout est connecté et c'est là que les problèmes commencent. Ces équipements connectés (téléphones, réfrigérateurs, caméras, serrures, montres, vêtements,...) peu, mal voire pas du tout sécurisés sont devenus les cibles privilégiées

des "hackers" pour en faire des relais capables de lancer des attaques comme, tout récemment, la cyberattaque menée contre le réseau interministériel de l'État à l'aide d'un énième botnet [8].

Généralement, ces cyberattaques sont véhiculées par le réseau d'interconnexion de l'ensemble de ces équipements et se manifestent par des changements de comportements que nous pouvons qualifier d'*anomalies réseaux*. Par exemple, un malware cherchant à se propager sur un réseau peut générer un trafic anormal en scannant des plages d'adresses IP, de ports ou en essayant de se connecter à des systèmes vulnérables. De même, un attaquant cherchant à exfiltrer des données sensibles peut induire des anomalies en transférant de grandes quantités de données vers des destinations externes. Ces anomalies réseaux peuvent indiquer des comportements inhabituels ou malveillants dans un environnement informatique. Cela peut inclure des schémas de trafic inhabituels, des pics soudains d'activité, des connexions inattendues ou des communications suspectes entre des systèmes. L'analyse des modèles de trafic et de comportement réseau peut également révéler les indicateurs d'une cyberattaque imminente ou en cours. Par exemple, la détection de schémas de communication non autorisés ou de tentatives répétées d'accès à des ressources sensibles peut révéler une activité malveillante.

Détecter ces anomalies est donc devenu un enjeu crucial dans le domaine de la cybersécurité. C'est dans cette optique de détection des anomalies réseaux que s'inscrivent nos travaux, travaux présentés dans ce document. Mais, avant d'entrer dans le vif du sujet, qu'est-ce qu'une anomalie ?

1.2 Anomalies

Dans les contextes de l'analyse de données et de la théorie des événements rares, une anomalie (souvent appelée point aberrant ou valeur atypique) est une observation qui dévie tellement des autres qu'elle suscite des soupçons quant à la génération par un mécanisme ou une origine différente. Dans les sections 1.2 et 1.3, nous nous attachons à définir les notions d'anomalies et plus spécifiquement d'anomalies réseaux mais également à poser les différents concepts gravitant autour de celles-ci.

Cette présentation se veut exhaustive mais dans le cadre de cette thèse, nous nous sommes focalisés sur la détection de l'activité de certaines familles de botnets, de quelques unes de leurs attaques les plus courantes ainsi qu'à la détection d'anomalies liées à des causes externes comme des événements particuliers ayant un impact sur l'activité réseau.

1.2.1 Définition

Statistiquement, une anomalie est une *déviation par rapport à la norme*. Elle est souvent une observation qui est significativement différente de la majorité des données incluses dans un jeu de données. Par exemple, dans une distribution normale, les observations qui se situent à plus ou moins trois écarts-types de la moyenne sont souvent considérées comme des anomalies. Dans ce cas, les anomalies peuvent être mises en évidence en regardant les mesures de centralité telles que la moyenne ou la médiane et en évaluant combien une observation est éloignée de ces valeurs.

D'autres mesures comme les écarts-types ou les inter-quartiles peuvent aussi aider à identifier les anomalies en mesurant la dispersion des données.

1.2.2 Types & Causes

Il existe différents types d'anomalie en fonction du point vue par lequel elles sont appréhendées. Une anomalie peut être *ponctuelle*. Dans ce cas, il n'y a qu'une seule observation anormale par rapport à l'ensemble des données. Une anomalie peut également être *contextuelle*. Une observation sera considérée comme anormale uniquement dans un certain contexte particulier (par exemple, une température élevée peut être normale en été, mais anormale en hiver). Enfin, une anomalie peut être *collective*. C'est-à-dire qu'un ensemble d'observations peut ne pas être anormal par lui-même, mais peut être considéré comme anormal lorsqu'il est considéré dans son ensemble (par exemple, une série ininterrompue de pertes financières peut être une anomalie alors qu'une seule perte ne l'est pas).

Ces anomalies peuvent être dues à plusieurs causes, causes pas nécessairement malveillantes. Il peut tout d'abord s'agir d'*erreurs* de mesure ou de collecte des données. En effet, une erreur dans le processus de mesure, de recueil ou d'enregistrement des données peut entraîner des anomalies. Autre cause probable, un *changement* ou une *évolution* dans le système observé peut entraîner une variation. Par exemple, un changement dans le comportement des consommateurs peut entraîner des anomalies dans des données de vente. Dernière cause possible, une anomalie peut indiquer une *fraude* ou une *activité malveillante*. Les anomalies indiquent dans ce cas des comportements déviants ou des cyberattaques.

1.2.3 Importance de la détection

La détection de ces anomalies a toujours été un sujet de recherches. Plusieurs méthodes basées sur différentes approches ou techniques ont été et sont toujours explorées. L'approche traditionnelle utilise des *techniques statistiques* pour modéliser ce qui est considéré comme normal et ensuite tester les observations par rapport à ce modèle. Ont été ensuite développées des méthodes basées sur la *proximité*. Des techniques telles que l'analyse des k-plus proches voisins (k-NN) peuvent identifier les points de données qui sont inhabituellement éloignés de leurs voisins. D'autres méthodes comparent la *densité* autour d'une observation par rapport à la densité de ses voisins. Un point avec une densité significativement différente peut être considéré comme une anomalie. Des approches utilisent le *clustering* pour regrouper les données similaires et former des ensembles ou 'clusters'. Les points qui ne s'intègrent pas dans un cluster sont considérés comme étant des anomalies. Enfin, ces dernières années ont vu émerger des méthodes s'appuyant sur l'*apprentissage automatique*. Des modèles comme les forêts d'isolation ou les réseaux de neurones sont entraînés pour détecter des anomalies particulières.

Ce domaine d'études est très important dans le traitement et l'analyse des données, qui trouve des applications dans presque tous les secteurs, de la finance à la santé en passant par l'ingénierie et la sécurité informatique. En effet, identifier les anomalies peut aider à prévenir les fraudes, à réduire les coûts opérationnels ou les pertes financières, par exemple en détectant les transactions financières suspectes. Elles peuvent également fournir des indices sur de nouveaux phénomènes ou sur des erreurs dans une nouvelle théorie ce qui peut aider à mieux les comprendre ou les modéliser. Enfin et surtout, dans les systèmes d'information, la détection des anomalies est cruciale pour identifier les menaces potentielles et donc améliorer la sécurité.

1.3 Anomalies réseaux

La détection d'anomalies réseau fait référence aux méthodes et processus utilisés pour identifier des comportements inhabituels ou suspects sur un réseau informatique.

1.3.1 Définition

Une *anomalie réseau* peut être définie comme une activité ou un événement qui ne correspond pas au comportement de trafic attendu ou établi. Elles peuvent indiquer une variété d'événements

tels que des défaillances techniques (panne d'un équipement réseau), des congestions de trafic, des erreurs opérationnelles (mauvaise configuration par exemple) ou des variations de l'activité. Cela inclut également le trafic dû à des cyberattaques (intrusions, malwares, phishing, DDoS, exfiltration de données,...). À mesure que les réseaux deviennent plus complexes et que les attaques deviennent plus sophistiquées, les techniques de détection doivent constamment être améliorées.

Dans le domaine de la détection des anomalies réseaux, nous trouvons également les notions de *trafic normal* et *trafic anormal*. La distinction entre trafic normal et anormal est essentielle. Le trafic normal est généralement défini par une ligne de base statistique ou un modèle de comportement prévu, alors que le trafic anormal dévie de cette norme.

1.3.2 Méthodes de détection

Il existe plusieurs méthodes de détection car basées sur des approches différentes, souvent complémentaires, utilisant diverses techniques.

L'approche par *détection de signatures* consiste à comparer le trafic réseau à des signatures connues d'attaques ou de malwares. C'est une méthode très efficace capable de détecter des menaces connues. Par contre, elle s'avère inadaptée pour ce qui est de la détection de nouvelles attaques (Zero-Day) ou des variantes d'attaques déjà existantes pour lesquelles la signature est inconnue.

L'approche dite *statistique* utilise des seuils statistiques pour identifier les écarts par rapport à la norme. Différentes techniques sont mises en œuvre comme l'analyse des séries temporelles, les modèles de régression et les tests d'hypothèse. Des modèles de trafic normaux sont établis et les écarts par rapport à ces modèles sont signalés comme des anomalies. Ces méthodes sont capables de détecter des menaces inconnues, mais elles peuvent produire plus de faux positifs.

Une dernière approche, plus récente devenue très populaire, consiste à exploiter la puissance des *algorithmes d'apprentissage automatique*. En effet, l'intelligence artificielle en général et le machine learning en particulier jouent un rôle croissant dans la détection d'anomalies réseau. S'agissant de l'apprentissage *supervisé*, des algorithmes sont entraînés avec des données marquées comme normales ou anormales. Cette approche est très efficace, mais nécessite un grand nombre de données différentes et correctement labellisées pour être efficace. De plus, les modèles doivent être maintenus à jour pour qu'ils soient pertinents. Dans le cadre de l'apprentissage *non supervisé*, des algorithmes de

regroupement (clustering) ou de classification tentent de trouver des structures ou des caractéristiques sous-jacentes dans des données non labellisées afin de modéliser le trafic et identifier des anomalies en détectant des groupes de données inhabituels ou reconnaître des classes d'activité réseau.

1.3.3 Solutions & Défis

Des Systèmes de Détection ou de Prévention d'Intrusions (IDS/IPS) comme `Snort` ou `Suricata` utilisent à la fois des méthodes de détection basées sur la signature et la statistique. D'autres outils comme `NetFlow` ou `sFlow` capturent des méta-données sur le trafic réseau pour analyse ultérieure. Une solution plus globale est le déploiement dans le système d'information d'un SIEM [1]. Celui-ci permet de collecter, normaliser et analyser les données de sécurité (journaux systèmes, applicatifs ou remontées de sondes) à travers un réseau pour détecter des activités suspectes.

Un grand défi est de réduire le nombre de faux positifs sans augmenter les faux négatifs, afin de ne pas surcharger les équipes de sécurité avec des alertes inutiles. Il faut pouvoir équilibrer la sensibilité de la détection sans générer trop de fausses alertes ce qui est une tâche difficile. Avec l'augmentation du volume de trafic réseau et leur vitesse de transit, il devient de plus en plus difficile d'analyser les données en temps réel. En outre, les menaces évoluent et les attaquants modifient constamment leurs techniques, rendant les méthodes de détection obsolètes rapidement.

Dans cette optique, nos contributions ; décrites dans les chapitres 3 et 4 ; proposent respectivement un métamodèle capable de mettre en lumière le trafic anomal dû à l'activité malveillante de botnets, un framework permettant de détecter des variations de l'activité réseaux liées aux changements de comportement des utilisateurs suite à des événement extérieurs ou des pannes.

1.3.4 Perspectives

Avec la démocratisation d'outils comme `TensorFlow` et la mise à disposition de ressources de calcul ou stockage, l'utilisation de réseaux de neurones profonds pour détecter des schémas complexes et subtils dans les données de trafic réseau se généralise.

Une autre tendance actuelle est de fournir une réponse *automatisée et en temps réel* aux anomalies détectées pour accélérer la mitigation des menaces potentielles. Notre dernière contribution, détaillée dans la chapitre 5, a pour objectifs de répondre à ces deux dernières exigences.

1. Security Information and Event Management

1.4 Événements rares

Bien qu'en dehors du cadre de nos travaux, dans cette section 1.4, nous nous intéressons à un sous-ensemble des anomalies, à savoir les événements rares qui est un domaine d'étude à part entière.

1.4.1 Définition & Théories sous-jacentes

La théorie des événements rares est intrinsèquement liée à la gestion des risques et à la prise de décision dans des conditions d'incertitude. Il s'agit d'un domaine spécialisé des mathématiques et de la statistique qui étudie la probabilité et le comportement des événements qui surviennent avec une très faible fréquence, mais qui peuvent avoir des conséquences significatives ou catastrophiques lorsqu'ils se produisent. Ces événements rares peuvent inclure des catastrophes naturelles comme les tremblements de terre et les inondations, des incidents financiers tels que les krachs boursiers, des pannes dans des systèmes complexes comme les centrales nucléaires ou des activités suspectes dans les réseaux informatiques.

Il existe deux grandes théories sous-jacentes à ce domaine d'étude. Nous trouvons tout d'abord, la *théorie des valeurs extrêmes* (EVT). Elle est employée pour modéliser la probabilité des événements extrêmes [9]. Elle s'appuie sur des lois de probabilité qui ne suivent pas les modèles gaussiens habituels.

L'EVT est divisée en deux approches principales que sont l'approche dite du *bloc maxima* (ou minima), où on s'intéresse aux maxima (ou minima) d'un échantillon sur une période donnée et l'approche du *seuil excédentaire* qui elle se concentre sur les données qui dépassent un certain seuil.

La seconde est la *théorie des grandes déviations* ("Large Deviations Theory" ou LDT) qui fournit des outils pour évaluer la probabilité de déviations importantes par rapport à la moyenne ou à l'état typique d'un système. Elle est particulièrement utile dans le contexte des processus stochastiques où l'on cherche à estimer la probabilité de trajectoires improbables.

1.4.2 Applications & Défis

La théorie des événements rares est appliquée dans de nombreux domaines. En ingénierie, elle aide à la conception de systèmes capables de résister à des conditions opérationnelles anormales. Les événements rares sont également importants dans l'analyse de la fiabilité des réseaux et de la qualité de service, où l'on s'intéresse par exemple à la probabilité de congestion.

Comme les événements rares sont par définition difficiles à observer dans les données réelles ou expérimentales, les techniques de simulation sont cruciales pour leur étude comme le Monte Carlo d'échantillonnage d'importance ou les méthodes de changement de mesure qui permettent de concentrer l'effort de simulation sur les régions de l'espace d'état où les événements rares sont susceptibles de se produire. La simulation *Monte Carlo* est une méthode utilisée pour estimer la probabilité d'événements rares en générant un grand nombre de scénarios. Les techniques de Monte Carlo par chaînes de *Markov* (MCMC) et d'échantillonnage d'importance ont été développées pour améliorer l'efficacité des estimations dans le cas d'événements rares.

1.5 Formulation du problème & Contributions

N'ayant pas la possibilité de simuler ou de modéliser des événements rares, nous avons choisi d'employer des jeux de données existants et donc d'orienter nos recherches vers la détection d'anomalies réseaux en général. Dans l'idée de proposer des approches ou des méthodologies génériques, automatisables, voire capables de détecter des anomalies en temps réel, nous nous sommes tournés vers l'apprentissage automatique pour ses capacités de généralisation et de mise en œuvre dans des environnements réels. Comme exposé dans la section 1.2, ces anomalies peuvent être dues à différentes causes que nous pourrions qualifier d'*internes* ou d'*externes* au réseau supervisé. Par conséquent, dans le cadre de nos travaux de recherche, nous avons apporté trois contributions principales.

Nous avons tout d'abord souhaité détecter des anomalies liées à une cause *interne*, à savoir des cyberattaques. Pour ce faire, nous nous sommes donnés pour objectif de proposer un modèle capable de détecter l'activité maligne de différents botnets et donc différentes attaques notamment 'DDoS', 'Spams' ou encore 'Click Fraud', attaques expliquées dans la section 2.1. Pour ce faire, nous avons construit un métamodèle *performant* et *évolutif* baptisé "Forêt combinée" à l'aide de différents algorithmes d'apprentissage supervisé.

Notre deuxième contribution est un "Framework de détection d'anomalies par signature numérique" que nous avons voulu *générique, cyclique et fractal*, basé sur l'apprentissage automatique non supervisé et notre concept des *signatures numériques*. Il permet de détecter des anomalies dues à des causes *externes* ayant un impact sur le comportement des utilisateurs et donc le trafic réseau.

Notre dernière contribution, la méthodologie "DiNATrA$\mathcal{X}$" basée sur notre concept de DNA, est une évolution du framework précédent permettant de détecter des anomalies réseaux mais cette fois de façon *automatisée*. Méthodologie qui pourrait, par conséquent, être mise en œuvre en *temps réel*.

1.6 Organisation du document

Ce document est organisé autour de sept chapitres. Le chapitre 1 introduit les notions d'anomalie, d'anomalies réseaux, la théorie des événements rares et le contexte dans lequel s'inscrivent nos contributions à leur détection.

Nous proposons ensuite, dans le chapitre 2, un état de l'art au sujet des différentes attaques réseaux qui sont très souvent à l'origine de ces anomalies, sur les diverses méthodes de détection existantes ainsi que sur l'apprentissage automatique qui est devenu un des outils les plus utilisés car particulièrement efficace. Nous décrivons également dans ce chapitre les deux jeux de données employés pour mener nos recherches.

Notre première contribution, la "Forêt combinée", est présentée et expliquée dans le chapitre 3. Puis, nous détaillons dans le chapitre 4 notre "Framework de détection d'anomalies par signature numérique". Le chapitre 5 est une description fonctionnelle détaillée de notre méthodologie "DiNATrA$\mathcal{X}$". Celle-ci a été testée et validée en ayant été mise en œuvre avec deux jeux de données totalement différents. Ces deux implémentations et l'ensemble des résultats obtenus sont décrits dans le chapitre 6.

Enfin, le chapitre 7 conclut ce document en synthétisant l'ensemble de nos contributions et propose des pistes de recherches alternatives qui pourraient être suivies afin d'améliorer notre métamodèle ainsi que notre méthodologie voire même pour l'employer à d'autres fins.

Chapitre 2

État de l'art

Dans la section 2.1, nous commençons par détailler et expliquer les différents types d'attaque véhiculées via les réseaux informatiques. Puis dans la section 2.2, nous présentons quelques-unes des cyberattaques s'étant déroulées au cours des vingt dernières années et qui, malheureusement, sont devenues célèbres. Ensuite, dans la section 2.3, nous proposons un état des lieux de la menace informatique, les différentes méthodes permettant de les détecter et de les prévenir ainsi que l'apport, dans cette lutte quotidienne, de l'apprentissage automatique. Celui-ci est devenu un outil très puissant de détection des anomalies en général et un remède très efficace contre les cybermenaces en particulier. Les sections 2.4 à 2.5 brossent les différentes approches inhérentes à l'apprentissage automatique. Dans le cadre de nos travaux de recherche, nous avons utilisé deux jeux de données, un public et un propriétaire, que nous détaillons dans les sections 2.6 et 2.7. La section 2.8 synthétise cet état de l'art et explique les différents choix que nous avons été amenés à faire.

2.1 Attaques réseaux

2.1.1 DoS & DDoS

Déni de Service (DoS) : L'objectif est de rendre une ressource informatique (serveur, site web, service en ligne) indisponible pour les utilisateurs auxquels elle est destinée. Celle-ci est souvent accomplie en surchargeant la ressource avec un flot de paquets de données ou en ouvrant un grand nombre de connexions, la submergeant au point qu'elle ne peut répondre aux requêtes légitimes.

Déni de Service Distribué (DDoS) : Il s'agit d'une forme avancée d'attaque 'DoS' impliquant plusieurs systèmes compromis, souvent un 'botnet' (voir section 3.1.2), qui sont utilisés pour lancer l'attaque de surcharge [10]. Ces attaques sont plus difficiles à mitiger, car elles proviennent de plusieurs sources.

2.1.2 Click Fraud

Dans le cadre de la publicité en ligne, la fraude par clic se réfère à l'action de cliquer de manière répétitive sur une publicité en ligne avec l'intention de générer une visibilité non méritée pour l'annonceur [11]. Cela peut être réalisé manuellement par des personnes, ou plus couramment, par des bots ou des logiciels automatiques. Les motivations peuvent inclure l'augmentation des revenus pour les sites hébergeant les publicités ou l'épuisement des budgets publicitaires des concurrents.

2.1.3 Ransomware

Il s'agit ici d'un type de malware qui chiffre les fichiers de l'utilisateur ou des systèmes informatiques, empêchant l'accès aux données jusqu'à ce qu'une rançon soit payée pour obtenir la clef de déchiffrement [12]. Même après paiement de la rançon, il n'y a aucune garantie que les utilisateurs récupéreront l'accès à leurs données. Des attaques ont été dues à des ransomwares célèbres comme `WannaCry`.

2.1.4 Pourriel ou Spam

Une campagne de spam consiste en l'envoi massif et répété de messages non sollicités, généralement à des fins publicitaires ou malveillantes, à un grand nombre de destinataires via le courrier électronique, mais aussi à travers d'autres moyens de communication numérique comme les messages textes (SMS), les réseaux sociaux ou les forums en ligne [13]. Les objectifs et les méthodes des campagnes de spam peuvent varier considérablement :

Objectifs : Ces campagnes de pourriels visent essentiellement quatre objectifs qui sont de : *(i)* promouvoir des produits ou services, souvent de manière agressive ou trompeuse (publicité commerciale) *(ii)* inciter les destinataires à divulguer des informations personnelles ou financières (hameçonnage ou phishing) ou à participer à des escroqueries comme les arnaques à l'avance de frais (arnaques et fraudes) *(iii)* diffuser des virus, des chevaux de Troie, des ransomwares ou d'autres logiciels malveillants pour compromettre les ordinateurs des destinataires (propagation de logiciels malveillants) *(iv)* améliorer

artificiellement le classement d'un site web dans les résultats de recherche en générant des liens entrants via des commentaires de blog ou des messages de forum non sollicités (Search Engine Optimization).

Méthodes employées : Ces campagnes sont automatisées soit via l'utilisation de bots et de scripts pour envoyer des messages à grande échelle soit par l'emploi de réseaux d'ordinateurs infectés (botnets) pour distribuer le spam, rendant ainsi plus difficile le traçage de l'origine des messages. D'autres approches consistent à recueillir ou acheter des listes d'adresses email, de numéros de téléphone, ou d'autres identifiants pour cibler les destinataires tout en falsifiant les adresses d'expéditeur via des techniques de masquage afin d'éviter la détection par les filtres anti-spam. Ces filtres peuvent également être contournés grâce à des techniques de dissimulation dans le contenu des messages (comme modifier l'orthographe des mots clefs) pour en éviter la détection.

Conséquences : Ces campagnes de pourriels, outre le fait de piéger leurs destinataires, ont un impact important sur les systèmes d'information qui les subissent. Les principales conséquences sont : *(i)* la surcharge des systèmes de messagerie ce qui peut ralentir ou perturber les services de messagerie pour les utilisateurs et les fournisseurs *(ii)* une augmentation du risque d'exposition à des logiciels malveillants et aux fraudes *(iii)* la perte de productivité car les destinataires perdent du temps à trier et à supprimer des messages non sollicités *(iv)* la détérioration de la confiance ce qui peut nuire à la réputation des entreprises légitimes associées, même involontairement, à ces campagnes.

Les campagnes continuent d'évoluer avec de nouvelles techniques pour contourner les mesures de sécurité et de filtrage, ce qui nécessite une vigilance constante de la part des utilisateurs et des fournisseurs de services en ligne.

Campagnes majeures : Il y a eu plusieurs campagnes de pourriels célèbres au fil des ans, chacune marquante pour différentes raisons, soit de part l'ampleur de leur diffusion ou de leur impact et des dégâts causés soit du fait de leur ingéniosité technique. Voici quelques exemples notables :

ILOVEYOU (2000) Connu aussi sous le nom de "Love Bug", ce spam était un ver informatique qui s'est propagé via un courriel [14]. Il se présentait sous la forme d'une lettre d'amour d'un admirateur secret. Lorsque l'utilisateur ouvrait la pièce jointe, le ver se copiait sur les lecteurs de l'ordinateur et s'envoyait à tous les contacts dans le carnet d'adresses de l'utilisateur.

Pump and Dump (2000) Des spams boursiers encourageaient les destinataires à investir dans certaines actions à bas prix [15]. Après que suffisamment d'investisseurs aient acheté les actions et fait monter leur prix, les escrocs vendaient leurs actions à un prix élevé, entraînant une chute des prix et des pertes pour les autres investisseurs.

Spam pharmaceutique (2005) Une grande quantité de spam était consacrée à la vente de médicaments, souvent des contrefaçons ou des médicaments vendus sans ordonnance [16]. Ces spams ont souvent été associés à des pharmacies en ligne douteuses.

Storm Worm (2007) Ce ver a été diffusé via des courriels contenant des titres provocateurs comme "230 morts dans la tempête en Europe" [17]. Lorsqu'un utilisateur ouvrait la pièce jointe, son ordinateur était infecté et ajouté à un botnet.

Nigerian Scam ou "419 scam" Bien que pas strictement un spam, cette escroquerie par courriel est très célèbre [18]. Elle implique un prétendu fonctionnaire ou membre de la royauté nigériane offrant une part importante d'une somme d'argent en échange d'une petite avance pour couvrir les frais administratifs. '419' fait référence au numéro de l'article du code nigérian sanctionnant ce type de fraude.

2.1.5 Cross-Site Request Forgery

L'attaque CSRF est une vulnérabilité de sécurité web qui permet à un attaquant de forcer l'envoi de requêtes non autorisées par un utilisateur authentifié vers une application web à laquelle il est actuellement connecté [19].

Fonctionnement : Par exemple, étant connecté au site d'une banque en ligne, dans un autre onglet, je navigue sur un autre site. Si ce site contient un script malveillant (`JavaScript`), il pourrait, à mon insu, envoyer une requête à ma banque pour effectuer une action (comme un virement d'argent) en utilisant mes informations d'identification de session actuellement actives. Puisque la banque pense que la requête provient d'un utilisateur authentifié (moi), elle exécutera la commande.

Exemple : Un utilisateur se connecte à un site web (comme une banque), et sans se déconnecter, navigue sur un autre site malveillant. Ce site malveillant peut exécuter une action telle que :
<img src="https ://bank.com/transfer.do ?amount=1000&destination=attacker">

qui pourrait, théoriquement, déclencher un transfert d'argent si le site de la banque n'est pas protégé.

Pourquoi c'est dangereux ? L'attaque CSRF exploite la confiance qu'un site a envers l'ordinateur de l'utilisateur. Elle peut être utilisée pour changer l'adresse email associée à un compte, changer le mot de passe, ou même transférer des fonds, selon le site web ciblé. L'utilisateur n'a généralement pas conscience de l'attaque, ce qui la rend particulièrement insidieuse.

Préventions : Il existe plusieurs méthodes pour prévenir ce type d'attaque mais les applications web doivent être conçues avec une protection contre les attaques CSRF dès le départ. Les développeurs doivent être conscients de cette menace et implémenter des mécanismes de défense comme ceux décrits ci-dessous pour protéger les utilisateurs.

Jeton anti-CSRF La méthode la plus courante pour prévenir les attaques CSRF est l'utilisation de jetons (tokens) anti-CSRF. Ces tokens sont des valeurs uniques et imprévisibles qui sont générées par le serveur et doivent être incluses dans chaque requête soumise. Si la requête ne contient pas le token, ou si le token ne correspond pas à celui attendu par le serveur, la requête est rejetée.

Entête "Referer" Les serveurs peuvent également vérifier l'en-tête "Referer" des requêtes HTTP pour s'assurer qu'elles proviennent de la même origine.

Attribut "SameSite" pour les cookies Cet attribut peut être utilisé pour contrôler si un cookie doit être envoyé avec les requêtes cross-origin. En réglant l'attribut sur 'Strict' ou 'Lax', il peut empêcher les cookies d'être envoyés lors des requêtes inter-sites, ce qui réduit le risque d'exposition.

Double soumission de cookie Une variante de l'utilisation de tokens, où le token est stocké à la fois dans un cookie et dans un paramètre de requête, assurant que seul un utilisateur ayant accès à ce cookie spécifique peut faire une requête valide.

2.1.6 Autres types d'attaques

Phishing : L'attaquant envoie des courriels semblant provenir de sources fiables pour inciter les utilisateurs à révéler des informations confidentielles, telles que des mots de passe et des détails de cartes de crédit [20].

Man-in-the-Middle (MitM) : L'attaquant intercepte la communication entre deux parties sans que celles-ci s'en rendent compte pour modifier les échanges ou voler des informations.

Injection SQL : Une attaque qui vise les bases de données en injectant des commandes SQL malveillantes à travers, par exemple, des formulaires web, pour lire ou modifier les données de la base.

Cross-Site Scripting (XSS) : L'attaquant injecte du code `JavaScript` malveillant dans les pages web. Lorsqu'un utilisateur visite la page web, le script est exécuté et peut accéder aux cookies, créer des requêtes à l'insu de l'utilisateur,...

Attaque par force brute : Elle consiste à tenter de deviner un mot de passe ou une clef de chiffrement en essayant systématiquement toutes les combinaisons possibles jusqu'à ce que la combinaison correcte soit trouvée.

Attaque par ingénierie sociale : Celle-ci implique la manipulation des gens pour qu'ils divulguent des informations confidentielles. Ce n'est pas nécessairement technologique, mais c'est souvent le prélude à des attaques plus sophistiquées techniquement.

Eavesdropping (écoute clandestine) : L'attaquant écoute secrètement les communications privées, souvent réalisées par des connexions non sécurisées, pour recueillir des informations sensibles.

2.2 Quelques attaques célèbres

Nous avons choisi de présenter ci-dessous plusieurs attaques réseaux "célèbres" car ayant marqué l'histoire de la cybersécurité par leur ampleur ou leur ingéniosité.

2.2.1 MyDoom (2004)

Cette attaque a eu lieu en 2004. Considéré comme l'un des vers informatiques à la croissance la plus rapide, `MyDoom` s'est propagé principalement via du pourrriel [21]. Il ouvrait une porte dérobée dans les systèmes infectés permettant aux attaquants d'accéder à distance aux ordinateurs infectés.

Le plus souvent, les utilisateurs ne se rendaient pas compte que leur ordinateur était infecté. Ils remarquaient tout au plus le ralentissement du système ou des interruptions de service intermittentes, mais ne recevaient probablement pas d'alerte ou d'avertissement signalant un problème de fonctionnement de leur ordinateur. Pendant ce temps le code, agissant silencieusement dans l'environnement `Windows`, permettait au ver de se propager.

L'attaque `MyDoom` se déroule en quatre étapes : 1. Téléchargement : L'ouverture de la pièce jointe permet au code de se déplacer dans l'environnement `Windows`. Aucun autre environnement n'a été touché. 2. Propagation : Le code puise dans les contacts enregistrés sur l'ordinateur de la victime. Chaque adresse trouvée reçoit une nouvelle version du ver en tant que pièce jointe à un message. 3. Exécution : À une date fixée, les ordinateurs infectés envoient des demandes au site web de `SCO Group` ou à celui de `Microsoft`. 4. Persistance : Les attaquants laissent une porte dérobée (backdoor) ouverte, dans l'éventualité de leur retour.

2.2.2 Stuxnet (2010)

`Stuxnet`, découvert en 2010, est un exemple précoce de cyberarme [22]. Il s'agissait d'un ver informatique complexe qui a été conçu spécifiquement pour cibler les systèmes de contrôle industriel utilisés dans les centrales nucléaires iraniennes. Il a causé des dommages importants aux centrifugeuses iraniennes utilisées dans l'enrichissement de l'uranium, retardant ainsi le programme nucléaire iranien.

2.2.3 Mirai (octobre 2016)

L'attaque `Mirai` a eu lieu en octobre 2016 et est devenue célèbre pour avoir utilisé un botnet constitué de dispositifs IoT (Internet des Objets) infectés [23]. Ces dispositifs, tels que des caméras de surveillance et des routeurs, ont été transformés en un botnet qui a ensuite été utilisé pour lancer une série d'attaques par déni de service distribué (DDoS) massives. La plus notable a été contre le fournisseur de `DNS Dyn`, ce qui a entraîné l'indisponibilité de sites comme `Twitter` ou `Netflix`.

2.2.4 WannaCry (mai 2017)

Une attaque célèbre basée sur l'exploitation d'une faille de sécurité est le ransomware `WannaCry` [24]. En mai 2017, `WannaCry` a infecté des centaines de milliers d'ordinateurs dans plus de 150 pays. Ce logiciel malveillant exploitait une faille dans les systèmes `Windows`, initialement découverte par la

NSA américaine, puis divulguée par le groupe de hackers `Shadow Brokers`. `WannaCry` verrouillait les fichiers des utilisateurs et demandait une rançon en 'Bitcoin' pour les déverrouiller comme présenté par la figure 2.1. Cette attaque a causé des perturbations majeures et a souligné l'importance de la mise à jour régulière des systèmes de sécurité informatique. Le virus a touché de nombreuses entreprises et de nombreux organismes, mais on en a surtout parlé car il a eu un impact important sur les hôpitaux, notamment au Royaume-Uni [25] (NHS) mais aussi en France.

FIG. 2.1. Aperçu de WannaCry – L'utilisateur doit payer la rançon

2.2.5 NotPetya (juin 2017)

`NotPetya`, qui a frappé en juin 2017, est souvent considéré comme l'une des cyberattaques les plus dévastatrices de l'histoire [26]. Initialement considéré comme un ransomware, `NotPetya` semblait plus intéressé par la destruction des systèmes que par la collecte de rançons. Il a utilisé une variété de méthodes pour se propager à travers les réseaux d'entreprise et a causé des dommages importants, surtout à des entreprises en Ukraine et à de grandes multinationales.

2.2.6 Equifax (2017)

`Equifax` est une agence privée qui évalue la cote de crédit des individus et entreprises aux États-Unis. Elle analyse différents paramètres, comme les crédits déjà en cours, pour analyser si la personne est capable de rembourser de nouvelles dettes ou non. En somme, `Equifax` agrège des informations privées, les analyse et crée ensuite une note qu'elle revend. Si vous avez une bonne note chez `Equifax`, vous pouvez souscrire à un crédit. Elle fût, en 2017, victime de l'une des plus grandes violations de données dans l'histoire, affectant environ 147 millions de consommateurs [27]. Des pirates informatiques ont exploité une faille de sécurité dans une application web pour accéder à des données sensibles, y compris des numéros de sécurité sociale, dates de naissance, adresses,. . ..

2.2.7 DDoS sur GitHub (2018)

En 2018, `GitHub` a subi la plus grande attaque DDoS enregistrée à l'époque, atteignant un pic de trafic de 1.35 Tbps [28]. Cette attaque a utilisé une vulnérabilité des serveurs `Memcached` exposés à Internet pour amplifier massivement le volume de données envoyé à la cible.

2.2.8 SolarWinds (2020)

L'attaque `SolarWinds`, découverte fin 2020, a impliqué un compromis de la chaîne d'approvisionnement logicielle qui a affecté des dizaines de milliers de clients de `SolarWinds` [29]. Des acteurs étatiques présumés ont inséré une porte dérobée (backdoor) dans le logiciel de gestion réseau `Orion` de `SolarWinds`, ce qui leur a permis d'accéder à des réseaux sécurisés.

2.2.9 Viamedis & Almerys (2024)

Deux sociétés ,`Viamedis` et `Almerys`, intermédiaires entre les professionnels de santé et les complémentaires santé, ont été la cible d'une cyberattaque. Plus de 33 millions de Français ont été victimes d'un vol de données [30]. La CNIL [1] a précisé que les données concernées par cette vaste cyberattaque sont notamment les informations sur l'état civil, la date de naissance, le numéro de sécurité sociale, le nom de l'assureur santé et les garanties du contrat souscrit.

1. Commission Nationale de l'Informatique et des Libertés

L'attaque s'est produite après l'usurpation d'identifiants et mots de passe de professionnels de santé. L'alerte a été donnée le 1er février par la société `Viamedis`, qui a détecté l'attaque et averti les autres plateformes de tiers payant. Son directeur général a expliqué qu'il ne s'agissait pas d'une attaque par rançongiciel mais d'une intrusion dans la plateforme via le compte d'un professionnel de santé qui a été hameçonné. Quelques jours plus tard, *Almerys* a annoncé avoir également détecté une intrusion [31], [32].

2.2.10 Anonymous Sudan (2024)

Des hackers pro-russes et islamistes ont mené une cyberattaque contre le réseau informatique entre les ministères. Les agents de la fonction publique ont rencontré des perturbations [8]. Le 11 mars 2024, le RIE [2] a fait l'objet d'une attaque DDoS (voir section 2.1.1) lancée par le groupe `Anonymous Sudan`. La chaîne `Telegram` d'`Anonymous Sudan` revendique des perturbations au sein du ministère de la Culture, de la santé et des affaires sociales, de l'Économie et des finances, de la Transition écologique et les services du premier ministre comme illustré par la figure 2.2. En effet, le RIE raccorde l'ensemble des services de l'État sur le territoire national. Il assure, pour ces services, le transport des flux internes aux entités, les échanges sécurisés entre entités, ainsi que les échanges sécurisés avec les réseaux tiers, notamment Internet et le réseau Inter-États membres de la Communauté européenne.

FIG. 2.2. Cyberattaque contre le RIE

2. Réseau Interministériel de l'État

2.3 Prévention des attaques & Détection

2.3.1 Évolution des menaces

Nous proposons dans cette section 2.3 un aperçu des statistiques et tendances concernant les différents types d'attaques informatiques, issues de divers rapports de sécurité et d'analyses de l'industrie. Il est important de noter que ces chiffres évoluent constamment et peuvent varier d'une source à l'autre.

Les attaques DDoS ont continué d'augmenter en fréquence, en complexité et en volume. `Akamai` a rapporté une augmentation de 57 % des attaques DDoS d'une année sur l'autre dans son rapport de sécurité de 2023 [33]. De plus, `Akamai` a également relevé dans son rapport "State of the Internet / Security" plus de 193 milliards d'attaques par force brute en 2023, soit une moyenne de plus de 3,4 milliards d'attaques par semaine.

Selon le rapport "State of Ransomware 2023" de `Sophos`, environ 37 % des organisations à travers le monde ont été touchées par un ransomware en 2022 [34]. Le coût moyen du rétablissement après une attaque de ransomware a plus que doublé en un an, passant de 761 106 dollars en 2022 à 1,85 million de dollars en 2023.

En 2023, `Malwarebytes` a rapporté une augmentation des détections de malware de 68 % sur les ordinateurs de bureau et une augmentation de 15 % sur les appareils mobiles pour atteindre 4 475 attaques [35]. Le nombre total de nouveaux malwares détectés a diminué en 2023, mais leurs variantes sont devenues plus sophistiquées.

Le nombre d'attaques contre les dispositifs IoT a continué de croître, avec `Kaspersky` détectant 1,5 milliard d'attaques sur des appareils IoT en la première moitié de 2021, un chiffre en augmentation par rapport à l'année précédente [36].

Enfin, le rapport "2022 Phishing Insights" de `PhishLabs` a indiqué que les attaques de phishing ont augmenté de 28 % en 2021 par rapport à 2020 [37]. `Microsoft` a signalé que les tentatives d'hameçonnage par courriel représentaient 70 % des attaques observées dans ses divers services.

Ces statistiques soulignent l'importance de rester vigilants et d'adopter de bonnes pratiques de sécurité pour se prémunir de ces menaces en constante évolution.

2.3.2 Mesures de prévention

Après cet aperçu des attaques par protocole et des plus courantes, il est essentiel de se rappeler que le paysage des menaces évolue constamment. Chacune de ces attaques a souligné l'importance de la cybersécurité et a incité entreprises et gouvernements à repenser leurs stratégies de défense contre les cybermenaces. Celles-ci peuvent être prévenues ou atténuées par une combinaison de mesures de sécurité techniques et de formation à la sensibilisation des utilisateurs.

Les principales mesures de prévention sont : *(i)* l'utilisation de pare-feux, de sondes IDS/IPS, de logiciels antivirus et anti-malware et les filtres anti-spam *(ii)* la mise à jour régulière des logiciels et du firmware des équipements *(iii)* la surveillance du réseau et détection d'anomalies *(iv)* l'éducation et sensibilisation des utilisateurs *(v)* les programmes de formation à la sécurité informatique et la vigilance constante.

2.3.3 Détection des attaques

Certaines attaques sont particulièrement difficiles à détecter en raison de leur discrétion, de leur complexité ou de leur capacité à imiter un comportement légitime. Les attaques suivantes sont réputées pour être parmi les plus difficiles à détecter.

Man-in-the-Middle (MitM) : Elles se produisent au sein de la communication légitime entre deux parties. Sans une surveillance ou une protection appropriée, comme le chiffrement TLS/SSL, il peut être très difficile pour les victimes de réaliser que leurs données sont interceptées.

Zero-Day : Ces attaques exploitent des vulnérabilités inconnues des fabricants de logiciels et des équipes de sécurité jusqu'à ce qu'elles soient exploitées, rendant leur détection extrêmement difficile. Les correctifs de sécurité ne sont pas disponibles avant que l'exploit ne soit découvert, offrant une fenêtre d'opportunité aux attaquants.

Malware polymorphique : Ces types de virus changent leur code à chaque infection ou peuvent se modifier eux-mêmes au fil du temps, rendant la détection par les signatures antivirus traditionnelles très difficile. Ils nécessitent des solutions de sécurité avancées, capables d'analyser les comportements plutôt que de se fier uniquement aux signatures.

Phishing et phishing ciblé (spear phishing) : Bien que les filtres anti-phishing et la sensibilisation puissent aider à réduire le risque, les attaques d'ingénierie sociale et le spear phishing sont difficiles à détecter, car elles exploitent la confiance humaine. Les e-mails de phishing ciblés peuvent être très convaincants et semblent provenir de sources légitimes.

La détection et la prévention de ces attaques nécessitent une combinaison de technologies, une veille sécuritaire continue, des analyses comportementales et une formation approfondie des utilisateurs pour reconnaître les tentatives d'ingénierie sociale. La complexité et la sophistication croissantes des menaces soulignent l'importance d'une approche multicouche en matière de cybersécurité.

2.3.4 Apports de l'apprentissage automatique

L'apprentissage automatique est devenu un outil puissant pour améliorer la détection et la prévention des attaques informatiques. Il s'agit d'un sous-ensemble de l'intelligence artificielle qui est devenu une composante incontournable de l'analyse de données. L'apprentissage automatique ou Machine Learning (ML) est une technique qui permet aux machines d'apprendre à partir de données et de s'améliorer avec l'expérience, sans être explicitement programmées pour chaque tâche.

Ces tâches d'apprentissage peuvent consister en la définition de la meilleure fonction de transfert pour faire correspondre l'entrée et la sortie, la découverte de la structure sous-jacente de données non étiquetées ou le regroupement d'échantillons de sorte que les objets d'un même groupe soient plus similaires les uns aux autres que les objets d'un autre groupe [38].

Le type le plus courant de processus d'apprentissage automatique consiste à apprendre la fonction de transfert : $Y = f(X) + e$ afin de prédire des Y pour de nouveaux X avec une erreur irréductible e en fonction du manque d'attributs permettant de caractériser la meilleure correspondance. C'est ce que l'on appelle la modélisation prédictive. L'objectif est ici de faire les prédictions les plus précises possibles, quelle que soit la forme de la fonction.

Différents algorithmes posent différentes hypothèses sur la forme et la structure de la fonction de transfert. Les hypothèses de départ peuvent grandement simplifier le processus d'apprentissage, mais peuvent également limiter ce qui peut être appris. C'est pourquoi il est très important d'essayer différents algorithmes sur un problème d'apprentissage automatique. La figure 2.3 est une représentation générale de l'organisation de ces différents algorithmes d'apprentissage automatique (MLA).

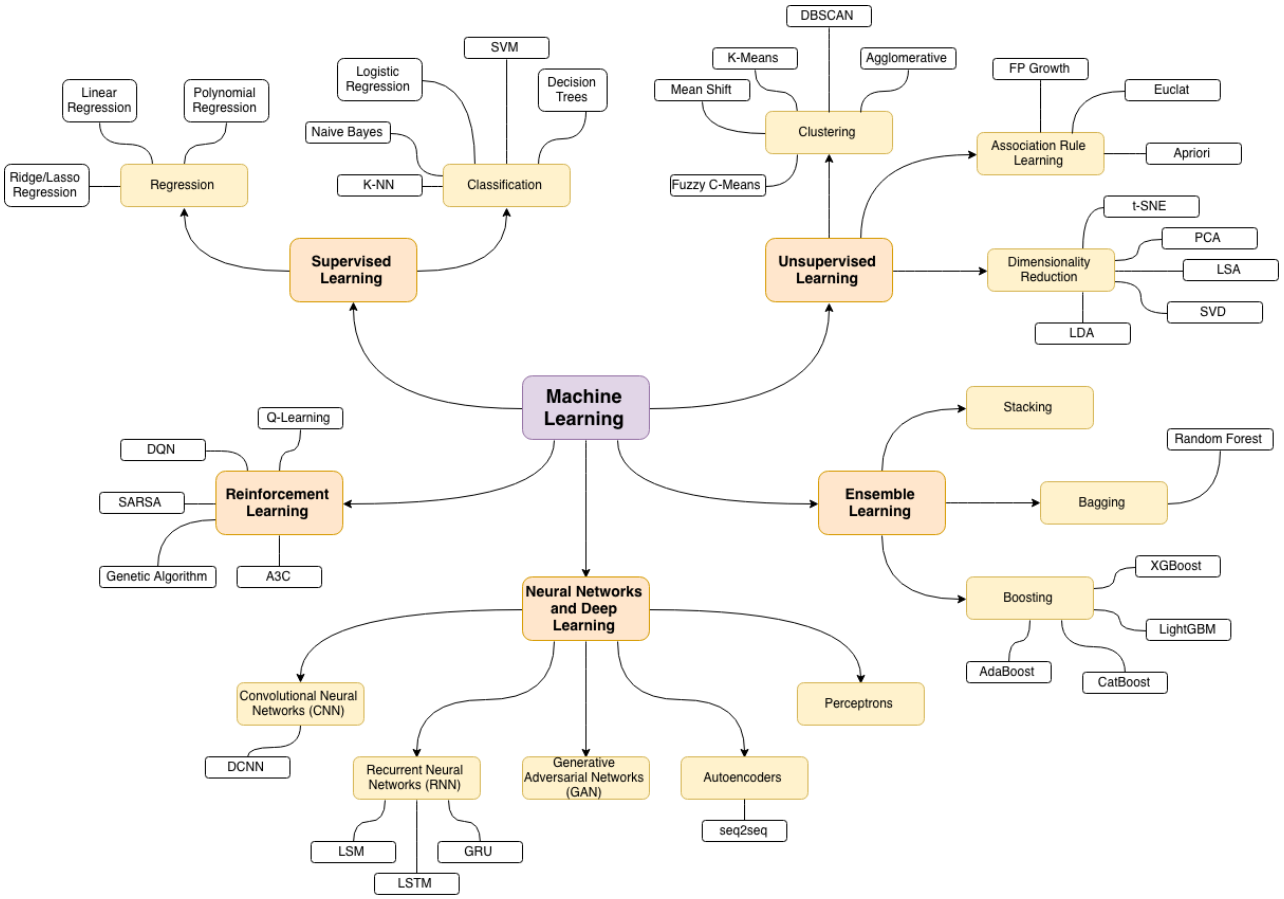


FIG. 2.3. Carte des algorithmes d'apprentissage automatique (MLAs)

En s'appuyant sur des algorithmes statistiques, les systèmes d'apprentissage automatique sont capables de reconnaître des modèles dans les données et de prendre des décisions avec un minimum d'intervention humaine. L'apprentissage automatique est généralement divisé en trois catégories ou approches principales :

1. *supervisé* où les algorithmes prédisent des résultats à partir de données *étiquetées ou labellisées*. Les applications typiques incluent la classification et la régression.
2. *non supervisé* lorsque les algorithmes découvrent des modèles cachés ou des structures dans des données *non étiquetées*. Les techniques communes comprennent le clustering et la réduction de dimensionnalité.
3. *par renforcement* avec des modèles qui apprennent à effectuer des tâches en essayant de *maximiser une récompense* donnée. Il est souvent utilisé dans les systèmes de navigation ou les jeux.

Grâce à sa capacité à apprendre de grandes quantités de données et à identifier des modèles complexes, le ML peut aider à détecter divers types d'attaques, y compris certaines qui sont difficiles à identifier avec les méthodes traditionnelles (IDS par exemple) comme celles listées ci-dessous.

MitM : Bien que plus difficiles à détecter, certains systèmes basés sur des algorithmes de ML peuvent identifier des anomalies dans le trafic réseau qui pourraient suggérer une interception ou une altération des données transmises.

Zero-Day : En analysant les comportements des applications et du réseau, les systèmes basés sur l'IA peuvent détecter des anomalies qui indiquent l'exploitation de vulnérabilités inconnues, même en l'absence de signatures ou de modèles connus.

Malware polymorphique : Les systèmes basés sur le ML peuvent être entraînés à reconnaître les signatures de malware ainsi que les comportements malveillants typiques de ces logiciels malveillants, ce qui les rend capables de détecter les variants polymorphiques qui échappent souvent aux antivirus traditionnels.

Phishing et spear phishing : Les algorithmes d'apprentissage automatique peuvent analyser les courriels pour détecter des signes de phishing, en examinant des éléments tels que le corps du message, les méta-données et les URL incluses. Ils peuvent apprendre à distinguer les caractéristiques des e-mails légitimes et des tentatives de phishing avec une précision élevée.

DoS et DDoS : L'apprentissage automatique peut aider à identifier les modèles de trafic anormal qui indiquent une attaque DoS ou DDoS en cours, permettant une intervention rapide pour mitiger l'attaque.

Remplissage d'identifiants ou brute force : Les modèles d'apprentissage automatique peuvent détecter les tentatives de connexion suspectes qui s'écartent des comportements normaux des utilisateurs, ce qui peut indiquer une attaque de remplissage d'identifiants.

Exfiltration ou vol de données : L'apprentissage automatique peut aider à identifier des modèles de trafic suspect qui indiquent une exfiltration de données, tels que des volumes inhabituels de téléchargements ou de transferts de données vers des destinations suspectes.

Anomalies et comportements réseaux suspects : En général, l'apprentissage automatique est très efficace pour détecter des comportements anormaux dans les réseaux qui pourraient indiquer une variété d'attaques, en se basant sur la surveillance continue du trafic et l'analyse comportementale.

L'un des principaux avantages de l'apprentissage automatique est sa capacité à s'adapter et à apprendre continuellement, améliorant ainsi sa précision et sa capacité à détecter de nouvelles menaces au fil du temps.

Toutefois, il est important de noter que les techniques basées sur l'apprentissage automatique ne sont pas infaillibles et doivent être utilisées en complément d'autres méthodes de sécurité pour offrir la meilleure protection possible contre les cyberattaques.

2.4 Approche supervisée

L'apprentissage automatique *supervisé* est un pilier essentiel de l'intelligence artificielle qui a révolutionné la manière dont nous analysons les données et en extrayons de la valeur. Ce domaine se concentre sur la construction de modèles qui peuvent faire des prédictions précises basées sur des ensembles de données contenant des entrées/sorties connues. Dans cette section 2.4, nous explorons les concepts fondamentaux de l'apprentissage supervisé et présentons les principaux algorithmes.

2.4.1 Principe

L'apprentissage supervisé utilise des ensembles de données labellisées, ce qui signifie que chaque exemple de l'ensemble de données d'entraînement est associé à une étiquette ou un résultat.

Le modèle est dit "supervisé" dans le sens où il apprend de ces exemples étiquetés pour pouvoir faire des prédictions sur des données non étiquetées. Les deux tâches principales de l'apprentissage supervisé sont la *classification* et la *régression*. La classification concerne les prédictions catégorielles, tandis que la régression est utilisée pour prédire des valeurs continues.

2.4.2 Principaux algorithmes

La *régression linéaire* est un des algorithmes les plus simples et les plus anciens. La régression linéaire est utilisée pour prédire des valeurs continues. Elle suppose une relation linéaire entre les variables indépendantes et la variable dépendante et cherche à minimiser la somme des carrés des erreurs entre les prédictions et les vraies valeurs.

La *régression logistique*, malgré son nom, est utilisée pour la classification binaire. Elle modélise la probabilité qu'une entrée appartienne à une certaine classe (habituellement 0 ou 1) ou catégorie en utilisant la fonction logistique [39].

Les *Machines à Vecteurs de Support ou SVM* est un algorithme puissant pour la classification et la régression. Pour la classification, il cherche à trouver l'hyperplan qui maximise la marge entre les différentes classes. Il utilise également des fonctions noyau pour traiter des cas non linéaires [40].

Les *arbres de décision ou CART* sont utilisés à la fois pour la classification et la régression. Ils apprennent une série de règles d'inférence basées sur les caractéristiques des données. L'algorithme divise l'ensemble de données en branches pour aboutir à des décisions (feuilles) [41].

Les *forêts aléatoires* sont une extension des arbres de décision. Les forêts aléatoires construisent un grand nombre d'arbres en sélectionnant des racines de façon aléatoire et fusionnent leurs résultats. C'est une méthode d'ensemble qui offre une meilleure performance en réduisant le risque de sur ajustement. Ce modèle est basé sur un ensemble d'*arbres de décision*. Il s'agit de l'un des algorithmes les plus populaires et les plus puissants [42]. Il sélectionne la variable à diviser à l'aide d'un algorithme qui minimise l'erreur. Les arbres peuvent présenter de nombreuses similitudes structurelles, ce qui se traduit par une corrélation élevée de leurs prédictions. La combinaison des prédictions de plusieurs modèles dans des ensembles fonctionne mieux si les prédictions des sous-modèles sont non corrélées ou faiblement corrélées. Avec la *forêt aléatoire*, l'algorithme d'apprentissage est limité à un échantillon aléatoire de caractéristiques sur lesquelles il peut effectuer une recherche. Le nombre de caractéristiques pouvant être recherchées à chaque point de séparation doit être spécifié comme un paramètre de l'algorithme.

Le *K-Plus Proches Voisins ou K-NN* est une méthode de classification non paramétrique qui prédit la classe d'une instance en fonction des classes des K instances les plus proches dans l'espace des caractéristiques. L'algorithme regroupe les données en un nombre prédéfini de clusters [43].

Enfin, les algorithmes de *Boosting* comme `AdaBoost` et `Gradient Boosting` créent une séquence de modèles qui apprennent à corriger les erreurs des modèles précédents dans la séquence [44].

2.4.3 Évaluation des modèles

Suivant les données à analyser ou le contexte dans lequel est employé l'apprentissage automatique, il n'est pas toujours facile ou évident de savoir à l'avance quel sera l'algorithme le plus performant ou adéquat. Pour pouvoir le déterminer, il faut en tester plusieurs et évaluer les performances des modèles grâce à la *validation croisée*. C'est une méthode utilisée pour évaluer la performance des modèles d'apprentissage supervisé sur un ensemble de données limité.

Les performances des algorithmes peuvent être définies en fonction de la vitesse à laquelle les modèles sont construits et de la qualité de leurs prédictions. Des métriques comme la *Précision*, le *Rappel*, la *F-Mesure* ou l'*aire sous la courbe ROC* (AUC) permettent d'évaluer les performances de ces modèles. Les mesures sont souvent présentées sous la forme de *matrices de confusion*. La qualité des prédictions peut être mesurée à l'aide de différents indicateurs de performances expliqués ci-dessous.

Caractéristiques de bases :

True Positives (TP) correctement prédis 'Vrai' **False Positives (FP)** prédis Vrai, mais 'Faux'

True Negatives (TN) correctement prédis 'Faux' **False Negatives (FN)** prédis Faux, mais 'Vrai'

Mesures de performance :

À partir des caractéristiques de base décrites précédemment, nous pouvons calculer les différentes mesures de performance suivantes.

Précision $\frac{TP}{TP+FP} \Rightarrow$ exprime la fraction de prédictions vraies réellement vraies

True-Positive Rate (TPR) ou Recall ou Sensibilité $\frac{TP}{TP+FN} \Rightarrow$ fraction de 'Vrai' correctement prédite

True-Negative Rate (TNR) ou Spécificité $\frac{TN}{TN+FP} \Rightarrow$ fraction de 'Faux' correctement prédite

False-Positive Rate (FPR) $\frac{FP}{TN+FP} = 1 - \text{TNR}$

False-Negative Rate (FNR) $\frac{FN}{TP+FN} = 1 - \text{TPR}$

F-Measure (FM) or F1 Score $2\frac{\text{Recall.Precision}}{\text{Precision+Recall}}$

$\Rightarrow$ moyenne de *Précision* et *Recall* où 1 est considéré comme 'bon' et 0 comme 'mauvais'

Matthews Correlation Coefficient (MCC)

$$\frac{TP.TN - FP.FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Sensibilité et *Spécificité* sont inversement proportionnelles. Ainsi, lorsque nous augmentons la sensibilité, la spécificité diminue et vice-versa.

2.4.4 Défis & Limitations

Le principal défi avec l'apprentissage automatique supervisé est ce que l'on appelle le *sur-ajustement* ou 'overfitting'. Il se produit lorsque le modèle apprend trop bien les détails et le bruit dans l'ensemble d'entraînement au détriment de sa capacité à généraliser.

Un autre défaut est le *biais de sélection* qui se produit lorsque les exemples d'entraînement ne représentent pas correctement la population générale. De ce fait, le modèle peut ne pas bien fonctionner sur des données non vues.

Enfin, il est indispensable de bien équilibrer la complexité du modèle pour éviter le sur-ajustement (haute variance) et le sous-ajustement (biais élevé).

2.5 Approche non supervisée

2.5.1 Principe

L'apprentissage automatique *non supervisé* est une méthode d'analyse de données qui s'articule autour de la découverte de caractéristiques communes ou de structures cachées dans des données non étiquetées. C'est-à-dire, que contrairement à l'apprentissage supervisé, les données ne sont pas accompagnées de réponses correctes au préalable. L'objectif est de modéliser la structure sous-jacente ou la distribution dans les données afin de pouvoir en tirer des informations utiles ou prendre des décisions sur d'autres données.

2.5.2 Principaux algorithmes

Un des algorithmes le plus employé en apprentissage automatique non supervisé est *K-Means*. Son principe consiste à partager les données en K groupes en minimisant la variance intra-cluster et maximisant la variance inter-cluster [45].

La technique *Density-Based Spatial Clustering of Applications with Noise ou DBSCAN* est employée pour identifier les clusters en tant que zones de forte densité séparées par des zones de faible densité. Cet algorithme est principalement utilisé pour la géolocalisation ou la détection d'anomalies [46].

Une autre technique est le *clustering hiérarchique* qui permet de construire une hiérarchie de clusters en fusionnant ou en divisant successivement les groupes selon une mesure de distance [47].

Pour terminer, les données peuvent être catégorisées via l'*Analyse en Composantes Principales ou PCA*. Le principe consiste ici en la réduction de la dimensionnalité des données en les projetant sur les composantes principales qui maximisent la variance. La PCA est notamment employée pour la visualisation de données ou le prétraitement pour d'autres algorithmes de ML [48].

2.5.3 Domaine d'application

L'apprentissage non supervisé est utilisé dans de nombreux domaines plus particulièrement la détection d'anomalies pour la maintenance prédictive ou la détection de fraudes. Il peut également être utilisé en finance pour réaliser de la segmentation de marché en regroupant les clients ayant des comportements similaires.

L'apprentissage automatique non supervisé est un domaine puissant de l'IA qui permet de découvrir des informations cachées dans de vastes ensembles de données non étiquetées. Les algorithmes non supervisés offrent des moyens d'explorer la structure intrinsèque des données et de révéler des caractéristiques non évidentes au premier abord.

2.5.4 Défis & Limitations

L'apprentissage non supervisé présente des défis particuliers. Par exemple, l'interprétation des résultats peut être subjective, car il n'y a pas de "réponses correctes". De plus, la performance de ces algorithmes peut être difficile à évaluer puisqu'il n'y a pas de mesure standard comme la précision ou la matrice de confusion utilisable dans l'apprentissage supervisé.

Quelle que soit l'approche, supervisée ou non, le « nerf de la guerre » en apprentissage automatique ce n'est pas l'argent mais les données. Obtenir un jeu données à analyser est toujours une difficulté.

2.6 Jeu de données CTU-13

2.6.1 Contexte

Avec l'augmentation incessante des cyberattaques, la nécessité de systèmes de détection d'intrusion (IDS) fiables et efficaces n'a jamais été aussi critique. Les IDS s'appuient sur des jeux de données précis pour identifier le trafic réseau anormal, qui peut indiquer une attaque. Le CTU-13 est un jeu de données publiquement disponible, comprenant une série de scénarios d'attaque réels, ce qui le rend idéal pour tester et améliorer les IDS. Il a été créé par le groupe de recherche *Stratosphere* du département d'informatique de l'*Université Technique de Tchéquie* (Czech Technical University ou CTU) [49]. Il est devenu une référence dans la recherche en cybersécurité, notamment dans le domaine de la détection d'intrusion.

Le CTU-13 comprend treize scénarios d'attaques, chacun contenant des enregistrements de trafic malveillant et bénin, capturés dans des environnements contrôlés. Ces scénarios simulent diverses attaques (DDoS, Click Fraud, Brut Force, spam,...) réalisées par des malwares ou des botnets. Le jeu de données est accompagné d'étiquettes de classification et de métriques, ce qui permet d'évaluer les performances des différents systèmes de détection.

2.6.2 Applications & Limitations

Les données du CTU-13 ont été utilisées dans un large éventail de recherches, comme l'évaluation comparative des performances des IDS, le développement d'algorithmes d'apprentissage pour la classification du trafic réseau ou encore l'étude de la dynamique des attaques de réseau et de la propagation des malwares [50]-[53]. Ces scénarios peuvent également être utilisés pour tester l'efficacité des IDS et développer de nouvelles techniques pour identifier voire neutraliser les botnets.

Bien que le CTU-13 soit largement reconnu pour sa valeur, il présente des défis, notamment : *(i)* la nécessité de mettre à jour constamment le jeu de données pour refléter les nouvelles attaque *(ii)* la gestion de la grande taille du jeu de données, qui peut être difficile à traiter *(iii)* la possibilité de biais dans les scénarios en raison de la simulation d'attaque dans un environnement contrôlé.

2.6.3 Description des scénarios

Chacun des scénarios est conçu pour étudier et tester la détection de différentes formes de comportement malveillant, en particulier ceux associés aux botnets (réseau de machines malveillantes). Les treize scénarios représentent une gamme complète de comportements malveillants, dont : *(i)* plusieurs variantes d'attaques de botnets *(ii)* différentes stratégies d'attaques DDoS *(iii)* attaques de force brute SSH *(iv)* activités de scan réseau *(v)* téléchargement *(vi)* campagne de pourriels (spam) *(vii)* propagation de malwares.

Chaque scénario est documenté avec des informations détaillées sur le contexte de l'attaque, son déroulé et la nature du trafic malveillant. Un scénario contient des enregistrements détaillés des activités réseau associées à ces botnets, y compris les paquets envoyés et reçus, les tentatives de connexion à des serveurs de Commande et Contrôle (C&C), ainsi que le trafic normal généré par les utilisateurs du réseau pour fournir un contexte réaliste. La relation entre la durée du scénario, le nombre de paquets, le nombre de `NetFlows` et la taille du fichier 'pcap' est illustrée dans le tableau 2.1. Ce tableau montre également les logiciels malveillants (bots) utilisés afin de générer le trafic réseau pour la capture, le nombre de machines infectées dans chaque scénario et leurs activités malveillantes.

TABLE 2.1
Détails des données par scénario CTU-13

Id	Capture	Bot	#Bots	#Packets	#NetFlows	Duration (hrs)	.pcap (Go)	Taille (Mo)	Activités
1	42	`Neris`	1	71 971 482	2 824 637	6.15	52	377	IRC, SPAM, CF
2	43	`Neris`	1	71 851 300	1 808 123	4.21	60	241	IRC, SPAM, CF
3	44	`Rbot`	1	167 730 395	4 710 639	66.85	121	625	IRC, PS
4	45	`Rbot`	1	62 089 135	1 121 077	4.21	53	150	IRC, DDoS
5	46	`Virut`	1	4 481 167	129 833	11.63	37.6	17	SPAM, PS, HTTP
6	47	`Menti`	1	38 764 357	558 920	2.18	30	75	PS
7	48	`Sogou`	1	7 467 139	114 078	0.38	5.8	15	HTTP
8	49	`Murlo`	1	155 207 799	2 954 231	19.5	123	395	PS
9	50	`Neris`	10	115 415 321	2 753 885	5.18	94	279	IRC, SPAM, CF, PS
10	51	`Rbot`	10	90 389 782	1 309 792	4.75	73	175	IRC, DDoS
11	52	`Rbot`	3	6 337 202	107 252	0.26	5.2	14	IRC, DDoS
12	53	`NSIS.ay`	3	13 212 268	325 472	1.21	8.3	43	P2P
13	54	`Virut`	1	50 888 256	1 925 150	16.36	34	256	SPAM, PS, HTTP

Les valeurs avec # représentent le nombre d'éléments
CF=ClickFraud, PS=PortScan									

Scénarios 1 & 2 (infection par `Neris`) : Ces scénarios illustrent une infection par le bot `Neris`. Le trafic capturé comprend une communication typique entre le bot et son C&C serveur, ainsi que des tentatives de propagation et d'activités malveillantes, telles que des campagnes de pourriels ou de "Click Fraud".

Scénarios 3 & 4 (infection par `Rbot`) : Ces scénarios montrent comment `Rbot` scanne le réseau (scénario 3) ou réalise des attaques DDoS (scénario 4) en constituant un botnet et la manière dont il communique avec son C&C serveur pour obtenir des instructions ou télécharger ses mises à jour.

Scénarios 5 & 13 (infection par `Virut`) : Ces scénarios matérialisent l'activité du bot `Virut` et d'un réseau de machines infectées par ce même bot. Celui-ci réalise différentes activités malveillantes dont la participation à des réseaux de distribution de fichiers illégaux, le scan de port et l'envoi de pourriels.

Scénario 6 (infection par `Menti`) : `Menti` est un exemple de bot beaucoup moins répandu. Ce scénario se concentre sur l'activité de ce bot spécifique, en particulier ses échanges avec le serveur de commande et de contrôle ou son scan des ports.

Scénario 7 (infection par `Sogou`) : `Sogou` est un bot qui détourne les navigateurs Web pour effectuer des redirections non désirées, souvent vers des sites de 'phishing' (hameçonnage) ou des publicités.

Scénario 8 (infection par `Murlo`) Ce scénario dépeint l'activité d'un bot effectuant du scan de ports.

Scénario 9 (variante d'infection par `Neris`) Il s'agit d'une variante du premier scénario, offrant des exemples de trafic légèrement différents. Le trafic est généré par dix machines infectées.

Scénarios 10 & 11 (infection par `Rbot`) : Ces scénarios combinent le trafic généré par trois bots.

Scénario 12 (infection par `NSIS.ay`) : C'est un bot qui a été inclus dans le jeu de données afin de matérialiser l'activité d'un botnet impliquant des téléchargements (P2P) et les mises à jour de son logiciel malveillant.

2.7 Jeu de données CANCAN

2.7.1 Contexte

CANCAN est l'acronyme de "Content and Context based Adaptation in Mobile Networks". Le projet éponyme [54] vise à atteindre les objectifs suivants :

1. Recueillir de nouveaux ensembles de données de mesure qui décrivent le trafic de données des réseaux mobiles à des niveaux de précision spatiale et temporelle sans précédent, et pour différents services mobiles séparément. Les ensembles de données seront collectés dans un réseau national opérationnel.
2. Évaluer les outils analytiques existants pour la classification, la prédiction et la détection d'anomalies dans les données de réseau mobile réelles très détaillées par service, et les adapter aux spécifications de la gestion des ressources à différents niveaux du réseau.
3. Démonstration de l'intégration de l'analyse des données dans les architectures de réseaux cognitifs de la prochaine génération dans trois études de cas pratiques, à savoir : *(i)* la programmation radio prédictive et différenciée dans le réseau d'accès radio virtualisé (vRAN) *(ii)* la gestion dynamique des machines virtuelles et des conteneurs dans l'informatique mobile de périphérie (MEC) *(iii)* les accords de niveau de service dynamiques et la génération de classes de service pour le partage des ressources et l'atténuation des goulets d'étranglement.

Les données mobiles ont été collectées à grande échelle grâce à des sondes passives déployées dans le réseau mobile de l'opérateur français *Orange*. Elles ont ensuite été anonymisées conformément aux règles du RGPD [3] [55] et au Code de Conduite National des Télécommunications [56]. Le jeu de données ainsi créé est composé de nouveaux *flux de trafic anonymisés très détaillés* (xDR) contextualisés sur la mobilité des utilisateurs et leur utilisation des services mobiles. Il caractérise le *trafic mobile* et l'*utilisation des applications* sur une période de trois mois pour l'ensemble du territoire national français. Les flux réseaux obtenus sont composés de onze champs différents détaillés ci-dessous :

0/ PortApp Type d'application mobile

1/ LocInfo Identifiant de l'antenne

2/ Coord_X/Y Coordonnées du site

3/ SiteName Nom du site

4/ TimeSlot Horodatage capture

5/ nPktUp Paquets montants session TCP

3. Règlement Général sur la Protection des Données

6/ nPktDn Paquets descendants session TCP **7/ Duration** Durée session TCP

8/ Users Nombre utilisateurs distincts sous cellule **9/ Flows** Flux échangés pendant session TCP

2.7.2 Applications & Limitations

Les caractéristiques des flux ainsi obtenues seront une donnée fondamentale pour les fonctions réseaux en charge d'organiser le trafic et d'allouer les ressources en fonction des besoins des utilisateurs finaux, et ce, de façon automatique. Cette vision s'appuie principalement sur l'orchestration efficace des ressources ou des services, et ce, à tous les niveaux d'un réseau mobile. Un effort substantiel est, de ce fait, actuellement en cours afin de définir des architectures pour l'allocation dynamique des ressources, qui comprennent toutes des composants d'orchestration chargés de prendre des décisions automatisées sur la reconfiguration des ressources. Cependant, les algorithmes et les politiques de gestion dynamique des ressources qui fonctionneront à l'intérieur des orchestrateurs de réseau doivent être entièrement étudiés et définis. Un autre défi est d'analyser cette masse de données très détaillées afin d'essayer d'en extraire des informations sous-jacentes ou de matérialiser des comportements particuliers ou spécifiques.

Le principal inconvénient de ces données fournies dans la cadre du projet CANCAN est justement cette anonymisation des traces. Il en résulte des flux réseaux sans adresse IP source ou destination, ni port source ou destination, au sens de la couche transport. Nous n'avons également pas accès aux informations liées aux numéros IMSI ou IMEI des utilisateurs mobiles. Tout ceci limite grandement les possibilités d'agrégation des flux ou d'étude comportementale de l'activité réseau. De plus, comme expliqué précédemment, ce jeu de données a été fourni par *Orange* et reste donc la propriété de l'opérateur ce qui nous interdit de le mettre à disposition. Par conséquent, seuls les fichiers agrégés par secteur sont disponibles au téléchargement.

2.8 Conclusion

2.8.1 Attaques réseaux

Les attaques réseau varient en fonction des protocoles. Le tableau 2.2 est un aperçu des attaques les plus courantes en fonction des protocoles réseaux les plus communs. Nous constatons qu'il existe des attaques pour chacunes des couches du modèle OSI.

TABLE 2.2
Attaques réseaux en fonction des protocoles

Couche	Protocole	Attaque	Principe
Liaison	ARP	Spoofing/Poisoning	- Associer adresse MAC de l'attaquant à une adresse IP légitime pour détourner le trafic
	ICMP	Ping Flood	- Inonder la cible avec des paquets pings pour réaliser un déni de service
	ICMP	Ping of Death	- Envoyer des paquets ICMP plus grands que la taille maximale autorisée pour faire 'crasher' le destinataire
	ICMP	Smurf	- Exploiter des réseaux qui répondent aux requêtes de diffusion pour inonder une cible de réponses ICMP
Réseau	IP	Spoofing	- Falsifier adresse IP source pour masquer identité de l'attaquant ou imiter une autre machine
		Teardrop	- Transmettre des paquets IP fragmentés dans un ordre incorrect pour faire 'crasher' la cible
Transport	TCP	SYN Flood	- Inonder la cible avec demandes SYN sans jamais compléter poignée de main en 3 étapes pour provoquer déni de service
	TCP	RST	- Envoyer paquet RST pour interrompre session TCP établie
	TCP	Hijacking	- Usurper session TCP active en prédisant num. séquence
	TCP	Fragmentation	- Fragmenter paquet malveillant pour éviter la détection
	UDP	Flood	- Envoyer de nombreux paquets UDP à des ports aléatoires
	UDP	Réflexion	- Exploiter des serveurs vulnérables pour envoyer du trafic à une victime avec une fausse adresse IP source
Présentation	SSL	Heartbleed	- Exploiter faille `OpenSSL` pour lire mémoire du serveur
		Poodle	- Exploiter faille `SSLv3` pour déchiffrer info. de session
Application	DNS	Amplification	- Inonder une victime de réponses DNS
	DNS	Cache Poisoning	- Introduire données corrompues dans cache pour rediriger utilisateurs vers des sites malveillants
	DHCP	Starvation	- Épuise toutes les adresses IP disponibles
	DHCP	Rogue Server	- Configurer un serveur malveillant pour distribuer des adresses IP et re-router le trafic

2.8.2 Jeu de données idéal

Il existe de nombreux jeux de données dont certains sont plus tournés vers la cybersécurité. Ces jeux de données spécialisés sont utilisés dans la recherche et par les professionnels de la cybersécurité pour développer et évaluer des méthodes de détection d'intrusion, des solutions de sécurité, des systèmes de détection d'anomalies, et bien plus encore. Ils fournissent une base de données réaliste pour tester et améliorer la résilience des systèmes informatiques contre les menaces et les attaques.

La sélection du jeu de données dépend de plusieurs facteurs, notamment le type d'anomalies recherché, la variété des scénarios d'attaque couverts, la qualité et la représentativité des données, ainsi que la capacité du jeu de données à refléter des conditions réelles de trafic réseau. Nous avons listé les principaux d'entre-eux et décrit leurs particularités dans le tableau 2.3.

TABLE 2.3
Principaux jeux de données utilisés en cybersécurité

Nom	Origine	Capture	Date	Taille	Labellisé	Description des données	Références
KDD'99	UCI	Réseau militaire	1999	743 Mo	Oui	Simulation d'attaques	[57], [58]
NSL-KDD	UNB	Réseau privé labo.	2000		Oui	Amélioration du KDD	[59], [60]
CNU/CDMA	CRAWDAD	Internet mobile	2005-2007	3 Mo (zip)	Non	Fichiers `tcpdump`	[61], [62]
UCSD	CAIDA	Réseau privé labo.	2007	21 Go	Non	Attaque DDoS 1 heure	[53], [63]
CTU-13	CTU	Réseau privé labo.	2011	2.5 Go	Oui	Activités botnets	[51], [64]
MalGenome	NCSU	Collection malwares	2012		Oui	Fichiers APK	[65], [66]
CANCAN	OrangeLab	Réseau mobile	2019	190 Go	Non	Réelles anonymisées	[54], [67]
MAWI	WIDE	Réseau Internet	2006-2024	"Live"	Non	Réelles, lien transpacific	[68], [69]
AndroZoo	UNILU	Collection malwares	2023	2.7 Go	Oui	Fichiers APK	[70], [71]
ISCX	CIC	Simulation	2009-2016	75 Go	Oui	VPN, Botnets	[72], [73]
CIC/IDS	CIC	Simulation	2017-2020	50 Go	Oui	Malwares Androïd	[65], [72]

CRAWDAD : Community Resource for Archiving Wireless Data at Dartmouth – CTU : Czech Technical University
CIC : Canadian Institut of Cybersecurity – UCI : University of California – UNB : University of New Brunswick
NCSU : North Carolina State University – UNILU : Université du Luxembourg

Les jeux de données CTU-13, CANCAN et NSL-KDD offrent chacun des caractéristiques uniques qui peuvent les rendre plus ou moins adaptés en fonction des objectifs que nous recherchons.

Le CTU-13 est spécialisé dans le trafic des botnets. Il propose des scénarios d'attaque variés, bien documentés et détaillés. Basé sur du trafic réseau réel généré dans un environnement contrôlé, il offre une représentation réaliste des comportements malveillants et bénins de différents malwares. Par contre, bien que riche en données de botnets, il peut manquer de variété pour détecter d'autres types d'attaques ou d'activités malveillantes.

Les traces fournies par `Orange` dans le cadre du projet CANCAN sont des données réelles décrivant l'activité des utilisateurs et le type d'application mobile utilisé sur son réseau mobile 2G et 3G. Les captures sont très détaillées et sont caractérisées par des champs spécifiques et modernes. Par contre, les traces sont très génériques du fait de l'anonymisation et non labellisées ce qui rend difficile la création de modèles. De plus, le jeu de données extrait pour les besoins de nos travaux est particulièrement volumineux (environ 190 Go) ce qui est un défi supplémentaire pour son exploitation.

NSL-KDD résout les problèmes de redondance et de déséquilibre de son prédécesseur, le KDD'99, offrant un jeu de données plus équilibré pour l'entraînement et les tests. Il couvre une variété d'attaques, rendant les modèles formés plus généralisables. D'un autre côté, malgré ses améliorations, il peut toujours manquer de représentativité pour les attaques les plus récentes ou complexes.

Pour conclure sur les jeux de données, nous pourrions essayer de définir *le jeu de données idéal* permettant la détection d'anomalies par apprentissage automatique. Celui-ci pourrait répondre aux caractéristiques suivantes :

Diversité Il devrait couvrir une large gamme d'attaques réelles et hypothétiques, y compris les plus récentes et complexes.

Équilibre Un bon équilibre entre les données d'attaques et les données normales pour éviter les biais d'apprentissage.

Qualité Les données devraient être de haute qualité, avec peu de bruit et correctement labellisées.

Détaillé Les traces devraient être détaillées par des caractéristiques précises et très faiblement corrélées.

Représentativité Il devrait refléter fidèlement le trafic réseau actuel, y compris les protocoles, les comportements utilisateurs et les configurations système variées.

Accessibilité Facilement accessible afin de favoriser la collaboration, la reproductibilité des expériences et la comparaison des résultats.

Aucun des jeux de données mentionnés précédemment ne remplit parfaitement tous ces critères. Le choix dépend donc des objectifs spécifiques de recherche et des types d'anomalies ciblées. Une approche mixte, utilisant des caractéristiques de plusieurs jeux de données ou leur combinaison, pourrait s'avérer être une stratégie efficace pour développer des modèles de détection d'anomalies robustes et généralisables.

2.8.3 Algorithmes populaires en cybersécurité

Dans le domaine de la sécurité informatique, et plus spécifiquement dans la détection d'anomalies réseau, plusieurs algorithmes d'apprentissage automatique ont été largement étudiés et utilisés. Leur popularité repose sur leur capacité à identifier des comportements non standards ou malveillants dans le trafic réseau, souvent en apprenant à distinguer les activités normales des activités suspectes ou malveillantes. Dans cette section 2.8.3, nous donnons un aperçu des algorithmes les plus usités.

1. Les forêts aléatoires sont sans aucun doute la méthode d'apprentissage ensembliste la plus utilisée. En effet, elles sont réputées pour leur robustesse et leur capacité à gérer de grandes quantités de données et de caractéristiques.

   Inconvénients : Peuvent être assez gourmandes en ressources et en temps de calcul, surtout avec un grand nombre d'arbres et une grande profondeur. Elles peuvent également souffrir d'un manque de transparence dans le processus de prise de décision (boîte noire).

2. Les SVM sont particulièrement appréciées pour leur efficacité dans les espaces de grande dimension et les situations où le nombre de dimensions dépasse le nombre d'échantillons. Elles sont utilisées pour la classification et la régression en trouvant l'hyperplan qui sépare le mieux les différentes classes.

   Inconvénients : Le choix du noyau et le réglage des paramètres peuvent être difficiles et influencer grandement la performance du modèle. Les SVM peuvent également être inefficaces sur des jeux de données très volumineux et sont moins efficaces pour des ensembles de données où le nombre de caractéristiques est beaucoup plus grand que le nombre d'échantillons.

3. Les techniques de deep learning, y compris les réseaux de neurones artificiels (ANN) et convolutifs (CNN), sont de plus en plus utilisées pour la détection d'anomalies réseau en raison de leur capacité à apprendre des représentations complexes des données. Les modèles profonds peuvent automatiquement extraire des caractéristiques pertinentes à partir des données brutes, ce qui est un avantage significatif pour la détection d'anomalies.

   Inconvénients : Ils nécessitent de grandes quantités de données d'entraînement et sont coûteux en termes de calcul et de temps. Ils peuvent aussi être difficiles à interpréter en raison de leur nature de "boîte noire" et sont sujets au sur apprentissage sans une régularisation appropriée.

4. Les algorithmes de clustering, tels que K-Means, sont utilisés pour regrouper les données en plusieurs clusters en fonction de leur similarité. Dans le contexte de la détection d'anomalies, le clustering peut aider à identifier les groupes d'activités réseau qui se comportent de manière similaire, facilitant ainsi la détection des comportements anormaux ou des outliers.

   Inconvénients : L'algorithme peut devenir significativement plus lent à mesure que la taille du jeu de données augmente, car il doit calculer la distance entre chaque paire de points. De plus, la performance de k-NN peut être fortement affectée par la présence de caractéristiques non

pertinentes ou bruitées.

5. L'Isolation Forest est un algorithme spécifiquement conçu pour la détection d'anomalies. Il isole les observations en sélectionnant aléatoirement une caractéristique et en choisissant aléatoirement un seuil de séparation entre les valeurs maximales et minimales de la caractéristique sélectionnée. Les anomalies sont isolées plus rapidement, ce qui rend cet algorithme efficace pour détecter les anomalies dans les données.

   Inconvénients : Bien que performant pour la détection d'anomalies, cet algorithme peut parfois isoler des points normaux comme étant anormaux, surtout dans des données très bruitées ou dans des situations où les anomalies ne sont pas significativement différentes des points normaux.

6. Pour les données de séries temporelles, comme le trafic réseau, les RNN et en particulier les variantes telles que les LSTM (Long Short-Term Memory) sont utiles pour modéliser la dépendance temporelle entre les observations. Ces modèles sont capables de capturer des séquences temporelles longues et variables, ce qui est essentiel pour détecter des comportements anormaux qui se manifestent sur de longues périodes.

   Inconvénients : Peuvent être sujets au sur apprentissage et à des problèmes d'explosion des gradients, bien que les LSTM soient spécifiquement conçus pour les atténuer. Ils nécessitent également beaucoup de ressources, sont complexes à configurer et à entraîner.

Le choix de l'algorithme dépend des spécificités du problème à résoudre, notamment le type et la quantité de données disponibles, la nature des anomalies à détecter, et les exigences en termes de performance et de temps de réponse. Dans la pratique, une combinaison de plusieurs techniques peut être utilisée pour améliorer la précision et la robustesse de la détection d'anomalies réseaux.

2.8.4 Anomalies & Détection

Le CTU-13, au travers de ses différents scénarios, nous permet d'entraîner des algorithmes d'apprentissage automatique et de générer des modèles capable de détecter différentes familles de botnets. Le problème ici est qu'un modèle, construit sur la base d'un scénario, permet de détecter l'activité d'une et d'une seule famille de botnets bien particulière notamment grâce à la modélisation des échanges inter-bots et des échanges bots-C&C serveur, mais également du trafic généré lors des attaques (spam, scan de ports,...). Les algorithmes mettant en œuvre des arbres de décision et des

forêts aléatoires sont très efficaces pour cette détection, mais la création d'un modèle générique s'avère être difficile et inefficace. L'idée sous-jacente de notre première contribution, inspirée par ces forêts, a donc été de *concaténer* des modèles entrainés pour détecter des familles de botnets spécifiques afin de créer un *métamodèle* capable de détecter plusieurs familles de botnets, métamodèle baptisé "Forêt combinée" et présenté dans le chapitre 3. Celui-ci est basé sur l'apprentissage automatique *supervisé*, car le CTU-13 fournit des données *labellisées*.

Le projet CANCAN nous a permis d'avoir accès aux traces mobiles éponymes. Ces captures décrivent l'activité des usagers d'un réseau mobile, activité représentée par le type d'application mobile utilisé, le volume ainsi que le nombre de données échangées. Les données étant *non labellisées*, nous avons orienté nos recherches vers l'apprentissage *non supervisé* et le *clustering* pour essayer d'*extraire des tendances* et de *matérialiser le comportement* des utilisateurs en fonction du type d'application mobile utilisé. Pour ce faire, nous avons défini ce que nous avons nommé des *signatures numériques*. Ces signatures nous ont permis de *catégoriser* des secteurs réseaux et de faire ressortir des *changements de comportement*, ou anomalies réseaux, relatifs à des événements grâce à notre seconde proposition baptisée "Framework de détection d'anomalies par signature numérique" et expliqué dans le chapitre 4. Le principal inconvénient de cette seconde approche est le fait que ces signatures doivent être comparées entre elles *manuellement*, c'est-à-dire interprétées par un "œil humain", d'où notre troisième contribution.

La chapitre 5 détaille notre dernière contribution, la méthodologie DiNATrA$\mathcal{X}$, qui est une évolution de notre "Framework de détection d'anomalies par signature numérique". Celle-ci permet de comparer de façon *automatisée* et pratiquement en *temps reél* les signatures numérique grâce à notre concept de *Digital Network Assessment* ou ADN. Cette comparaison nous permet de mettre en évidence des variations de l'activité réseau et donc des anomalies pouvant caractériser des attaques ou des événements particuliers sortant de l'ordinaire.

2.8.5 Outils mis en œuvre

Avant de mener nos travaux, nous avons réalisé un tour d'horizon des différents outils logiciels dédiés à l'apprentissage automatique. De cette étude, restituée dans l'annexe A, nous pouvons dire que : *(i)* `MLR3` et `Weka`, bien que bénéficiant d'un engagement plus modeste de la part des contributeurs, maintiennent une qualité et une spécialisation élevées grâce à leurs communautés

dédiées. *(ii)* `Scikit-learn` se distingue par sa grande base de contributeurs ce qui témoigne de son adoption généralisée, de son support communautaire robuste et de son évolution rapide.

Le tableau 2.4 est une synthèse des principaux outils dédiés à l'apprentissage automatique que nous avons étudiés. Les cinq critères d'évaluation employés sont pondérés de '1 à 3' où '1' est le poids le plus faible et '3' le plus représentatif.

TABLE 2.4
Synthèse pondérée des outils d'apprentissage automatique

Outil	Performance	Facilité	Popularité	Communauté	Déploiement	TOTAL
`Weka`	1	3	1	1	1	7
`MLR3`	2	1	2	2	2	9
`Scikit-learn`	3	2	3	3	3	14
Points : 1=Faible – 2=Moyen – 3=Fort						

Nos premiers travaux de recherche ont été réalisés grâce à la suite logicielle `Weka`, car elle permet de très facilement créer des pipelines de traitement notamment d'assemblage de modèles et de "boosting" contrairement au langage R même avec `MLR3`. De plus, la quantité de données à traiter était raisonnable, de l'ordre de quelques centaines de mégaoctets par scénario. Par conséquent, notre "Forêt combinée" a été entièrement testée et modélisée avec la suite `Weka`.

Pour ce qui est de l'étude du jeu de données CANCAN, nous avons également pu obtenir des résultats en utilisant `Weka`, car les données étaient prétraitées et agrégées avec des analyseurs syntaxiques (parsers) codés en `Perl`. Notre "Framework de détection d'anomalies par signature numérique" a lui aussi été implémenté avec l'aide `Weka`. Malgré tout, nous avons été assez vite limités par les performances de cette solution notamment à cause du volume de données à analyser.

S'agissant de DiNATrA$\mathcal{X}$, nous avons choisi d'utiliser le langage `Python` et la librairie `Scikit-learn` pour ses performances, sa facilité de mise en œuvre et ses nombreuses fonctionnalités comme expliquées dans l'annexe A.4 et synthétisées par le tableau 2.4.

Chapitre 3

Métamodèle supervisé & évolutif de détection de botnets

3.1 Introduction

3.1.1 Contexte

La multiplication et la prolifération des botnets constituent aujourd'hui une des menaces les plus sérieuses contre les systèmes d'information en général et les réseaux informatiques en particulier. L'attaque majeure récente subie par les sociétés françaises `Viamedis` et `Almerys`, décrite dans la section 2.2.9, a fait plus de 33 millions de victimes suite aux vols d'identifiants et de mots de passe de professionnels de santé. Ces vols ont probablement été perpétrés par hameçonnage via un email reçu lors d'une campagne de pourriels comme expliqué dans la section 2.1.4. Ces campagnes de spams sont très souvent réalisées par des *botnets*. La dernière cyberattaque en date, présentée dans la section 2.2.10, a également été réalisée à l'aide d'un botnet. Elle a consisté en une tentative de rendre inutilisable le RIE via une attaque DDoS comme expliqué dans la section 2.1.1. La détection de ces botnets est donc devenue une *nécessité* et un *défi majeur* en matière de cybersécurité [74].

Certaines techniques de détection d'attaques informatiques exploitant des approches supervisées ou non ont été développées, détections généralement basées sur l'analyse du trafic réseau comme présentées et détaillées par [44], [48]. La technique exposée dans cette première étude, employant l'approche supervisée, nécessite d'extraire *trente-neuf caractéristiques réseaux différentes*, et ce, pour chacun des flux de trafic définis par les couples adresse IP & port identifiant la source et la destination ainsi que le protocole utilisé et les données utiles ('payload'). Les inconvénients induits ici sont que :

(i) le port source peut être amené à *évoluer (ii)* le système doit nécessairement pouvoir *analyser* le corps de la requête ce qui peut induire des contraintes liées à la confidentialité des données. La seconde étude, exploitant les données du CTU-13 détaillé dans la section 2.6, est quant à elle basée sur l'*Analyse en Composantes Principales*, un algorithme non supervisé présenté dans la section 2.5.2. La difficulté est ici de pouvoir *catégoriser* chacun des groupes obtenus comme 'normal' ou 'malveillant' et ce en fonction de l'*analyse statistique* réalisée, ce qui peut aboutir à des erreurs de classification.

3.1.2 Qu'est-ce qu'un botnet ?

Un botnet est un ensemble d'équipements compromis (bots) par un logiciel malveillant (malware) qui sont sous le contrôle d'un attaquant distant. Contrairement à d'autres malwares plus conventionnels comme les virus, les chevaux de Troie ou encore les vers, les bots échangent des données soit entre eux, soit avec l'attaquant par le biais d'un canal de communication ouvert employant des protocoles tels qu'IRC, HTTP ou le Peer-to-Peer (P2P).

Au cours des dix dernières années, les botnets ont représenté une menace croissante. Ils sont devenus le moyen privilégié pour lancer diverses attaques de type DDoS ou bien des campagnes de pourriels. Au fur et à mesure de leurs attaques, des malwares comme `Zeus`, `Stuxnet`, `Emotet` ou `Retadup` ont évolué pour constituer des botnets de plus en plus sophistiqués, furtifs et malveillants.

La topologie des botnets, illustrée par la figure 3.1, repose essentiellement sur quatre éléments clefs : *(i)* les machines compromises connues sous le nom de *Bots* ou *Zombies (ii)* le *Botmaster* qui contrôle le botnet et qui parfois a développé *(iii)* le *code du bot* ou malware *(iv)* et le *Serveur de Commande et de Contrôle* (C&C serveur) utilisé pour diriger le botnet. Tous ces éléments communiquent entre eux.

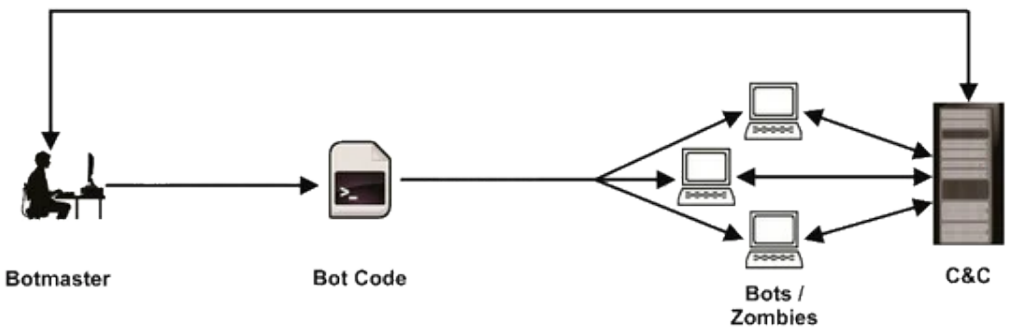


FIG. 3.1. Topologie d'un botnet

L'architecture de commande et de contrôle est le point clef de tout botnet. Les bots interagissent avec le botmaster pour recevoir des ordres et envoyer des journaux ou entre eux pour échanger des données en utilisant des canaux de communication légitimes afin de s'inscrire dans un réseau plus large de dispositifs infectés [75]. La résilience et l'efficacité d'un botnet résident dans la structure de son architecture de commande et de contrôle. Il existe actuellement trois principales architectures [76] représentées par la figure 3.2 : *(a)* les bots et le botmaster échangent *directement* en utilisant un canal IRC *(b)* un *C&C serveur* est utilisé comme lien HTTP entre les bots et le botmaster *(c)* le botmaster utilise un *protocole P2P* pour atteindre un ou certains bots sélectionnés qui, à leur tour, contacteront d'autres bots.

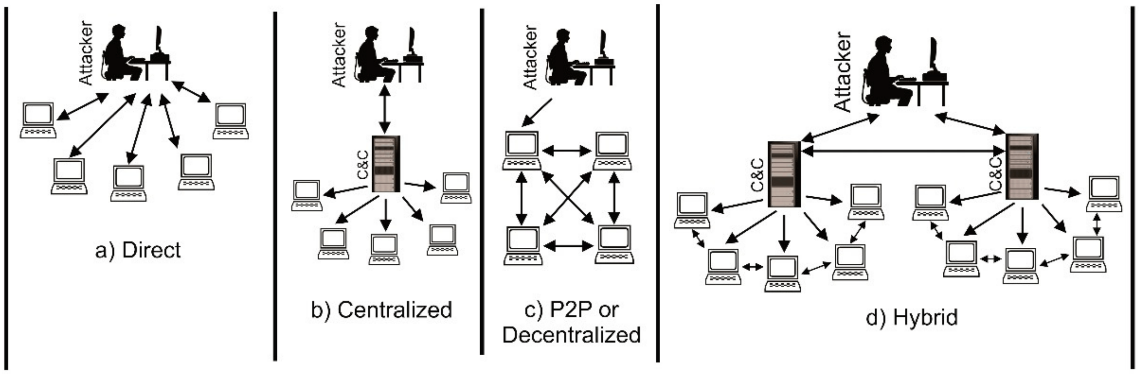


FIG. 3.2. Architectures de C&C

Certaines architectures hybrides (d) fusionnent les avantages du mode direct, centralisé ou P2P pour créer une topologie de botnet plus robuste, les rendant ainsi assez difficiles à détecter et à contrecarrer. Le canal de communication est créé une fois qu'un équipement est compromis. Le bot essaiera de contacter et d'informer le botmaster que l'appareil a été pris en contrôle avec succès.

Ces échanges ; que nous avons définis comme étant *intrinsèques* ; sont véhiculés via différents canaux de communication et par conséquent génèrent du trafic réseau, trafic qui peut être *détecté*. En effet, la mise en lumière de ces échanges, qui peuvent être considérés comme anormaux et donc comme des anomalies réseaux, peut permettre de détecter ces botnets et de prévenir, voire limiter leurs attaques. Le canal de communication est de ce fait le principal point faible ou SPoF (Single Point of Failure) d'un botnet. Sans lui, il n'est qu'un ensemble inutile de bots. Interrompre les échanges en coupant ce canal est une contre-mesure efficace [75].

3.1.3 Contribution

Pour nos premiers travaux de recherche sur les anomalies réseaux, nous avons choisi de travailler sur la détection de l'activité de différentes familles de botnets. Pour ce faire, nous nous sommes orientés vers l'apprentissage automatique supervisé. Notre première étude nous a permis de proposer et de publier une nouvelle approche baptisée "Forêt combinée".

L'apprentissage automatique supervisé se base sur des algorithmes qui permettent de créer des modèles à partir de données d'apprentissage labellisées. Une fois définis, ces modèles sont ensuite utilisés pour classifier ou catégoriser des nouvelles entrées. La grande force de ces algorithmes supervisés est qu'ils sont très performants. Par contre, ils sont capables de classifier *uniquement* les nouvelles données pour lesquelles ils ont été *explicitement entraînés*. Pour synthétiser, un modèle permet de classifier un type et un seul d'activité réseau. Or, il existe différentes "familles" de botnets, c'est-à-dire utilisant des méthodes différentes pour leurs échanges internes ou *intrinsèques*, générant par conséquent différents types de trafic réseau. Il nous faut donc définir autant de modèles qu'il existe de familles de botnets si l'on veut espérer conserver l'efficacité de l'apprentissage supervisé. En outre, même s'il était possible de concevoir un modèle unique performant capable de faire ressortir plusieurs types de familles, celui-ci nécessiterait d'être reconstruit entièrement dès la découverte d'une nouvelle famille.

Notre idée a alors été d'assembler plusieurs modèles pré-entraînés ou *prédéfinis* pour classifier les flux réseaux issus de l'activité de différentes familles de botnets et construire ce que nous avons nommé une *forêt combinée*. Il s'agit d'un *métamodèle* de type forêt d'apprentissage automatique, construit en combinant plusieurs *modèles de décision prédéfinis* comme présenté par la figure 3.3. Chaque modèle est généré à partir d'un ensemble de données étiquetées grâce à une méthode d'analyse supervisée. Ils nous permettent de mettre en évidence différentes familles de botnets en détectant leurs échanges intrinsèques, échanges spécifiques à chaque famille.

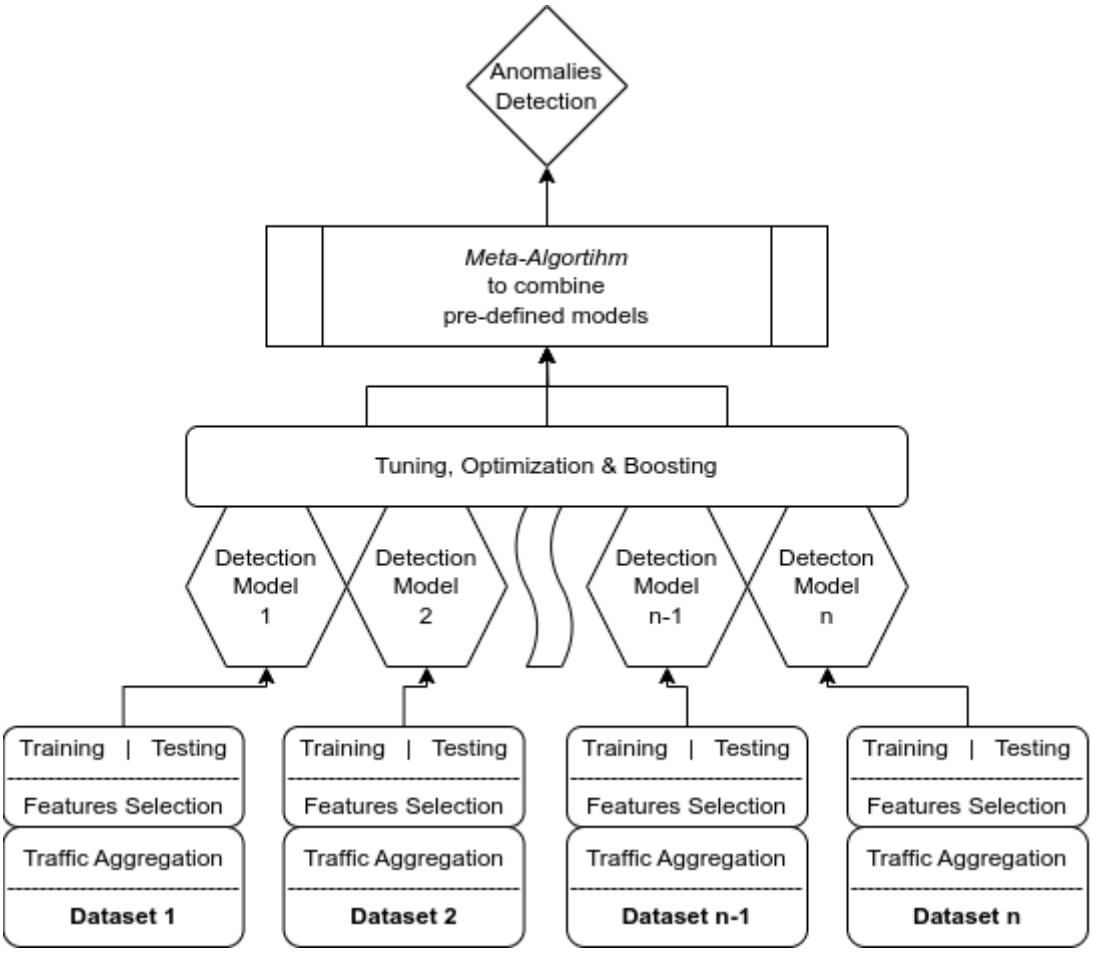


FIG. 3.3. Schéma fonctionnel métamodèle "Forêt combinée"

Pour ce faire, nous avons tout d'abord regroupé tous les *flux IP* en *flux de trafic* afin d'en extraire les caractéristiques clefs, indépendantes notamment du port source, pour éviter les sur-ajustements, notion abordée dans la section 2.4.4. Puis, nous avons testé différents algorithmes supervisés afin de déterminer le plus adapté pour construire nos différents modèles à partir des données extraites avant de les combiner en un métamodèle capable de détecter plusieurs familles d'anomalies. Pour finir, de nombreuses expériences et mesures ont été menées pour affiner les paramètres et concevoir le métamodèle le plus précis, efficace et évolutif possible.

Les trois principaux avantages de notre "Forêt combinée" par rapport aux travaux existants sont : *(i)* ses performances en termes de précision et de rapidité *(ii)* sa souplesse en terme de configuration et de choix des algorithmes d'apprentissage *(iii)* son évolutivité qui permet de facilement la mettre à jour en ajoutant d'autres modèles afin de détecter de nouvelles familles de botnets.

Ce chapitre 3 présente notre contribution à la détection de ces botnets. Nous commençons par décrire fonctionnellement notre "Forêt combinée" dans la section 3.2. Puis, la section 3.3 détaille la mise œuvre de notre méthode de détection avec le CTU-13, jeu de données utilisé pour entraîner et tester notre métamodèle ainsi que les différents résultats que nous avons obtenus. Enfin, la section 3.4 conclut ces premiers travaux de recherches sur l'apprentissage automatique supervisé et son application à la détection des anomalies réseaux engendrées par les botnets.

3.2 Description fonctionnelle

De nombreuses recherches ont été menées à propos de la détection de l'activité malveillante des botnets. Celles-ci consistent soit à surveiller le trafic réseau soit à analyser les entêtes des messages ou le contenu des échanges, avec les problèmes de confidentialité que cela engendre [44], [48].

Dans le cadre de nos travaux, nous avons choisi de nous orienter vers l'apprentissage automatique. Notre but est de proposer un modèle capable de détecter des comportements réseaux anormaux qui pourraient indiquer l'existence d'un canal de *Command & Control* au sein d'un logiciel malveillant et donc, mettre en évidence un membre d'un botnet.

L'hypothèse principale des Systèmes de Détection d'Anomalies (ADS) basés sur l'apprentissage automatique est que les réseaux de zombies ou bots génèrent des modèles de trafic spécifiques qui peuvent être détectés efficacement. Cette approche offre une méthode de détection flexible, indépendante des technologies de communication et des stratégies de résilience employées par les botnets. Différents algorithmes d'apprentissage automatique ont été développés et mis en œuvre pour concevoir diverses méthodes de détection comme dans les travaux [47], [77].

Pour obtenir un bon taux de détection avec de l'apprentissage automatique supervisé, les méthodes citées précédemment ont nécessité trois étapes indispensables : *(i)* les données brutes labellisées issues de la capture ont été *agrégées* afin de modéliser l'ensemble des *échanges réseaux (ii)* des *éléments particuliers* ont été extraits de ces flux agrégés pour les *caractériser (iii)* des algorithmes ont été employés dans le but de construire des *modèles de décision*.

Une fois créés, ces modèles sont utilisés pour analyser de nouvelles données et en discriminer les activités malveillantes du trafic bénin, en fonction des données d'apprentissage utilisées.

3.2.1 Analyse & Agrégation des données

Une grande quantité de données peut être collectée lors d'un processus de capture réseau. Ces trames, matérialisant différentes transactions, sont principalement définies par l'adresse IP et le port (socket TCP) de la source et le socket destination. Afin de réduire le volume de données à analyser et mettre en évidence les différents échanges, nous avons *agrégé les flux IP* issues de la capture en *flux de trafic*. Ces flux nous permettent de matérialiser un échange complet entre une source et une destination.

3.2.2 Extraction des caractéristiques

De nombreuses caractéristiques peuvent être collectées lors de la capture et sont donc disponibles pour construire, définir les modèles de décision. En plus des adresses IP, des ports sources et destinations, sont généralement disponibles la durée de la transaction, le sens du flux, le nombre de paquets échangés, la taille des paquets, les données utiles, l'état de la connexion et le protocole de transport.

L'objectif ici est d'extraire des informations clefs afin de réduire la quantité d'informations à analyser, mais en conservant la capacité de détecter l'ensemble des comportements malveillants. La sélection d'un ensemble adéquat de caractéristiques pour la modélisation des données améliore les performances du processus d'apprentissage, réduit les temps de calcul et par conséquent les ressources nécessaires. À partir de ces caractéristiques, il nous est alors possible de modéliser l'activité réseau.

3.2.3 Modélisation du trafic

L'entraînement des algorithmes supervisés nécessite un ensemble de données étiquetées et définies à partir des caractéristiques extraites précédemment. Pour ce faire, le jeu de données peut être créé en capturant et en étiquetant du trafic légitime issu d'un réseau connu. Puis, ce trafic légitime est fusionné avec du trafic malveillant, généré dans un environnement contrôlé à l'aide de bots connus et maîtrisés. L'autre solution consiste à employer un jeu de données pré-existant pour les phases d'entraînement et de test. En effet, certains jeux de données étiquetés sont disponibles publiquement. Dans le cadre de nos recherches, nous avons choisi d'utiliser le jeu de données CTU-13, présenté dans la section 2.6.

3.3 Mise en œuvre avec le jeu de données CTU-13

3.3.1 Présentation

Le processus d'entraînement nécessite des données correctement définies pour pouvoir apprendre certaines caractéristiques bien spécifiques à chaque famille de botnets et définir un modèle capable de détecter leur comportement. Par conséquent, nous avons sélectionné les scénarios marqués d'un astérisque dans le tableau 3.1, scénarios issus du CTU-13 présenté dans la section 2.6, pour générer nos différents modèles. Nous avons choisi ces scénarios en particulier pour l'entraînement, car chacun d'entre eux ne contient l'activité que d'un seul bot (`Neris`, `Rbot`, `Virut`, `Menti`, `Sogou` et `Murlo`) générant différentes attaques et tentant de communiquer avec son C&C serveur ou de se propager.

TABLE 3.1
Scénarios du CTU-13

Nmr/Id	Dur (hrs)	Bots	Bot	Size (GB)
42/1*	6.15	1	Neris	52
43/2	4.21	1	Neris	60
44/3*	66.85	1	Rbot	121
45/4	4.21	1	Rbot	53
46/5*	11.63	1	Virut	37.6
47/6*	2.18	1	Menti	30
48/7*	0.38	1	Sogou	5.8
49/8*	19.5	1	Murlo	123
50/9	5.18	10	Neris	94
51/10	4.75	10	Rbot	73
52/11	0.26	3	Rbot	5.2
53/12	1.21	3	NSIS.ay	8.3
54/13	16.36	1	Virut	34
*Scénarios utilisés pour la phase d'entraînement – Bot = *147.32.84.165*				

Les autres scénarios ont été utilisés comme jeux de test pour notre métamodèle, construit en combinant les différents modèles prédéfinis à partir des six scénarios d'entraînement. Chaque scénario contient des flux IP constitués de plusieurs champs, séparés par une virgule. Un échantillon des données contenues dans le scénario S42 est présenté dans le tableau 3.2.

TABLE 3.2
ÉCHANTILLON DE FLUX IP ISSUS DU SCÉNARIO S42

StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport
2011/08/10 11 :04 :24	0.002290	udp	195.64.148.210	61497	<->	147.32.84.229	13363
2011/08/10 11 :04 :24	0.003364	udp	178.125.149.171	13131	<->	147.32.84.229	13363
2011/08/10 11 :04 :24	3 241,145	udp	41.78.78.1	21849	<->	147.32.84.229	13363
2011/08/10 11 :04 :24	0.000278	udp	147.32.84.165	1025	<->	147.32.80.9	53
2011/08/10 11 :04 :24	0.206033	udp	147.32.86.148	38051	<->	147.32.80.9	53

State	sTos	dTos	TotPkts	TotBytes	SrcBytes	Label	
CON	0	0	2	532	472	flow=Background-UDP-Established	
CON	0	0	2	527	467	flow=Background-UDP-Established	
FSPA_FSPA	0	0	2	203	64	flow=From-Botnet-V42-UDP-DNS	
CON	0	0	18	3 235	675	flow=Background-UDP-Established	
CON	0	0	2	540	72	flow=To-Background-UDP-CVUT-DNS-Server	

Pour pouvoir mener nos travaux, nous avons développé un analyseur syntaxique et employé la suite logicielle *Weka Toolbox* [78] présentée dans l'annexe A.2. Ces outils nous ont permis de : *(i)* agréger les données de chacun des scénarios étudiés en flux de trafic *(ii)* sélectionner l'algorithme le plus à même de détecter les anomalies qui nous intéressent dans le cadre de nos recherches *(iii)* déterminer les réglages les plus appropriés afin d'obtenir les meilleures performances *(iv)* combiner les différents modèles générés et *(v)* valider notre métamodèle.

3.3.2 Agrégation en flux de trafic & Extraction des caractéristiques

Les *flux IP* doivent être agrégés en *flux de trafic* pour pouvoir en extraire les modèles spécifiques et détecter les activités malveillantes. Pour mettre en évidence cette étape importante, nous avons expérimenté deux approches différentes à l'aide de deux parsers écrits en `Perl` : *(i)* un parser *basique* permettant de transposer *directement* ces flux IP et *(ii)* un parser dit *avancé* utilisé pour agréger ces flux IP et en déduire les *flux de trafic*.

3.3.2.1 Analyse basique

Une première approche naïve consiste simplement à parser les données brutes afin d'en vérifier leur format et supprimer les éventuelles trames non conformes puis à transposer les fichiers CSV du CTU-13 en fichiers ARFF (voir annexe A.2) décrivant simplement les flux IP mais en ne conservant que les champs significatifs et sans y apporter aucun enrichissement. Pour ce faire, nous avons écrit le parser 'basic.pl' [79] dont le fonctionnement est présenté par l'algorithme 1. Un échantillon des données générées suite à cette analyse basique est présenté dans le tableau 3.3.

TABLE 3.3
ÉCHANTILLON DU FICHIER 'BASIC.ARFF'

Dur	Proto	SrcAddr	Sport	DstAddr	Dport	TotPkts	TotBytes	SrcBytes	Bot
20.020	tcp	147032084059	58542	209085149113	80	31	12392	4802	false
3 001,041	udp	217080208116	3028	147032084229	13363	10	1241	931	false
0.001	udp	147032084165	1025	147032080009	53	2	203	64	true
0.206	udp	147032086148	38051	147032080009	53	2	540	72	false
Flux IP générés à partir du scénario S42									

Cette approche basique a été testée avec le scénario S42 [80] et l'algorithme `J48` [41], une des implémentations `Weka` des arbres de décision (DTs) présentés dans la section 2.4.2. Ces premiers résultats, détaillés dans le tableau 3.5, semblent très prometteurs avec des mesures de performances extrêmement flatteuses : **TPR, Recall, Precision et F-Measure = 1 et FPR = 0** ! Ces mesures sont expliquées à la section 2.4.3. Malheureusement, ces excellentes performances s'expliquent par le fait que l'algorithme `J48` sur-ajuste les données comme expliqué dans la section 2.4.4. En effet, les feuilles et donc la classification des données sont basées sur le port source et l'adresse IP du bot (147.32.84.165).

3.3.2.2 Analyse avancée

Pour éviter ce sur-ajustement et par conséquent de biaiser l'algorithme, nous avons développé notre parser avancé. Celui-ci vise à *agréger les flux IP en flux de trafic*, flux construits sur la base d'une clef primaire de type 4-uplet. En effet, les échanges client-serveur sont identifiés par l'adresse IP/port source et destination. Or, le port source est versatile et peut de ce fait évoluer. Nous avons donc choisi de construire notre clef d'agrégation (AgK) non pas à partir du port source mais avec le protocole de transport comme expliqué par [81], [82].

Notre 4-uplet d'agrégation 'AgK' est construit comme suit : 'SrcAddr-DstAddr :Dport_Protocol'. Il est utilisé par notre analyseur syntaxique avancé [83], dont le code est détaillé par l'algorithme 2, pour agréger l'ensemble des flux IP liés à une même transaction en un seul flux de trafic, et ce, pour l'ensemble des données brutes. Ces flux de trafic sont définis *uniquement* par le port destination, leurs caractéristiques numériques, les drapeaux TCP et la classe 'Bot' valant 'true' ou 'false'. Ils sont exploités par l'algorithme supervisé pour construire le modèle.

Un échantillon de ces flux de trafic générés à partir du scénario S42 est fourni dans le tableau 3.4.

TABLE 3.4
ÉCHANTILLON DU FICHIER 'ADVANCED.ARFF'

Proto	PSH	ACK	RST	SYN	FIN	Dport	TotReq	TotDur	TotPkts	TotBytes	TotSrcBytes	Bot
udp	0	0	0	0	0	13363	1	1995.481079	7	3407	209	false
tcp	0	0	0	1	0	48375	1	0.63595	3	192	132	false
udp	0	0	0	0	0	13363	1	0.001	2	550	78	false
udp	0	0	0	0	0	13363	1	230.991653	6	1205	1025	false
tcp	1	1	0	1	1	80	1	41.230164	12	2171	1099	false
udp	0	0	0	0	0	53	10	0.549831	20	1676	700	true
Flux de trafic générés à partir du scénario S42												

Les résultats obtenus avec le même algorithme `J48` à partir du même scénario S42 mais cette fois ci, prétraité avec notre analyseur syntaxique avancé, sont détaillés dans le tableau 3.6. Le modèle d'arbre de décision construit à partir de ces données agrégées en flux de trafic est composé de '18' feuilles et est de taille '35'. Les performances du modèle ainsi obtenu sont très bonnes avec les mesures suivantes : **TPR et Recall = 0.935, FPR = 0, Precision = 0.959, F-Measure et MCC = 0.947**.

TABLE 3.5
MATRICE DE CONFUSION BASIQUE

	Vrai	Faux
Vrai	28 122	4
Faux	2	2 584 844
99.999% des instances *S42* correctement classifiées !		

TABLE 3.6
MATRICE DE CONFUSION AVANCÉE

	Vrai	Faux
Vrai	2 154	150
Faux	91	694 143
99.965% des instances *S42* correctement classifiées		

La particularité la plus importante ici est que ce second modèle dit "avancé" ne fait référence à *aucune adresse IP ou port source*. Le tableau 3.7 résume le nombre total de flux IP et de trafic suite aux analyses de base et avancée ainsi que le nombre respectif de flux marqués comme étant malveillants avec les ratios correspondants.

TABLE 3.7
TOTAL FLUX IP & TRAFIC

Scénario	Flux IP Total/Bot (‰)	Flux Trafic Total/Bot (‰)	Coefficient de réduction
S42	2 612 972/28 126 (10.65)	695 538/2 305 (3.3)	3.8
S44	4 156 939/2 556 (0.61)	684 175/2 464 (3.6)	6
S46	121 994/749 (6.1)	43 116/65 (1.5)	2.8
S47	520 225/230 (0.442)	119 375/4 (0.034)	4.4
S48	106 298/59 (0.555)	40 725/13 (0.32)	2.6
S49	2 762 306/1 414 (0.512)	462 824/63 (0.14)	6
Nombre total flux/Nombre flux étiquetés comme bot (Ratio)			

3.3.3 Modélisation du trafic

3.3.3.1 Sélection de l'algorithme

`Weka` fournit de nombreux algorithmes et différentes implémentations pour chacun d'eux. Nous avons choisi de tester un réseau de neurones de type Multi-Layers Perceptron (MLP) sachant que ce type d'approche est plus adapté à l'analyse d'images, l'algorithme Naive Bayes (NB) et différentes implémentations d'arbres de décision. Les algorithmes disponibles pour construire ces arbres sont `J48` et plusieurs de ses dérivés, à savoir `J48Consolidate`, `DTGraft` et `BFTree`.

Des tableaux 3.7 à 3.10, nous pouvons déduire que : *(i)* nous n'avons pas pu extraire de modèle d'arbre fiable des scénarios S47 et S48 car ils ne comportaient pas assez de données étiquetées comme malveillantes avec respectivement seulement 4 (0,034‰) et 13 (0,32‰) flux de trafic attribués au bot *(ii)* l'algorithme `J48` est celui qui nécessite le temps de test minimum et *(iii)* fournit les meilleurs *MCC* et *Spécificité* par rapport aux autres implémentation d'arbres de décision.

TABLE 3.8
Performances des algorithmes

Scénario	J48	NB	MLP
S42	0.93/1.00/**0.95**	**0.99**/0.89/0.16	0.54/1.00/0.79
S44	1.00/1.00/**0.99**	1.00/0.96/0.33	0.00/1.00/X
S46	0.30/1.00/**0.58**	0.11/0.98/0.03	0.00/1.00/X
S47	X	X	X
S48	0.00/1.00/X	**0.68**/0.95/ ?	0.00/1.00/X
S49	0.52/1.00/**0.68**	**0.93**/0.80.02/ ?	0.00/1.00/X
TPR/TNR/MCC			

TABLE 3.9
Temps Entraînement/Test

Scénario	J48	NB	MLP
S42	34.8/**0.04**	**2.5**/ ?	?/ ?
S44	20.2/**0.05**	**2.5**/0.36	708.6/0.12
S46	0.5/**0.00**	**0.1**/0.02	44.8/0.01
S47	2.0/**0.01**	**0.3**/0.06	127.6/0.02
S48	0.2/**0.00**	**0.1**/0.02	43.9/0.01
S49	13.8/**0.02**	**1.9**/0.19	500.5/0.09
Temps en secondes			

TABLE 3.10
TPR/MCC pour différentes implémentations d'arbres

Scénario	J48	J48Consolidate	DTGraft	BFTree
S42	0.93/**0.95**	**0.97**/0.73	0.93/0.94	0.89/0.93
S44	1.00/**0.99**	**1.00**/0.95	0.99/0.99	1.00/0.99
S46	0.31/**0.72**	**0.68**/0.19	0.26/0.52	0.04/0.48
S49	0.52/**0.68**	**0.65**/0.47	0.50/0.69	?
Tous les modèles ont un TNR égal '1'				

Nous avons donc choisi de construire notre métamodèle à partir de quatre arbres de décision, dans leur implémentation `Weka J48`, générés à partir des scénarios S42, S44, S46 et S49.

Ce type de modèle en arbre peut être utilisé pour des problèmes de classification ou de régression bien qu'il soit mieux adapté aux problèmes de classification. Il est appelé arbre de décision ou DT car, comme un arbre, il commence par le nœud racine (en vert), qui ensuite s'étend vers d'autres branches pour construire une structure arborescente décrite par la figure 3.4. Ces arbres permettent d'obtenir toutes les solutions possibles à un problème en fonction de conditions données.

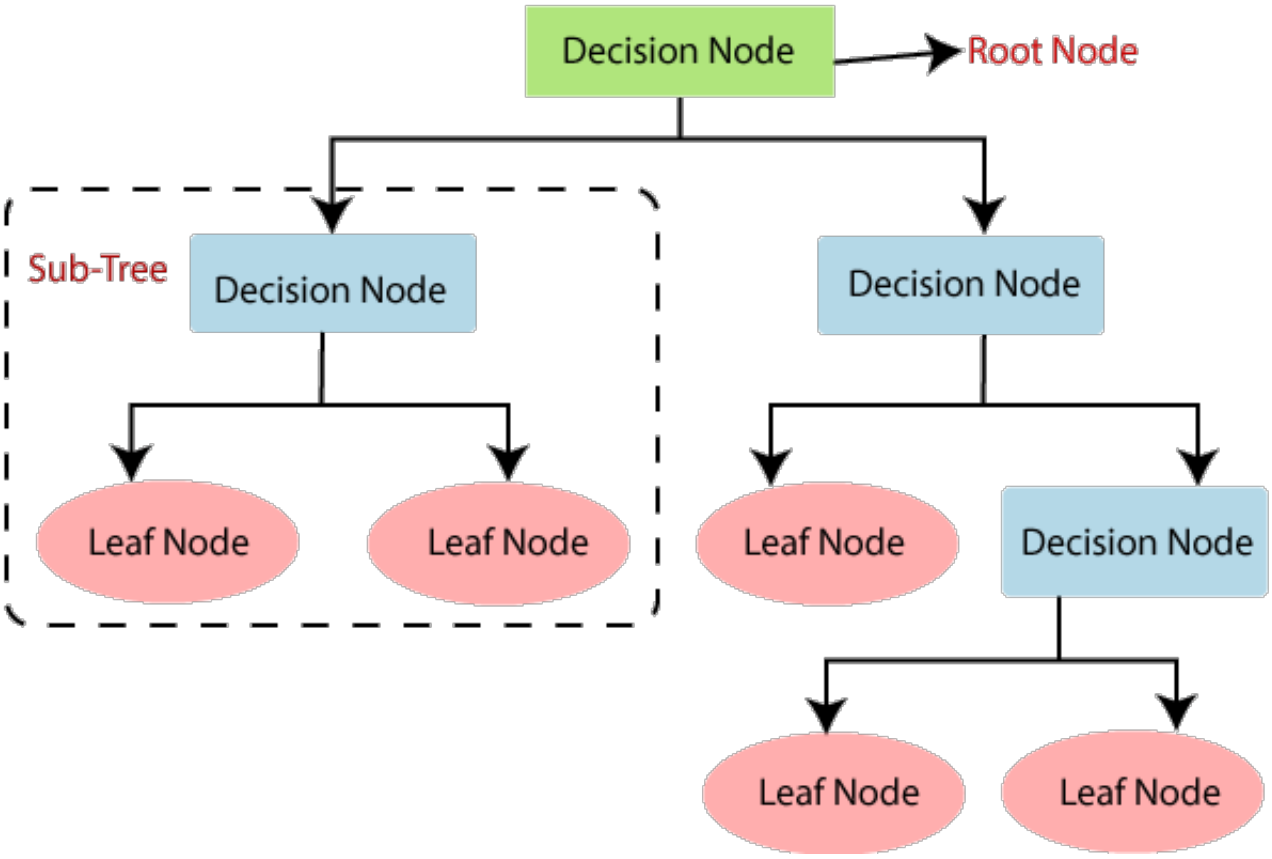


FIG. 3.4. Représentation d'un arbre de décision (DT)

Ils sont composés de deux types de nœuds à savoir les *décisions* et les *feuilles*. Il s'agit d'un classificateur structuré en arbre, où les nœuds internes représentent les caractéristiques d'un ensemble de données, les branches sont les règles de décision et chaque nœud feuille représente le résultat. Les nœuds de décision (en bleu) sont utilisés pour faire des choix et ont plusieurs branches, tandis que les nœuds feuilles (en rose) sont la réponse à ces décisions et ceux-ci ne contiennent pas d'autres branches. Les décisions sont prises sur la base des caractéristiques de l'ensemble des données en entrée.

3.3.3.2 Optimisation des modèles

L'arbre de décision est créé par l'algorithme d'apprentissage automatique. Selon son implémentation, un processus d'élagage supplémentaire est mis en œuvre pour déterminer quels nœuds peuvent être supprimés sans en affecter les performances. Un paramètre, le *Facteur de Confiance* (CF), est utilisé pour déterminer l'élagage de l'arbre. Cette étape réduit le risque de sur-ajustement du modèle aux données d'apprentissage, mais diminue également le nombre de décisions et de réponses possibles.

Un sur-ajustement excessif signifie que le modèle est capable de classifier parfaitement les données d'apprentissage, mais rien d'autre. Au lieu d'apprendre le concept sous-jacent, le modèle a appris les propriétés spécifiques des données d'apprentissage, ce qui le rend inutilisable avec d'autres données. Des valeurs plus petites entraînent un élagage plus important, mais également des variations du MCC. Par conséquent, des tests doivent être réalisés afin de déterminer le CF le plus adéquat. Les résultats de ces tests sont données par le tableau 3.11.

TABLE 3.11
MCC EN FONCTION DU CF

Scénario	**0.20**	**0.23**	**0.25**	**0.28**	**0.30**	**0.35**	**0.38**
S42	0.95	0.95	0.95	0.95	0.95	**0.95**	0.94
S44	0.99	0.99	0.99	0.99	0.99	**0.99**	0.99
S46	0.57	0.57	**0.57**	0.55	0.55	0.56	0.55
S49	0.68	0.70	0.70	0.70	0.70	**0.72**	0.70
Modulation du CF de 0.20 à 0.38							

Des résultats obtenus, nous pouvons conclure qu'un CF de '0.35' offre les meilleures performances bien que la documentation de *Weka* recommande un CF de '0.2'. Nous avons gardé un CF plus élevé afin d'obtenir des arbres plus précis, comportant un nombre plus important de nœuds et de feuilles. Mais, ces DTs plus profonds sont compensés lors de l'étape suivante.

3.3.3.3 Amélioration des modèles

Les étapes précédentes ont consisté à construire nos modèles d'arbres à partir de chaque scénario d'entraînement puis à les optimiser grâce à l'élagage. Des expériences réalisées, nous avons conclu que les meilleurs résultats étaient obtenus avec l'algorithme *J48* et un *Facteur de confiance* égale à '0.35'.

Parallèlement aux algorithmes classiques, ont été développés des "méta-algorithmes" comme *AdaBoost*, *Bagging* ou *IterativeClassifierOptimizer* [84]-[86] pour améliorer ou "booster" les classificateurs nominaux. Ces méta-algorithmes sont utilisés en conjonction avec d'autres algorithmes pour en améliorer leurs performances globales.

Le méta-algorithme *Bagging* ne nous a pas permis d'améliorer les performances de façon significative, nous avons alors choisi d'employer *AdaBoost* [87] pour le boosting des modèles. Le tableau 3.13 synthétise les résultats obtenus après avoir utilisé l'algorithme *J48*, boosté par *AdaBoost*. Ceci nous a permis d'améliorer le MCC par rapport aux résultats obtenus précédemment avec les modèles nominaux comme détaillés par le tableau 3.12.

TABLE 3.12
MODÈLES *J48* DE BASE

Scé.	TPR	FM	MCC	ETT	Taille*	Root
S42	0.932	0.945	0.945	36.5	35/18	Dp53
S44	0.997	0.993	0.993	17.6	25/13	Dp22
S46	0.308	0.455	0.517	0.5	33/17	Dp6774
S49	0.532	0.673	0.698	13.9	17/9	TotBytes
CF = 0.35 – *Nœuds/Feuilles						

TABLE 3.13
MODÈLES *J48* OPTIMISÉS

Scé.	TPR	FM	MCC	ETT	Precision
S42	0.931	0.947	0.947	625	0.962
S44	0.991	0.991	0.991	329	0.992
S46	0.385	0.543	0.596	22	0.926
S49	0.597	0.747	0.772	188	1.00
CF = 0.35 & `AdaBoost` – Elapsed Time Training (s)					

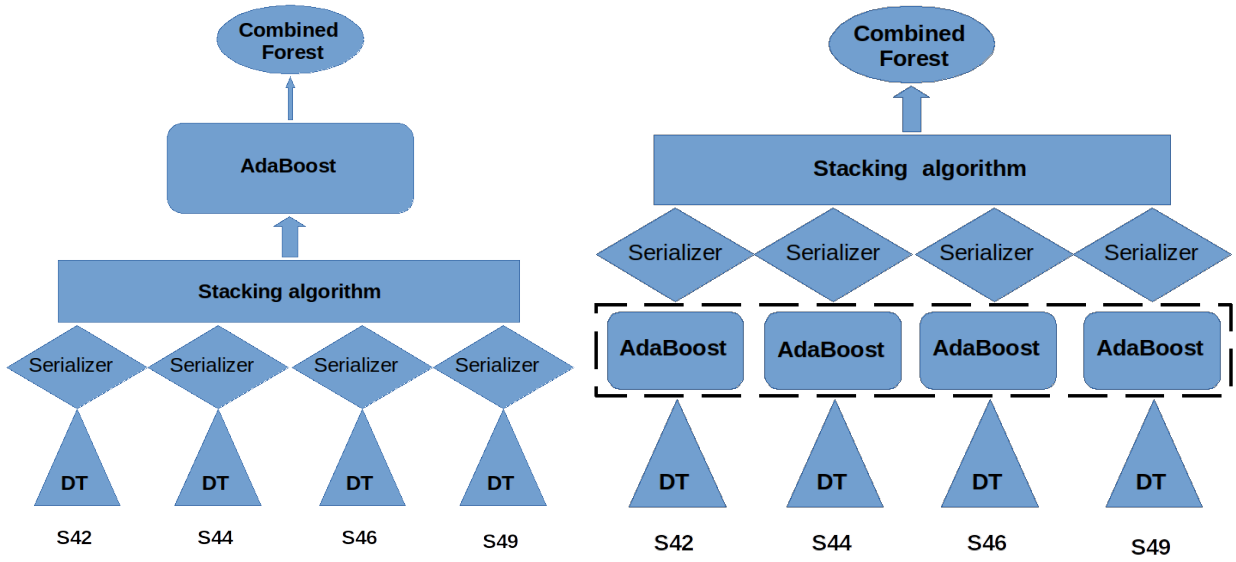
3.3.4 Construction de notre métamodèle

Une première implémentation de notre métamodèle avec l'algorithme d'amélioration positionné juste après la combinaison des modèles (figure 3.5a) n'a pas permis d'augmenter les performances de manière significative. Par conséquent, nous avons implémenté la phase de boosting juste après la définition des modèles.

Il existe également d'autres méta-algorithmes comme *Vote* ou *Stacking* [88], [89] destinés à assembler les classificateurs basiques. Pour construire notre métamodèle, nous avons choisi de combiner les différents arbres de décision prédéfinis à l'aide du méta-algorithme *Stacking* proposé par [90] car l'algorithme *Voting* n'était pas adapté à notre modèle. En effet, pour chaque scénario de test, seul un arbre est capable de détecter l'activité des botnets. Par conséquent, lors de la phase de test, le résultat du vote était toujours de 1 contre 3 ce qui avait pour conséquence que notre "Forêt combinée" considérait qu'il n'y avait pas d'activité réseau malveillante.

Enfin, pour pouvoir charger des modèles prédéfinis dans `Weka`, il faut qu'ils soient préalablement encapsulés, sérialisés à l'aide d'un "wrapper". Pour ce faire, un connecteur générique nommé *Serializer* est nécessaire. Ce classificateur charge un modèle puis le sérialise avant de l'appeler pour déterminer les prédictions comme présenté par la figure 3.5.

Le schéma complet de notre métamodèle "Forêt combinée" est représenté par la figure 3.5b.



(a) Première version (b) Forêt combinée

FIG. 3.5. Notre métamodèle

3.3.5 Validation de notre métamodèle

Pour tester notre "Forêt combinée", nous avons utilisé les sept autres scénarios du CTU-13, à savoir les scénarios S43, S45 et S50 à S54. Comme détaillé dans le tableau 3.1, le S43 matérialise l'activité d'un botnet constitué du bot `Neris`, S45 d'un botnet avec un bot `Rbot`, S50 un botnet constitué de dix bots *Neris*, S51 et S52 représentent des botnets constitués de respectivement dix et trois bots *Rbot*, S53 un botnet avec trois bots `NSYS.ay` et enfin, le S54 est un botnet `Virut` d'un seul bot.

Le tableau 3.14 synthétise l'ensemble des résultats obtenus par notre métamodèle. Des différentes mesures de performance pour chacun des scénarios, nous en avons calculé la moyenne et l'écart-type afin de déterminer les performances globales de notre approche à savoir :

TPR = 0,374/0,123, FM = 0,528/0,121, MCC = 0,595/0,089 et Precision = 0,976/0,054.

La mise en parallèle avec les résultats obtenus par les différents modèles pris isolément sur la base des données d'entraînement détaillés dans le tableau 3.13, nous permet d'affirmer que les performances de notre métamodèle sont très bonnes avec une *Précision* proche de '0.98' et un *F1-Score* de '0.53'. Pour comparaison, le tableau 3.15 synthétise les résultats de notre "Forêt combinée", mais sans le boosting apporté par le méta-algorithme.

TABLE 3.14
MESURES DE PERFORMANCE

Scé.	TPR	FM	MCC	Precision	ETT (s)
S43	0.324	0.489	0.569	1.00	5
S45	0.200	0.333	0.447	1.00	2.5
S50	0.558	0.678	0.693	0.866	5
S51	0.457	0.627	0.676	1.00	3
S52	0.333	0.5	0.577	1.0	0.5
S53	?	?	?	?	2
S54	0.374	0.54	0.609	0.991	5
métamodèle "Forêt combinée"					

TABLE 3.15
MESURES DE PERFORMANCE **sans** ADABOOST

Scé.	TPR	FM	MCC	Precision	Flux Trafic Total/Bot
S43	0.356	0.520	0.586	0.965	427 810/309
S45	0.133	0.222	0.298	0.667	201 329/15
S50	0.471	0.636	0.678	0.979	436 460/3 303
S51	0.395	0.557	0.610	0.941	236 082/81
S52	0.333	0.5	0.577	1.0	43 330/9
S53	?	?	?	?	92 482/977
S54	0.414	0.577	0.628	0.954	370 614/302
métamodèle "Forêt combinée" sans amélioration					

Il est également intéressant de noter, mais surtout important de préciser que notre "Forêt combinée" n'a pas été en mesure de détecter l'activité générée par le botnet `NSYS.ay` (scénario S53). Ceci est parfaitement normal, car *aucun arbre de décision* n'a été *modélisé* pour reconnaître ce type de trafic ce qui, bien entendu, pénalise les performances globales. De plus, ces résultats auraient pu encore être améliorés si les scénarios utilisés avaient eu un ratio 'Flux de trafic total/Flux de botnet' plus élevé.

Pour compléter l'analyse, à partir des flux de trafic générés par notre parser avancé, nous avons pu mettre en évidence le fait que les bots *Neris* (S42) échangent soit avec le protocole TCP sur le port 6667 soit en UDP sur le port 53 (service DNS). Nous avons également matérialisé que le canal C&C de *Rbot* (S44) est basé sur le protocole ssh (TCP/22). Les échanges intrinsèques de ces deux types de botnets sont représentés par la figure 3.6.

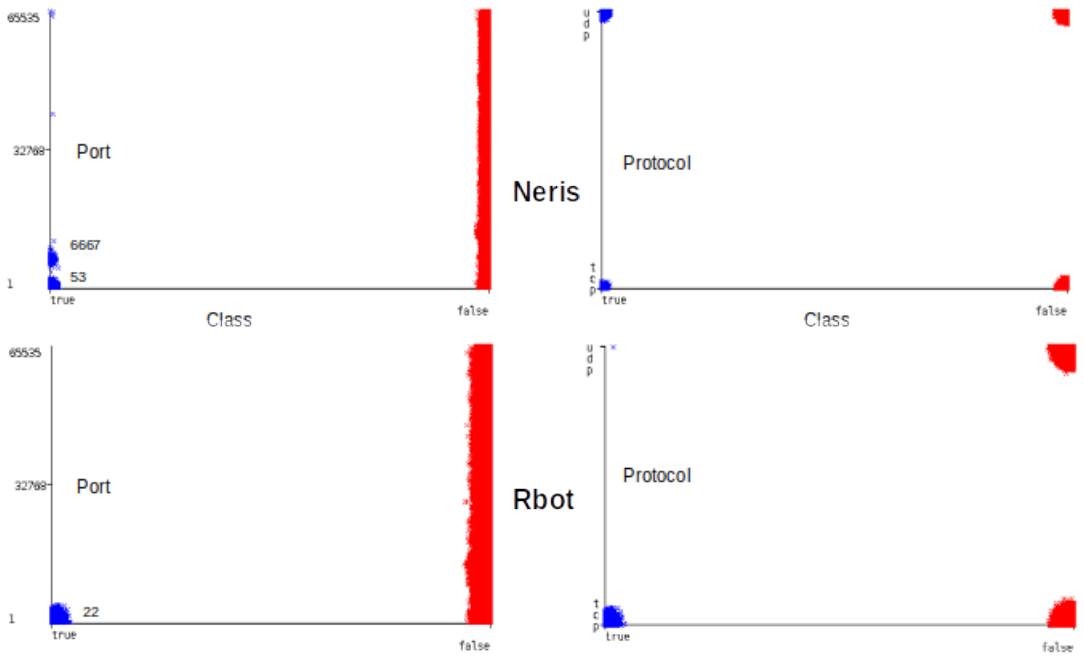


FIG. 3.6. Activité de *Neris* & *Rbot*

Les mêmes projections nous montrent que *Virut* échange en utilisant TCP sur les ports 80, 443 et 6667 – une porte dérobée bien connue – ou en UDP sur le port 53 comme *Neris*. Une analyse du code source de ces bots nous permettrait de le confirmer. Nous pouvons en déduire que *Neris*, *Rbot* et *Virut* mixent leurs échanges avec du trafic DNS ou `ssh` pour contourner les pare-feux et masquer leurs canaux de Command & Control afin d'en empêcher la détection.

3.4 Conclusion

3.4.1 Contribution

Notre approche de détection des botnets consiste non seulement à mettre en lumière leurs activités malveillante mais également leurs échanges via les canaux de C&C. Celle-ci n'emploie pas les outils traditionnels comme `R/MLR3` ou `Python/Scikit-learn`, présentés dans les annexes A.3 et A.4.

Pour construire notre métamodèle, nous avons : *(i)* développé en `Perl` deux analyseurs syntaxiques, un premier dit "basique" que nous avons rapidement écarté et un second dit "avancé" qui a été utilisé pour agréger les flux IP contenus dans les fichiers de capture (scénarios) du CTU-13 en flux de trafic mis sous forme de fichiers ARFF plus petits afin de pouvoir les exploiter avec `Weka` *(ii)* cette plateforme

`Java` nous a permis de tester plusieurs algorithmes d'apprentissage automatique supervisé afin de déterminer le plus approprié *(iii)* quatre arbres de décision ont été construits à l'aide de l'algorithme `J48` et ce, à partir de quatre scénarios d'entraînement *(iv)* ces DTs ont été élagués avec un *Facteur de Confiance* de '0,35' afin d'optimiser les modèles *(v)* ces DTs ont ensuite été améliorés à l'aide du méta-classificateur *AdaBoost* présenté dans la section 2.4.2 *(vi)* pour terminer, ces arbres de décisions *optimisés* et *améliorés* ont été combinés à l'aide de la méthode *Stacking*.

3.4.2 Analyse

L'agrégation en flux de trafic des flux IP utilisés pour l'apprentissage, via sur une clef primaire composée des quatre éléments que sont les adresses IP source et destination, le port de destination et le protocole, a permis de mettre en évidence le canal C&C de différentes familles de botnet. La dernière colonne du tableau 3.12, intitulée 'Root', indique la racine de chaque modèle.

Notre modèle est capable de mettre en évidence les échanges intrinsèques et le canal C&C d'une famille de botnets, bien entendu, si un modèle de décision correspondant a été généré et combiné. Le tableau 3.14 montre que l'activité des bots du scénario S53 n'a pas été détectée car aucun scénario d'entraînement correspondant à cette famille n'a été exploité.

À contrario, notre approche consistant à détecter les échanges intrinsèques et les canaux C&C des botnets est plus efficace que la mise en évidence des attaques elles-mêmes qui reste difficile et souvent trop tardive. De plus, notre métamodèle peut facilement être amélioré pour détecter de nouvelles familles de botnets en l'amendant avec de nouveaux modèles sans avoir à reconstruire l'ensemble.

Chapitre 4

Framework de détection d'anomalies par signature numérique

4.1 Introduction

4.1.1 Contexte

Le spectre des menaces informatiques est très vaste. Nous avons vu qu'il existe divers types de cyberattaques comme le vol de données personnelles [24] ou d'informations d'identification dans le but de propager des logiciels malveillants [27], les attaques DDoS [91] ou encore les campagnes de pourriel pour tenter d'accéder au système avec un compte ayant des privilèges plus élevés [92]. Ces menaces sont essentiellement d'origine *interne* au réseau. Outre ces cyberattaques, il est possible que d'autres menaces tout aussi sournoises voire néfastes surviennent et se manifestent, elles aussi, sous forme de variations de l'activité réseau. Ces modifications peuvent indiquer des pannes ou des événements particuliers tels des déploiements de mises à jour ou une augmentation/diminution anormale du nombre d'utilisateurs à un instant donné. Ces événements également significatifs et parfois révélateurs de phénomènes particuliers en cours sont, par conséquent, importants à détecter. Ces phénomènes, qu'ils soient internes ou externes, ont en commun le fait que des flux réseaux sont échangés ou modifiés.

Pour anticiper et contrer ou tout du moins tenter d'atténuer les conséquences de ces événements ayant un impact sur le réseau et son activité, il est devenu primordial de pouvoir détecter les variations ou les *flux suspects* qu'ils induisent. Flux qui, par définition, ne correspondent pas à une activité dite "normale" au sein du réseau supervisé. Le but est alors ici d'analyser le trafic afin d'en extraire ces *anomalies*, matérialisées par la modification de l'activité réseau consécutive à ces événements.

Ces anomalies réseaux peuvent être définies comme un "comportement qui s'écarte de ce qui est normal, standard ou attendu". Pour pouvoir les détecter, il faut alors avoir défini ce qu'est un comportement "normal" ou attendu. Des travaux basés sur la surveillance du trafic réseau, l'analyse des en-têtes des messages échangés ou encore du contenu de leur corps (DPI [1]) ont été menés par [48] et [93]. Ces méthodes d'analyse du trafic réseau nécessitent de déployer des sondes telles que des IDS [2] ou des ADS [3] comme décrit par [94] et [95]. Ces approches présentent cinq inconvénients majeurs :

1. Le processus de détection des intrusions est basé sur des règles élaborées à partir de caractéristiques ou d'attributs réseaux de base comme les méthodes HTTP, les ports sources ou destinations ou encore les drapeaux TCP. De surcroît, les règles doivent être mises à jour en fonction de l'évolution du réseau ou des cybermenaces.
2. L'utilisation de méthodes basées sur la DPI peut induire des problèmes liés à la préservation de la vie privée des utilisateurs.
3. L'ADS implique la nécessité d'entraîner le système à partir d'une référence de base normalisée, puis de comparer l'activité par rapport à cette référence. Ce n'est que lorsqu'un événement sort de l'ordinaire qu'une alerte est déclenchée.
4. Chaque solution est déployée en des points particuliers ou de défaillance unique (SPoF) spécifiques qui doivent être bien sélectionnés.
5. Chaque SPoF ou point de prélèvement est isolé des autres et n'est donc pas relié à ses homologues. De ce fait, les données collectées doivent être agrégées et corrélées par un système externe afin de mettre en évidence des flux relatifs à une même attaque.

Le premier point faible peut être atténué en souscrivant à un service de mises à jour ou de veille. Mais vous serez toujours dépendant de la fréquence, de la pertinence et surtout du fournisseur de ces mises à jour. La sélection des SPoF et la répartition des points de prélèvement n'est pas toujours intuitive ni exhaustive et il peut être parfois difficile de choisir convenablement et efficacement où les disposer en fonction de l'architecture ou de la complexité du réseau à superviser. Autre inconvénient majeur de ce type d'approche est l'absence de vue d'ensemble sur tous ces points de collecte.

1. Deep Packet Inspection
2. Systèmes de Détection d'Intrusion
3. Systèmes de Détection d'Anomalies

Certaines menaces ou attaques sont sournoises et la seule façon de les détecter est d'agréger les flux provenant de différents IDS. Pour ce faire, il est possible de déployer une plateforme efficace de gestion des incidents et des événements de sécurité (SIEM), qui est un ensemble de composants de cybersécurité utilisés pour surveiller le trafic et les ressources du réseau en collectant des indicateurs de performance clefs (KPI) de bas niveau. Ensuite, le centre des opérations de sécurité (SOC) analyse ces remontées alarmes pour valider l'anomalie et confirmer la menace de sécurité [96].

La détection des anomalies est primordiale pour identifier des comportements déviants ou inattendus parfois matérialisés par des valeurs aberrantes dans les données ou "outliers". Pour ce faire, des techniques de détection basées sur l'apprentissage automatique sont de plus en plus déployées. Plusieurs approches ont été proposées dans la littérature. Par exemple, les auteurs de [97] ont proposé un algorithme de classification du trafic réseau et de détection des anomalies. Un algorithme de reconnaissance a également été utilisé pour valider le modèle. Les auteurs de [98] ont proposé une solution basée à la fois sur une structure de réseau neuronal conventionnel résiduel profond (CNN) pour la modélisation d'un ensemble de données et sur l'apprentissage par transfert pour la création du modèle afin de détecter des anomalies dans les systèmes de contrôle industriels. Ces techniques d'apprentissage automatique (ML) peuvent également être utilisées pour détecter les valeurs aberrantes dans les données transmises par des capteurs. Les auteurs de [99] ont décrit et catégorisé plusieurs techniques d'apprentissage automatique pour pouvoir détecter des valeurs aberrantes dans l'Internet des Objets (IoT). Enfin, dans leurs travaux respectifs, les auteurs de [100], [101] ont proposé un cadre de travail basé à la fois sur un algorithme d'apprentissage profond et sur l'algorithme *K-Means*.

4.1.2 Contribution

Contrairement à notre approche *supervisée* présentée et utilisée dans le chapitre 3, l'apprentissage *non supervisé* ne requiert ni données labellisées, ni phase d'entraînement préalable pour la création d'un modèle. Comme présenté dans section 4.1.1, à partir des données générées lors des différents échanges au sein d'un réseau informatique, diverses techniques utilisant du ML non supervisé ont été mises en œuvre afin de mettre en lumière des activités réseaux anormales ou particulières.

Dans le cadre du projet CANCAN [54], nous avons été amenés à analyser des données brutes réelles *non labellisées*. Par conséquent, pour pouvoir détecter des éventuelles anomalies, une approche basée sur de l'apprentissage automatique non supervisée s'est tout naturellement imposée à nous.

Comme expliqué dans la section 2.5, les algorithmes non supervisés sont employés à des fins de regroupement des données. Ces regroupements sont calculés par l'algorithme sur la base des différentes caractéristiques numériques constituant les données analysées.

S'agissant de notre contribution, nous cherchons à détecter des anomalies relatives à des changements de comportement utilisateurs en fonction de leur utilisation des applications mobiles. Cette détection est donc réalisée à l'aide d'algorithmes non supervisés et sur la base des champs décrits dans section 2.7. Pour ce faire, nous avons défini le concept de *signature numérique*. Celui-ci pourrait être assimilé à une "photographie" de l'activité réseau prise à un instant précis et pour un emplacement, segment ou secteur particulier du réseau. Grâce à ces signatures numériques, notre approche baptisée "Framework de détection d'anomalies par signature numérique" ne requiert ni mise à jour, ni sonde ou encore de SIEM à la différence des travaux [48], [93], [94].

Notre méthode de détection consiste à : *(i)* agréger les flux de données réseaux caractérisant l'activité des utilisateurs non pas dans leur globalité, mais en délimitant ce que nous avons dénommé des secteurs *(ii)* scinder les données agrégées en différentes périodes de temps de même durée (tranches) et ce, pour chacun des secteurs réseaux définis précédemment *(iii)* classifier les données constituant ces tranches à l'aide d'un algorithme de regroupement non supervisé *(iv)* comparer la répartition des données au sein des différents clusters par rapport aux autres tranches. Si elle diffère, cela signifie qu'une anomalie relative à l'activité réseau a été mise en évidence *(v)* isoler les données relative aux anomalies détectées pour pouvoir les corréler avec des événements extérieurs.

Nos expériences ont été menées avec les données CANCAN relatives à l'*Île-de-France*. Les flux réseaux correspondants aux différents sites d'accès radio couvrants cette zone sont répartis en vingt-cinq fichiers CSV et triés par nom des sites dans l'ordre alphabétique (df_A.csv à df_Z.csv car il n'y a aucun site relais dont le nom commence par 'X'). L'ensemble de ces fichiers représente un volume total de *184 Go* de données brutes. Celui-ci contient des coordonnées GPS qui nous ont permis de localiser avec précision les stations de base 2G et 3G à partir desquelles ont été collectées les données.

Les secteurs, permettant de regrouper les flux réseaux prétraités, ont été définis sur la base des coordonnées géographiques contenues dans le jeu de données. Ensuite, pour chaque secteur, les données correspondantes sont scindées en périodes de temps identiques. Puis, un algorithme de regroupement non supervisé est utilisé pour extraire les *signatures numériques*. Si la signature d'un secteur spécifique pour une période donnée diffère des autres, cela signifie qu'une anomalie a été mise en évidence.

De ces données, nous avons pu extraire des anomalies caractérisant une modification du comportement des utilisateurs sur le réseau mobile. C'est-à-dire, un changement dans le type d'application mobile utilisé à un instant donné. Notre approche nous a permis de détecter des anomalies que nous avons pu corréler avec des événements externes et donc expliquer.

Ce chapitre 4 présente notre "Framework de détection d'anomalies par signature numérique". Décrit dans la section 4.2, celui-ci s'appuie sur de l'apprentissage automatique non supervisé pour générer des *signatures numériques* puis en extraire les *valeurs aberrantes*. Ensuite, dans la section 4.3, nous expliquons la mise en œuvre de notre framework avec les données CANCAN et présentons les résultats obtenus. Le projet dans lequel se sont inscrits nos travaux ainsi que le jeu de *données mobiles réelles* utilisé pour nos recherches ont été décrits dans la section 2.7. Enfin, la section 4.4 résume notre contribution, synthétise nos résultats tout en proposant des travaux ultérieurs pouvant être conduits et, enfin, propose une analyse critique de notre framework.

4.2 Description fonctionnelle

La figure 4.1 présente notre "Framework de détection d'anomalies par signature numérique". Celui-ci se décompose en quatre phases distinctes : *(i)* analyse et agrégation des données via des *analyseurs syntaxiques (ii)* extraction, à partir des données agrégées, des *signatures numériques* à l'aide d'un algorithme d'apprentissage automatique *non supervisé (iii)* détection des *éléments aberrants* à l'aide d'un second algorithme non supervisé *(iv)* corrélation des anomalies avec des *événements extérieurs*.

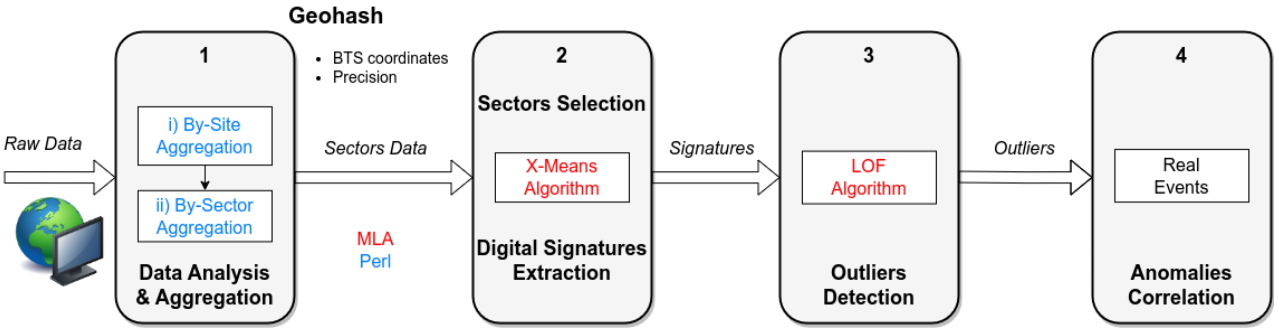


FIG. 4.1. Framework de détection d'anomalies par signature numérique

4.2.1 Analyse & Agrégation des données

Notre étude porte sur le trafic mobile capturé à partir des relais `Orange` couvrant la région de l'*Île-de-France*. Le jeu de données correspondant est présenté dans la section 2.7. Pour cette région uniquement, le volume total de données à analyser représente plus de 180 Go ! De ce premier constat, s'impose de façon évidente le fait que l'ensemble de ces données ne peut pas être traité et analysé directement sauf, bien-sûr, à disposer d'une infrastructure conséquente offrant suffisamment d'espace de stockage et de ressources tant en mémoire que de puissance de calcul pour répondre à ce besoin.

Cette nécessaire phase de prétraitement a donc pour but d'analyser, vérifier, nettoyer les données brutes puis de les enrichir pour pouvoir ensuite les agréger en des flux réseaux. Cette phase vise notamment à réduire le volume total de données à analyser, et ce, sans perte d'informations, pour pouvoir ensuite en extraire les comportements sous-jacents des utilisateurs.

Cette phase d'agrégation est réalisée en deux étapes distinctes : *(i) fusionner par station de transmission de base* (BTS) les données réseaux brutes capturées depuis les différentes antennes constituant ces BTS : c'est l'étape dite d'*agrégation par site (ii) agréger les flux réseaux* correspondant au trafic réseau généré et capturé à partir de chaque site pour définir l'activité au niveau de ce que nous avons nommé des *secteurs* : il s'agit de l'étape d'*agrégation par secteur*.

Agrégation par site

Cette première étape d'*agrégation par site* consiste donc à trier, nettoyer, transposer, enrichir et fusionner les données réseaux brutes capturées depuis chacune des BTS déployées par `Orange` en *Ile-de-France*. Ces données, capturées dans le cadre du projet CANCAN [54], sont mises à disposition sous forme de 25 fichiers '.csv'. Les données contenues dans ces fichiers ont été capturées à partir de chacune des antennes 2G et 3G installées au niveau de chaque *station de transmission de base*. En effet, une BTS peut comporter plusieurs antennes radio.

Nous commençons par fusionner les traces issues des antennes appartenant à la même station de base pour définir ce que nous avons appelé des *sites* et obtenir de cette façon des *flux agrégés* représentant l'*activité réseau du site*. Le type d'activité réseau généré par les utilisateurs dépend des applications mobiles utilisées. Cette information est donnée par le champ 'PortApp'. Pour notre cas d'usage, l'application utilisée est une information trop fine. Nous avons, de fait, choisi de focaliser

notre étude sur le groupe applicatif ou 'AppGroup'. En effet, nous avons simplement besoin de savoir si l'utilisateur navigue sur Internet (Web) ou envoie des messages électroniques (mail) mais pas de connaître le type de navigateur ou de client de messagerie employé. Cette étape de fusionnement des données brutes par site est représentée par les triangles rouges dans la figure 4.4.

Agrégation par secteur

Cette deuxième étape consiste à agréger en flux réseaux les données fusionnées de chacune des BTS identifiées comme étant *voisines* afin de former des *secteurs*. Cette étape est représentée par le rectangle bleu dans la figure 4.4.

L'*agrégation par secteur* vise à pouvoir sélectionner certains secteurs ou zones spécifiques en vue d'une analyse plus détaillée. Cette étape d'agrégation des données fusionnées issues des sites voisins pour former des secteurs présente trois difficultés :

1. Comment définir les secteurs (taille, forme, sur quelles bases) ?
2. Comment savoir si des sites sont voisins (distance) ?
3. Comment déterminer si le site doit être inclus ou pas dans le secteur (étendue du secteur) ?

Un secteur est par conséquent une zone constituée de plusieurs BTS proches l'une de l'autre et vue comme une seule et même source. Une première approche naïve permettant de déterminer les BTS voisines serait de calculer les distances séparant chacune des BTS. Le problème reviendrait alors à calculer la distance entre tous les arrangements possibles de l'ensemble des BTS pris par paire. Le nombre d'arrangements $\mathcal{A}$ d'un ensemble $\mathcal{E}$ comprenant n éléments pris k à la fois est donné par la équation (4.1).

$$A_n^k = \frac{n!}{(n-k)!} \tag{4.1}$$

Dans le cadre de nos travaux, $n = 2736$ et $k = 2$ (voir section 4.3.2) ce qui nous donne $\frac{2736!}{2734!}$ et génère un nombre infini de combinaisons qui est, par définition, impossible à évaluer ! Pour répondre à ces trois difficultés, nous avons opté pour le système `Geohash` [102]. Il s'agit d'un *système de géocodage* inventé en 2008 par Gustavo Niemeyer puis reversé au domaine public qui se base sur les coordonnées GPS pour encoder un emplacement géographique en une courte chaîne de lettres et de chiffres. Il s'agit d'une *structure de données spatiales hiérarchiques* qui subdivise la surface terrestre selon une grille hiérarchique comme représentée par la figure 4.2. Les geohashes offrent des propriétés telles que

la précision arbitraire et la possibilité d'éliminer progressivement les caractères de la fin du code pour réduire sa taille, en perdant peu à peu de la précision. Conséquence de la dégradation progressive de précision, les lieux à proximité présentent souvent (mais pas toujours) des préfixes similaires : plus un préfixe partagé est similaire, plus les deux endroits sont proches.

FIG. 4.2. Système `Geohash` – La France est dans le secteur 'u0'

Une limitation de l'algorithme du `Geohash` apparaît lorsque, justement, il est utilisé pour trouver des points situés à proximité les uns des autres en se basant sur un préfixe commun. Il existe des cas limites où des endroits proches, mais situés sur les côtés opposés de l'équateur ou d'un méridien, peuvent générer des codes geohashes sans préfixe commun. Ce qui n'est pas notre cas ici.

Afin de déterminer le geohash d'un point, il faut disposer de ses coordonnées de latitude et de longitude. Or, les coordonnées disponibles dans le jeu de données CANCAN sont données dans le système `LambertII`. Donc, pour pouvoir calculer le geohash associé à chaque site, nous avons converti les coordonnées dans le système GPS à l'aide d'un convertisseur en ligne [4].

Avec le système `Geohash`, la zone définissant une cellule dépend de la longueur du geohash associé (voir figure 4.3). Cette longueur est également connue sous le nom de *précision*. Plus la précision du geohash est élevée, plus la zone correspondante est petite comme exposée par les figures 4.3a à 4.3c. Pour déterminer la précision la plus adéquate, différentes longueurs de geohashes allant de

4. `https://geofree.fr/gf/coordinateconv.asp`

5 à 7 caractères ont été testées. Un geohash de 6 caractères a été choisi parce que ses dimensions correspondent le mieux à des lieux d'étude connus, bien définis et compréhensibles comme *Notre-Dame de Paris* (N-d-P) ou le *Stade de France* (S-d-F). Une précision de '6' définit une cellule d'une hauteur de 1,2 km et d'une largeur d'environ 0,6 km comme présenté par la figure 4.3b.

Le tableau 4.1 détaille les dimensions des cellules définies par le système `Geohash` en fonction de la précision (longueur) utilisée.

(a) Précision 5 (u09wm) (b) Précision 6 (u09wmd) (c) Précision 7 (u09wmdw)

FIG. 4.3. Cellules du système `Geohash` en fonction de la Précision

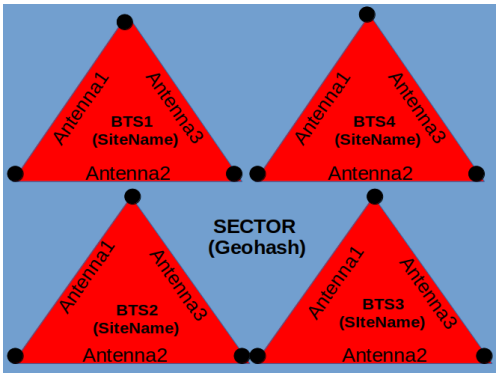


FIG. 4.4. Agrégations par site & secteur

TABLE 4.1
PRÉCISIONS & DIMENSIONS

Précision	Longueur	Largeur
1	5 000	5 000
2	1 250	625
3	156	156
4	39	19.5
5	4.9	4.9
6	1.2	0.6
7	0.153	0.153
8	0.038	0.019
Distances en km		

Cette première phase d'enrichissement des données et d'agrégation par sites puis par secteurs, et ce, tout en conservant les caractéristiques réseaux des flux échangés, nous permet ensuite d'extraire de ces données agrégées détaillées ce que nous avons nommé les "signatures numériques". Signatures qui nous permettrons, lors des phases suivantes, de mettre en évidence les anomalies réseaux.

4.2.2 Sélection des secteurs & Extraction des signatures numériques

Cette deuxième phase consiste, pour des *secteurs particuliers*, à regrouper les données agrégées lors de l'étape précédente pour constituer ce que l'on appelle des regroupements ou "clusters". Ce regroupement en clusters des données agrégées par secteur est répété périodiquement et ce pour des intervalles de temps de longueur équivalentes. Les données sont regroupées sur la base des attributs calculés lors de l'étape d'agrégation. La façon dont sont réparties les données, pour chaque période de temps, au sein des différents clusters est ce que nous avons appelé la *signature numérique* d'un secteur. Si, pour un même secteur, la signature d'une période diffère des autres, elle est considérée comme suspecte et met en évidence un comportement des usagers sur le réseau considéré comme *anormal*, car déviant des autres. Cette étape est réalisée à l'aide de l'algorithme non supervisé `X-Means`, une variante de l'algorithme de regroupement `K-Means` dont l'inconvénient est qu'il requiert de préciser le nombre attendu de clusters K avant de commencer le regroupement des données. Or, nous ne pouvons déterminer à l'avance combien de clusters seront constitués. Il existe différentes techniques permettant de déterminer la valeur optimale de K.

Comme présenté par la figure 4.5, `K-Means` commence par positionner aléatoirement des points centraux nommés *starter means* ou points moyens initiaux. Il associe ensuite au même groupe, les observations les plus proches de ces points initiaux puis calcule la moyenne des observations de chacun des groupes et déplace ces points moyens vers la nouvelle position calculée. Il continue à réaffecter les observations aux moyennes les plus proches et ainsi de suite jusqu'à convergence de l'algorithme. Pour garantir la stabilité des groupes ainsi définis, la sélection des points moyens de départ est répétée plusieurs fois afin d'éviter certains tirages initiaux qui pourraient aboutir à une configuration différente de la majorité des cas [45].

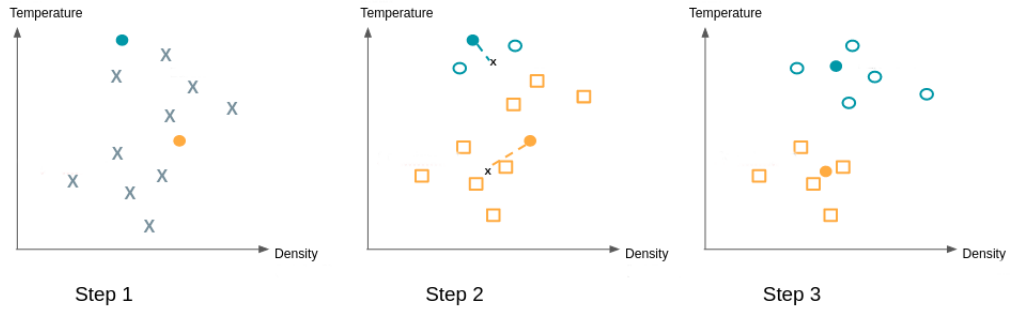


FIG. 4.5. Fonctionnement de K-MEANS

La variante `X-Means` permet d'affiner l'affectation des groupes en essayant par itérations successives de les subdiviser et en ne conservant que les meilleurs résultats [103]. La répartition des données dans les K clusters mutuellement exclusifs est réalisée de manière que les données de chaque regroupement soient aussi proches que possible les unes des autres, mais aussi éloignées que possible des données formant les autres regroupements. Chaque cluster est caractérisé par son point central nommé *centroïde*. Il s'agit du point dont les coordonnées sont obtenues en calculant la moyenne de toutes les coordonnées des points d'échantillonnage attribués à chaque cluster.

4.2.3 Détection des valeurs aberrantes

La troisième phase permet quant à elle la *détection des valeurs aberrantes*. Elle vise à détecter les anomalies réseaux en faisant ressortir parmi l'ensemble des données agrégées détaillées des valeurs aberrantes ou *outliers*. Elle est réalisée à l'aide l'algorithme non supervisé `Local Outlier Factors` [104]. LOF est basé sur le concept de la densité locale où la localité est donnée par les k voisins les plus proches et dont la distance, qui peut être Euclidienne, de Mahalanobis ou de Chebyshev [48], est utilisée pour estimer la densité. En comparant la densité locale d'un objet aux densités locales de ses voisins, il est possible d'identifier des régions de densité similaire et des points dont la densité est nettement inférieure à celle de leurs voisins. Ces derniers sont alors considérés comme des *outliers*.

4.2.4 Corrélation des anomalies

Une fois les valeurs aberrantes identifiées, la dernière phase de notre framework consiste à *corréler* les anomalies extraites avec des faits ou *événements externes* pouvant justifier ces variations dans l'activité réseau tels que des pannes ou des mouvements de foules. Cette étape permet de confirmer les anomalies et éventuellement de les expliquer.

4.3 Mise en œuvre avec le jeu de données CANCAN

4.3.1 Présentation

Pour cette étude, nous avons extrait du jeu de données les traces brutes générées entre le 16/03/2019 et le 14/06/2019 inclus. Elles ont été capturées par `Orange` toutes les trente minutes grâce à des sondes disposées au niveau de chaque antenne. Un extrait décrivant quelques lignes de ces données brutes

fournies dans la cadre du projet CANCAN, présenté dans la section 2.7, est donné par le tableau 4.2. Certaines valeurs ont été tronquées pour des raisons de confidentialité.

TABLE 4.2
EXTRAIT DES DONNÉES BRUTES CANCAN

PortApp	LocInfo	COORD_X	COORD_Y	SiteName	TimeSlot	nPktUp	nPktDn	Duration	Users	Flows
65734	7cdb04	A2142	D3023	U_PEL	2019-05-14 21 :00	49	180	58.83	1	1
66333	7cdb02	A2142	D3023	U_PEL	2019-05-14 11 :30	4	83	2.28	1	1
65759	7cdb05	A2142	D3023	U_PEL	2019-05-13 08 :30	10	468	630.40	1	2
65745	7cdb05	A2142	D3023	U_PEL	2019-04-11 16 :30	7	153	0.4	1	1
65730	7cdb05	A2142	D3023	U_PEL	2019-04-11 17 :30	28	378	375.82	1	1
65734	7cd202	A2172	D2971	U_TRI_S	2019-05-13 21 :30	98	35	45.84	1	2
66326	7cd205	A2172	D2971	U_TRI_S	2019-05-13 07 :30	7	174	195.50	1	1
65807	7cd204	A2172	D2971	U_TRI_S	2019-04-11 18 :30	5	2 532	2.33	1	1
66422	7cd202	A2172	D2971	U_TRI_S	2019-04-11 19 :00	148	26 538	893.94	1	8
66327	07a280	A2172	D2971	U_TRI_S	2019-04-11 10 :30	11	849	5.81	1	3
65701	01cc5d	A5471	D3054	UNION_F	2019-04-11 16 :30	1	97	24.33	1	1
65745	01cc5d	A5471	D3054	UNION_F	2019-04-11 07 :30	8	338	23.42	1	2
65745	01cc5d	A5471	D3054	UNION_F	2019-04-11 12 :30	13	338	51.46	1	3
66358	01cc5d	A5471	D3054	UNION_F	2019-04-11 09 :30	5	419	27.82	1	1
Données extraites du fichier 'df_U.csv'										

Comme précisé dans la section 2.7, les données brutes sont constituées de onze champs, séparés par une virgule. Du tableau 4.2, nous pouvons alors tirer les cinq observations suivantes :

1. La colonne 'PortApp' ne correspond pas à des numéros de ports au sens de la couche 4 (transport) du modèle OSI. En effet, certaines valeurs sont supérieures à '65 535'.
2. Les coordonnées X & Y, exprimées dans le système LAMBERTII, sont identiques pour un même nom de site (colonne 'SiteName').
3. Pour un même site, nous pouvons avoir des identifiants d'antenne différents (colonne 'LocInfo'). En effet, un site peut servir plusieurs antennes radio
4. Les données ont été capturées à intervalles réguliers de 30 mn.
5. Le fichier df_U.csv contient l'ensemble des données brutes des sites dont le nom commence par la lettre 'U'.

4.3.2 Analyse & Agrégation des données

Cette phase d'analyse et d'agrégation, constituée des étapes d'*agrégation par site* et d'*agrégation par secteur*, vise à réduire la quantité de données brutes à traiter par les algorithmes d'apprentissage automatique tout en conservant la structure sous-jacente de ces données pour en permettre ensuite une

meilleure compréhension et ainsi effectuer des analyses plus approfondies. Les analyseurs syntaxiques et outils utilisés lors de ces étapes sont librement disponibles sur notre dépôt [105]. Ces parsers ont tous été écrits en `Perl`. Il s'agit d'un langage de script disposant de fonctions évoluées de traitement et d'analyse de texte ainsi qu'un puissant moteur d'expressions régulières. `Perl` dispose également de capacités d'analyse de chaînes de caractères ce qui en fait un langage particulièrement efficace pour le traitement de gros fichiers de données.

Agrégation par site

Lors de cette étape, les données brutes sont vérifiées, nettoyées, enrichies puis agrégées par site. Cette analyse est réalisée en se basant sur les différents champs décrits dans la section 2.7 : *(i)* si le nombre de champs est différent de 11 (soit un champ est manquant soit un champ supplémentaire est détecté), la ligne est supprimée *(ii)* le type d'application 'PortApp' est converti en groupe applicatif 'AppGroup' *(iii)* les champs 'nPktUp' et 'nPktDn' sont sommés pour obtenir les valeurs de la colonne 'Packets' *(iv)* les données brutes sont agrégés par 'TimeSlot' et BTS à l'aide de la clef *2-uplet* suivante : 'TimeSlot ;SiteName'. Les caractéristiques numériques des flux échangés ('Duration', 'Users' et 'Flows') sont sommées ce qui permet de conserver l'ensemble des informations. La transposition du champ 'PortApp' en 'AppGroup' est effectuée en utilisant le dictionnaire détaillé ci-dessous. Celui-ci a été généré à partir d'une table de correspondance fournie par `Orange` : 0. **Unknown** 1. **Web** 2. **P2P** 3. **Download** 4. **CloudStorage** 5. **Mail** 6. **DB** 7. **Others** 8. **Control** 9. **Games** 10. **Streaming** 11. **Chat** 12. **voIP** 13. **MailOperator** 14. **VPN** 15. **VVM** 16. **MMS** 17. **StreamAVSP** 18. **Portal**

L'algorithme 3 détaille comment cette première étape d'agrégation a été réalisée. À l'issue de celle-ci, les données brutes sont fusionnées en flux réseaux par site, les ports applicatifs similaires ont été convertis en un même groupe applicatif, les nombres de paquets montants et descendants ont été sommés pour former le champ 'Packets' ainsi que les champs 'Users' et 'Flows' et les champs non utilisés ou devenus inutiles supprimés. Ces données agrégées sont exportées sous la forme de 25 fichiers ARFF (Attribute-Relation File Format). Il s'agit d'un format de type texte ASCII qui décrit une liste d'instances partageant un ensemble d'attributs. Un échantillon des flux réseaux obtenus, extrait du fichier 'I.arff', est présenté par le tableau 4.3.

Après agrégation, la quantité de données à analyser est passée de 184 Go à 1,2 Go ce qui représente un taux de réduction proche de '*160*'. Les détails de chacun des fichiers obtenus après cette étape sont répertoriés dans le tableau 4.5. Il n'y a pas de fichier 'X.arff' car aucun site ne possède un nom commençant par la lettre 'X'. Le nombre total de sites obtenus est de '2 736'. Le détail du nombre de sites par fichier '.arff' est donné par le tableau 4.4.

TABLE 4.3
Échantillon de données agrégées par site

TimeSlot	SiteName	COORD_X	COORD_Y	AppGroup	GroupDesc	Packets	Duration	Users	Flows
2019-03-16 12 :30	INVALIDES	597939	2428477	1	Web	1500925	207415	635	2193
2019-03-16 12 :30	INVALIDES	597939	2428477	10	Streaming	2753567	117630	211	1747
2019-03-16 12 :30	INVALIDES	597939	2428477	11	Chat	40595	30550	63	177
2019-03-16 12 :30	INVALIDES	597939	2428477	12	VoIP	6631	21	1	4
2019-03-16 12 :30	INVALIDES	597939	2428477	16	MMS	3136	385	10	11
2019-03-16 12 :30	INVALIDES	597939	2428477	17	StreamAVSP	3790	14	1	2
2019-03-16 12 :30	INVALIDES	597939	2428477	3	Download	327743	4228	40	80
2019-03-16 12 :30	INVALIDES	597939	2428477	4	CloudStorage	269794	4915	39	149
2019-03-16 12 :30	INVALIDES	597939	2428477	5	Mail	37530	25793	39	57
2019-03-16 12 :30	INVALIDES	597939	2428477	6	DB	70	242	1	3
2019-03-16 12 :30	INVALIDES	597939	2428477	8	Control	1557	560	4	5
2019-03-16 12 :30	INVALIDES	597939	2428477	9	Games	6408	543	2	15
Échantillon issu du fichier 'I.csv'									

TABLE 4.4
Nombre de sites par fichier '.csv'

Fichier	Site	Fichier	Site	Fichier	Site
df_A	150	df_I	28	df_Q	261
df_B	176	df_J	20	df_R	155
df_C	304	df_K	10	df_S	268
df_D	51	df_L	214	df_T	112
df_E	51	df_M	196	df_U	13
df_F	67	df_N	70	df_V	168
df_G	98	df_O	250	df_W	5
df_H	51	df_P	261	df_Y/Z	1/1
Nombre total de sites = 2 736					

TABLE 4.5
Taille des fichiers '.csv'

Fichier	Brutes	Agrégées	Fichier	Brutes	Agrégées
df_A	9.7G	65M	df_M	13G	80M
df_B	14G	76M	df_N	6.6G	33M
df_C	23G	129M	df_O	5.9G	81M
df_D	3.2G	20M	df_P	19G	108M
df_E	3.6G	21M	df_Q	1.9G	7.1M
df_F	5.3G	28M	df_R	13G	67M
df_G	6.7G	43M	df_S	18G	112M
df_H	4.0G	22M	df_T	3.8G	44M
df_I	3.0G	13M	df_U	681M	3.9M
df_J	1.5G	8.1M	df_V	13G	75M
df_K	881M	4.6M	df_W	211M	1.2M
df_L	18G	94M	df_Y/Z	70M/316K	476K/40K
25 fichiers '.csv' (données brutes) / '.arff' (données agrégées)					

Agrégation par secteur

Cette deuxième étape consiste à fusionner les flux agrégés par site lors de l'étape précédente pour en faire ressortir l'activité réseau, mais au niveau des secteurs. Cette agrégation par secteur requiert de conserver les détails concernant l'activité par groupe applicatif pour permettre une analyse plus

approfondie lors de la phase suivante. De plus, elle nécessite de déterminer quels sites sont voisins pour pouvoir les regrouper en secteurs. Ce processus de fusion a été mis en œuvre en utilisant la librairie `Geohash` du langage `Perl` [106] que nous avons dû "packager" et rentrer dans la distribution `Debian` pour l'occasion. Grâce aux champs 'Coord_X' et 'Coord_Y', préalablement transposés dans le système GPS, nous avons calculé le geohash correspondant pour chacun des sites.

Comme expliqué dans la section 4.2.1, les sites dont le geohash est identique sont voisins. A partir de cette information, nous avons pu fusionner les données de l'ensemble des sites ayant le même geohash et donc considérés comme voisins en fonction de la précision choisie. Pour ce faire, nous avons employé une clef d'agrégation sous forme d'un *3-uplet* construit comme suit : 'TimeSlot ; Geohash ; AppGroup'. Les données calculées sont transposées sous la forme suivante : 'TimeSlot,Geohash,Packets,Duration,Users,Flows' en utilisant l'algorithme 4 puis exportées en un seul fichier ARFF par précision, dont la nomenclature est 'sectors_<Précision>.arff', afin d'être exploitées lors de la phase suivante. Cette étape est réalisée par l'analyseur syntaxique lors du processus d'agrégation. Le tableau 4.6 détaille le nombre de secteurs obtenus en fonction de la précision du geohash généré pour chacun des sites. Nous constatons fort logiquement que le nombre de secteurs obtenus augmente avec la précision.

Nous avons effectué différents essais pour évaluer l'impact de la *Précision*, et donc du nombre de secteurs obtenus, sur la taille du fichier ARFF généré. Parallèlement, dans le but de réduire encore d'avantage la taille du fichier des activités utilisateurs par secteurs, nous avons ajouté une option dans le code de nos parsers afin de pouvoir agréger les données non plus par tranches de trente minutes, mais par heure entière. Suite aux différentes tests réalisés, nous avons pu constater que le rapport de réduction entre l'agrégation par heure entière et celle par demi-heure n'était pas suffisamment intéressant. De ce fait, nous avons choisi de conserver un découpage horaire par demi-heure. Les résultats obtenus pour différentes précisions sont résumés dans le tableau 4.7.

TABLE 4.6
NOMBRE DE SECTEURS & PRÉCISION

Précision	**5**	**6**	**7**	**8**
Nombre de secteurs	109	1 233	2 545	2 707

TABLE 4.7
TAILLE DU FICHIER & PRÉCISION

Précision	**5**	**6**	**7**	**8**
Taille du fichier	29	207	327	340
Taille du fichier*	16	139	246	258
Ratio	1,8	1,5	1,3	1,3
* Flux agrégés par secteur & par heure entière (Taille en Mo)				

Suite à cette deuxième étape d'agrégation, nous obtenons un seul ficher ARFF décrivant l'activité des utilisateurs en fonction du temps (champ 'TimeSlot') et ce, par secteur dont la taille dépend de la 'Précision'. Pour les raisons évoquées dans la section 4.2.1, nous avons choisi de créer des secteurs en encodant les coordonnées de chaque site en un geohash de 6 caractères et de fusionner les données des sites ayant le même geohash. Ceci nous a fourni 1 233 secteurs comme précisé dans le tableau 4.6. Les geohashes obtenus sont de forme rectangulaire et de dimensions 1.2 km par 0.6 km comme indiqué dans le tableau 4.1. Le tableau 4.8 est un extrait du fichier "sectors_6.arff" obtenu après l'étape d'agrégation en secteurs en utilisant une précision de 6 et des tranches de temps de trente minutes. L'algorithme 5 nous a permis de générer dynamiquement les entêtes ARFF en fonction de la précision choisie.

TABLE 4.8
Échantillon de données agrégées par secteur

TimeSlot	Geohash	AppGroup	GroupDesc	Packets	Duration	Users	Flows
2019-06-06 16 :00	u09wj0	8	Control	17133	10827	51	55
2019-06-06 16 :00	u09wj0	9	Games	189790	47859	76	326
2019-06-06 16 :30	u09wj0	1	Web	19384116	2466830	7171	22041
2019-06-06 16 :30	u09wj0	10	Streaming	21781836	1290829	2709	15166
2019-06-06 16 :30	u09wj0	11	Chat	1487864	196473	523	1229
2019-06-06 16 :30	u09wj0	12	VoIP	5367	2041	8	10
2019-06-06 16 :30	u09wj0	13	MailOrange	8367	2917	23	37
2019-06-06 16 :30	u09wj0	14	VPN	436	165	3	3
2019-06-06 16 :30	u09wj0	15	VVM	8500	3790	34	34
2019-06-06 16 :30	u09wj0	16	MMS	23141	1709	71	88
2019-06-06 16 :30	u09wj0	17	StreamAVSP	785709	2881	7	89
2019-06-06 16 :30	u09wj0	18	OrangePortal	8339	610	7	36
2019-06-06 16 :30	u09wj0	3	Download	4822262	103490	440	1083
2019-06-06 16 :30	u09wj0	4	CloudStorage	1624903	117040	518	1278
Extrait du fichier 'sectors_6.arff'							

4.3.3 Sélection des secteurs & Extraction des signatures numériques

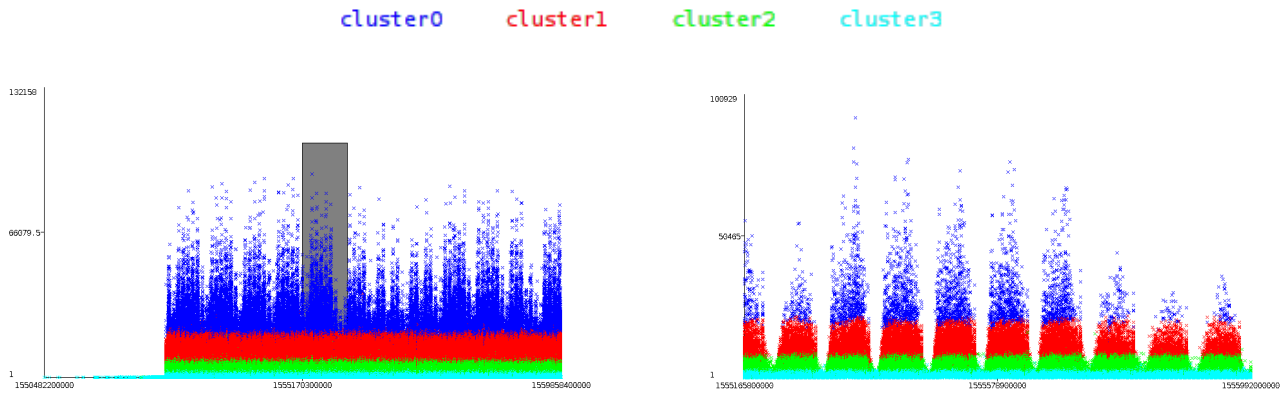
Une fois généré, le fichier 'sectors_6.arff' est chargé pour exploitation dans le logiciel `Weka`. Une première approche d'analyse de ces données agrégées par secteurs fut d'employer l'algorithme `X-Means` pour en comprendre leur répartition et essayer d'en extraire des tendances. `X-Means` nous a permis de classifier les secteurs sur la base des données caractérisant l'activité des utilisateurs, à savoir les champs 'TimeSlot', 'Packets', 'Duration', 'Users' et 'Flows' (créneau horaire, nombre de paquets, durée totale, nombre d'utilisateurs et de flux échangés). Pour ce faire, nous avons dû supprimer le champ 'GroupDesc' car `X-Means`, comme nombre d'algorithmes, n'est pas capable d'interpréter les données de type chaîne de caractères.

Pour réaliser cette première analyse par regroupement, nous avons choisi d'utiliser comme métrique la distance *Euclidienne*, car elle offre les meilleures performances en temps de calcul pour des résultats de répartition dans les clusters pratiquement identiques, comme détaillé dans le tableau 4.9. Il est intéressant de constater que l'algorithme `X-Means` répartit systématiquement les données dans *quatre* clusters, et ce, dans quasiment les mêmes proportions quelle que soit la métrique utilisée.

TABLE 4.9
RÉPARTITION DES DONNÉES PAR CLUSTER

Distance	**Clusters**				**Temps de calcul (s)**
	0	***1***	***2***	***3***	
Euclidean	1%	7%	28%	64%	45
Chebyshev	1%	8%	29%	62%	47
Manhattan	1%	7%	27%	65%	49
Minkowski	1%	7%	28%	64%	66
Nombre total de points = 2 997 893					

Puis, sur deux axes, nous projetons les quatre clusters obtenus avec en abscisse le champ 'TimeSlot' et en ordonnée le champ 'Flows' comme présentés par la figure 4.6. La figure 4.6a matérialise l'activité des utilisateurs pour l'ensemble de la région de l'*Île-de-France* sur trois mois (12 semaines). Nous remarquons que les points maximums de cette semaine sont plus élevés que les maximums des semaines précédentes ou suivantes. La figure 4.6b est une vue détaillée sur la troisième semaine d'avril 2019 (samedi 13/04/2019, dimanche 14/04/2019, ..., lundi 22/04/2019). Nous nous sommes focalisés sur le mois d'avril, car des événements de grande ampleur et ayant drainés énormément de foule se sont produits au cours de cette période. De plus, le 01/04/2019 étant un lundi, ceci facilite l'extraction des données par semaines entières.



(a) 12 semaines (3e semaine avril 2019 en grisé) (b) Avril 2019 (3e pic : 15/04/2019)

FIG. 4.6. Activité des utilisateurs sur l'ensemble des secteurs

Étant donné que nous cherchions à isoler des anomalies correspondant à de grands événements, nous avons sélectionné des points extrêmes isolés dans le *Cluster0* autour du lundi 15 avril 2019. Les données relatives à quelques-uns de ces points sont listées dans le tableau 4.10. Celles-ci font ressortir trois secteurs ayant un niveau d'activité anormalement plus élevé que les autres. Ils ont pour geohash respectif *u09tvm*, *u09ty5* et *u09wjd*. Les cellules correspondant à ces geohashes sont projetées sur un fond cartographique [5] et matérialisés par la figure 4.7. Elles coïncident au secteur de *Notre-Dame de Paris* et à deux gares parisiennes, à savoir la *Gare de Lyon* et la *Gare du Nord* comme représentées par les figures 4.7a à 4.7c.

TABLE 4.10
EXTRAIT DES POINTS EXTRÊMES ISOLÉS

TimeSlot	Geohash	AppGroup	Packets	Duration	Users	Flows
2019-04-15 16 :00	u09wjd	7	3157	47859	76	326
2019-04-15 16 :30	u09wjd	3	239790	7893	35	62
2019-04-15 22 :00	u09ty5	1	91506	741461	221	141
2019-04-15 22 :00	u09tvm	3	1712	1818461	183	183
2019-04-15 22 :00	u09ty5	1	16285	1297180	2	4
2019-04-16 00 :30	u09tvm	2	72511	10827	51	55

5. `https://www.movable-type.co.uk/scripts/geohash.html`

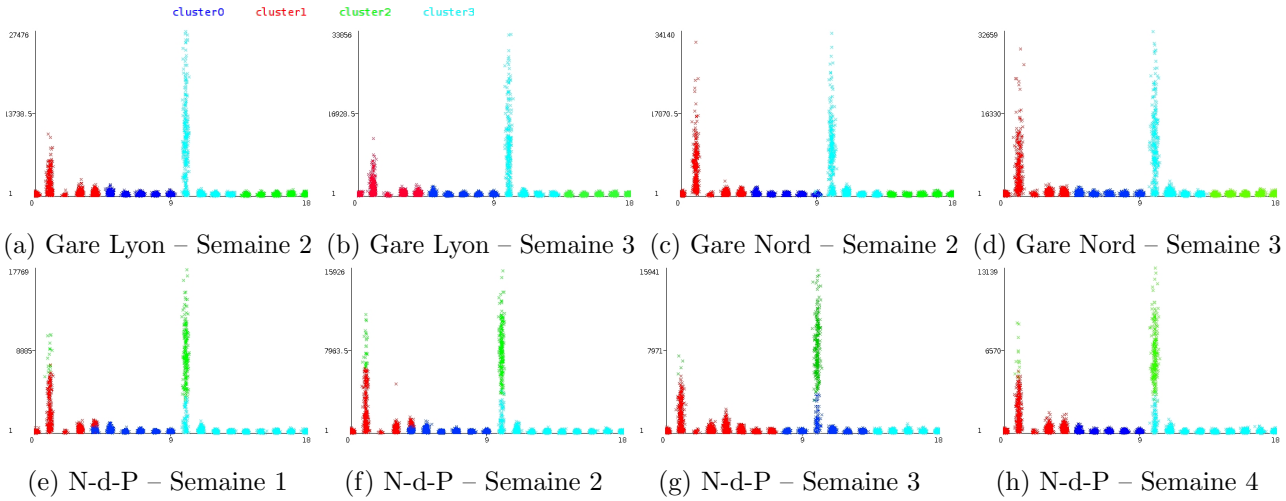
(a) Notre-Dame de Paris (b) Gare de Lyon (c) Gare du Nord

FIG. 4.7. Secteurs à forte activité

Les faits de : *(i)* pouvoir regrouper les données selon différents clusters *(ii)* constater qu'une modulation de l'activité était visible lors de la projection dans le plan des données, nous ont confortés dans l'idée d'employer une approche non supervisée pour modéliser cette répartition de l'activité en clusters et ensuite la comparer semaine par semaine.

Pour valider notre approche, nous nous sommes intéressés à la période allant du lundi 1er avril au dimanche 28 avril 2019. Les données incluses dans cette période ont été filtrées et scindées par semaines à l'aide d'expressions régulières sur le créneau horaire ('TimeSlot') et nous avons conservé uniquement les données relatives aux secteurs que nous souhaitions analyser grâce à leur geohash.

Cette phase consistant en l'extraction des *signatures numériques* est également réalisée à l'aide de l'algorithme *X-Means* [107]. Cet algorithme non supervisé est utilisé pour regrouper les données détaillées de chacun des secteurs et ce pour chaque semaine. La figure 4.8 est une représentation de l'affectation des flux réseaux dans les clusters par semaine et par secteur avec en abscisse le groupe applicatif et en ordonnée le nombre d'utilisateurs.



(a) Gare Lyon – Semaine 2 (b) Gare Lyon – Semaine 3 (c) Gare Nord – Semaine 2 (d) Gare Nord – Semaine 3

(e) N-d-P – Semaine 1 (f) N-d-P – Semaine 2 (g) N-d-P – Semaine 3 (h) N-d-P – Semaine 4

FIG. 4.8. Exemples de signatures numériques

Des figures 4.8a et 4.8b, nous pouvons déduire que les signatures de la *Gare de Lyon* sont similaires pour les semaines 2 & 3. C'est également le cas pour la première et la quatrième semaine d'avril 2019, non représentées ici. En conséquence, nous pouvons dire que les signatures des quatre semaines d'avril 2019 pour le secteur de la *Gare de Lyon* sont identiques. Nous pouvons en conclure qu'il n'y a pas eu d'anomalie et de ce fait qu'aucun événement particulier ne s'est produit sur ce secteur en avril 2019. La même conclusion peut être déduite des figures 4.8c et 4.8d à propos du secteur *Gare du Nord*. La distribution et la répartition des données dans les différents clusters sont identiques pour les quatre semaines d'avril 2019, ce qui signifie que les signatures numériques sont identiques.

Une observation plus intéressante est que les signatures des secteurs *Gare de Lyon* et *Gare du Nord* sont identiques quelle que soit la semaine étudiée. Les affectations dans les clusters sont les mêmes pour les figures 4.8a à 4.8d. Grâce à notre concept des *signatures numériques*, nous serions possiblement en mesure de classifier les secteurs par type d'activité (gares, gares TGV, stade, cimetières,...).

Les figures 4.8e à 4.8h décrivent le secteur de *Notre-Dame de Paris*. Nous pouvons observer que les signatures des semaines 1, 2 & 4 sont uniformes. Une exception peut être trouvée dans la figure 4.8h présentant les clusters de la quatrième semaine où une évolution insignifiante apparaît en ce qui concerne la distribution du champ 'AppGroup' au sein des quatre clusters. Cependant, en se focalisant sur la figure 4.8g, il est évident que la signature est très différente des autres signatures numériques. Les 'AppGroup' 5 (Mail), 6 (DB), 7 (Others), 10 (Streaming), 11 (Chat), 12 (VoIP) et 13 (MailOperator) ne sont pas assignés aux mêmes clusters. Cela met en évidence une variation de l'activité au niveau réseau ce qui nécessite une analyse plus approfondie. Celle-ci est réalisée au cours de la dernière phase au moyen d'un deuxième algorithme non supervisé.

4.3.4 Détection des valeurs aberrantes

La dernière phase de notre "Framework de détection d'anomalies par signature numérique" emploie l'algorithme Local Outlier Factor. Son implémentation Weka nécessite un attribut de classe pour pouvoir les représenter graphiquement. Nous avons donc ajouté une classe "Outlier" et toutes les instances ont été marquées comme 'False'. Il est également possible de configurer la distance mesurée par celui-ci lors du processus de regroupement. Nous avons sélectionné la distance de *Chebyshev* car c'est celle qui a donné les meilleurs résultats, c'est-à-dire nous ayant permis d'obtenir le plus de valeurs aberrantes tout en ayant une répartition homogène.

L'algorithme LOF a été employé pour extraire les valeurs aberrantes.Des 5 075 instances contenues dans la troisième semaine d'avril, LOF a fait ressortir une quinzaine d'outliers, listés dans le tableau 4.11, ce qui réprésente un ratio de 0.03% ; contre seulement deux valeurs aberrantes extraites des 5 098 instances de la première semaine et de la quatrième semaine d'avril soit 0,06.

4.3.5 Corrélation des anomalies

Du tableau 4.11, nous pouvons déduire qu'une anomalie relative à l'activité réseau des utilisateurs mobiles d'*Orange* s'est produite du 15 avril 2019 à compter de 18h00 jusqu'au 17 avril 2019 à 10h00

sur le secteur de *Notre-Dame de Paris*. Cette variation, liée au type d'application mobile utilisé, peut être corrélée à un événement externe, à savoir l'*incendie de Notre-Dame de Paris* [108].

TABLE 4.11
LISTE DES VALEURS ABERRANTES EXTRAITES PAR LOF

Instance	TimeSlot	AppGroup	Packets	Users	Flows
528	2019-04-15 18 :00	12	1745	XX	9
714	2019-04-15 23 :30	12	360	XX	2
808	2019-04-16 02 :30	4	8347	XX	68
889	2019-04-16 05 :30	5	7491	XX	43
926	2019-04-16 07 :00	4	9425	XX	175
1012	2019-04-16 10 :00	1	29726	XX	235
1355	2019-04-16 20 :00	10	731148	XX	9194
1758	2019-04-17 09 :30	16	1241	XX	46
1767	2019-04-17 10 :00	0	4372	XX	15
3e semaine d'avril 2019					

D'après la figure 4.8g et la section 4.3.2, nous sommes capable de dire que cet événement majeur a généré un afflux important d'usagers qui se sont inscrits sous les BTS couvrant cette zone. En outre, les principaux types d'application utilisés à cette occasion étaient : 1 (Web), 4 (CloudStorage), 3 (Download), 5 (Mail) et 10 (Streaming) à la différence des autres semaines où les applications utilisées étaient plutôt : 1 (Web), 3 (Download), 4 (CloudStorage), 10 (Streaming) et 11 (Chat).

4.3.6 Résultats

De cette étude, nous pouvons affirmer que notre framework permet de détecter des anomalies liées à des événement externes. Ces anomalies sont liées au changement de comportement des utilisateurs inscrits sous les BTS de l'opérateur mobile. Ce comportement fait écho aux différents types d'application employés, à différentes périodes de temps et sur des secteurs géographiques bien déterminés.

Nous avons été à même de détecter l'*incendie de Notre-Dame de Paris* qui a eu lieu dans la nuit du 15 avril 2019, événement majeur faisant se déplacer de nombreux curieux pendant plusieurs jours. Nous avons également pu mettre en lumière d'autres anomalies et donc événements comme le *match de finale de la Coupe de France de football* qui opposait Rennes à Paris le 28 avril 2019 ou le concert de *Metallica* qui s'est déroulé le 12 mai 2019 également au *Stade de France*.

4.4 Conclusion

4.4.1 Contribution

Notre "Framework de détection d'anomalies par signature numérique" permet de corréler des anomalies à des événements externes. Il ne nécessite ni donnée labellisée ni entraînement préalable ni service de mises à jour régulières pour être efficace. Indépendant du type de trame ou de leur payload, il détecte les variations de l'activité réseau sans recourir à une infrastructure tierce.

Notre cadre de travail consomme en entrée des données brutes qui : *(i)* sont enrichies puis fusionnées par site pour former des flux réseaux *(ii)* des secteurs sont définis grâce au système de géocodage `Geohash` [102] en utilisant les coordonnées géographiques incluses dans le jeu données *(iii)* les flux réseaux relatifs à chacun des sites constituant un secteur sont agrégés *(iv)* les données ainsi agrégées par secteur sont ensuite regroupées pour former des ensembles cohérents dénommés "clusters" grâce à l'algorithme `X-Means` *(v)* la répartition des données au sein de ces clusters définit ce que nous avons baptisée la *signature numérique* du secteur *(vi)* elle est générée à des intervalles de temps réguliers *(vii)* cette "photo" est la représentation numérique du type d'application utilisé ce qui permet la mise en lumière des variations de comportement au sein du secteur réseau supervisé à un instant donné *(viii)* si un changement est constaté entre deux signatures, cela signifie qu'une anomalie a été détectée *(ix)* l'extraction des valeurs aberrantes à l'aide de l'algorithme `LOF` nous permet de corréler cette anomalie avec un événement externe.

Il est important de préciser que les secteurs ont été définis et délimités à l'aide des coordonnées `LambertII` des différentes BTS car, dans le cadre du projet CANCAN [54], les traces mises à disposition par `Orange` comportaient cette information. Bien entendu, la façon de formaliser ces secteurs reste libre et dépend uniquement des données collectées ou disponibles.

4.4.2 Analyse

4.4.2.1 Méthodologie

Nous avons choisi de détecter des anomalies relatives au comportement des utilisateurs, car l'activité liée à l'application mobile utilisée était une des caractéristiques disponibles dans notre jeu de données. Par extension, notre framework pourrait être transposé à d'autres réseaux présentant des trames ayant des caractéristiques différentes, des périodes de temps et de captures autres ou encore une méthode

d'agrégation sectorielle basée sur d'autres notions comme le masque de sous-réseau, un tag VPN (Virtual Private Network) ou le type de lien emprunté par les données (Ethernet, fibre optique,...).

Nous pouvons donc affirmer que notre "Framework de détection d'anomalies par signature numérique" est *générique*, *cyclique* et *fractal*, car adaptable voire transposable, itératif et peut être plus ou moins détaillé ou précis en fonction de la taille des secteurs constitués et à analyser.

4.4.2.2 Évolutions

Une autre approche consisterait à appliquer notre concept à des secteurs de taille plus petite, secteurs définis par exemple par des geohashes de huit caractères afin de détecter des événements ou des types d'activité moins importants.

Des études supplémentaires pourraient porter sur la possibilité de détecter l'absence d'un trafic réseau particulier lorsqu'un niveau d'activité élevé est la norme ou requis. Nous avions également pour projet d'étudier la manière dont l'activité se répartit au sein du réseau mobile en analysant des secteurs définis par des geohashes de dimensions variables afin d'essayer de suivre l'activité réseau au fur et à mesure qu'elle change de secteurs.

Enfin, grâce à notre concept de *signature numérique*, il semble que nous soyons capables de classifier des secteurs en fonction de l'activité réseau générée. Cette idée doit être confirmée en sélectionnant davantage de secteurs et en comparant leurs signatures respectives pour identifier par exemple des centres commerciaux, culturels, des écoles ou autres.

4.4.2.3 Contraintes

L'étape de "clustering" ou regroupement est réalisée avec l'aide d'un MLA non supervisé. Le clustering est une méthode d'apprentissage automatique non supervisée qui permet d'identifier et de regrouper des points de données similaires dans des ensembles de données plus vastes, sans se soucier du résultat spécifique. Le regroupement (parfois appelé analyse de regroupement) est généralement utilisé pour classer les données dans des structures plus faciles à comprendre et à manipuler [109]. Les algorithmes de clustering recherchent donc des caractéristiques communes intrinsèques aux données afin d'en extraire des similitudes et les regrouper. Par conséquent, toutes les données appartenant à un même "cluster" sont réputées être très semblables en fonction des caractéristiques qui les composent.

Ces algorithmes disposent de plus ou moins de paramètres appelés ”hyper-paramètres” qui permettent de contrôler, d'adapter ou de modifier leur comportement et ainsi de moduler les résultats obtenus. Dans le cadre de nos travaux, deux de ces hyper-paramètres nous importent plus particulièrement, à savoir : *(i)* le nombre de clusters attendus (k) *(ii)* la graine permettant d'initialiser un centre aléatoire pour chacun des clusters (centroïde).

Suite aux résultats obtenus dans la section 4.3.3 et le tableau 4.9, nous avons choisi de produire des signatures numériques construites sur la base de quatre clusters d'où une valeur par défaut définie à '4' pour le nombre de clusters attendus. Pour chacun de ces clusters, une graine fixe a été définie de façon arbitraire afin que l'algorithme de regroupement produise toujours les mêmes clusters, et ce, dans le même ordre lorsque les mêmes données sont analysées. Par conséquent, ce seront toujours les mêmes données qui seront regroupées ensemble et qui produiront donc le même cluster représenté par la même couleur, avec le même identifiant ou ”ClusterId”.

Les données analysées peuvent comporter un nombre indéterminé de caractéristiques réseaux. La projection dans le plan de ces signatures peut être réalisée en utilisant différents couples x et y de caractéristiques. Nous avons choisi de projeter le paramètre 'AppGroup' en abscisse et le nombre de flux générés par les utilisateurs 'Flows' en ordonnée comme présenté par la figure 4.9 qui est une signature numérique constituée de quatre clusters. Le fait de fixer la graine nous permet de toujours obtenir les mêmes clusters pour les mêmes données en entrée.

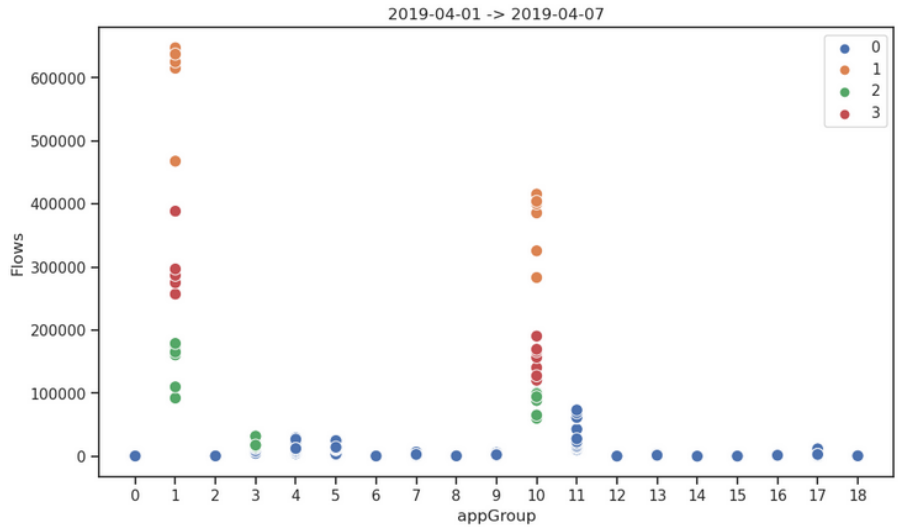
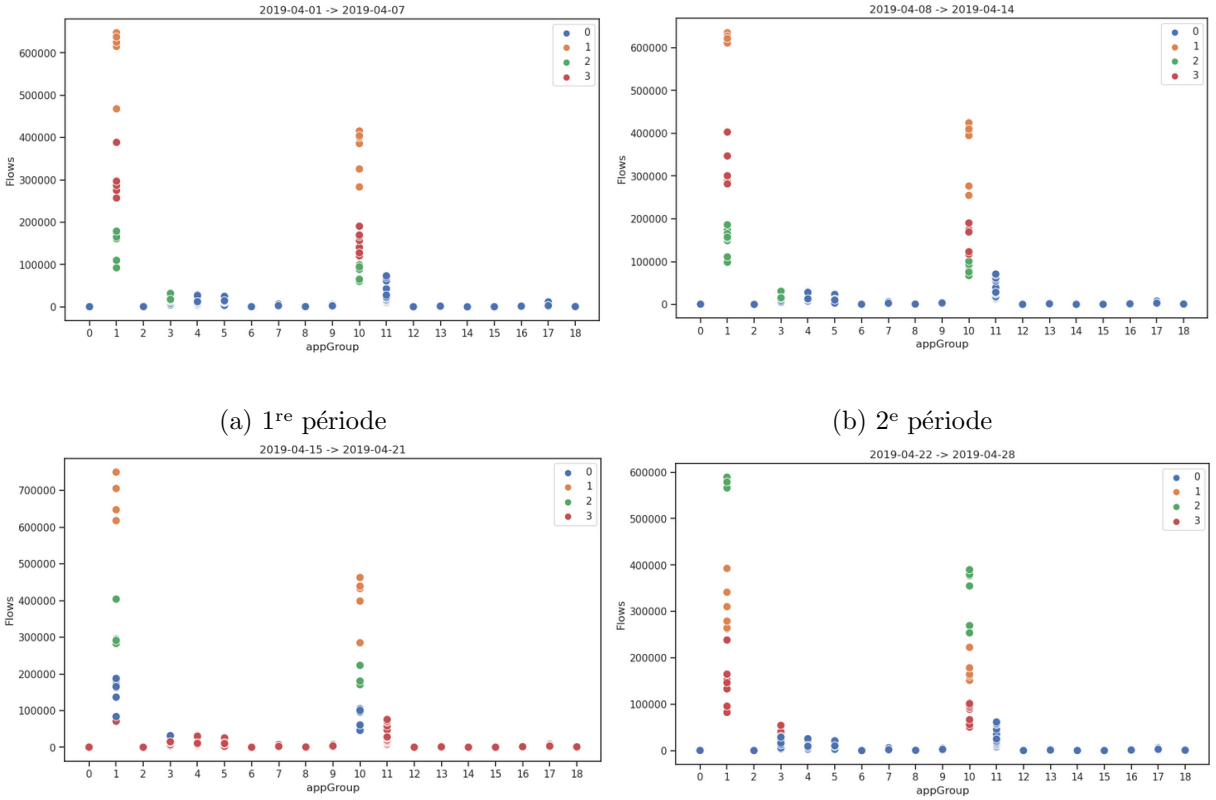


FIG. 4.9. Exemple de signature numérique constituée de 4 clusters

4.4.2.4 Limites

Afin de détecter des éventuelles anomalies concernant un secteur réseau donné, la signature numérique dudit secteur est recalculée régulièrement à différents intervalles de temps. Une évolution dans les proportions des caractéristiques réseaux décrivant le trafic entraîne un changement dans la répartition des clusters entre deux signatures ce qui révèle une évolution différente de l'activité et donc potentiellement une anomalie réseau.

Or, étant donné que les signatures sont calculées à des intervalles de temps réguliers ; mais de manière indépendante ; même en fixant la graine, nous ne pouvons pas être sûr que les clusters seront calculés dans le même ordre entre deux signatures numériques. Par conséquent, des données semblables entre deux signatures seront réparties dans les mêmes clusters, mais les clusters n'auront potentiellement pas le même identifiant. La figure 4.10 illustre ce problème.



(a) 1re période

(b) 2e période

(c) 3e période

(d) 4e période

FIG. 4.10. Signatures numériques sur 1 mois pour le secteur *Gare du Nord*

Hormis la dispersion des points, la répartition des clusters en fonction du paramètre 'AppGroup' est strictement identique pour les 1re, 3e et 4e périodes comme présentée par les figures 4.10a, 4.10c et 4.10d. Par contre, pour ce qui est de la 2e période dont la signature est présentée par la figure 4.10b, la répartition des clusters n'est pas identique sauf à permuter les identifiants des clusters '2' (rouge) et '3' (vert). En prenant en compte cette permutation, la signature numérique est identique pour les quatre périodes, ce qui nous permet d'en déduire qu'il n'y a pas eu de modification de l'activité réseau au cours de cette période, sur ce secteur particulier.

Une autre limitation quant à l'analyse des signatures numériques est le nécessaire traitement visuel et donc humain que la méthode requiert pour déterminer si la distribution des clusters est réellement différente ou non. Cet inconvénient majeur interdit de fait un traitement par une machine et par conséquent l'automatisation du processus d'analyse. D'où la mise au point du concept de *DNA* implémenté dans DiNATrA$\mathcal{X}$ pour pallier à ces deux difficultés, concept défini, expliqué et employé dans le chapitre 5 suivant.

Chapitre 5

Méthodologie DiNATrA$\mathcal{X}$

5.1 Introduction

5.1.1 Présentation

DiNATrA$\mathcal{X}$ est l'acronyme de "**Di**gital **N**etwork **A**ssessment & s**TR**and based **A**nomalies e**X**traction". Il s'agit d'une méthodologie de détection *automatisée* d'anomalies réseaux formalisant : *(i)* la collecte, la préparation, l'enrichissement et les différentes étapes d'agrégation des traces réseaux *(ii)* le découpage et l'analyse par DNA de ces données agrégées en zones d'intérêt *(iii)* le calcul des distances d'anormalité et la remontée d'une éventuelle alerte si une anomalie est cwconfirmée.

DiNATrA$\mathcal{X}$ est basée sur nos concepts de *Digital Network Assessment* ou DNA et de *brin associé*. Le DNA est la transcodification de la *signature numérique*, notion détaillée dans la section 4.3.3 et utilisée par notre "Framework de détection d'anomalies par signature numérique", en une chaîne de caractères décrivant la signature. À partir de ce DNA, nous générons ce que nous avons nommé le brin associé qui est lui une liste ordonnée des nucléotides composant le DNA. Le brin ainsi formé permet également de caractériser la signature mais à l'aide d'un autre paramètre décorrélé de celui utlisé pour coder le DNA. Enfin, nous calculons les distances séparant deux DNAs et brins adjacents ce qui nous permet de quantifier la variation entre deux signatures numériques consécutives.

Fort logiquement, au vu de la description précédente, DiNATrA$\mathcal{X}$ est articulé autour de trois composants principaux nommés *Blocs Fonctionnels* ou BF. Ces composants BFI, BFII et BFIII sont représentés dans la figure 5.1.

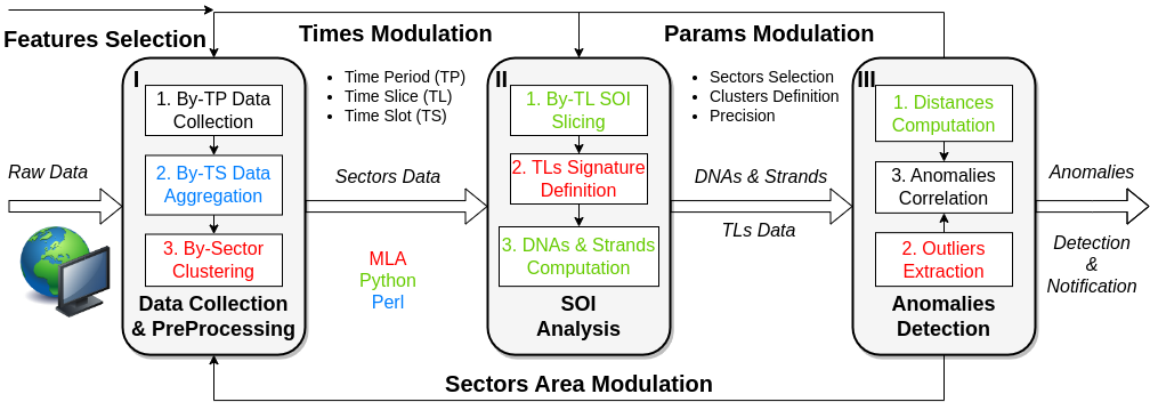


FIG. 5.1. Description fonctionnelle du système DiNATrA$\mathcal{X}$

Dans la section 5.1, nous commençons par présenter l'organisation générale de DiNATrA$\mathcal{X}$. La section 5.2 décrit le premier bloc fonctionnel chargé de la collecte et de la préparation des données. La section 5.3 détaille le deuxième bloc qui constitue le cœur métier de DiNATrA$\mathcal{X}$, bloc où sont générées les signatures, les DNAs et leur brin associé. Le calcul des distances et la détection des anomalies à proprement parler est assuré par le troisième bloc présenté dans la section 5.4. Enfin, les sections 6.1 et 6.2 sont des mises en œuvre complètes de DiNATrA$\mathcal{X}$ avec respectivement les jeux de données CANCAN et CTU-13 qui ont été décrits dans les sections 2.6 et 2.7.

5.1.2 Définitions

DiNATrA$\mathcal{X}$ définit les notions de *secteur* et *Secteur d'Intérêt* (SOI), et emploie trois périodes de temps différentes : *Time Period* (TP), *Time Slice* (TS) et *Time Slot* (TS) représentées par la figure 5.2, de la plus englobante (TP) vers la plus fine (TS).

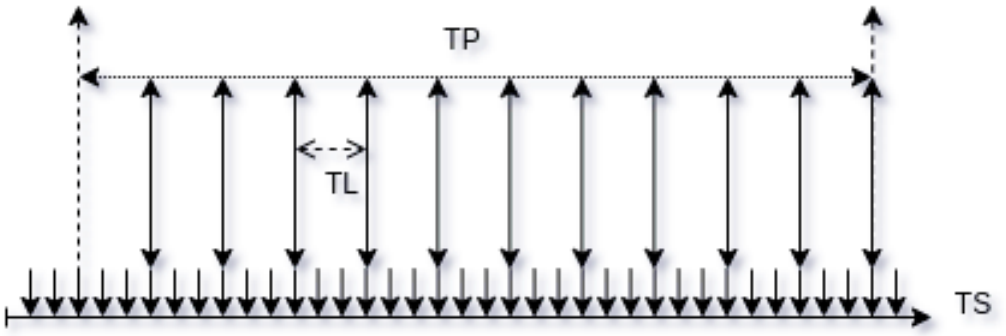


FIG. 5.2. Périodes de temps employées par DiNATrA$\mathcal{X}$

Time Period (TP)

Il s'agit de la période de temps pendant laquelle les données réseaux sont collectées pour ensuite être analysées. Par conséquent, c'est la période sur laquelle portera l'étude. Si une détection d'anomalies en temps réel est envisagée, une anomalie ne pourra pas être remontée en un temps inférieur à cette période. Les ordres de grandeurs pour la période TP sont la journée, la semaine, voire le mois.

Time Slice (TL)

La période de temps TP est divisée en tranches de temps de longueur TL. Pour chacune de ces TLs est générée une signature numérique puis calculé le DNA et son brin associé. Il s'agit de l'unité de temps élémentaire permettant la détection d'une variation dans l'usage du réseau informatique et donc détecter une éventuelle anomalie réseau. Un ordre de grandeur pour cette période TL est l'heure, la journée, voire la semaine. La définition de la période TL nécessite d'avoir une connaissance "métier" de la répartition des zones, de l'activité inhérente à celles-ci et des événements qui peuvent s'y produire. Il peut être nécessaire d'effectuer plusieurs essais pour en déterminer la valeur adéquate.

Time Slot (TS)

La période TS correspond à la fréquence d'échantillonnage de l'utilisation du réseau. Plus TS est élevée, plus la granularité et par conséquence le volume de données collectées augmente et donc meilleure est la précision avec laquelle l'activité réseau est décrite. Son corollaire est que la quantité de données à traiter et à stocker est également plus importante. TS est la base de temps permettant d'agréger les données capturées par des équipements (sondes) répartis en différents points d'un réseau de type LAN mais surtout MAN. Cette agrégation par TS requiert que les traces soient horodatées.

Secteurs & Secteurs d'Intérêt (SOI)

Un réseau informatique permet à des équipements inter-connectés de communiquer entre eux en échangeant des données. Un réseau peut être de taille assez réduite, quelques centaines de mètres, nous parlons alors de "Local Area Netwotk" (LAN). Il peut s'étendre à l'échelle d'une métropole, c'est-à-dire sur quelques kilomètres. Il est alors ici notion de "Metropolitan Area Network" (MAN). Enfin, un réseau peut couvrir l'ensemble du globe terrestre à l'instar de l'Internet qui est, dans ce cas, un Wide

Area Network (WAN). Détecter des anomalies dans un réseau de type LAN peut être envisagé de façon directe et globale, c'est-à-dire en analysant l'ensemble du trafic brut généré par les équipements. Par contre, cette approche n'est pas envisageable si l'on souhaite détecter des anomalies dans un réseau d'envergure de type MAN. Il n'est pas possible d'appréhender le réseau dans sa globalité pour traiter directement et en totalité l'ensemble des données circulant dans les liens. L'analyse ne peut alors être menée qu'en implémentant une approche parcellaire, sectorielle afin de mettre en exergue des interactions entre différents points répartis sur plusieurs milliers de kilomètres carrés.

Pour ce faire, la topologie réseau doit être scindée en zones logiques ou *secteurs*. Il n'y a pas de règles pré-établies ou universelles pour définir ces zones logiques et réaliser ce découpage en secteurs. Ils peuvent être librement délimités en fonction de la topologie ou des caractéristiques disponibles lors de la capture des trames réseaux comme le masque de sous-réseau, les tags de VLAN, les plages d'adresses IP ou encore les éventuelles coordonnées géographiques permettant d'identifier l'origine des trames réseaux. Une fois ces zones d'abstraction logique bien définies ; en fonction de l'étendu du réseau étudié, des équipements le composant ou des données générées ; il n'est pas toujours possible de procéder à une analyse exhaustive de toutes d'entre elles.

Parmi tous les secteurs disponibles, seuls certains d'entre eux peuvent présenter un intérêt particulier (en bordure de réseau, un lieu géographique, un service réseau critique,...). Ces secteurs présentant un intérêt particulier sont dénommés *Secteurs d'Intérêt* ou SOI. En fonction du découpage logique réalisé, de sa granularité ou de l'anomalie réseau à rechercher, il peut être intéressant de regrouper certains de ces secteurs entre eux. Il est donc important de garder à l'esprit qu'*un SOI peut être constitué de plusieurs secteurs*. Ceci implique qu'un SOI peut être considéré comme une fonctionnalité, un service ou un segment réseau indépendamment du ou des secteurs qui le composent. Cette notion de SOI permet d'avoir un niveau d'abstraction par-dessus la topologie réseau pour décrire le trafic de manière *fonctionnelle*. La détection d'anomalies au sein d'un réseau, en particulier de type MAN, se basera alors sur les données relatives à ces SOI et leurs interactions réseaux.

5.1.3 Blocs fonctionnels

BFI — Data Collection & PreProcessing

Ce premier bloc fonctionnel permet tout d'abord la collecte des données réseaux au format brut avec des outils ou équipements spécifiques. Il prend de ce fait en entrée des trames réseaux brutes.

Lors de la capture, seules certaines caractéristiques réseaux ou données sont collectées. Le choix ou la possibilité de collecte des données disponibles est lié aux : *(i)* limitations imposées par les outils eux-mêmes *(ii)* obligations d'anonymisation réglementaires *(iii)* contraintes physiques comme la performance ou le stockage par exemples. Ensuite, il assure le prétraitement des données collectées en vue de leur analyse. Celui-ci consiste principalement à nettoyer, consolider, éventuellement sélectionner (paramètre "Features Selection") voire enrichir les traces brutes (entrée "Raw Data") puis à les agréger.

Ces traces agrégées sont ensuite fusionnées par SOI afin de permettre une analyse par zones logiques bien délimitées et identifiées. En sortie de ce premier bloc fonctionnel, nous obtenons alors des données agrégées et fusionnées décrivant l'utilisation du réseau pour chacune de ces zones logiques ou secteurs (sortie "Sectors Data") et ce pour la période TP considérée.

Ce composant BFI peut être adapté aux besoins spécifiques des utilisateurs du système DiNATrA$\mathcal{X}$ en personnalisant la liste des caractéristiques réseaux à capturer, à traiter puis à analyser. Il est également possible de paramétrer les différentes bases de temps TP, TL et TS via les paramètres "Times Modulation" ainsi que la méthode de découpage des SOI (geohashes, clustering,...). Ces méthodes peuvent être adaptées en fonction des résultats obtenus en sortie du système DiNATrA$\mathcal{X}$. En effet, il existe une boucle de "rétro-action" entre les blocs BFIII et BFI permettant de moduler la définition des secteurs au niveau de BFI afin d'affiner la détection (paramètre "Sectors Area Modulation").

BFII — SOI Analysis

Le composant BFII représente le cœur du système DiNATrA$\mathcal{X}$. Ce bloc fonctionnel est en charge de l'analyse des données agrégées correspondant à la période TP supervisée et relatives au secteur d'intérêt sélectionné (entrée "Sectors Data").

Cette analyse sectorielle se déroule selon quatre phases distinctes : *(i)* l'extraction des données de chacun des secteurs constituant le SOI à analyser *(ii)* le découpage des données du SOI en tranches de temps (TL) (ces deux premières phases sont réalisées simultanément pour des raisons de performance) *(iii)* la définition des signatures numériques pour chacune périodes TL *(iv)* le calcul pour chaque signature numérique du DNA et du brin qui lui est associé.

Ce deuxième bloc fournit alors au bloc BFIII la liste des DNAs et de leur brin associé pour chaque tranche TL de la période TP considérée, et ce, pour le SOI sélectionné (sortie "DNAs & Strands"). BFII est exécuté pour chacun des SOI à analyser. Il peut être configuré par l'intermédiaire de plusieurs paramètres à savoir :

Precision pour fixer la longueur des séquences constituant les DNAs

Sectors Selection pour sélectionner le ou les secteurs composant le SOI à analyser

Clusters Definition pour configurer le nombre de clusters générés par le MLA de regroupement utilisé pour définir les signatures numériques

BFIII — Anomalies Detection

Il s'agit du dernier bloc fonctionnel constitutif du système DiNATrA$\mathcal{X}$. Le BFIII a pour rôle de détecter les anomalies réseaux qui auraient pu survenir durant la période TP et d'éventuellement envoyer une alerte si une anomalie est détectée.

Les éléments exploités par ce composant fonctionnel sont les DNAs et les brins décrivant l'utilisation réseau de chaque SOI (entrée "DNAs & STrands") ainsi que les données agrégées par SOI (entrée "TLs Data"). L'utilisation des données agrégées en parallèle des DNAs et des brins permet d'affiner la détection et de limiter le nombre de faux positifs. Ceci améliore les performances globales de détection d'anomalies du système DiNATrA$\mathcal{X}$.

La détection des anomalies à proprement parler est donc réalisée à l'aide de deux traitements différents et complémentaires à savoir : *(i)* le calcul de la distance entre deux DNAs consécutifs ainsi que le calcul de la distance entre les deux brins correspondants et ce pour l'ensemble des DNAs et des brins. A partir de ces distances, nous calculons les distances d'anormalité correspondantes à partir desquelles nous définissons une sévérité pour chaque anomalie détectée. *(ii)* l'utilisation d'un MLA non supervisé qui nous permet d'extraire des données agrégées par TL les différents outliers. L'étude de la distribution des différents paramètres disponibles dans ces données réseaux permet de confirmer les "outliers" extraits.

Le dernier élément composant ce bloc BFIII a pour but, si possible, de corréler les éventuelles anomalies détectées avec des événements pouvant expliquer ce ou ces changements de comportant au sein du réseau. Cette étape de corrélation nécessite soit d'avoir mis en œuvre une analyse temps réelle soit d'avoir une base de connaissance des événements qui auraient pu survenir.

Ce dernier bloc fonctionnel peut également lever des alertes et transmettre à un outil de supervision le détail des données agrégées qui ont permis la détection de l'anomalie. En fonction des résultats retournés par ce bloc, c'est-à-dire de leur pertinence ou de leur précision, il est possible de moduler les différents paramètres impactant le fonctionnement des blocs BFI et BFII. Cet "asservissement" est matérialisé par les boucles de rétro-action BFIII vers BFII et BFI.

5.1.4 Spécificités

Cette architecture du système DiNATrA$\mathcal{X}$ organisée en trois blocs fonctionnels distincts et indépendants nous assure un cadre de travail transposable quelle que soit la topologie du réseau à analyser et agnostique au type d'anomalies recherchées : *(i)* les caractéristiques réseaux extraites et analysées sont choisies librement et dépendent uniquement du contexte dans le lequel il est déployé *(ii)* les périodes de temps, la valeur des paramètre peuvent être adaptées pour répondre au mieux aux besoins *(iii)* la granularité des secteurs peut être modulée pour s'adapter au réseau concerné.

L'ensemble de ces trois blocs fonctionnels permet de définir le système DiNATrA$\mathcal{X}$ selon cinq critères fondamentaux intrinsèques ou "Framework Fondamental Features (3F)" à savoir :

Temps réel (3F1) TP est définie de manière à analyser le trafic en continu et en temps réel

Générique (3F2) Les caractéristiques réseaux exploitées dépendent uniquement des données collectées

Cyclique (3F3) Le processus complet est répété pour chaque TP

Fractal (3F4) La taille ou la définition des secteurs peut être modulée pour s'adapter aux besoins

Récurrent (3F5) BFII est répété pour chaque SOI. BFII et BFIII sont répétées pour chaque TL

5.2 BFI – Data Collection & PreProcessing

La collecte et le prétraitement des données brutes capturées est une étape primordiale surtout lorsque des outils d'apprentissage automatique sont employés dans la suite du processus d'analyse. En effet, celles-ci conditionnent la qualité et la pertinence des résultats qui seront obtenus. La figure 5.3 est une vue détaillée de ce premier bloc fonctionnel.

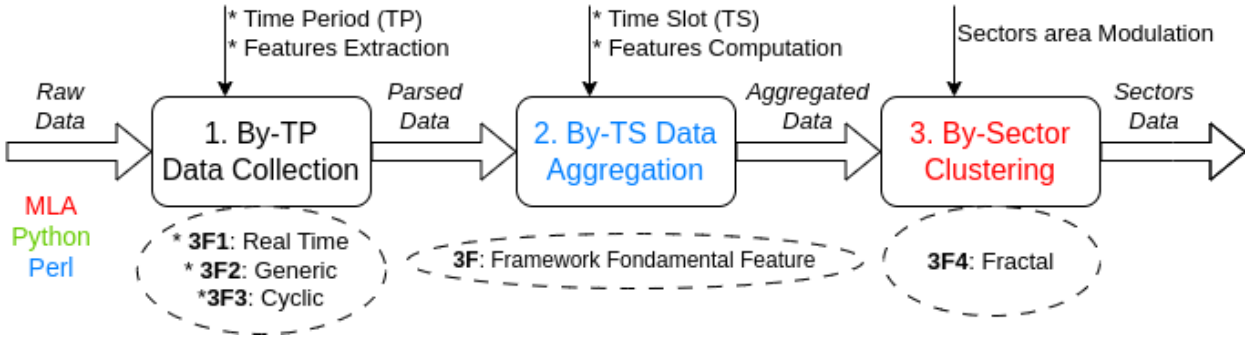


FIG. 5.3. DiNATrA$\mathcal{X}$ — Collecte des données & Pré-traitement

5.2.1 By-TP Data Collection

Cette étape est réalisée par des équipements réseaux comme des sondes ou des ordinateurs avec l'aide d'utilitaires comme `tcpdump`, `Argus` ou encore `WireShark`. Ces outils matériels ou logiciels permettent de capturer en temps réel et en continu toutes les trames réseaux qui circulent sur le médium même si celles-ci ne sont pas destinées explicitement à l'équipement de capture. C'est ce que l'on appelle écouter le trafic en mode *promiscuité*. En effet, chaque paquet réseau envoyé inclut l'adresse MAC du destinataire. Lorsqu'un équipement réseau "voit passer" un paquet, il vérifie s'il en est le destinataire. Si le paquet ne lui est pas adressé, celui-ci est ignoré. En mode "promiscuous", il traitera le paquet dans tous les cas ce qui permet de capturer l'ensemble des trames. Les paquets sont ensuite réassemblés pour reformer les flux réseaux qui pourront être exploités.

Cette phase permet également de sélectionner les caractéristiques principales qui serviront pour l'analyse et la détection des anomalies. Comme précisé par [110], le but de la sélection des caractéristiques est double : *(i)* augmenter les performances des différents algorithmes en diminuant le volume de données à analyser *(ii)* améliorer les résultats fournis en supprimant les variables corrélées *(iii)* mieux comprendre et faire ressortir les processus sous-jacents qui ont généré les données.

Cette étape de collecte et de sélection des données est primordiale pour la suite de l'analyse. Une fois capturées, les données sont très souvent traitées pour être nettoyées, enrichies ou normalisées. En effet, les données brutes ne peuvent généralement pas être utilisées directement. Cela s'explique notamment par les raisons suivantes : *(i)* les algorithmes d'apprentissage automatique exigent que les données soient des nombres *(ii)* il peut s'avérer nécessaire de corriger le bruit statistique et les erreurs dans les données *(iii)* ou inférer les valeurs manquantes.

Cette étape de prétraitement est organisée autour de cinq phases comme explicitée par [111], [112] :

Nettoyage des données Identification et correction des erreurs dans les données (doublons, valeurs manquantes, format erroné)

Sélection des caractéristiques Identification des variables d'entrée les plus pertinentes pour le traitement qui sera appliqué avec éventuellement suppression des variables corrélées

Transformation des données Modification de l'échelle ou de leur distribution (normalisation)

Ingénierie des caractéristiques Dérivation de nouvelles variables à partir des données disponibles

Réduction de la dimensionnalité Création de projections compactes des données

La durée de cette phase de collecte (TP) détermine la période sur laquelle portera la détection d'anomalies sachant que cette période impacte la réactivité du système de détection.

5.2.2 By-TS Data Aggregation

Lors de la capture, un horodatage peut-être ajouté par l'outil de capture, un "TimeStamp". Les protocoles réseaux étant extrêmement verbeux, l'activité réseau n'est habituellement pas capturée en continu, mais plutôt à intervalles réguliers. On parle dans ce cas de "TimeSlot" ou TS qui pourrait être assimilé à une fréquence d'échantillonnage. Plus les TS sont courts, plus la précision de l'analyse sera grande et le volume de données à traiter important. Le fait de moduler, dynamiquement ou non, cette fenêtre d'échantillonnage permet de traiter de très gros volume de données sans nuire à l'analyse [113].

De plus, cette activité peut être prélevée à différents points du réseau. L'ensemble des données capturées peut être agrégé en fonction de cet horodatage. Les valeurs des données caractérisant les échanges qui ont eu lieu pendant un même TS sont sommées entre elles afin d'en réduire le volume et permettre une analyse plus macroscopique. Le fait de sommer les valeurs permet de conserver l'ensemble de l'information, mais avec une vision plus large.

5.2.3 By-Sector Clustering

L'idée sous-jacente de cette étape est de "scinder" le réseau en différentes zones ou secteurs logiques à analyser comme décrit par [114]. Cette approche permet de détecter des anomalies comportementales surtout lorsque le réseau à analyser est du type MAN, voire WAN. Une approche par segment a été proposée par [115] mais celle-ci est surtout adaptée à des réseaux de type LAN du fait de son approche statistique.

La méthode de découpage de ces zones, c'est-à-dire la logique utilisée pour délimiter ces secteurs, dépend des anomalies recherchées, des données disponibles et de la topologie du réseau à analyser. Ce découpage peut-être abordé soit sous un aspect plutôt fonctionnel soit plus orienté technique. Cette étape nécessite une connaissance fine de l'architecture du réseau, des équipements qui le composent et de son usage. Une fois cette logique de découpage arrêtée, sa mise en œuvre est basée sur les attributs disponibles après la phase de prétraitement.

Aspect fonctionnel

Le regroupement fonctionnel des équipements réseaux et donc des flux qu'ils génèrent peut être basé sur le service rendu (serveurs Web, serveurs Proxy, serveurs DNS, base de données, annuaire,...). Une autre approche peut être leur position dans la topologie (bordure, cœur, DMZ, en entrée, en sortie,...).

Aspect technique

La formalisation technique du découpage logique peut être effectuée sur la base des adresses IPs en regroupant par exemple tous les équipements appartenant à une même plage d'adresses. Ce regroupement peut être également basé sur un masque de sous-réseau si celui-ci est disponible dans les traces. Les adresses MAC peuvent également permettre de scinder le réseau en différents secteurs à condition de connaître leurs répartitions. Les "tags" des VLAN pourraient également servir de clef pour l'agrégation des flux. Enfin, les zones ou secteurs d'un réseau peuvent être définis en se basant sur des données géographiques permettant de localiser les sources ayant émis les paquets.

5.3 BFII – SOI Analysis

Les secteurs réseaux à présent déterminés, les données correspondantes sont analysées en vue de détecter des changements de comportement susceptibles de mettre en évidence des anomalies réseaux comme des incidents, des attaques ou des événements particuliers.

Une approche plus fine par secteurs d'intérêt (SOI) décrivant soit des fonctions ou des zones réseaux soit des ensembles d'équipements particuliers permet de détecter des attaques distribuées ciblées (plusieurs sources vers une destination ou fonction). Cette approche permet également de mettre en lumière des pannes localisées (un ou plusieurs SOI particuliers) voire des services défaillants (variation de l'activité DNS, latence de bases de données...) ou des variations anormales de l'activité en certains points du réseau. Ce bloc fonctionnel nécessite donc de sélectionner les SOI à analyser. La figure 5.4 détaille les différentes étapes réalisées par ce deuxième bloc.

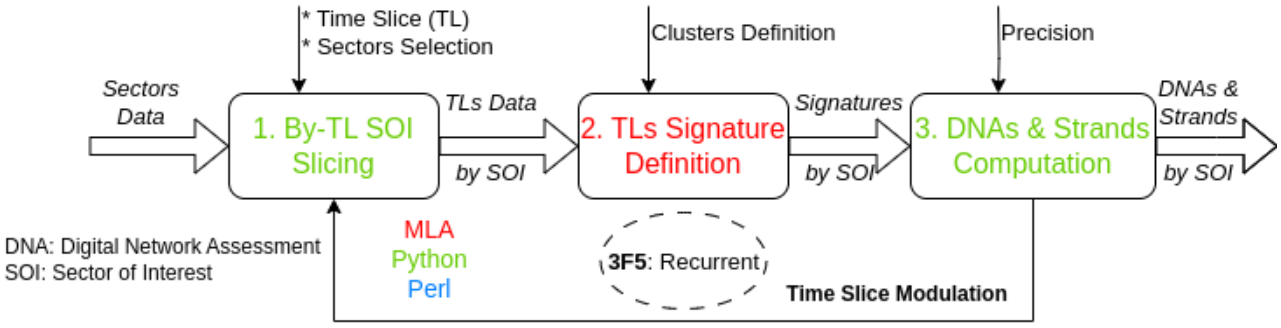


FIG. 5.4. DiNATrA$\mathcal{X}$ — Analyse des secteurs

5.3.1 By-TL SOI Slicing

Cette étape permet de découper la période considérée (TP) en fenêtres ou tranches de temps équivalentes ("TimeSlice" ou TL) et ce pour chacun des SOIs analysés. Le fait de découper la période TP en tranches TL permet d'effectuer l'analyse même dans le cas où les données sont capturées de façon continue [53]. Une anomalie réseau sera alors ici formalisée par une variation de l'activité survenue entre deux tranches de temps adjacentes. La difficulté est de matérialiser cette variation lors du passage d'une tranche TL_n à la tranche TL_{n+1}. Cette variation de l'activité réseau est mise en exergue en générant, par SOI, la signature numérique de chacune des tranches de temps TL : "TimeSlice Digital Signature" ou TLDS. Ces signatures numériques sont définies lors de l'étape suivante.

5.3.2 TLs Signature Definition

Ce concept de *signature numérique* a été présenté et utilisé par [116] dont les travaux sont issus des principes posés par [115]. Les signatures numériques DSNS (Digital Signature Network Segment) et dérivées évoquées dans ces travaux sont générées sur la base de l'analyse statistique d'un segment particulier du réseau, et ce, pendant une journée. Cette analyse repose sur les caractéristiques réseaux décrivant le trafic comme le volume de données échangées, le nombre d'erreurs, les types de protocoles ou services utilisés et est répétée à intervalles réguliers.

Avec DiNATrA$\mathcal{X}$, la signature numérique d'une tranche de temps consiste en un regroupement des données décrivant le trafic réseaux en fonction de leurs caractéristiques intrinsèques. Ce regroupement est réalisé à l'aide d'un MLA non supervisé, à savoir l'algorithme `K-Means`. Comme expliqué dans la section 4.3.3, le nombre de clusters par TLDS a été fixé à '4' suite aux tests réalisés précédemment.

Les TLDS définies, il faut pouvoir les comparer entre elles afin de déterminer si une variation dans l'usage du réseau est survenue. Cette comparaison peut être réalisée visuellement en matérialisant graphiquement les clusters comme présenté par la figure 4.9. Pour des questions de facilité de lecture et d'interprétation, les clusters constituant les signatures numériques sont projetés en deux dimensions. Le choix des paramètres utilisés pour la représentation des clusters n'influe pas sur la répartition des données au sein des clusters.

Ce traitement visuel de chaque TLDS serait éventuellement possible pour étudier un nombre limité de SOIs et nécessiterait de veiller à l'ordre dans lequel les clusters ont été déterminés et identifiés par l'algorithme de clustering. Pour l'analyse d'un réseau de plus grande envergure, ce traitement visuel par un humain n'est pas envisageable et requiert qu'il soit automatisé. D'où le concept de *DNA et de brin associé* utilisé par la suite. Ces éléments sont calculés pour chaque TLDS.

5.3.3 DNAs & Strands Computation

Les signatures numériques TLDS permettent de représenter la répartition des données dans les clusters, répartition qui est effectuée par l'algorithme *K-Means* en se basant sur l'ensemble des caractéristiques disponibles dans le jeu de données. De ces clusters, nous déduisions ce que nous appelons les DNAs (Digital Network Assessments). Une autre notion de DNA a été proposée et utilisée pour étudier des jeux de données et en extraire leurs éléments de base par [117]. Dans le cadre de

nos travaux, les DNAs sont la représentation sous forme de chaînes de caractères des clusters créés par le MLA en se basant sur une caractéristique particulière identifiée par la lettre $\mathcal{F}$. $\mathcal{F}$ est une caractéristique matérialisant le comportement réseau que l'on souhaite superviser et dont on veut mettre en évidence l'évolution, la variation dans le temps (le volume de données échangées, l'utilisation par type de protocole, les ports concernés lors des communications, les codes HTTP retournés, les drapeaux TCP, etc). Toutes les valeurs ou plages de valeurs que peut prendre cette caractéristique $\mathcal{F}$ sont reportées sur l'axe des abscisses et l'occurrence de chacune de ces valeurs est représentée en ordonnée lors de la projection des TLDS. Le choix du critère matérialisant l'occurrence de $\mathcal{F}$ est libre.

Les DNAs sont constitués d'autant de séquences qu'il y a de clusters. Une séquence est une chaîne de caractères décrivant la constitution d'un cluster où chaque caractère (*nucléotide*) identifie une valeur de $\mathcal{F}$ incluse dans le cluster (Seq_1 = ADC & Seq_2 = BC). La longueur maximale d'une séquence est définie comme étant la *précision*. Les nucléotides constituant la séquence appartiennent à l'ensemble constitué par les différentes valeurs ou plages de valeurs possibles de $\mathcal{F}$. Elles sont triées en fonction de leur occurrence ce qui permet d'obtenir une "image" codifiée des clusters. Le DNA est ensuite construit en concaténant avec le caractère '-' les séquences ordonnées en fonction de l'occurrence de la première valeur de $\mathcal{F}$ de la séquence. En cas d'égalité, c'est le total des occurrences ou "Volumes" constituant la séquence qui permettra le classement. En effet, plus le volume d'une séquence est élevé, plus le cluster décrit par cette séquence sera représentatif de $\mathcal{F}$.

Pour affiner l'analyse et donc améliorer la pertinence de la détection, est calculé le *brin associé* de chaque DNA. Ils sont déduits des DNAs et représentent la liste des nucléotides constituant le DNA, ordonnés en fonction de leur occurrence. Le critère de mesure de l'occurrence des nucléotides est *différent* de celui utilisé pour la mesure de l'occurrence des DNAs ce qui permet de combiner plusieurs caractéristiques et de *confirmer ou infirmer* la variation.

L'exemple suivant illustre comment sont construits les DNAs et leur brin associé. Nous supposons ici que $\mathcal{F}$ peut prendre les valeurs de A à E et que l'occurrence de ces nucléotides est : A = 10, B = 100, C = 5, D = 50 et E = 1. Les séquences sont constituées d'après les clusters comme suit : Seq_1 = BDA, Seq_2 = BD, Seq_3 = AC & Seq_4 = DAC. Le DNA correspondant à cette TLDS sera alors : BDA-BD-DAC-AC et son brin associé sera donc : BDAC car 'E' n'apparaît pas dans le DNA.

Une fois constitué, l'ensemble des DNAs et des brins relatifs à un SOI est passé avec les données agrégées par TLDS au troisième bloc fonctionnel qui est en charge de la détection des anomalies.

5.4 BFIII – Anomalies Detection

La détection d'anomalies consiste, à partir des DNAs et des brins décrivant l'évolution du trafic réseau de chaque SOI, à déterminer l'importance de la variation entre deux DNAs et brins adjacents. Pour ce faire, nous déterminons la distance qui sépare ces chaînes de caractères pour mesurer l'importance de celle-ci. Si la déviation est jugée suffisamment significative, l'anomalie est confirmée grâce à l'extraction des valeurs aberrantes et l'étude de la distribution des données. L'anomalie peut éventuellement être corrélée avec un événement puis une alerte est levée et un message émis. La figure 5.5 est une vue détaillée des étapes constituant ce troisième et dernier bloc fonctionnel.

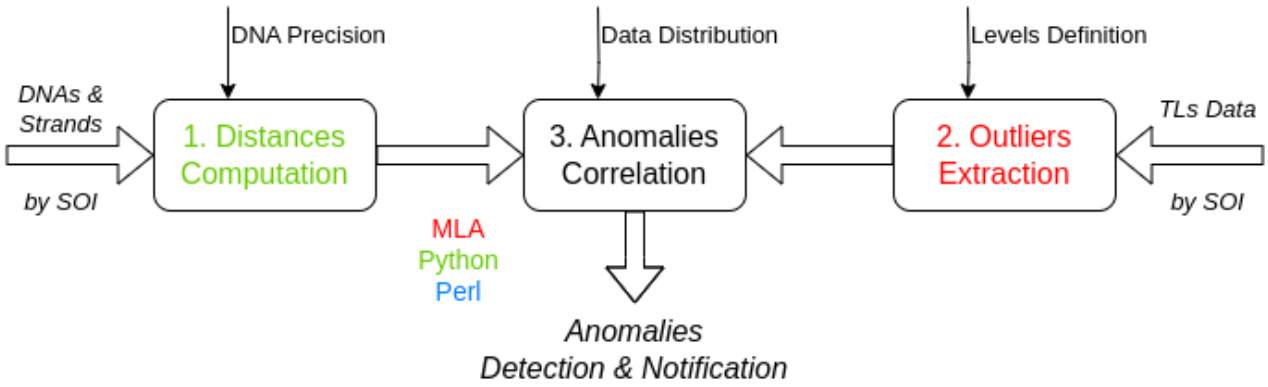


FIG. 5.5. DiNATrA$\mathcal{X}$ — Détection des anomalies

5.4.1 Distances Computation

Il existe différentes distances permettant de mesurer le nombre de modifications entre deux chaînes de caractères comme les distances de JARO-WINKLER ou de LEVENSHTEIN [118]. Nous avons choisi d'utiliser la distance de DAMERAU-LEVENSHTEIN car plus adaptée à notre cas d'usage [119]. En effet, elle a largement été utilisée pour effectuer de l'exploration de données, comparer des paquets réseaux, quantifier la similarité entre des séquences d'ADN ou l'identification de gènes [120]. Cette distance représente le *nombre minimum d'opérations* nécessaires pour transformer une chaîne de caractères en une autre. La distance de DAMERAU-LEVENSHTEIN définit quatre opérations possibles : l'*insertion*, la *suppression* ou la *substitution* d'un *simple caractère* et la *transposition* de *deux caractères adjacents*. Par exemple, la distance DL('CA', 'ABC') vaut '2' car 'CA' devient 'AC' par la transposition de deux caractères adjacents puis 'ABC' par l'insertion du caractère 'B'.

Pour valider l'existence d'une anomalie, pour chaque DNA_n où 'n' représente l'indice de la tranche TL concernée, nous calculons plusieurs distances d'anormalité :

(i) la *distance d'anormalité par DNA* nommée *DNAD* :

$DNAD_n = \mathrm{DL}(DNA_{n-1}, DNA_n)$ où $DNAD_n$ représente le DNA de la période TL_n

(ii) la *distance d'anormalité par brin* nommé *STAD* :

$STAD_n = \mathrm{DL}(Strand_{n-1}, Strand_n)$ où $STAD_n$ est le brin associé à DNA_n

(iii) la *distance d'anormalité* pour la tanche TL_n considérée :

$ANOD_n = ANOD_n + STAD_n$

Une fois obtenu l'ensemble des distances d'anormalité pour chacune des tranches TLs d'un SOI, il faut pouvoir classer, catégoriser les anomalies en fonction de leur importance ou gravité. Cette notion de gravité de l'anomalie, la *Severity*, est définie par l'équation (5.1) où *Precision* représente la longueur maximale d'une séquence.

$$Severity_n = \begin{cases} 0 & \text{si } ANOD_n < Precision \\ (\frac{ANOD_n}{Precision})^2 & \text{sinon} \end{cases} \tag{5.1}$$

Sachant que les DNAs sont constitués de plusieurs séquences, nous avons posé la *Severity* comme étant le carré du rapport de la *distance d'anormalité* sur la *précision* car :

(i) le fait de diviser la *Distance d'anormalité* par la *Precision* fait que la *Severity* est égale à '1' si les deux valeurs sont égales ce qui correspond au niveau le plus faible

(ii) si $ANOD_n$ et *Precision* sont très proches, $Severity_n$ reste très faible et il est difficile de faire ressortir des écarts d'où l'élévation au carré

(iii) si *Precision* diminue, mais que $ANOD_n$ reste importante, $Severity_n$ augmente ce qui est cohérent, car l'anormalité est importante malgré une faible précision

(iv) inversement, si *Precision* augmente, mais que $ANOD_n$ reste proportionnellement faible, $Severity_n$ diminue ce qui reflète bien que l'anormalité est faible malgré une précision importante.

Le tableau 5.1 illustre cette variation de la $Severity_n$ en fonction de $ANOD_n$ et *Precision*.

TABLE 5.1
ÉVOLUTION $Severity_n = f(ANOD_N, Precision)$

$ANOD_n$	5	6	7	9	10	12	15	15	15	15	15
Precision	6	6	6	6	6	6	6	7	8	10	12
$Severity_n$	0	1	1.36	2.25	2.77	4	6.25	4.59	3.51	2.25	1.56

Cette notion de *Severity* est une mesure permettant de quantifier l'anormalité. À partir de cette mesure, il est possible d'établir ce que nous appelons des seuils d'alerte (1 à 4 = Faible, 5 à 9 = Modéré et Élevé à partir de 10 par exemple), seuils qui correspondent à des niveaux de gravité. De ces niveaux de gravité, nous pouvons choisir de n'analyser que certaines TLs ayant révélées une anomalie.

5.4.2 Outliers Extraction

À partir des données agrégées de chacune de ces TLs analysées, sont extraites les outliers. Cette étape est réalisée à l'aide d'un autre MLA non supervisé à savoir l'algorithme `Local Outliers Factor` ou `LOF`. Les valeurs aberrantes extraites par `LOF` sont confirmées grâce à l'étude de la distribution de leurs différentes caractéristiques dans le jeu données de la TL considérée. Cette confirmation par l'étude de la distribution nous permet de déterminer le nombre de caractéristiques réseaux constituant la valeur aberrante dont la valeur est au-delà d'un seuil acceptable (90% par défaut). De ces outliers, nous pouvons ensuite remonter jusqu'aux flux réseaux ayant générés cette anomalie grâce à leur index.

5.4.3 Anomalies Correlation

Cette dernière étape a pour but de corréler les anomalies détectées par le système DiNATrA$\mathcal{X}$ avec des événements pouvant expliquer le ou les changements observés dans le trafic réseau. Ce rapprochement peut permettre d'expliquer ou justifier l'alerte levée par le système détection. Ce recoupement n'est pas toujours possible posteriori.

Ce dernier module peut également transmettre un message ou rapport d'anomalie contenant les différentes caractéristiques des flux concernés et les distances calculées. Ce rapport pourrait ensuite être exploité par un système de gestion des informations et des événements de sécurité ou SIEM.

Chapitre 6

DiNATrA$\mathcal{X}$– Mises en œuvre

6.1 Jeu de données CANCAN

Pour cette étude, DiNATrA$\mathcal{X}$ a été testé avec les données CANCAN que nous avons présentées dans la section 2.7 et implémenté en `Python` sous la forme d'un NoteBook `Jupyter` librement téléchargeable depuis notre dépôt `Kaggle` [121], une plateforme web interactive qui permet, entre autres, l'hébergement de ces NoteBook. Pour ce faire, nous avons eu recours à différentes bibliothèques `Python` listées et expliquées dans l'annexe B afin d'étendre les fonctionnalités du langage et répondre à des besoins spécifiques.

6.1.1 Bloc fonctionnel I

6.1.1.1 Collecte des données brutes & Agrégation par TS

DiNATrA$\mathcal{X}$ est une évolution des travaux exposés dans la section 4.4.1. Nous sommes donc repartis des 25 fichiers ARFF présentés dans la section 4.3.2. Ces fichiers n'étant plus exploités avec *Weka* mais analysés à l'aide d'outils `Python`, ils ont été transposés au format '.csv' en supprimant l'entête ARFF avant d'être importés dans un 'dataframe' `Pandas` à l'aide de la bibliothèque `Glob`. Une fois chargés en RAM (10 Go), les champs inutiles sont supprimés et le type de chaque colonne défini au mieux pour réduire l'empreinte mémoire du dataframe (5 Go).

Lors de l'étude de ces données agrégées, il a été constaté que des captures étaient manquantes avant le 1er avril et après le 20 mai 2019. Nous avons alors choisi de ne conserver que les données comprises entre ces dates. Pour permettre une analyse pertinente en ayant suffisamment de données par TS, les données ont à nouveau été agrégées, toujours par groupe applicatif 'appGroup' et BTS

grâce aux coordonnées LambertII, mais cette fois par journée entière. En effet, pour certain TS notamment la nuit, nous ne disposions que de quelques lignes. Un échantillon de ces données agrégées par TS d'une journée, BTS & 'AppGroup' est présenté par le tableau 6.1.

TABLE 6.1
Échantillon des données CANCAN agrégées par jour, BTS & groupe applicatif

TimeSlot	CoordX	CoordY	AppGroup	grpDesc	Duration	Users	Flows	Packets	nPacketUp	nPacketDn
2019-04-01	575120	2425620	0	Unknown	138966.1	45	73	119662	19528	100134
2019-04-01	575120	2425620	1	Web	7651970	15473	50790	39338709	3104910	36233799
2019-04-01	575120	2425620	3	Download	314951.2	1332	4169	18372236	183143	18189093
2019-04-01	575120	2425620	4	CloudStorage	99806.87	762	2775	7627116	1327320	6299796
2019-04-01	575120	2425620	5	Mail	620817.2	1042	1648	1309897	377130	932767
2019-04-01	575120	2425620	8	Control	25130.46	139	174	116759	70070	46689
2019-04-01	575120	2425620	9	Games	360709.7	300	1235	13374142	156102	13218040
2019-04-01	575120	2425620	10	Streaming	2797403	4997	32530	63941887	1125491	62816396
Nombre total de lignes : 2 154 287										

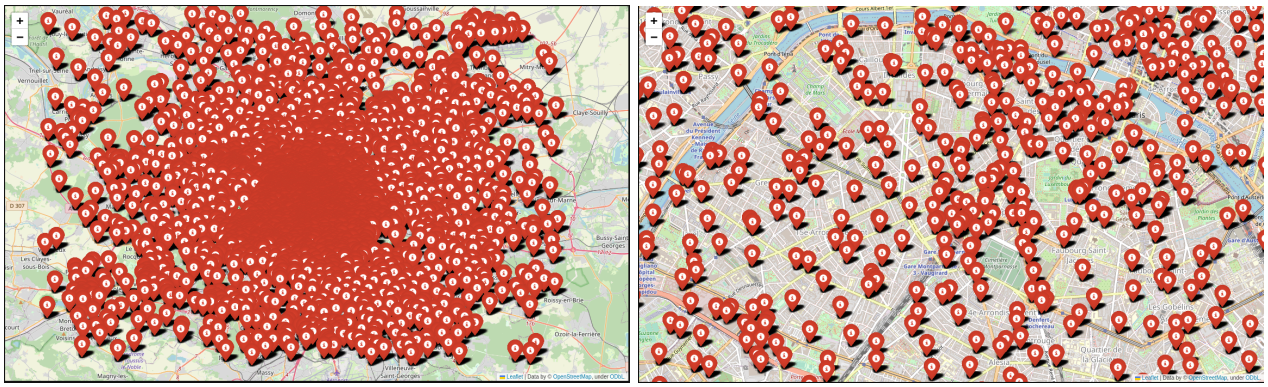
Le tableau 6.2 détaille la correspondance entre les références numériques (utilisées pour les signatures numériques) ou alphabétiques (utilisées pour les DNAs) des groupes applicatifs et leur équivalence en terme d'applications. Ces correspondances ont été fournies par l'opérateur mobile.

TABLE 6.2
Correspondances 'AppGroup' – Type d'application

AppGroup	Type	AppGroup	Type	AppGroup	Type
0 ou A	Unknown	7 ou H	Others	14 ou O	VPN
1 ou B	Web	8 ou I	Control	15 ou P	VVM
2 ou C	P2P	9 ou J	Games	16 ou Q	MMS
3 ou D	Download	10 ou K	Streaming	17 ou R	StreamAVSP
4 ou E	CloudStorage	11 ou L	Chat	18 ou S	Portal
5 ou F	Mail	12 ou M	VoIP		
6 ou G	DB	13 ou N	MailOperator		

6.1.1.2 Regroupement en secteurs

Comme expliqué dans la section 4.2.1, ces données agrégées par TS d'une journée, BTS et 'AppGroup' représentent l'activité générée par les abonnés d'`Orange` sur la partie de son réseau GSM couvrant la région Ile-de-France (voir figure 6.1). La figure 6.1a est la projection sur une carte de l'ensemble des 2 736 BTS incluses dans cette zone et depuis lesquelles le trafic a été capturé. La figure 6.1b est une vue restreinte au centre de Paris. Ces projections cartographiques sont réalisées à l'aide de la bibliothèque `Folium`.



(a) Région *Ile-de-France* (b) Zone *Paris Centre*

FIG. 6.1. Répartition des BTS Orange

La solution retenue avec DiNATrA$\mathcal{X}$ a été d'utiliser *DBSCAN*, un algorithme fourni par la bibliothèque `Scikit-learn` permettant d'agréger des données géospatiales. Celui-ci a été implémenté par [122] dans le cadre de leurs travaux sur la détection de points à forte activité à partir des données géographiques de différentes sources. Son fonctionnent est détaillé dans la section 2.5.2. Une mise en œuvre de `DBSCAN` afin d'agréger des données à partir de leurs coordonnées géographique a été proposée, expliquée et implémentée dans l'article [123]. Dans son implémentation `Python` [124], `DBSCAN` requiert de définir plusieurs hyper-paramètres dont :

min_samples Le nombre d'échantillons dans un voisinage pour qu'un point soit considéré comme un point central. Cela inclut le point lui-même.

metric La métrique à utiliser pour calculer la distance entre les instances.

algorithm L'algorithme utilisé pour calculer les distances et déterminer les plus proches voisins.

epsilon (ϵ) La distance maximale entre deux échantillons pour que l'un soit considéré comme étant dans le voisinage de l'autre. Il ne s'agit pas d'une limite maximale pour les distances entre les points d'un groupe. Il s'agit du paramètre de `DBSCAN` le plus important à choisir en fonction du jeu de données et de la fonction de distance utilisée.

Le paramètre *min_samples* correspondant à la taille minimale des clusters, tout le reste étant considéré comme du bruit, a été fixé à '1' afin que chaque point de données soit assigné à un cluster ou forme son propre cluster de taille '1' ce que l'on nomme un *singleton*. Dans le cas d'usage de DiNATrA$\mathcal{X}$, aucune donnée ne sera classée comme bruit, de sorte que nous conservons les observations

éloignées comme des points importants et non du bruit.

Nous utilisons la métrique de `Haversine` avec un partitionnement de l'espace du type 'BallTree' avec l'algorithme `k-NearestNeighbors` pour calculer les distances en grand cercle entre les différents points comme détaillé par [125]. La formule de `Haversine` permet de déterminer la distance du grand cercle entre deux points d'une sphère ce qui correspond à la plus courte distance entre ces deux points. Elle est généralement employée pour mesurer la distance entre deux points situés à la surface du globe terrestre, à partir de leur longitude et leur latitude.

La mise en œuvre de `DBSCAN` avec `Scikit-learn` et la métrique HAVERSINE requiert des coordonnées géographiques exprimées en radians. L'algorithme 6 présente comment il a été employé dans DiNATrA$\mathcal{X}$ avec *radius*, le rayon de la zone à considérer et *df*, le dataframe contenant les données agrégées par jour, BTS et 'AppGroup'.

Le principe de fonctionnement de l'algorithme `k-NearestNeighbors` consiste en ce qu'un objet soit classé par un vote à la pluralité de ses voisins. L'objet est affecté à la classe la plus fréquente parmi ses k voisins les plus proches où k est un nombre entier positif souvent petit. Un partitionnement de l'espace de type 'BallTree' constitue une structure de données permettant d'organiser des points dans un espace multidimensionnel sous forme d'un ensemble imbriqué de boules.

Nous terminons le paramétrage de `DBSCAN` avec le paramètre ϵ. Il détermine la distance maximale à laquelle les points peuvent être éloignés les uns des autres pour qu'ils soient considérés comme un groupe. Celle-ci a été fixée à '0.15', ce qui correspond à '150' mètres. Nous avons choisi cette valeur d'après l'étude faite à la section 4.2.1. En effet, il a été constaté qu'une précision de '6' pour les geohashes était la plus pertinente. Le quadrillage géographique résultant permettait d'obtenir des zones adaptées à l'étude de secteurs présentant un intérêt particulier comme le *Stade de France* ou *Notre-Dame de Paris*, ce qui convenait parfaitement à notre cas d'usage. Le tableau 6.3 précise les dimensions des geohashes en fonction de leur précision ainsi que le nombre de secteurs obtenus avec le jeu de données agrégées. Quant au tableau 6.4, il détaille le nombre de clusters formés par `DBSCAN` avec les mêmes données agrégées, et ce, en fonction du rayon utilisé ; rayon défini par le paramètre ϵ ainsi que la précision théorique équivalente pour les geohashes pour obtenir à peu près le même nombre de secteurs.

TABLE 6.3
DÉCOUPAGE EN GEOSHASHES

Précision	Dimensions (km)	Surface	Secteurs
8	.038 / .019	722 m^2	2 700
7	.150 / .150	22 500 m^2	2 550
6	1.20 / 0.60	0.72 km^2	1 230
5	4.90 / 4.90	24 km^2	110

TABLE 6.4
REGROUPEMENT AVEC `DBSCAN`

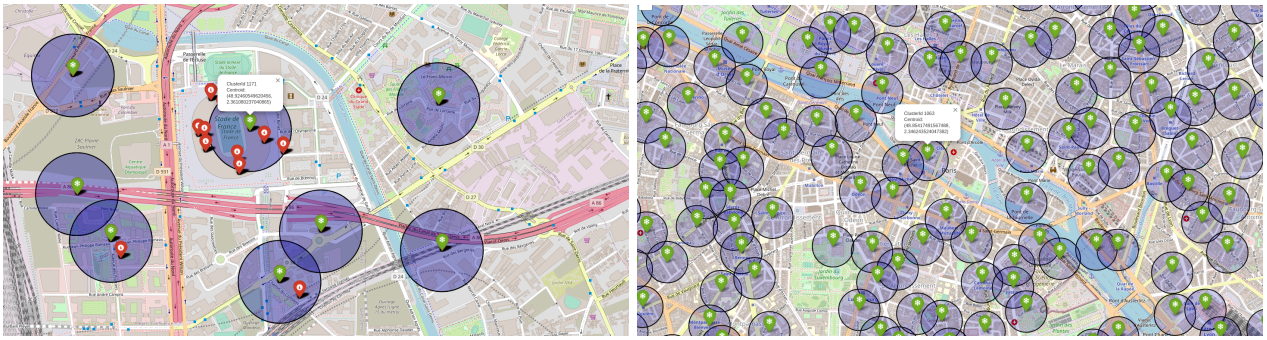
Rayon (km)	Secteurs	Précision théorique
0.010	2 660	8
0.015	2 650	8
0.075	2 450	7
0.085	2 400	7
0.150	2 020	6.5
0.200	1 729	6.5
0.300	1 310	6
0.480	880	6
2.500	4	5
2.800	4	5

En comparant le nombre de secteurs obtenus avec les deux systèmes : *(i)* découpage par le système `Geohash` avec une précision de '6' *(ii)* regroupement par `DBSCAN` avec un rayon de 300 mètres soit $\epsilon = 0.3$, nous constatons que les résultats obtenus sont similaires en nombre de secteurs. Or, cette résolution et donc le découpage obtenu ne permet pas d'étudier des secteurs de faible dimension.

Nous avons testé un regroupement par `DBSCAN` en utilisant un rayon de 200 mètres. Mais, les clusters obtenus étaient encore trop étendus pour pouvoir analyser des secteurs particuliers comme celui correspondant à *Notre-Dame de Paris*. La figure 6.2 présente les clusters obtenus avec un rayon de 200 mètres où nous pouvons constater qu'aucun cluster ne couvre l'*Ile de la Cité* située au centre de la carte. D'où, le choix d'un rayon de 150 mètres sachant que les clusters générés ne sont pas des cercles parfaits.

FIG. 6.2. Regroupement en secteurs par `DBSCAN` (rayon = 200 m)

La figure 6.3 représente l'agrégation des BTS en secteurs ayant un rayon de 150 mètres. Cette projection n'est pas une représentation exacte de l'affectation des BTS dans les clusters, mais permet de situer les clusters et donc les secteurs en fonction de leur centroïde. Le centroïde d'un cluster est un point réel ou imaginaire représentant son centre. Le centroïde de chaque secteur est calculé grâce aux fonctions `MultiPoint` et `great_circle` fournies par les bibliothèques `Shapely` et `Geopy` comme expliqué par l'algorithme 7.



(a) Secteur Stade de France avec quelques BTS (marqueurs rouges)

(b) Secteurs zone Notre-Dame de Paris et leur centroïde (marqueurs verts)

FIG. 6.3. Regroupement en secteurs par `DBSCAN` (rayon = 150m)

La figure 6.3a détaille le secteur du *Stade de France* et quelques BTS qui le composent et dont les données ont été agrégées. La figure 6.3b est une vue de quelques secteurs situés dans la zone de *Notre-Dame de Paris*. Comme indiqué dans les détails décrivant chaque secteur, un numéro unique identifie les clusters ou secteurs, le *ClusterId*. Cet identifiant permet, lors de l'étape suivante, de sélectionner des secteurs particuliers et de constituer les SOIs. Un échantillon de ces données agrégées par TS, 'Cluster' et 'AppGroup' est présenté dans le tableau 6.5.

TABLE 6.5
DONNÉES AGRÉGÉES PAR TS, 'CLUSTERID' & 'APPGROUP'

TimeSlot	ClusterId	grpDesc	AppGroup	Duration	Users	Flows	Packets	nPacketUp	nPacketDn
2019-05-07	2020	Chat	11	2 230 320	4365	13 480	10 331 799	1 271 757	9 060 042
2019-05-09	2020	Chat	11	2 342 478	4324	15406	11 660 577	1 745 193	9 915 384
2019-05-08	2021	Web	1	595.825	2	4	765	87	678
2019-05-01	2021	Web	1	1 546,769	3	3	971	677	294
2019-04-29	2021	VVM	15	18.084	1	1	145	15	130
2019-05-04	2021	VVM	15	184.789	2	2	448	54	394
2019-04-29	2021	Unknown	0	24 563,9	3	3	2 376	1 323	1 053
2019-05-04	2021	Unknown	0	126 536,4	3	3	3 105	1 939	1 166
2019-05-03	2021	Streaming	10	382.641	1	1	93	6	87
Total lignes : 1 716 657									

6.1.2 Bloc fonctionnel II

L'algorithme `DBSCAN` est *déterministe* car il génère toujours les mêmes clusters s'il reçoit les mêmes données et dans le même ordre. Cependant, les résultats peuvent différer lorsque les données sont fournies dans un ordre différent. Même si les échantillons principaux sont toujours assignés aux mêmes clusters, les 'ClusterId' dépendent de l'ordre dans lequel ces échantillons sont rencontrés dans les données. De plus, les clusters auxquels les échantillons non fondamentaux sont affectés peuvent différer en fonction de l'ordre dans lequel les données sont traitées.

Étant donné que les données sont traitées par TP, et ce, pour l'ensemble des BTS, celles-ci sont chargées en une seule fois dans un dataframe puis ne sont plus modifiées (voir la section 6.1.1.1). Le fait que `DBSCAN` soit déterministe nous permet donc de sélectionner des secteurs particuliers par leur 'ClusterId' et donc de définir les SOIs qui seront analysés par la suite. Ce processus d'analyse peut, de fait, être réitéré afin d'analyser plusieurs SOIs. De plus, si la phase de regroupement en secteurs doit être répétée, la répartition des BTS dans les secteurs sera identique à condition bien-sûr que le dataframe contenant les données ne soit pas modifié.

6.1.2.1 Découpage du SOI par TL

Un SOI peut être constitué d'un ou plusieurs secteurs définis par leur 'ClusterId' comme expliqué dans la section 5.3. Pour définir les SOIs, il faut par conséquent extraire uniquement les données du ou des secteurs concernés. Cette sélection des clusters peut être opérée soit visuellement en projetant les secteurs sur un fond cartographique soit en recherchant les 'ClusterId' des secteurs à conserver à partir des coordonnées GPS de leur centroïde.

Une fois les SOIs à analyser sélectionnés, la période d'analyse considérée (TP) est découpée en tranches de temps équivalentes (TL). Bien évidemment, la durée TL est inférieure à la période TP (voir figure 5.2). Les colonnes 'TimeSlot' et 'grpDesc' sont supprimées pour respectivement : *(i)* éviter de biaiser l'algorithme `LOF` *(ii)* utiliser le MLA `LOF` qui ne sait pas traiter les chaînes de caractères. Cette extraction des données à conserver pour analyse et ce découpage en TL sont basés sur le principe des compréhensions de listes offert par `Pandas` [126]. Un exemple d'implémentation PYTHON est donné par l'algorithme 8. Nous avons sélectionné pour cette étude une dizaine de SOIs composés d'un ou plusieurs secteurs. L'ensemble de ces SOIs et des secteurs les composants est détaillé par le tableau 6.6.

La figure 6.4 est la projection des secteurs composant deux de ces SOIs.

TABLE 6.6
QUELQUES SOIS & SECTEURS LES COMPOSANT

SOI	Secteur(s)
Stade de France	1168
Notre-Dame de Paris	1060 + 1037
Gare Montparnasse	844 + 848 + 855 (figure 6.4a)
Gare du Nord	1125 + 1149 + 1160
Gare de Lyon	1286 + 1306 + 1338
Gare de l'Est	1146 + 1208
Cimetière du Montparnasse	948
Cimetière de Montmartre	962 (figure 6.4b)

(a) SOI *Gare de Lyon* (3 secteurs) (b) SOI *Cimetière de Montmartre* (1 secteur)

FIG. 6.4. Projection & Définition de deux SOIs (rayon = 150 mètres)

Les données agrégées relatives à chacun de ces SOIs ont été découpées par TL pour en définir leur signature numérique. Le tableau 6.7 est un échantillon des données agrégées représentant une des TLs du secteur du *Stade de France*, données qui seront exploitées par les MLAs `K-Means` et `LOF` lors de l'étape suivante. Le détermination de l'amplitude des périodes TL dépend du type fonctionnel de SOI à analyser (voir section 5.2.3) et de la fréquence de l'éventuelle anomalie à rechercher.

TABLE 6.7
TL_1 SOI *Stade de France* (2019-04-06)

index	AppGroup	Duration	Users	Flows	Packets	nPacketUp	nPacketDn
987803	1	4 443 470	7 946	24 736	21 448 230	3 225 036	18 223 194
987850	12	18 108	11	11	11 051	3 135	7 916
987899	15	1 054	10	24	5 322	380	4 942
987953	14	342.247	4	4	772	688	84
988014	0	49 690	18	22	100 683	27 812	72 871
988046	10	1 036 163	2 187	12 936	26 584 005	528 087	26 055 918
988118	17	11 649	15	766	2 761 480	2 946	2 758 534
988675	11	821 286	1 068	6 583	5 811 070	617 084	5 193 986

6.1.2.2 Définition de la signature des TLs

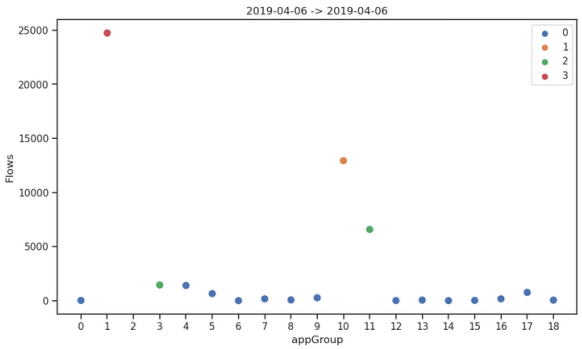
Pour chaque SOI, est calculée la signature numérique de chacune des TLs grâce au MLA non supervisé `K-Means`. L'apprentissage automatique est une composante importante de la science des données. Grâce à l'utilisation de méthodes statistiques, ces algorithmes peuvent effectuer des classifications ou des régressions, ce qui permet de découvrir des informations essentielles dans le cadre de projets d'exploration de données.

Dans leurs travaux présentés dans les articles [127], [128], les auteurs doivent effectuer une analyse statistique pour chacun des paramètres contenus dans les captures réseaux. Dans le cadre de DiNATrA$\mathcal{X}$, ces analyses sont effectuées par l'algorithme `K-Means`. Les données de chaque TL sont regroupées en clusters. La répartition des données dans les clusters ainsi formés sur la base de la caractéristique $\mathcal{F}$ constitue la signature numérique de chacune de ces TLs, les TLDS (*TimeSlice Digital Signature*).

Pour l'analyse des données CANCAN, nous avons choisi d'étudier le paramètre 'AppGroup' afin de détecter des anomalies relatives au changement de comportement des utilisateurs, c'est-à-dire le type d'application mobile utilisé (Web, Chat, Streaming,...). L'ensemble des figures 6.5 à 6.6 sont les signatures numériques de quelques-uns des SOIs sélectionnés pour $\mathcal{F}$ = 'AppGroup' avec, reporté en ordonnée, le total de flux échangés pour chacun des groupes applicatifs.

Stade de France Les figures 6.5a à 6.5e matérialisent les TLDS des trois derniers samedis d'avril et du premier samedi de mai 2019 pour le SOI du *Stade de France* avec la répartition de $\mathcal{F}$ par clusters. Nous pouvons déduire de ces signatures que la répartition de $\mathcal{F}$ dans chacun des clusters est identique pour les figures 6.5a à 6.5c à condition de *permuter* les clusters 2 et 3 et d'*exclure* le groupe applicatif '17'. La permutation des clusters est due au fonctionnement de `K-Mean`. De plus, l'AppGroup' '17' est un groupe à très faible activité donc négligeable ici.

Les figures 6.5c et 6.5d sont différentes dans leurs répartitions des clusters notamment par les groupes applicatifs 3, 4 et 11 alors que les figures 6.5c et 6.5e sont quasiment identiques à l'exception du 'AppGroup' '4' et à condition de réorganiser la numérotation des clusters 1, 2 et 3. Nous pouvons en conclure que les signatures $TLDS_1$, $TLDS_2$ et $TLDS_4$ sont quasiment identiques contrairement à $TLDS_3$ qui est différente des trois autres, mais *nous ne pouvons pas quantifier cette différence.*



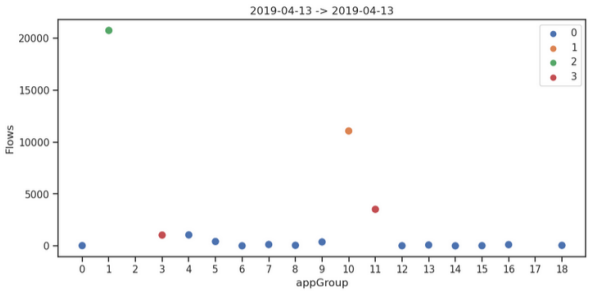
(a) 1er samedi d'avril

Cluster0 (bleu) 0, 4 à 9, 12 à 18

Cluster1 (orange) 10 uniquement

Cluster2 (vert) 3 & 11

Cluster3 (rouge) 1 uniquement



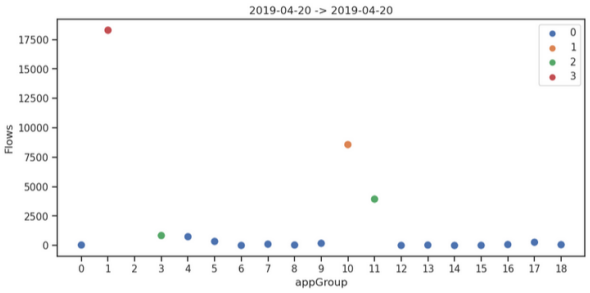
(b) 2e samedi d'avril

Cluster0 (bleu) 0, 4 à 9, 12 à 18 (sauf 17)

Cluster1 (orange) 10 uniquement

Cluster2 (vert) 1 uniquement

Cluster3 (rouge) 3 & 11



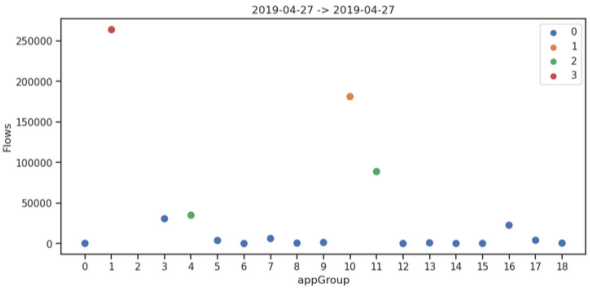
(c) 3e samedi d'avril

Cluster0 (bleu) 0, 4 à 9, 12 à 18

Cluster1 (orange) 10 uniquement

Cluster2 (vert) 3 & 11

Cluster3 (rouge) 1 uniquement



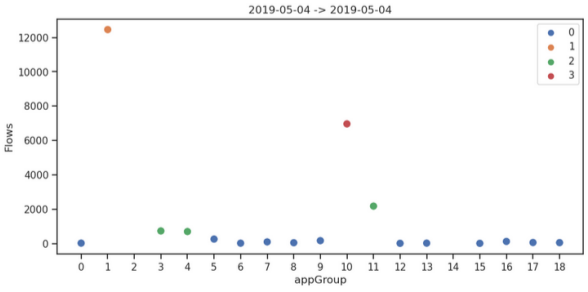
(d) 4e samedi d'avril

Cluster0 (bleu) 0, 3 à 9 (sauf 4), 12 à 18

Cluster1 (orange) 10 uniquement

Cluster2 (vert) 4 uniquement

Cluster3 (rouge) 1 uniquement



(e) 1er samedi de mai

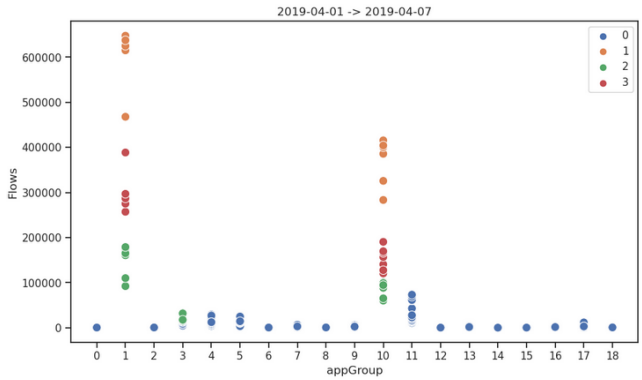
Cluster0 (bleu) 0, 5 à 9, 12 à 18 (sauf 14)

Cluster1 (orange) 1 uniquement

Cluster2 (vert) 3, 4 & 11

Cluster3 (rouge) 10 uniquement

FIG. 6.5. TLDS du SOI *Stade de France*
(TP = 1 mois & TL = 1 journée)



(a) 1re semaine d'avril 2019

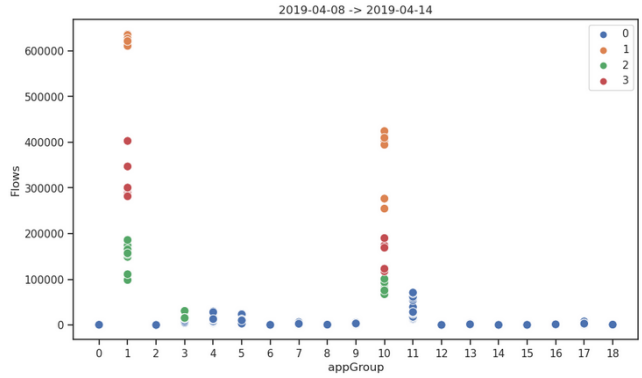
Gare du Nord

Cluster0 (bleu) 0 à 18 (sauf 1 & 10)

Cluster1 (orange) 1 & 10

Cluster2 (vert) 1, 3 & 10

Cluster3 (rouge) 1 & 10



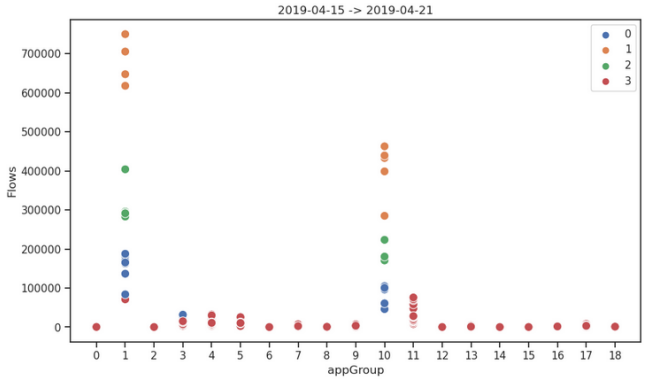
(b) 2e semaine d'avril 2019

Cluster0 (bleu) 0 à 18 (sauf 1 & 10)

Cluster1 (orange) 1 & 10

Cluster2 (vert) 1, 3 & 10

Cluster3 (rouge) 1 & 10



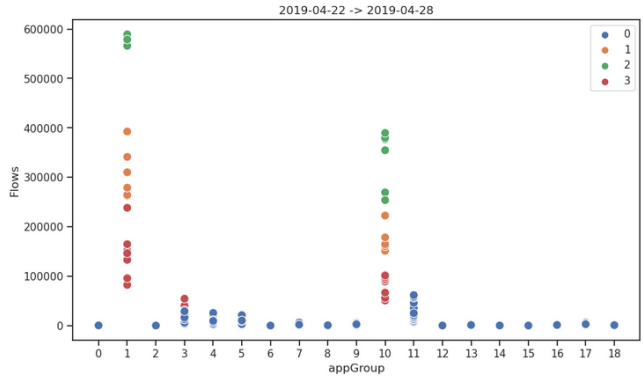
(c) 3e semaine d'avril 2019

Cluster0 (bleu) 1, 3 & 10

Cluster1 (orange) 1 & 10

Cluster2 (vert) 1 & 10

Cluster3 (rouge) 0 à 18 (sauf 10)



(d) 4e semaine d'avril 2019

Cluster0 (bleu) 0 à 18 (sauf 1 & 10)

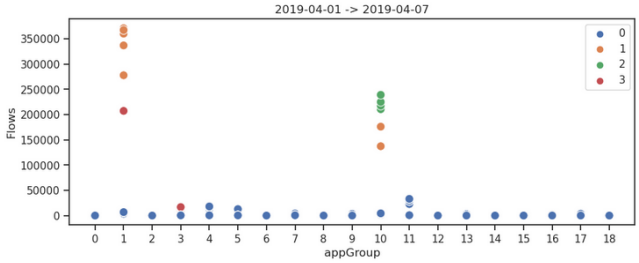
Cluster1 (orange) 1 & 10

Cluster2 (vert) 1 & 10

Cluster3 (rouge) 1, 3 & 10

FIG. 6.6. TLDS SOI *Gare du Nord*
(TP = 1 mois & TL = 1 semaine)

Des figures 6.6a et 6.6d, nous pouvons déduire que la répartition des clusters est parfaitement identique pour les signatures numériques $TLDS_1$ à $TLDS_4$ à condition de permuter la numérotation des clusters 0 et 3 pour la signature $TLDS_3$ comme exposé par la figure 6.7c.



(a) 1re semaine d'avril

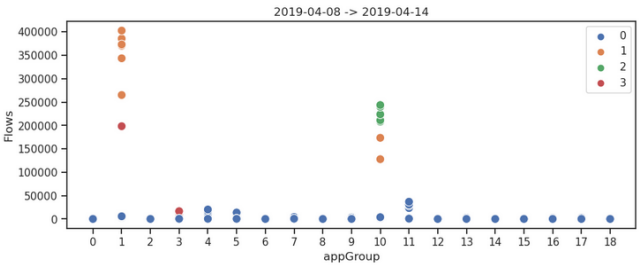
Notre-Dame de Paris

Cluster0 (bleu) 0 à 18

Cluster1 (orange) 1 & 10

Cluster2 (vert) 10 uniquement

Cluster3 (rouge) 1 & 3



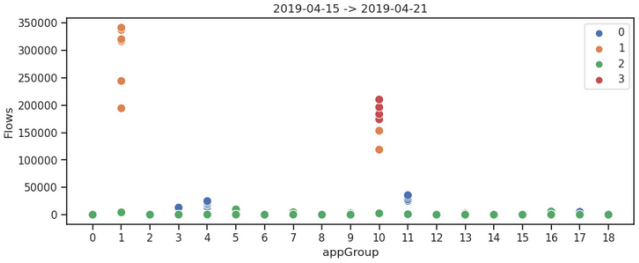
(b) 2e semaine d'avril

Cluster0 (bleu) 0 à 18

Cluster1 (orange) 1 & 10

Cluster2 (vert) 10 uniquement

Cluster3 (rouge) 1 & 3



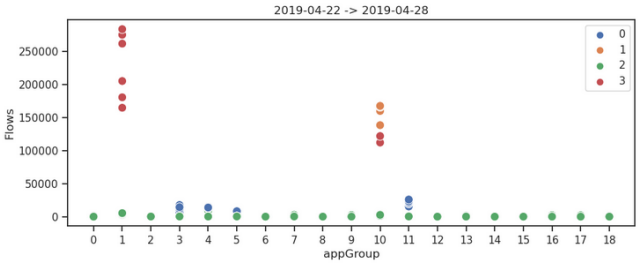
(c) 3e semaine d'avril

Cluster0 (bleu) 3, 4 & 11

Cluster1 (orange) 1 & 10

Cluster2 (vert) 0 à 18

Cluster3 (rouge) 10 uniquement



(d) 4e semaine d'avril

Cluster0 (bleu) 3, 4 & 11

Cluster1 (orange) 10 uniquement

Cluster2 (vert) 0 à 18

Cluster3 (rouge) 1 & 10

FIG. 6.7. TLDS du SOI *Notre-Dame de Paris*
(TP = 1 mois & TL = 1 semaine)

Les figures 6.7a à 6.7d matérialisent les TLDSs des quatre semaines d'avril 2019 pour le SOI *Notre-Dame de Paris*. Des figures 6.7a et 6.7d, nous pouvons déduire que la répartition des clusters est parfaitement identique pour les signatures numériques $TLDS_1$ et $TLDS_2$. Nous constatons un changement d'activité à partir de la troisième semaine. $TLDS_3$ (figure 6.7c) est différente de $TLDS_2$ (figure 6.7b) mais cette répartition est conservée pour la dernière semaine car $TLDS_3$ et $TLDS_4$ sont identiques à condition de permuter les clusters 1 et 3 comme exposé par les figures 6.7c et 6.7d.

Une fois toutes les signatures numériques générées, l'étape suivante consiste à les comparer deux par deux de la première jusqu'à la dernière. Si une évolution de la répartition des clusters est constatée entre deux signatures numériques consécutives, cela implique qu'il y a eu une variation de la caractéristique $\mathcal{F}$, ici le type d'application mobile utilisée par les abonnés Orange ('AppGroup').

Cette comparaison des TLDS impose une intervention humaine, car requiert une interprétation visuelle. En effet, comme exposé dans la section 4.4.2.4, la numérotation des clusters et donc la couleur qui leur est associée peut varier d'une signature à l'autre. De ce fait, bien que la répartition des données soit identique entre deux signatures, celles-ci pourraient être interprétées comme différentes si une méthode de comparaison automatisée était employée. D'où les concepts de 'Digital Network Assessment' ou *DNA* et de 'Strand' ou *brin associé* définis et utilisés par le processus suivant.

6.1.2.3 Calcul des DNAs & Brin associé

Pour chaque signature est calculé ce que nous avons appelé le *DNA* et son *brin associé*. La notion de DNA a été proposée par [129] et employée dans les travaux [130], [131] pour modéliser la façon dont des modules plus petits, assurant chacun une certaine fonction notamment réseau, sont utilisés par des modules plus grands qui exécutent des fonctions plus complexes. Ce cadre de modélisation explique comment cette structure hiérarchique (qui est fondamentalement une propriété du réseau) émerge et comment elle évolue dans le temps. Nous nous sommes basés sur cette idée pour développer notre concept de DNA appliqué à la détection d'anomalies réseaux.

Le calcul des DNAs nous fournit une première indication sur l'existence ou pas d'une éventuelle anomalie. Ensuite, est calculée la distance d'anormalité de chaque brin. Comme détaillé dans la section 5.3.3, les nucléotides composant les brins sont triés suivant leur occurrence, définie par un paramètre autre que celui utilisé pour construire les DNAs, à savoir ici le nombre d'utilisateurs par 'AppGroup'. Ces deux informations nous permettent d'affirmer ou d'infirmer la présence d'une

anomalie sur le SOI considéré lors du passage de TL_n à TL_{n+1}. Ce calcul est effectué pour chaque TLDS de la période TP, et ce, pour le SOI considéré. Ce processus est répété pour chacun des SOIs à étudier à l'aide des algorithmes 9 et 10 où les variables passées aux fonctions sont :

dfs une liste de dataframes contenant les données agrégées par TS

ts la liste des t-uplets précisant les débuts et fins des différentes TS

nclusters le nombre de clusters utilisés pour construire les signatures numériques

labels la liste contenant la répartition des données de chaque TS aux seins des clusters composant les TLDS

precision la longueur des brins et des séquences composant les DNAs

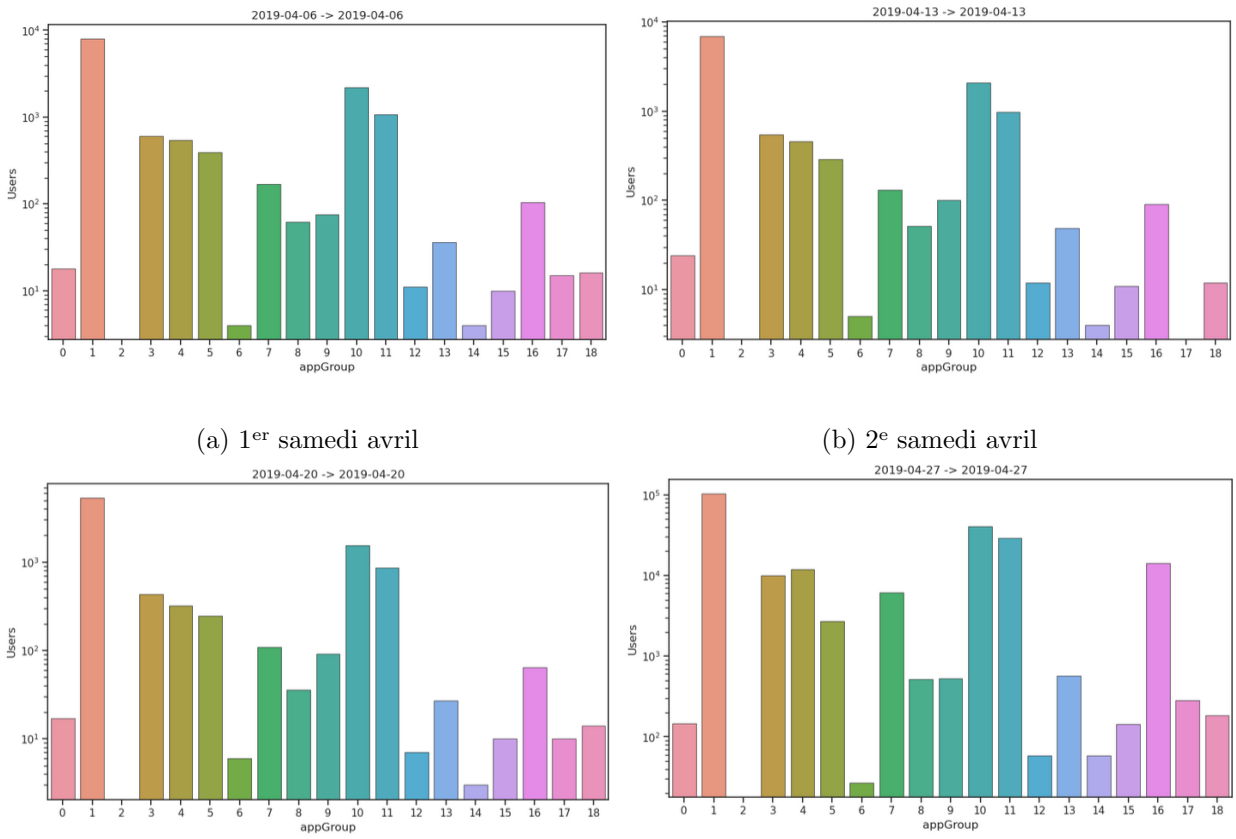
activity le paramètre utilisé pour mesurer la caractéristique $\mathcal{F}$ et donc trier les séquences ainsi que les lettres composant les brins

df une variable locale contenant successivement les dataframes de la liste dfs

Les résultats obtenus pour différents SOIs sont présentés et expliqués dans les sections suivantes sur la base des tableaux 6.8 à 6.11 et des figures 6.8 à 6.11.

TABLE 6.8
DNAs & Brins SOI *Stade de France*

n	Begin	End	DNA_n	$Strand_n$	Volumes
0	2019-04-06	2019-04-06	B-K-LD-EFHQJI	BKLDEF	59.93-16.49-12.57-10.15
1	2019-04-13	2019-04-13	B-K-LD-EFHJQI	BKLDEF	58.98-17.58-12.95-9.49
2	2019-04-20	2019-04-20	B-K-LD-EFHJQI	BKLDEF	58.33-16.93-14.19-9.52
3	2019-04-27	2019-04-27	B-LE-K-QDHFNJ	BKLQED	46.66-18.72-18.43-15.54
Précision=6, TP=Avril & TL=1 jour					



(a) 1er samedi avril

(b) 2e samedi avril

(c) 3e samedi avril

(d) 4e samedi avril

FIG. 6.8. Activité SOI *Stade de France*

Stade de France Pour l'analyse du SOI du *Stade de France*, nous avons découpé la période TP, correspondant au mois d'avril 2019, en tranches de 24 heures afin de détecter des anomalies liées à un changement dans l'activité quotidienne des utilisateurs inscrits dans cette zone du réseau mobile d'Orange. Nous avons choisi d'étudier uniquement les TLs des quatre samedis contenus dans la période TP.

TL_0 contient donc les données se rapportant à l'activité réseau du samedi 06 avril 2019. La figure 6.5a représente $TLDS_0$, la signature numérique de TL_0. DNA_0, le DNA correspondant à cette signature, est constitué de quatre séquences, car `K-Mean` est configuré pour construire des signatures numériques en répartissant les données dans quatre clusters différents.

Seq_3 est l'expression du $cluster_3$ en fonction de la caractéristique $\mathcal{F}$ ici 'AppGroup'. $Seq_3 = B$, car le $cluster_3$ est uniquement constitué de 'AppGroup' '1' qui correspond à la lettre 'B' qui représente

les applications de type 'Web' comme détaillé dans le tableau 6.2. $Seq_1 = K$, car le $cluster_3$ est uniquement constitué de 'AppGroup' '10'. $Seq_2 = LD$, car le $Cluster_2$ est constitué des 'AppGroup' '3' & '11'. $Seq_0 = EFHQJI$, car le $Cluster_0$ est constitué des 'AppGroup' '0', '4' à '9' & '12' à '18'. Les séquences sont tronquées à une longueur égale à $Precision$. Plus $Precision$ est grande, plus le nombre de valeurs différentes de $\mathcal{F}$ prises en compte est grand. Ce qui signifie que des groupes applicatifs dont l'activité est de plus en plus faible seront considérés.

Les nucléotides formant les séquences ('appGroupL') sont triées en fonction de la caractéristique décrivant l'activité des 'AppGroup', ici 'Users' qui représente le nombre d'utilisateurs par type d'application, activités détaillées par la figure 6.8a.

Enfin, pour chaque séquence est calculé le volume de données représentées par la séquence, exprimé en pourcentage (59.93-16.49-12.57-9.68). Les séquences sont ensuite triées dans l'ordre décroissant des volumes pour construire le DNA d'où : $DNA_0 = Seq_3 - Seq_1 - Seq_2 - Seq_0 = B - K - LD - EFHQJI$.

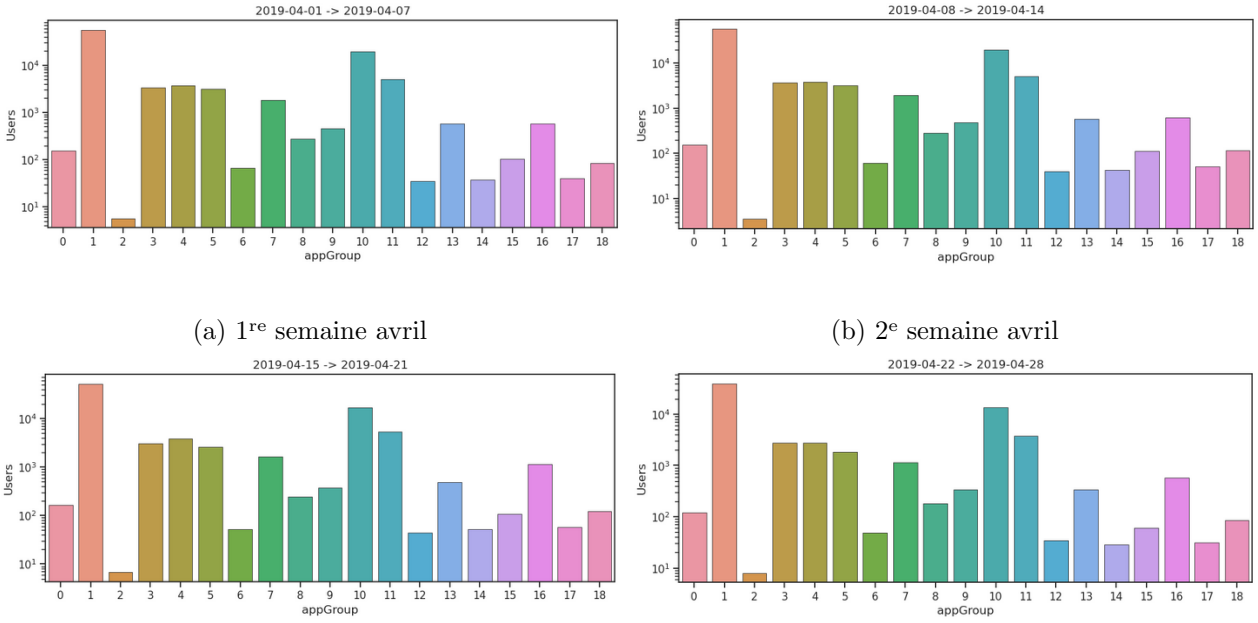
Le brin $Strand_0$, associé au DNA_0, est construit avec les nucléotides contenus dans DNA_0, triés dans l'ordre décroissant de l'activité et tronqué à la longueur $Precision$: $Strand_0 = BKLDEF$.

Ce processus est répété pour chacune des TLs. Les résultats obtenus sont détaillés dans le tableau 6.8. Nous pouvons en déduire que : *(i)* les brins $Strand_0$ à $Strand_2$ sont identiques *(ii)* les DNAs DNA_0 à DNA_2 sont quasiment identiques à l'exception des groupes applicatifs 'J' et 'Q' représentant respectivement 'Games' et 'MMS' permutés dans DNA_0 *(iii)* les volumes d'activité par séquences sont du même ordre de grandeur pour les périodes TL_0 à TL_2 *(iv)* le DNA, son brin et les volumes d'activité de la période TL_3 sont différents ce qui met en évidence une anomalie réseau pour cette période par rapport aux périodes TL_0 à TL_2.

Du tableau 6.2, nous pouvons conclure que les samedis 6, 13 et 20 avril 2019, sur le SOI du *Stade de France*, les utilisateurs employaient leur téléphone mobile essentiellement pour les activités 'BKLDEF' soit 'Web' (60%), 'Streaming' (17%), 'Chat' & 'Download' (13%), 'CloudStorage', 'Mail' et autres (pour 10%). Pour le samedi 27 avril 2019, l'activité correspondait à 'BKLQED' soit 'Web' (47%), 'Chat' & 'CloudStorage' (pour 19%), 'Streaming' (18%), 'MMS', 'Download' et autres (15.5%).

TABLE 6.9
DNAs & Brins SOI *Notre-Dame de Paris*

n	Begin	End	DNA_n	$Strand_n$	Volumes
0	2019-04-01	2019-04-07	BKLED-BK-BD-K	BKLED	92.22-79.42-62.57-20.41
1	2019-04-08	2019-04-14	BKLED-BK-BD-K	BKLED	92.1-78.99-62.5-20.25
2	2019-04-15	2019-04-21	BKLED-BK-K-LEDR	BKLED	91.96-77.99-19.3-14.02
3	2019-04-22	2019-04-28	BKLDE-BK-K-LDEF	BKLDE	92.88-79.15-20.17-16.46
4	2019-04-29	2019-05-05	BKLED-BK-K-LED	BKLED	93.02-79.78-20.52-13.24
5	2019-05-06	2019-05-12	BKLED-BK-BK-D	BKLED	92.65-79.86-79.86-3.68
6	2019-05-13	2019-05-20	BKLED-BK-BD-K	BKLED	92.13-79.63-63.46-20.05
Précision=5, TP=Avril+Mai & TL=1 semaine					



(a) 1re semaine avril

(b) 2e semaine avril

(c) 3e semaine avril

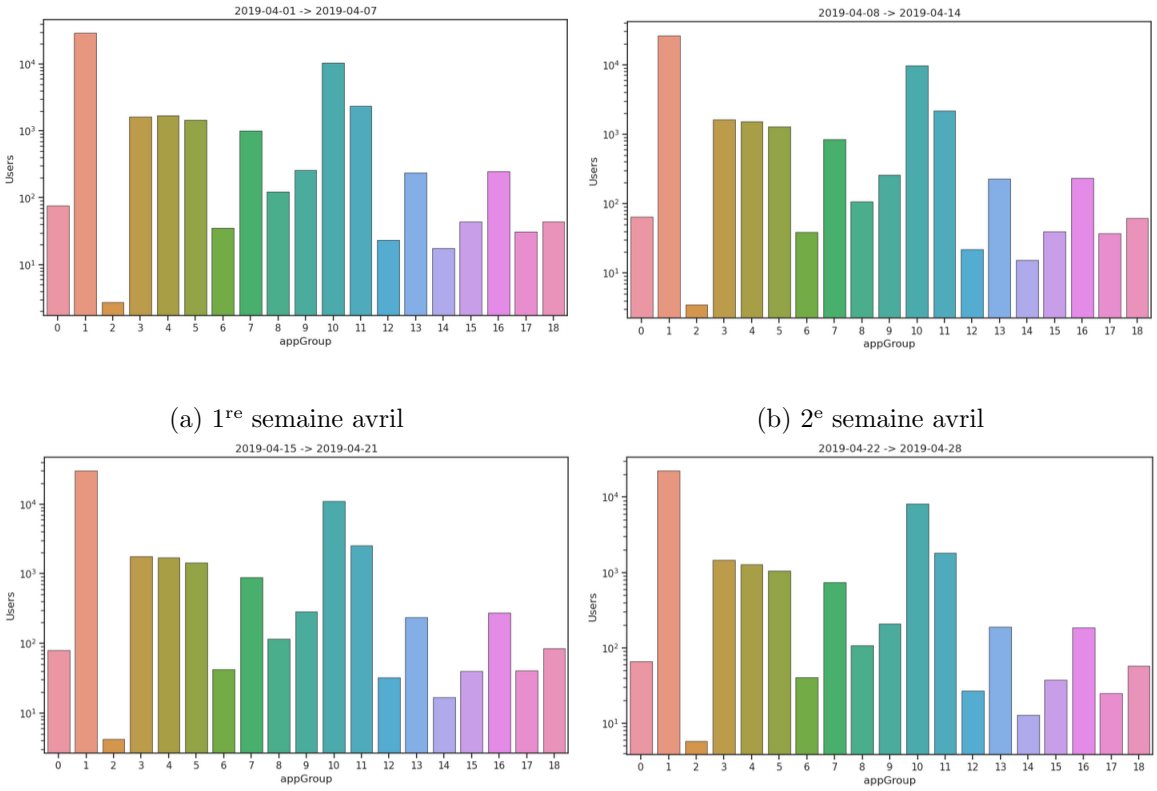
(d) 4e semaine avril

Fig. 6.9. Activité SOI *Notre-Dame de Paris*

Notre-Dame de Paris Pour l'analyse du SOI de *Notre-Dame de Paris*, nous avons choisi de diviser la période TP, correspondant aux mois d'avril et mai 2019, en semaines entières du lundi au dimanche sachant que le 1er avril 2019 était un lundi. Ce découpage nous permet de détecter des anomalies liées à un changement dans l'activité hebdomadaire des utilisateurs. Des résultats obtenus, présentés dans le tableau 6.9 et les figures 6.9a à 6.9d, nous pouvons en déduire que : *(i)* les brins sont quasiment tous identiques *(ii)* les DNAs DNA_0, DNA_1 et DNA_6 sont identiques *(iii)* les volumes d'activité par séquences sont du même ordre de grandeur pour les périodes TL_0, TL_1 et TL_6 *(iv)* les volumes

d'activité par séquences sont du même ordre de grandeur pour les périodes TL_2 à TL_4 *(v)* le DNA_5 semble marquer une transition entre les périodes TL_4 et TL_6 avec le retour progressif des 'AppGroup' B et D *(vi)* les DNAs, leur brin et les volumes d'activité des périodes TL_2 à TL_5 sont différents ce qui met en évidence une anomalie réseau pour ces périodes.

D'après le tableau 6.2, nous voyons apparaitre dans les DNA_2 et DNA_3, les groupes applicatifs de type 'Mail' (F) et 'StreamAVSP' (R) ainsi que 'Chat' (L) et 'CloudStorage' (E) dans les troisièmes séquences des DNAs.



(a) 1re semaine avril

(b) 2e semaine avril

(c) 3e semaine avril

(d) 4e semaine avril

FIG. 6.10. Activité SOI *Gare de l'Est*

TABLE 6.10
DNAs & Brins SOI *Gare de l'Est*

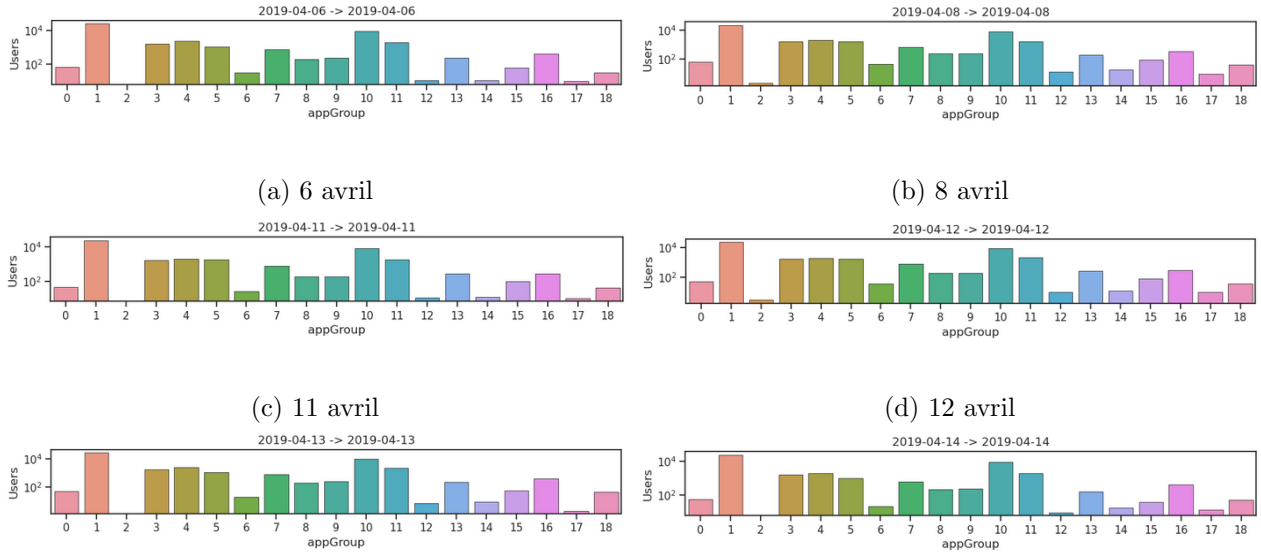
n	Begin	End	DNA_n	$Strand_n$	Volumes
0	2019-04-08	2019-04-14	BKLDEFH-BK-BD-K	BKLDEFH	97.61-80.9-62.8-21.71
1	2019-04-15	2019-04-21	BKLDEFH-BK-BK-K	BKLDEFH	97.7-81.29-81.29-21.68
2	2019-04-22	2019-04-28	BKLDEFH-BK-B-K	BKLDEFH	97.72-80.87-59.15-21.72
3	2019-04-29	2019-05-05	BKLDEFH-BK-BK-K	BKLDEFH	97.75-81.37-81.37-21.93
4	2019-05-06	2019-05-12	BKLDEFH-BK-B-K	BKLDEFH	97.71-81.5-59.69-21.81
Précision=7, TP=Avril+Mai & TL=1 semaine					

Gare de l'Est Pour le SOI de la *Gare de l'Est*, nous avons choisi de diviser le mois d'avril 2019 en semaines entières du lundi au dimanche et d'augmenter la précision à '7'. Des résultats obtenus, présentés dans le tableau 6.10 et les figures 6.10a à 6.9d, nous constatons que : *(i)* les brins sont tous identiques *(ii)* les DNAs sont pratiquement identiques sauf pour la séquence 2 des DNAs DNA_0, DNA_2 et DNA_4 *(iii)* les volumes d'activité par séquences sont du même ordre de grandeur sauf pour la deuxième séquence des DNAs DNA_0, DNA_2 et DNA_4

Nous pouvons constatons uniquement une faible modification des DNAs (la séquence 2 sur 3 des 5 DNAs) ce qui laisse penser que l'activité n'a pas subi de variation notable et donc qu'aucune anomalie ne peut être relevée sur ce SOI dans la période TP considérée.

TABLE 6.11
DNAs & Brins SOI *Cimetière de Montmartre*

n	Begin	End	DNA_n	$Strand_n$	Volumes
0	2019-04-01	2019-04-01	B-K-EFLHQIJ-D	BKEFLHQ	57.44-21.19-16.55-3.64
1	2019-04-02	2019-04-02	B-K-EFLHQIN-D	BKEFLHQ	56.76-21.24-16.8-4.07
2	2019-04-03	2019-04-03	B-K-EFLHQNJ-D	BKEFLHQ	57.75-19.73-17.34-4.02
3	2019-04-04	2019-04-04	B-K-EFLHQNI-D	BKEFLHQ	57.23-20.24-17.15-4.18
4	2019-04-05	2019-04-05	B-K-ELFHQNI-D	BKELFHQ	58.16-20.07-16.46-4.16
5	2019-04-06	2019-04-06	B-K-ELFHQJN-D	BKELFHQ	59.44-20.15-15.93-3.56
6	2019-04-07	2019-04-07	B-K-ELFHQIJ-D	BKELFHQ	59.23-21.37-15.2-3.4
7	2019-04-08	2019-04-08	B-K-EFD-LHQIJNP	BKEFDLH	57.84-20.49-12.99-8.21
8	2019-04-09	2019-04-09	B-K-EFLHQNJ-D	BKEFLHQ	57.54-19.94-16.75-4.55
9	2019-04-10	2019-04-10	B-K-EFLHQNI-D	BKEFLHQ	57.4-19.7-17.37-4.33
10	2019-04-11	2019-04-11	B-K-EFLHQNJ-D	BKEFLHQ	57.3-19.8-17.69-4.08
11	2019-04-12	2019-04-12	B-K-LEFHQNI-D	BKLEFHQ	56.83-20.26-17.58-4.31
12	2019-04-13	2019-04-13	B-K-LFHQJNI-ED	BKLFHQJ	59.29-20.37-11.04-8.9
13	2019-04-14	2019-04-14	B-K-ELFHQJI-D	BKELFHQ	58.26-21.71-15.41-3.79
Précision=7, TP=Avril & TL=1 jour					



(a) 6 avril (b) 8 avril

(c) 11 avril (d) 12 avril

(e) 13 avril (f) 14 avril

FIG. 6.11. Activité SOI *Cimetière de Montmartre*

Cimetière de Montmartre Concernant le SOI du *Cimetière de Montmartre*, il est intéressant de constater à partir du tableau 6.11 et des figures 6.11a à 6.11f que les DNAs, les brins et les volumes sont tous très semblables malgré une précision de 7 ; excepté pour les deux TLs du 8 et du 13 avril 2019 ce qui matérialise deux modifications de l'activité et, de ce fait, deux anomalies liées probablement à des cérémonies funéraires.

6.1.3 Bloc fonctionnel III

À présent que les DNAs et les brins ont été calculés, il faut pouvoir confirmer la ou les anomalies détectées. Pour éviter les faux positifs et lever sans ambiguïté des anomalies pertinentes, nous déterminons pour chaque DNA et chaque brin ce que nous appelons les *distances d'anormalité*. Ces distances nous permettent de quantifier les anomalies détectées et de n'envoyer une notification que pour certaines d'entre-elles en fonction de seuils prédéfinis.

6.1.3.1 Calcul des distances d'anormalité

Ces distances d'anormalité sont basées sur le principe de la distance de Damerau-Levenshtein notée DL($String_1, String_2$). Pour ce faire, nous utilisons la bibliothèque *JellyFish* qui, en plus de la distance de Damerau-Levenshtein, implémente d'autres mesures comme la distance de Jaro qui est principalement utilisée dans la détection de doublons ou la distance de Hamming plus employée dans le traitement du signal et les télécommunications [132].

Pour chaque DNA_n, sont calculées les distances d'anormalité spécifiques : $DNAD_n = DL(DNA_{n-1}, DNA_n)$ et $STAD_n = DL(Strand_{n-1}, Strand_n)$ où 'n' représente l'indice de la période TL concernée. De ces distances, sont déduites la distance totale d'anormalité : $ANOD_n = DNAD_n + STAD_n$.

Cette distance $ANOD_n$ matérialise, représente la *quantité d'anormalité* de la tranche de temps TL_n par rapport à la tranche de temps précédente TL_{n-1}. Cette mesure formalise donc le taux de variation de l'activité réseau par type d'applications des utilisateurs mobiles lors des transitions de TL_{n-1} vers TL_n pendant la période TP supervisée, et ce, pour le SOI considéré.

À partir des différentes distances d'anormalité $ANOD_n$, nous calculons la *Sévérité* de l'anomalie définie par l'équation (5.1) sur la base de laquelle peuvent être définis des niveaux ou seuils d'alerte. De ces seuils dépendrons la levée ou non d'une alerte puis l'envoi d'un message contenant les informations ayant justifié ce signalement.

Dans la section 6.1.3.1, les tableaux 6.12 à 6.21 détaillent l'ensemble des distances d'anormalité et les sévérités associées pour certains SOIs analysés et présentant une activité réseau qualifiée de forte ou faible en fonction du nombre total d'utilisateurs présent dans la zone couverte par le secteur d'intérêt.

SOIs à forte activité

Pour pouvoir calculer les DNA_0 et les $Strand_0$, il faut avoir calculé au préalable le DNA et le brin précédent ($DNA_{-1}, Strand-1$). Or, nous n'avons pas connaissance de la tranche TL avant la TL_0. Afin de résoudre ce problème "de l'œuf et de la poule", la distance d'anormalité DNA_0 est calculée entre le DNA de la tranche TL_O et le DNA de la dernière tranche TL de la période TP. Cette rotation de la liste des TLs est implémentée par la fonction Rotate détaillée par l'algorithme 9.

TABLE 6.12
Distances d'anormalité SOI *Stade de France*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-06	2019-04-07	BK-B-LD-EFHQIJ	BKLDEF	7	2	9	2.25
1	2019-04-13	2019-04-14	B-K-LD-EFHQJI	BKLDEF	3	0	3	0
2	2019-04-20	2019-04-21	B-K-LD-EFHJQI	BKLDEF	1	0	1	0
3	2019-04-27	2019-04-28	BK-B-LED-QEDHFI	BKLQED	7	2	9	2.25
4	2019-05-04	2019-05-05	BK-B-LDE-DEFHQJ	BKLDEF	5	2	7	1.36
5	2019-05-12	2019-05-13	BK-B-LQEDFH-LED	BKLQED	7	2	9	2.25
Précision=6, & TL=1 we								

TABLE 6.13
Distances d'anormalité SOI *Stade de France*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-06	2019-04-06	B-K-LD-EFHQ	BKLD	4	1	5	1.56
1	2019-04-13	2019-04-13	B-K-LD-EFHJ	BKLD	1	0	1	0
2	2019-04-20	2019-04-20	B-K-LD-EFHJ	BKLD	0	0	0	0
3	2019-04-27	2019-04-27	B-LE-K-QDHF	BKLQ	7	1	8	4
4	2019-05-04	2019-05-04	B-K-LDE-FHQJ	BKLD	8	1	9	5.06
5	2019-05-12	2019-05-12	B-K-LE-QDFH	BKLQ	5	1	6	2.25
Précision=6 & TL=1 jour								

A partir des tableaux 6.12 et 6.13, nous pouvons tirer les observations suivantes :

1. les tranches TL_0, TL_3, TL_4 & TL_5 font apparaître des sévérités moyennes de '2'
2. les tranches TL_0 à TL_2 & TL_4 ont des brins identiques 'BKLDEF'
3. les tranches TL_3 & TL_5 ont des brins identiques : 'BKLQED'
4. les tranches TL_0 à TL_2 ont des DNAs très semblables
5. le DNA_4 est un mixe entre les DNA_2 & DNA_3
6. les distances d'anomalies et les sévérités des TL_0 & TL_5 sont identiques ou proches
7. les tranches TL_3 & TL_5 ont des DNAs presque identiques. La différence est due au groupe applicatif 'Chat' (L) apparu dans la séquence Seq_3 du DNA_5

Des observations '2' et '4', nous pouvons conclure que les périodes TL_0, TL_1, TL_2 et TL_4 présentent une activité réseau identique. Des observations '1' à '3', nous pouvons déduire que les périodes TL_3 et TL_5 présentent une évolution majeure de l'activité des utilisateurs ce qui matérialise deux anomalies. Des observations '2' et '6', nous pouvons conclure que la transition entre les périodes TL_5 et TL_0 est un retour à l'état réputé "normal" ou de référence de l'activité réseau. Des observations '1' et '5', nous pouvons conclure que la période TL_4 présente une anomalie due à un retour à l'activité normalement

constatée pour les périodes TL_0, TL_1, TL_2. Enfin, d'après l'observation '7', l'activité réseau au cours de ces deux TLs a été identique. Ce qui signifie que les usagers mobiles ont utilisé quasiment les mêmes applications, et ce, dans les mêmes proportions au cours de ces périodes.

TABLE 6.14
Distances d'anormalité SOI *Notre-Dame de Paris*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-01	2019-04-07	BKLED-BK-BD-K	BKLED	0	0	0	0
1	2019-04-08	2019-04-14	BKLED-BK-BD-K	BKLED	0	0	0	0
2	2019-04-15	2019-04-21	BKLED-BK-K-LEDR	BKLED	5	0	5	1
3	2019-04-22	2019-04-28	BKLDE-BK-K-LDEF	BKLDE	3	1	4	0
4	2019-04-29	2019-05-05	BKLED-BK-K-LED	BKLED	3	1	4	0
5	2019-05-06	2019-05-12	BKLED-BK-BK-D	BKLED	3	0	3	0
6	2019-05-13	2019-05-20	BKLED-BK-BD-K	BKLED	2	0	2	0
Précision=5 & TL=1 semaine								

TABLE 6.15
Distances d'anormalité SOI *Notre-Dame de Paris*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-12	2019-04-12	BKLE-B-K-D	BKLE	0	0	0	0
1	2019-04-13	2019-04-13	BKLE-B-K-D	BKLE	0	0	0	0
2	2019-04-14	2019-04-14	BKLE-B-K-H	BKLE	1	0	1	0
3	2019-04-15	2019-04-15	BKLE-B-K-LEDR	BKLE	4	0	4	1
4	2019-04-16	2019-04-16	BKLE-BK-ED-H	BKLE	4	0	4	1
5	2019-04-17	2019-04-17	BKLE-B-K-ED	BKLE	3	0	3	0
6	2019-04-18	2019-04-18	BKLE-B-K-D	BKLE	1	0	1	0
Précision=4 & TL=1 jour								

Des tableaux 6.14 et 6.15, nous pouvons tirer les observations suivantes :

1. les tranches TL_0 à TL_6 ont des brins identiques sauf pour TL_3 avec une précision de '5' où les groupes 'E' et 'D' sont permutés
2. les DNAs DNA_0, DNA_1 & DNA_6 sont identiques
3. le DNA_2 marque un changement dans l'activité qui perdure avec les DNA_3 et DNA_4
4. les DNA_3 à DNA_6 marquent un retour progressif à l'activité de référence DNA_0 (tableau 6.15)
5. les sévérités sont de '1' pour TL_2 du tableau 6.14 et TL_2 & TL_3 du tableau 6.15
6. les distances d'anormalité sont nulles pour DNA_0

Des observations '1' à '3', nous pouvons conclure que les types d'application utilisée sont quasiment constants, mais que la troisième semaine d'avril 2019 marque une modification des proportions dans

lesquelles sont utilisés ces groupes applicatifs. L'observation '3' confirme l'apparition d'une anomalie réseau et l'observation '4' nous indique que cette anomalie perdure les deux semaines suivantes avec une tendance à un retour à la normale. L'observation '5' confirme que l'anomalie détectée en TL_2 (semaine du 15 au 21 avril 2019) est due aux variations de l'activité réseau constatées les 15 et 16 avril 2019. L'observation '6' confirme le retour à l'activité de référence à partir de TL_6.

TABLE 6.16
DISTANCES D'ANORMALITÉ SOI *Gare du Nord*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-01	2019-04-07	BKD-BK-BK-LDFEHQJ	BKDLFEH	2	1	3	0
1	2019-04-08	2019-04-14	BKD-BK-BK-LDEFHQJ	BKDLEFH	1	1	2	0
2	2019-04-15	2019-04-21	BKD-BK-BK-BLDEFHQ	BKDLEFH	2	0	2	0
3	2019-04-22	2019-04-28	BKD-BK-BK-LDEFHQJ	BKDLEFH	2	0	2	0
4	2019-04-29	2019-05-05	BKD-BK-BK-LDEFHQJ	BKDLEFH	0	0	0	0
5	2019-05-06	2019-05-12	BKD-BK-BK-LDEFHJQ	BKLDEFH	1	0	1	0
Précision=7 & TL=1 semaine								

TABLE 6.17
DISTANCES D'ANORMALITÉ SOI *Gare de Lyon*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-01	2019-04-07	BKLFDEH-BKD-BK-BK	BKLFDEH	1	1	2	0
1	2019-04-08	2019-04-14	BKLDFEH-BKD-BK-BK	BKLDFEH	1	1	2	0
2	2019-04-15	2019-04-21	BKLDFEH-BKD-BK-BK	BKLDFEH	0	0	0	0
3	2019-04-22	2019-04-28	BKLDFEH-BKD-BK-BK	BKLDFEH	0	0	0	0
4	2019-04-29	2019-05-05	BKLDEFH-BKD-BK-BK	BKLDEFH	1	1	2	0
5	2019-05-06	2019-05-12	BKLDFEH-BKD-BK-BK	BKLDFEH	1	1	2	0
Précision=7 & TL=1 semaine								

Des tableaux 6.16 et 6.17, nous pouvons tirer les observations suivantes :

1. bien que la Précision soit de '7', les distances d'anormalité restent très faibles
2. les DNAs et brins sont très similaires (distances faibles)
3. les sévérités sont nulles dans tous les cas
4. les DNAs de ces deux SOIs ont trois séquences en commun

Des observations '1' et '2', nous pouvons conclure que l'activité est stable, constante pour l'ensemble des TLs analysées. Ce que confirme l'observation '3' qui nous indique qu'aucune variation dans l'activité n'a été constatée et donc aucune anomalie détectée. L'observation '4' nous indique que la signature de ces deux SOIs est similaire ce qui semble cohérent de par leur aspect fonctionnel (gare) comme expliqué dans la section 5.2.3.

TABLE 6.18
Distances d'anormalité SOI *Gare de Lyon*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
10	2019-04-12	2019-04-12	BKLDF-BKD-B-K	BKLDF	2	2	4	0
11	2019-04-13	2019-04-13	BKLDE-BKLD-B-K	BKLDE	2	1	3	0
12	2019-04-14	2019-04-14	BKLDE-B-K-K	BKLDE	4	0	4	0
13	2019-04-15	2019-04-15	BKLFD-BKD-B-K	BKLFD	5	2	7	1.96
14	2019-04-16	2019-04-16	BKFLD-BKD-B-K	BKFLD	1	1	2	0
15	2019-04-17	2019-04-18	BKLFD-BKD-B-K	BKLFD	1	1	2	0
16	2019-04-19	2019-04-19	BKLDF-BKD-B-K	BKLDF	1	1	2	0
17	2019-04-20	2019-04-20	BKLDE-BKD-B-K	BKLDE	1	1	2	0
Précision=5 & TL=1 jour								

SOIs à faible activité

TABLE 6.19
Distances d'anormalité SOI *Cimetière du Père Lachaise*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
10	2019-04-12	2019-04-12	B-K-LDEFNQ-H	BKLDEF	4	0	4	0
11	2019-04-13	2019-04-13	B-K-LEFHQJ-D	BKLEDF	4	1	5	0
12	2019-04-14	2019-04-14	B-K-LEDR-FHQJIN	BKLEDF	5	0	5	0
13	2019-04-15	2019-04-15	B-K-LEDFQN-H	BKLEDF	6	0	6	1
14	2019-04-16	2019-04-16	B-K-LEFHNQ-D	BKLEDF	4	0	4	0
15	2019-04-17	2019-04-17	B-K-LEFHQJ-D	BKLEDF	2	0	2	0
16	2019-04-18	2019-04-18	B-K-LEFHNJ-D	BKLDEF	1	1	2	0
17	2019-04-19	2019-04-19	B-K-LEFHNQ-D	BKLEDF	1	1	2	0
Précision=6 & TL=1 jour								

Des tableaux 6.18 et 6.19, nous pouvons tirer les observations suivantes :

1. les sévérités sont nulles dans tous les cas sauf pour les deux TL_{13}
2. les DNAs de ces deux SOIs ont trois séquences en commun

Des observations précédentes, nous pouvons conclure qu'une anomalie est survenue le même jour sur ces deux SOIs géographiquement éloignés (sud-ouest pour le premier et nord-est pour le second).

TABLE 6.20
Distances d'anormalité SOI *Cimetière de Montmartre*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-01	2019-04-01	B-K-EFLHQI-D	BKEFLD	2	2	4	0
1	2019-04-02	2019-04-02	B-K-EFLHQI-D	BKEFLD	0	0	0	0
2	2019-04-03	2019-04-03	B-K-EFLHQN-D	BKEFDL	1	1	2	0
3	2019-04-04	2019-04-04	B-K-EFLHQN-D	BKEFDL	0	0	0	0
4	2019-04-05	2019-04-05	B-K-ELFHQN-D	BKELDF	1	2	3	0
5	2019-04-06	2019-04-06	B-K-ELFHQJ-D	BKELDF	1	0	1	0
6	2019-04-07	2019-04-07	B-K-ELFHQI-D	BKELDF	1	0	1	0
7	2019-04-08	2019-04-08	B-K-EFD-LHQIJN	BKEFDL	5	2	7	1.36
8	2019-04-09	2019-04-09	B-K-EFLHQN-D	BKEDFL	5	1	6	1
9	2019-04-10	2019-04-10	B-K-EFLHQN-D	BKEFDL	0	1	1	0
10	2019-04-11	2019-04-11	B-K-EFLHQN-D	BKEFLD	0	1	1	0
11	2019-04-12	2019-04-12	B-K-LEFHQN-D	BKLEDF	2	3	5	0
12	2019-04-13	2019-04-13	B-K-LFHQJN-ED	BKELDF	3	1	4	0
13	2019-04-14	2019-04-14	B-K-ELFHQJ-D	BKELDF	3	0	3	0
14	2019-04-15	2019-04-15	B-K-ELFHQN-D	BKELDF	1	0	1	0
15	2019-04-16	2019-04-16	B-K-EFLHQN-D	BKEFLD	1	2	3	0
16	2019-04-30	2019-04-30	B-K-ELFHQN-D	BKELFD	1	1	2	0
17	2019-05-01	2019-05-01	B-K-LFHQNI-ED	BKELDF	3	1	4	0
18	2019-05-02	2019-05-02	B-K-EFLHQN-D	BKEFLD	4	2	6	1
19	2019-05-03	2019-05-03	B-K-ELFHQN-D	BKELDF	1	2	3	0
20	2019-05-04	2019-05-04	B-K-ELFHQN-D	BKELDF	0	0	0	0
Précision=6 & TL=1 jour								

Du tableau 6.20, nous pouvons tirer les observations suivantes :

1. les tranches TL_0 à TL_{20} ont des brins quasiment identiques. Seul l'ordre des trois derniers groupes applicatifs (D, F & L) varie d'un brin à l'autre
2. les DNAs ont trois séquences communes sauf DNA_7 et DNA_{12}
3. seules les tranches TL_7, TL_8 & TL_{18} présentent une sévérité différente de '0'

Des observations '1' et '2', nous pouvons déduire que la répartition de l'activité par groupes applicatifs est constante. L'observation '3' nous indique une anomalie et de ce fait un changement dans la répartition de l'activité. De plus, l'anomalie levée en TL_8 est probablement liée au retour à l'activité normalement constatée sur ce SOI.

TABLE 6.21
DISTANCES D'ANORMALITÉ SOI *Cimetière de Montparnasse*

n	Begin	End	DNA	Strand	DNAD	STAD	ANOD	Severity
0	2019-04-02	2019-04-02	B-K-LFEHNJ-D	BKLFED	2	2	4	0
1	2019-04-03	2019-04-03	B-K-LFEHNJ-D	BKLFDE	0	1	1	0
2	2019-04-04	2019-04-04	B-K-LFEHNQ-D	BKLFED	1	1	2	0
3	2019-04-05	2019-04-05	B-K-LEFHNQ-D	BKLEFD	1	1	2	0
4	2019-04-06	2019-04-06	B-K-LEFHQN-D	BKLEDF	1	1	2	0
5	2019-04-07	2019-04-07	B-K-LEFHQJ-D	BKLEDF	1	0	1	0
6	2019-04-08	2019-04-08	B-K-LFEHNJ-D	BKLFED	2	2	4	0
7	2019-04-09	2019-04-09	B-K-LFEHNQ-D	BKLFDE	1	1	2	0
8	2019-04-10	2019-04-10	B-K-LFEHNJ-D	BKLDFE	1	1	2	0
9	2019-04-11	2019-04-11	B-K-LFEHNQ-D	BKLFDE	1	1	2	0
10	2019-04-12	2019-04-12	B-K-LEFHNQ-D	BKLDEF	1	2	3	0
Précision=6 & TL=1 jour								

Du tableau 6.21, nous pouvons tirer les observations suivantes :

1. les tranches TL_0 à TL_{10} ont des brins quasiment identiques. Seul l'ordre des trois derniers groupes applicatifs (D, E & F) varie d'un brin à l'autre
2. les DNAs sont quasiment identiques. Seul l'ordre des trois derniers groupes applicatifs de la séquence 3 varie (J, N & Q)
3. toutes les sévérités sont nulles

Des observations précédentes, nous pouvons conclure que l'activité est homogène pour l'ensemble des tranches TLs et aucune anomalie n'a été détectée. A partir des tableaux 6.20 et 6.21, nous pouvons constater que les DNAs de ces deux SOIs ont trois séquences communes et par conséquent que la signature de ces deux SOIs est similaire ce qui semble cohérent de par leur aspect fonctionnel (cimetière).

Synthèse

De l'analyse de ces quelques SOIs, il appert que DiNATrA$\mathcal{X}$ nous a permis d'extraire huit anomalies dont la liste est donnée par le tableau 6.22. D'après celui-ci, nous pouvons affirmer que nous avons été en mesure de détecter six anomalies liées à un changement de l'activité des utilisateurs sur le réseau mobile d'Orange. Les 7^{e} et 8^{e} anomalies sont, quant à elles, plus probablement liées à un incident réseau, car présentes sur deux SOIs différents, au même moment.

TABLE 6.22
ANOMALIES DÉTECTÉES

AnoId	SOI	Begin	End	ANOD	Severity
1	*Cimetière de Montmartre*	2019-04-08	2019-04-08	7	1.36
2	*Cimetière de Montmartre*	2019-05-02	2019-05-02	6	1
3	*Notre Dame de Paris*	2019-04-15	2019-04-15	4	1
4	*Notre Dame de Paris*	2019-04-16	2019-04-16	4	1
5	*Stade de France*	2019-04-27	2019-04-27	8	4
6	*Stade de France*	2019-05-12	2019-05-12	6	2.25
7	*Gare de Lyon*	2019-04-15	2019-04-15	7	1.96
8	*Cimetière du Père Lachaise*	2019-04-15	2019-04-15	6	1
Exemples basés sur les SOIs présentés					

6.1.3.2 Extractions des valeurs aberrantes

L'extraction des outliers à partir des données agrégées de chaque SOI est réalisée à l'aide du MLA non supervisé `Local Outlier Factor` ou `LOF`. Cet algorithme est couramment utilisé pour détecter des activités réseaux sortant de l'ordinaire comme expliqué et mis en œuvre par [133], [134] dans leurs travaux.

L'algorithme `LOF` a été présenté dans la section 4.2.3. Son implémentation `Python` disponible avec la bibliothèque `Scikit-learn` [135] dispose de plusieurs hyper-paramètres qu'il faut définir :

n_neighbors Le nombre de voisins à considérer lors des itérations

contamination Le degré de contamination de l'ensemble de données, c'est-à-dire la proportion de valeurs aberrantes dans l'ensemble de données

algorithm L'algorithme utilisé pour calculer les plus proches voisins à savoir `BallTree` ou `KDTree`

metric La métrique à employer pour le calcul de la distance qui peut être Euclidienne (EUC.), de MANHATTAN (MAN.) ou de MINKOWSKI (MIN.)

Afin que `LOF` prenne en compte l'ensemble des données contenues dans le jeu de données, nous fixons *n_neighbors* très grand.

S'agissant de la *contamination*, celle-ci est modulée en fonction des SOIs pour obtenir un nombre limité d'outliers afin de faire ressortir les plus significatifs.

Pour l'algorithme de calcul du plus proche voisin, nous avons conservé 'BallTree' qui est l'algorithme par défaut car, dans notre cas, nous obtenons les mêmes résultats indépendamment de l'algorithme utilisé.

Quant à la mesure de distance à employer, elle a été définie suite à différentes expérimentations et d'après l'étude de la distribution dans le but d'obtenir les résultats les plus cohérents possible par rapport à celle-ci.

Quelques exemples de valeurs aberrantes extraites par `LOF`, et ce avec différents paramètres, sont détaillées dans les tableaux C.1 à C.3. Pour confirmer que ces valeurs extraites des données de chaque SOI soient réellement en dehors des valeurs couramment observées, une étude de la distribution de ces données est effectuée.

Cette étude porte sur les données des TLs concernées et sur l'ensemble des données du SOI. L'ensemble des résultats obtenus sont disponibles en annexe C. Les figures C.1 à C.12 détaillent la distribution des différents paramètres réseaux issus des données agrégées pour quelques-uns des SOIs étudiés et à partir desquels nous avons pu extraire des anomalies. Les figures C.1, C.5 et C.8 représentent la distribution de l'ensemble des données agrégées des SOIs pour la période TP analysée. Les autres figures sont la représentation de la distribution par TLs.

Des tableaux C.1 à C.3, nous pouvons tirer les observations suivantes :

1. `LOF` a fait ressortir des valeurs aberrantes dont les caractéristiques étaient largement au-delà de la distribution gaussienne ou très peu représentées
2. `LOF` a également extrait des valeurs considérées comme aberrantes, car une ou plusieurs de leurs caractéristiques intrinsèques étaient anormalement faibles (voir tableau C.3)
3. utiliser la distance euclidienne comme métrique permet de confirmer plus de valeurs aberrantes par rapport aux autres distances disponibles avec `LOF`
4. l'extraction seule des valeurs aberrantes n'est pas suffisante pour détecter des anomalies dans l'activité réseau, car des outliers peuvent exister dans plusieurs tranches TL (voir tableau C.2)

Les observations '1' et '2' confirment la pertinence et l'efficacité de l'algorithme `LOF` pour isoler les outliers. L'observation '3' nous permet de dire que la distance euclidienne est la plus adaptée à notre méthodologie. L'observation '4' confirme que DiNATrA$\mathcal{X}$ permet de détecter des anomalies dues à des évolutions de la nature du trafic ou des variations de l'activité réseau qu'elles soient positives (augmentation anormale de l'activité) ou négatives (diminution dans des proportions anormales).

6.1.3.3 Corrélation des anomalies

Une fois les anomalies détectées confirmées par l'étude de la distribution, il est parfois possible de relier ces variations à des événements particuliers (mouvements de foules, incidents, ...) permettant de les expliquer. Or, `LOF` analyse puis met en exergue des valeurs aberrantes à partir des données agrégées de chaque TL, dont un échantillon est donné dans le tableau 6.7. Problème, ces données ne contiennent plus de champs 'TimeSlot' comme cela a été expliqué dans la section 6.1.2.1. Pour pouvoir retrouver cette information, nous utilisons l'index des dataframes `Pandas` pour réaliser une jointure avec les données agrégées avant la phase de 'slicing', c'est-à-dire de découpage en tranches de temps TLs.

6.1.4 Conclusion

Le tableau 6.23 synthétise les anomalies liées aux SOIs présentés précédemment et les probables événements qui en sont à l'origine. Parallèlement, l'ensemble des informations techniques et fonctionnelles relatives à ces anomalies peuvent être collationnées et transmises au format JSON à un outil de supervision de type SIEM [136].

TABLE 6.23
Anomalies & Événements corrélés

Index	**TimeSlot**	**SOI**	**ClusterId**	**Evénement**	**Ref.**
811179	2019-04-08	*Cimetière de Montmartre*	962	Funérailles ?	
894936	2019-04-15	*Notre-Dame de Paris*	1037, 1060	Incendie	[108]
894940	2019-04-16	*Notre-Dame de Paris*	1037, 1060	Incendie	[108]
988684, 987836	2019-04-27	*Stade de France*	1168	Final coupe de France	[137]
811185	2019-05-02	*Cimetière de Montmartre*	XXX	Obsèques de Dick Rivers	[138]
988078	2019-05-12	*Stade de France*	1168	Concert Metallica	[137]
??????	2019-04-15	Territoire national	??????	Panne réseau 4G Orange	[139]

6.2 Jeu de données CTU-13

6.2.1 Spécificités du CTU-13

Comme détaillé dans la section 5.1.4, notre méthodologie de détection présente cinq caractéristiques fondamentales. DiNATrA$\mathcal{X}$ est *(i)* générique *(ii)* cyclique *(iii)* fractale *(iv)* récurrente *(v)* et temps réel. Ces particularités font qu'il est possible de l'employer avec divers jeux de données.

Dans cette section 6.2, nous allons mettre en œuvre DiNATrA$\mathcal{X}$ afin d'analyser le trafic réseau généré lors du déroulé de différents scénarios issus du jeu de données CTU-13 (voir section 2.6). Le but ici est de vérifier si notre approche permet de détecter les étapes clefs caractérisant l'activité de différents botnets. L'ensemble des scénarios constituant le CTU-13 ont été décrits dans la section 2.6. Nous avons choisi de tester DiNATrA$\mathcal{X}$ avec les données issues des scénarios 9 à 11 (captures 50, 51 et 52) car ils sont représentatifs de l'activité réseau générée par des bots de natures différentes.

Afin de pouvoir analyser le trafic réseau caractérisant ces scénarios, nous avons tout d'abord dû définir une nouvelle méthode d'agrégation des flux réseaux contenus dans les différents scénarios du CTU-13. Puis, il nous a fallu adapter le code des différentes fonctions de DiNATrA$\mathcal{X}$ afin de les rendre agnostiques aux données analysées notamment en supprimant les références aux noms des champs qui étaient spécifiques au jeu de données CANCAN. Enfin, nous avons également ajouté une fonction nous permettant de découper les données en 'TimeSlice' (périodes TS).

L'ensemble de nos résultats pour les trois scénarios étudiés sont détaillés dans les sections 6.2.2 à 6.2.4. Pour chacun d'eux, nous présentons tout d'abord son déroulé tel que spécifié par leurs auteurs. Ceci afin de pouvoir faire un parallèle entre les résultats obtenus avec différents paramètres de configuration de DiNATrA$\mathcal{X}$ et les étapes clefs de l'exécution des attaques par les botnets.

6.2.1.1 Analyse des données

Les différents scénarios du CTU-13 ont consisté à faire générer du trafic réseau par des nœuds sains ("background traffic") et différents types de bots ("botnet traffic"). Ce trafic a été capturé à l'aide de l'utilitaire `Argus` puis labellisé. Pour chacun des scénarios est fourni, entre autres, un fichier au format 'binetflow' qui est le produit de ces captures. Un échantillon de ces données brutes est présenté dans le tableau 6.24.

TABLE 6.24
ÉCHANTILLON DE DONNÉES BRUTES SCÉNARIO 9

StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport
2011/08/17 12 :28 :41	0.750057	udp	113.161.65.193	61080	<->	147.32.84.229	13363
2011/08/17 12 :28 :41	3477.050049	udp	78.141.177.65	1574	<->	147.32.84.229	13363
2011/08/17 12 :28 :41	0.000304	udp	147.32.84.165	1025	<->	147.32.80.9	53
2011/08/17 12 :28 :41	3131.707520	udp	62.221.99.152	27123	<->	147.32.86.165	12114
2011/08/17 12 :28 :41	1792.383667	udp	217.172.89.244	21205	<->	147.32.84.229	13363
2011/08/17 12 :28 :41	0.000649	udp	125.202.60.177	33695	<->	147.32.86.165	443
2011/08/17 12 :28 :43	16.033258	tcp	78.102.233.175	54930	->	147.32.85.26	54145

State	sTos	dTos	TotPkts	TotBytes	SrcBytes	Label	
CON	0	0	6	1 314	209	flow=Background-UDP-Established	
CON	0	0	24	6 709	899	flow=Background-UDP-Established	
CON	0	0	2	203	64	flow=From-botnet-V50-1-UDP-DNS	
CON	0	0	6	405	219	flow=Background-UDP-Established	
CON	0	0	4	553	397	flow=Background-UDP-Established	
CON	0	0	2	128	60	flow=Background-UDP-Established	
FSPA_FSPA	0	0	20	2 854	1 503	flow=Background-TCP-Established	

Ces données brutes décrivent les flux réseaux échangés entre les différents nœuds pendant le déroulé de chacun des scénarios. Ces flux sont caractérisés par quinze champs détaillés ci-dessous :

StartTime Date et heure de la capture

Dur Durée de la session

Proto Type de protocole

SrcAddr Adresse IP source

Sport Port source

Dir Direction du flux

DstAddr Adresse IP destination

Dport Port destination

State Drapeaux TCP/UDP (SYN, ACK, RST, FIN, PSH, CON)

sTos Type de service source

dTos Type de service destination

TotPkts Nombre total de paquets échangés

TotBytes Nombre total d'octets échangés

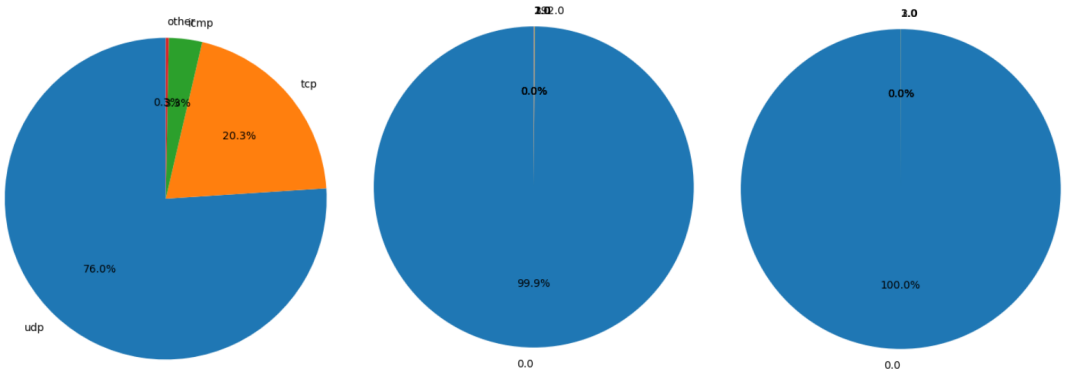
SrcBytes Nombre d'octets envoyés

Label Type de flux (background / botnet)

Afin de déterminer la pertinence de ceux-ci, une simple étude statistique a été réalisée sur l'ensemble des données des trois scénarios choisis. Le total cumulé de ces données représente un volume d'environ 480 Mo de flux réseaux à analyser. Les résultats de cette étude sont présentés par le tableau 6.25 et la figure 6.12. Nous pouvons en déduire que la répartition par protocole (figure 6.12a) est de 76% pour UDP, 20.3% pour TCP, 3.3% pour ICMP et 0.3% pour ce qui concerne les autres protocoles. Pour ce qui est des valeurs de 'sdTos' et 'sTos', elles sont à 99.9% égales à '0' (figures 6.12b et 6.12c) et donc sans intérêt pour nous.

TABLE 6.25
ÉTUDE STATISTIQUE DES SCÉNARIOS 9 À 11

	Dur	sTos	dTos	TotPkts	TotBytes	SrcBytes
count	6 458 780	6 402 979	5 927 405	6 458 780	6 458 780	6 458 780
mean	286.7301	0.05674983	0.0004195765	52.41833	40 815,79	8 559,823
std	826.6917	3.253154	0.03329776	8 203,014	4 887 700	1 746 985
min	0	0	0	1	60	0
25%	0.000285	0	0	2	214	78
50%	0.000708	0	0	2	271	83
75%	1.791903	0	0	5	723	417
max	3600.08	192	3	16 580 640	3 460 142 000	2 692 621 000



(a) Répartition par protocoles (b) Répartition par sTos (c) Répartition par dTos

FIG. 6.12. Étude statistique des scénarios 9 à 11

6.2.1.2 Agrégation des flux de trafic

Dans le cadre de notre analyse avec DiNATrA$\mathcal{X}$, les labels n'ont pas été utilisés pour conserver une approche *non supervisée*. Nous avons veillé à la conformité des données en remplaçant les valeurs absentes par '0' et en supprimant les éventuels flux dont les ports sources ou destinations ne sont pas compris dans la plage 0 à 65 535. De plus, les colonnes 'Dir', 'sTos' et 'dTos' ont été éliminées, car non significatives, et seuls les flux 'tcp' et 'udp' ont été conservés. En effet, le protocole 'ICMP' est très peu réprésenté et surtout ne se base pas sur la notion de port. Pourquoi cette notion de port est-elle importante ?

Afin de pouvoir générer les signatures numériques puis calculer les DNAs et les brins, il nous faut choisir une caractéristique $\mathcal{F}$. Or, les champs restants susceptibles d'être employés pour $\mathcal{F}$ sont : le type de protocole ('Proto'), les ports sources ou destinations ('Sport' et 'Dport') ou l'état des connexions ('State'). S'agissant de 'Proto', ce champ ne peut correspondre qu'à 2 valeurs ('tcp' ou 'udp'). Pour ce

qui est de l'attribut 'State', une fois converti en catégorie, il peut pendre 389 valeurs différentes. Or, ces valeurs n'ont aucun rapport entre elles et ne peuvent pas être corrélées ou factorisées. La solution retenue a donc été d'employer la notion de port. Sachant que les ports sources sont déterminés de façon presque aléatoire par le système d'exploitation, nous avons choisi d'utiliser la notion de *port destination*. Mais, les numéros de port sont compris entre 0 et 65 535. Afin de réduire le nombre de valeurs possibles pour $\mathcal{F}$, l'idée a été de regrouper les ports par plage ou bande, mais de quelle taille ou *largeur* ? La plage de 0 à 1023 est la bande de ports réservés. Nous sommes, de ce fait, partis sur des bandes de largeur multiple de 1 024, car 65 536 divisé par 1 024 nous laisse 64 valeurs pour $\mathcal{F}$ qui est un nombre codifiable à l'aide de symboles alphanumériques. Une largeur de bande de 2 048 nous donnerait 32 valeurs possibles ce qui est une valeur médiane correcte. Une largeur de 4 096 nous laisse seulement 16 valeurs pour caractériser $\mathcal{F}$ ce qui pourrait ne pas être suffisant.

Tout comme pour l'étude du jeu de données CANCAN, la première étape consiste à agréger les flux bruts afin de réduire la quantité totale de données à analyser et en faire ressortir des tendances grâce à l'algorithme non supervisé `K-Means`. Le t-uplet utilisé comme clef d'agrégation peut être constitué des adresses IP source, destination ; des ports sources, destination ; du protocole et de l'état de la connexion. À l'instar des ports sources et destination, nous avons cherché à fusionner les adresses IP source et destination. Pour ce faire, pour chacune des adresses IP, nous avons calculé l'*identifiant de sous-réseau* correspondant pour des valeurs de masque allant de /8 à /24 en notation CIDR (Classless Inter-Domain Routing [140]), et ce, par pas de 4 bits.

Pour résumer, la méthodologie DiNATrA$\mathcal{X}$ définit plusieurs notions ou concepts qu'il est possible de mettre en parallèle en fonction du jeu de données analysé. Le tableau 6.26 synthétise ces notions et leur correspondance si les données sont issues du jeu CANCAN ou CTU-13.

TABLE 6.26
CORRESPONDANCE DES NOTIONS DiNATrA$\mathcal{X}$

Dataset	TP	TL	TS	Secteur	SOI	$\mathcal{F}$	Activité
CANCAN	3 mois	1 jour / WE / semaine	30 mn	BTS rayon 200 m	Secteurs particuliers	AppGroup	TotBytes
CTU-13	1 jour	1 mn	0.1 s	Sous-réseau (CIDR)	Ensemble sous-réseaux	Bande Dports	SrcBytes

Pour terminer, la figure 6.13 est une représentation schématique de notre approche d'agrégation des flux bruts issus des scénarios du CTU-13. Dans les analyses qui suivent, différents t-uplets ont été utilisés et testés afin de déterminer la solution la plus efficiente.

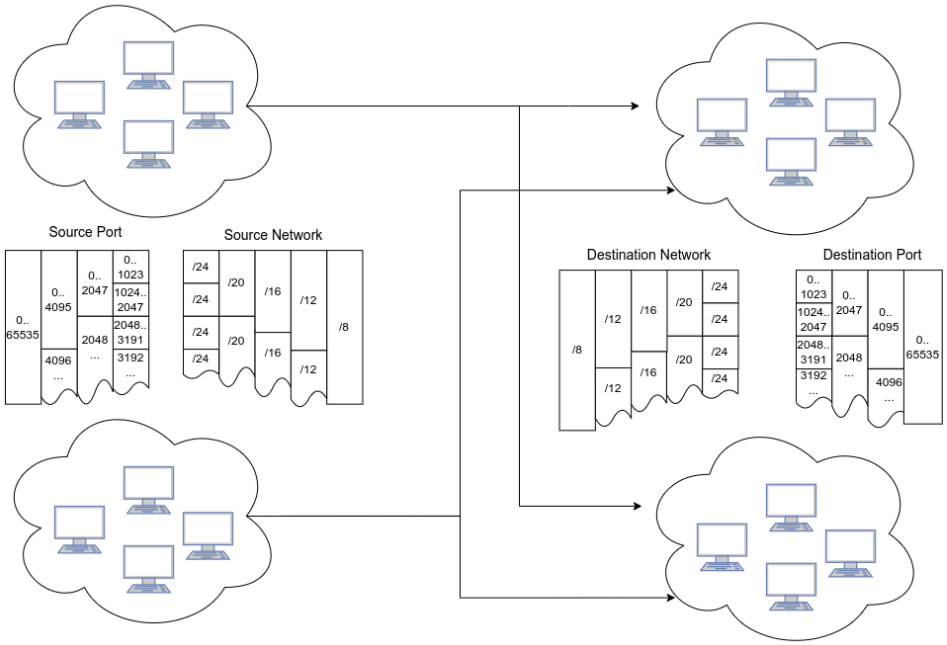


FIG. 6.13. Principe d'agrégation des flux réseaux du CTU-13

6.2.1.3 Adaptations pour l'étude des scénarios

L'ensemble du code (NoteBook Jupyter) et des scénarios composants le CTU-13 sont disponibles et librement téléchargeables depuis notre dépôt Kaggle [141]. Tous les résultats obtenus et présentés ci-dessous peuvent être vérifiés en exécutant notre NoteBook 'DiNATrA$\mathcal{X}$_CTU13'. Pour ce qui est des paramètres de configuration par défaut, nous avons choisi une 'Precision' égale à '5' et des signatures constituées de '4' clusters.

Afin d'évaluer l'importance de la sévérité des anomalies les unes par rapport aux autres ; nous avons introduit 'MeanGap', une mesure supplémentaire qui permet de mesurer l'écart des sévérités par rapport à la sévérité moyenne dont la méthode de calcul est donnée par l'équation (6.1). Pour les besoins de l'étude des différents scénarios CTU-13, les fonctions `split_data` et `extract_date` ont été généralisées. La fonction `appGroupToLetter` a été modifiée en `featureToLetter` afin de pouvoir gérer plus de catégories. Nous avons également ajouté la fonction `computeTS` pour générer les 'TimeSlice' utilisées dans le calcul des signatures numériques comme détaillé par les algorithmes 11 et 12.

$$MeanGap_n = Severity_n - \overline{Severity} \tag{6.1}$$

6.2.2 Étude du scénario 9 (CTU-Malware-Capture-botnet-50)

Les données brutes relatives à ce scénario sont librement téléchargeables depuis le site d'hébergement du 'Stratosphere Research Laboratory' [1].

6.2.2.1 Caractéristiques & Déroulé

Il consiste en la capture de l'activité d'un botnet constitué de dix bots Windows XP infectés par le malware `Neris`. Les dix machines sont démarrées puis infectées tour à tour. La chronologie des événements [2] est détaillée dans le tableau 6.27. Tous ces bots, nommés 'SARUMAN' à 'SARUMAN9', ont pour adresses IP respectives '147.32.84.165', '147.32.84.191 à .193' et '147.32.84.204 à .209'. En fonction du masque utilisé, ces adresses IP sont contenues dans les sous-réseaux suivants : 147.32.84.0/24, 147.32.80.0/20, 147.32.0.0/16, 147.32.0.0/12 ou 147.0.0.0/8.

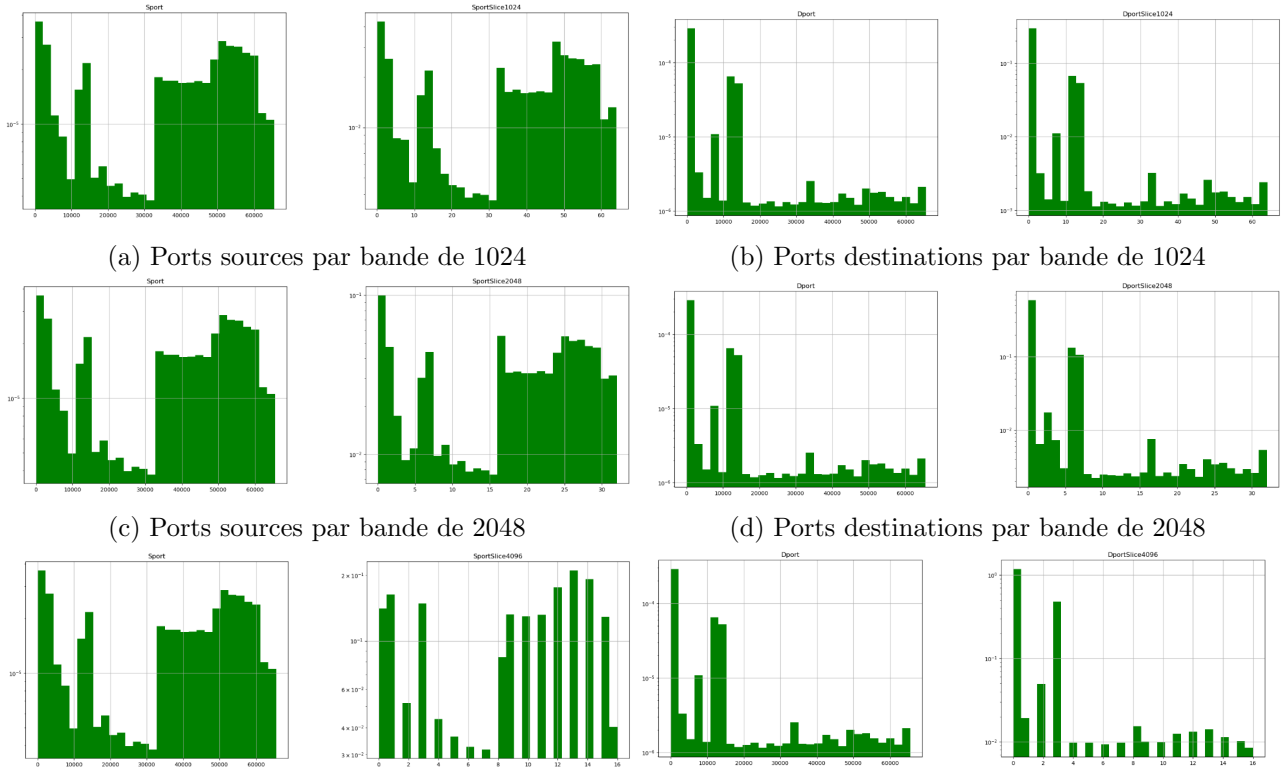
TABLE 6.27
Déroulé du scénario 9

TimeSlot	Événement	Bot	EventId
12 :01	Début capture		
12 :27	Début démarrage VMs	SARUMAN à SARUMAN8	
12 :42	Fin démarrage VMs	SARUMAN9	
13 :01	Début attaque TCP externe	82.162.140.147 → 147.32.84.192 :3389	1
13 :10	Fin attaque TCP externe	82.162.140.147 → 147.32.84.192 :3389	1
14 :24	Infection	SARUMAN	2
14 :43	Infection	SARUMAN1	3
14 :45	Infection	SARUMAN1	3
14 :48	Infection	SARUMAN2	4
14 :54	Infection	SARUMAN3	5
14 :56	Infection	SARUMAN4	6
14 :57	Infection	SARUMAN5	7
15 :00	Infection	SARUMAN6	8
15 :01	Infection	SARUMAN7	9
15 :03	Infection	SARUMAN8	10
15 :04	Infection	SARUMAN9	11
17 :06	Début arrêt VMs	SARUMAN à SARUMAN8	
17 :10	Fin arrêt VMs	SARUMAN9	
17 :12	Fin capture		
17/08/2011 – 5 heures de capture (279 Mo)			

1. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-50/detailed-bidirectional-flow-labels/capture20110817.binetflow`
2. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-50/`

6.2.2.2 Analyse & Résultats

Échantillonnage des ports La caractéristique $\mathcal{F}$ matérialise la bande à laquelle appartient le port destination du flux réseau. Pour déterminer la largeur de cette bande (1024, 2048 ou 4096), nous avons représenté la distribution des ports en fonction de chacune de ces bandes. Ces distributions sont présentées par la figure 6.14.



(a) Ports sources par bande de 1024 (b) Ports destinations par bande de 1024

(c) Ports sources par bande de 2048 (d) Ports destinations par bande de 2048

(e) Ports sources par bande de 4096 (f) Ports destinations par bande de 4096

FIG. 6.14. Distribution des ports sources & destinations

D'après les figures 6.14a et 6.14b, nous constatons que la courbe de distribution des ports sources et destinations par bandes de largeur '1 024' sont équivalentes. Ce qui signifie que cet "échantillonnage" respecte la distribution d'origine. Le problème avec ce découpage est que $\mathcal{F}$ nécessite '64' valeurs différentes pour pouvoir être codifiée et donc autant de nucléotides pour nos DNAs.

La figure 6.14c montre que l'échantillonnage par bandes de largeur '2 048' modifie la distribution pour les ports sources contenus dans la bande numéro '16'. Par contre, d'après la figure 6.14d, les courbes sont équivalentes pour les ports destinations.

Les figures 6.14e et 6.14f confirment qu'un échantillonnage en bandes de largeur '4 096' déforme trop les distributions.

Par conséquent, nous avons choisi un échantillonnage des ports sources et destinations par bandes de largeur '2 048'.

Échantillonnage des adresses IP Une rapide analyse des données contenues dans le scénario 9 nous indique que les flux, issus de 309 043 adresses IP et de 64600 ports sources, sont à destination de 90 198 adresses IP pour 41 653 ports. Les données contenues dans le champ 'State' correspondent à 237 catégories.

Pour déterminer le masque de sous-réseau à utiliser, nous avons calculé les adresses des sous-réseaux pour chacune des adresses IP sources et destinations. Le nombre de sous-réseaux correspondant pour chaque masque est détaillé dans le tableau 6.28.

TABLE 6.28
NOMBRE DE SOUS-RÉSEAUX EN FONCTION DU MASQUE

	/8	**/12**	**/16**	**/20**	**/24**
SrcNetwork	185	2 230	20 756	120 228	253 881
DstNetwork	190	2 120	15 309	51 357	76 350
Scénario 9					

Pour permettre l'utilisation de l'algorithme `K-Means`, le type de protocole est encodé sous forme de catégories numériques (tcp=1 & udp=2). L'ensemble des sous-réseaux et des bandes de ports sources et destinations sont ajoutés aux données brutes pour former les flux enrichis avant leur agrégation. Un échantillon de ces flux enrichis est présenté par le tableau 6.29.

TABLE 6.29
Échantillon de flux enrichis avant agrégation

TimeSlot	Dur	Proto	SrcAddr	Sport	DstAddr	Dport
11 :34 :49	1823.088379	2	46.196.43.16	6881	147.32.84.118	6881
11 :40 :19	1973.646729	2	151.28.221.122	6881	147.32.84.118	6881
12 :01 :01	2059.387451	1	119.252.172.92	59067	147.32.84.14	80
State	**TotPkts**	**TotBytes**	**SrcBytes**	**Label**		
7	2	214	107	flow=Background		
7	2	214	107	flow=Background		
126	224275	266462578	3199174	flow=Background		
SportSlice1024	**DportSlice1024**	**SportSlice2048**	**DportSlice2048**	**SportSlice4096**	**DportSlice4096**	
7	7	3	3	2	2	
7	7	3	3	2	2	
58	0	29	0	14	0	
SrcNetwork8	**DstNetwork8**	**SrcNetwork12**	**DstNetwork12**			
46.0.0.0/8	147.0.0.0/8	46.192.0.0/12	147.32.0.0/12			
151.0.0.0/8	147.0.0.0/8	151.16.0.0/12	147.32.0.0/12			
119.0.0.0/8	147.0.0.0/8	119.240.0.0/12	147.32.0.0/12			
SrcNetwork16	**DstNetwork16**	**SrcNetwork20**	**DstNetwork20**	**SrcNetwork24**	**DstNetwork24**	
46.196.0.0/16	147.32.0.0/16	46.196.32.0/20	147.32.80.0/20	46.196.43.0/24	147.32.84.0/24	
151.28.0.0/16	147.32.0.0/16	151.28.208.0/20	147.32.80.0/20	151.28.221.0/24	147.32.84.0/24	
119.252.0.0/16	147.32.0.0/16	119.252.160.0/20	147.32.80.0/20	119.252.172.0/24	147.32.84.0/24	
Proto : tcp=1 & udp=2						

Agrégation des flux & Analyses Une fois l'ensemble des bandes de ports et les adresses de sous-réseaux calculés, les flux enrichis sont agrégés par période TL de 1 minute en utilisant une clef primaire, un t-uplet, basé sur les adresses de sous-réseaux, les bandes de ports, le type de protocole et l'état des drapeaux HTTP. Les données numériques caractérisant chacun des flux enrichis sont sommées. Ces données agrégées seront ensuite analysées par l'algorithme `K-Means`. Pour cette phase d'agrégation, nous avons testé 2 types de t-uplet construits comme suit : '**TimeSlot ;SrcNetworkX ;(SportSlice2048 ;)DstNetworkX ;DportSlice2048 ;Proto ;State**' où le t-uplet de type 'A' n'utilise pas la bande de ports sources et où X peut prendre les valeurs : 8, 12, 16, 20 ou 24. Le tableau 6.30 synthétise le nombre de flux agrégés obtenus en fonction du type de t-uplet et du masque (/X) de sous-réseau utilisé.

TABLE 6.30
Nombre total de flux agrégés en fonction t-uplet & masque

	/8	/12	/16	/20	/24
Type 'A'	793 980	882 445	896 446	905 107	931 086
Type 'B'	368 252	661 055	808 784	843 795	872 002

Pour l'analyse du scénario 9, nous avons défini une période TP comprise entre le 17/08/2011 12 :00

et le 17/08/2011 17 :00. La fonction `computeTS` (algorithme 12) nous a permis de découper TP en TLs de 1 minute. Les tableaux 6.31 à 6.33 détaillent les résultats de l'analyse des signatures numériques menée avec DiNATrA$\mathcal{X}$ et ce pour différents paramètres ou type d'agrégations.

TABLE 6.31
DNA & Résultats (agrégation de type B & masque en /24)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
12 :54	12 :55	AHBGCE-A-A-A	AHBGCE	98.23-92.43-92.43-92.43	3	3	6	1	-1.38	
12 :55	12 :56	AHBGCE-A-A-A	AHBGCE	98.92-93.22-93.22-93.22	0	0	0	0	-2.38	
12 :56	12 :57	AHBGCD-A-A-A	AHBGCD	98.59-89.05-89.05-89.05	1	1	2	0	-2.38	
12 :57	12 :58	AHBGEC-A-A-A	AHBGEC	96.88-84.71-84.71-84.71	2	2	4	0	-2.38	
12 :58	12 :59	AWBGHE-A-A-A	AWBGHE	96.9-86.35-86.35-86.35	3	3	6	1	-1.38	
12 :59	13 :00	ABWHGD-AB-A-A	ABWHGD	96.63-86.66-81.73-81.73	4	3	7	1.36	-1.02	
13 :00	13 :01	BaRACI-BaRCI-B-A	BaRACI	94.49-87.18-57.82-7.31	12	6	**18**	9	6.62	1
13 :01	13 :02	BAaRCI-BaRCI-B-A	BAaRCI	93.22-84.45-55.94-8.77	2	2	4	0	-2.38	1
13 :02	13 :03	ABHGWY-AB-A-A	ABHGWY	88.77-66.56-48.99-48.99	11	5	**16**	7.11	4.73	1
13 :03	13 :04	ABHGWC-AB-A-BE	ABHGWC	95.04-81.54-66.49-15.2	3	1	4	0	-2.38	1
13 :04	13 :05	AHBGCf-Af-A-BE	AHBGCf	96.47-85.25-84.86-3.57	4	3	**7**	1.36	-1.02	1
13 :05	13 :06	ABHGEC-AB-AB-A	ABHGEC	94.6-82.62-82.62-52.6	6	3	**9**	2.25	-0.13	1
13 :06	13 :07	ABHGEC-AB-A-A	ABHGEC	93.98-80.67-48.3-48.3	1	0	1	0	-2.38	1
13 :07	13 :08	AHGBCS-A-A-A	AHGBCS	92.6-75.1-75.1-75.1	4	3	**7**	1.36	-1.02	1
13 :08	13 :09	ABHGXC-A-A-B	ABHGXC	95.66-45.31-45.31-39.93	4	3	**7**	1.36	-1.02	1
13 :09	13 :10	BAHGXC-B-AC-A	BAHGXC	95.78-47.46-38.53-36.89	4	1	5	0	-2.38	
14 :21	14 :22	AaEYHZ-AaEYZb-A-a	AaEYHZ	94.5-94.39-63.5-21.65	7	1	8	1.78	-0.6	
14 :22	14 :23	AEYHBa-AYaZb-AYa-EYaZb	AEYHBa	83.05-60.06-55.4-35.05	11	3	**14**	5.44	3.06	2
14 :23	14 :24	YAbEaB-YAbaZ-Y-A	YAbEaB	96.84-94.05-77.52-8.1	15	5	**20**	11.11	8.73	2
14 :24	14 :25	YAbEDG-YAbaZ-Y-A	YAbEDG	97.25-92.26-77.09-8.05	2	2	4	0	-2.38	2
14 :25	14 :26	YAEDGH-YA-Y-AZb	YAEDGH	91.1-61.59-34.68-27.84	7	2	**9**	2.25	-0.13	2
14 :26	14 :27	AEDBHG-Y-b-Z	AEDBHG	86.39-1.93-1.37-1.03	7	3	**10**	2.78	0.4	2
14 :27	14 :28	AEDHBG-A-Y-b	AEDHBG	87.92-35.54-1.76-1.25	4	1	5	0	-2.38	
14 :42	14 :43	AEDHGX-AD-A-A	AEDHGX	94.7-82.63-79.28-79.28	2	1	3	0	-2.38	
14 :43	14 :44	AENHDG-AENB-AD-A	AENHDG	92.9-86.74-75.47-73.08	7	3	**10**	2.78	0.4	3
14 :44	14 :45	AEYNHD-AEYN-A-A	AEYNHD	86.79-79.17-48.28-48.28	5	2	7	1.36	-1.02	3
14 :45	14 :46	AEYDBG-AEY-A-A	AEYDBG	89.81-78.37-50.92-50.92	4	3	7	1.36	-1.02	3
14 :46	14 :47	aAEDHB-a-A-A	aAEDHB	98.66-81.73-13.76-13.76	7	4	**11**	3.36	0.98	3
14 :47	14 :48	aAEDHG-a-A-A	aAEDHG	98.56-81.48-13.79-13.79	1	1	2	0	-2.38	
14 :48	14 :49	AEBDHG-AB-A-A	AEBDHG	93.82-76.26-74.0-74.0	4	2	**6**	1	-1.38	4
14 :49	14 :50	AEBDHG-AB-A-A	AEBDHG	94.54-81.6-79.11-79.11	0	0	0	0	-2.38	
14 :50	14 :51	AEDBHG-A-A-A	AEDBHG	89.89-59.22-59.22-59.22	2	1	3	0	-2.38	
14 :51	14 :52	AEDHGB-AW-A-A	AEDHGB	95.67-84.14-83.83-83.83	3	2	5	0	-2.38	
14 :52	14 :53	AEDGHX-AW-A-A	AEDGHX	95.64-83.34-83.06-83.06	2	2	4	0	-2.38	
14 :53	14 :54	AEGHDB-A-A-A	AEGHDB	92.12-56.66-56.66-56.66	4	3	**7**	1.36	-1.02	5
14 :54	14 :55	AEGHBD-ABZC-A-A	AEGHBD	92.41-67.47-63.43-63.43	4	1	5	0	-2.38	
14 :55	14 :56	AEHGDB-ADBCYZ-AY-E	AEHGDB	91.17-60.18-53.6-25.19	7	2	**9**	2.25	-0.13	6
14 :56	14 :57	WAaEHD-Wa-A-E	WAaEHD	94.05-53.77-24.49-11.61	11	4	**15**	6.25	3.87	7
14 :57	14 :58	aWAEDH-a-A-A	aWAEDH	95.46-45.0-21.15-21.15	5	3	8	1.78	-0.6	
14 :58	14 :59	aAYEBG-a-A-A	aAYEBG	94.22-37.72-29.67-29.67	4	4	8	1.78	-0.6	
14 :59	15 :00	AYEGHa-AYa-A-A	AYEGHa	92.23-82.75-48.71-48.71	6	4	**10**	2.78	0.4	8
15 :00	15 :01	ABaRCE-AaREYb-Aa-B	ABaRCE	86.64-57.32-46.26-27.62	11	5	**16**	7.11	4.73	9
15 :01	15 :02	ABaRHC-AB-A-B	ABaRHC	90.18-78.19-63.12-15.07	8	2	**10**	2.78	0.4	9
15 :02	15 :03	aAYHEB-a-AY-A	aAYHEB	99.7-94.89-4.61-3.7	9	5	**14**	5.44	3.06	10
15 :03	15 :04	aAYbHG-a-AY-AbB	aAYbHG	99.64-95.59-3.42-3.01	5	3	8	1.78	-0.6	
15 :04	15 :05	AbBHEG-AbBEa-Ab-AB	AbBHEG	92.95-85.83-76.64-63.81	10	4	**14**	5.44	3.06	11
15 :05	15 :06	AbZBHE-AbBaY-AZ-b	AbZBHE	93.11-74.21-59.57-21.49	7	2	9	2.25	-0.13	
Capture : 2011-08-17 – $\mathcal{F}$ = 'DportSlice2048' – Activity = 'SrcBytes' – Precision = 6										
$ANOD_{Max} = 20$ – $\overline{ANOD} = 8.33$ – $Severity_{Max} = 11.11$ – $\overline{Severity} = 2.37$ – $MeanGap_{Max} = 8.73$										

TABLE 6.32
DNA & Résultats (agrégation de type B & masque en /16)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
12 :54	12 :55	AHBGCE-A-A-A	AHBGCE	98.23-92.43-92.43-92.43	3	3	6	1	-1.34	
12 :55	12 :56	AHBGCE-A-A-A	AHBGCE	98.92-93.22-93.22-93.22	0	0	0	0	-2.34	
12 :56	12 :57	AHBGCD-A-A-A	AHBGCD	98.59-89.05-89.05-89.05	1	1	2	0	-2.34	
12 :57	12 :58	AHBGEC-A-A-A	AHBGEC	96.88-84.71-84.71-84.71	2	2	4	0	-2.34	
12 :58	12 :59	AWBGHE-A-A-A	AWBGHE	96.9-86.35-86.35-86.35	3	3	6	1	-1.34	
12 :59	13 :00	ABWHGD-A-A-B	ABWHGD	96.63-81.73-81.73-4.93	4	3	7	1.36	-0.98	
13 :00	13 :01	BaRACI-BaRCI-B-A	BaRACI	94.49-87.18-57.82-7.31	13	6	**19**	10.03	7.69	1
13 :01	13 :02	BAaRCI-BaRCI-B-A	BAaRCI	93.22-84.45-55.94-8.77	2	2	4	0	-2.34	1
13 :02	13 :03	ABHGWY-AB-A-A	ABHGWY	88.77-66.56-48.99-48.99	11	5	**16**	7.11	4.77	1
13 :03	13 :04	ABHGWC-A-A-BE	ABHGWC	95.04-66.49-66.49-15.2	4	1	5	0	-2.34	1
13 :04	13 :05	AHBGCf-Af-A-BE	AHBGCf	96.47-85.25-84.86-3.57	4	3	**7**	1.36	-0.98	1
13 :05	13 :06	ABHGEC-AB-AB-A	ABHGEC	94.6-82.62-82.62-52.6	6	3	**9**	2.25	-0.09	1
13 :06	13 :07	ABHGEC-AB-AB-A	ABHGEC	93.98-80.67-80.67-48.3	0	0	0	0	-2.34	1
13 :07	13 :08	AHGBCS-A-A-A	AHGBCS	92.6-75.1-75.1-75.1	5	3	**8**	1.78	-0.56	1
13 :08	13 :09	ABHGXC-A-A-B	ABHGXC	95.66-45.31-45.31-39.93	4	3	**7**	1.36	-0.98	1
13 :09	13 :10	BAHGXC-B-AC-A	BAHGXC	95.78-47.46-38.53-36.89	4	1	5	0	-2.34	
14 :21	14 :22	AaEYHZ-AaEYZb-A-Y	AaEYHZ	94.5-94.39-63.5-2.41	3	1	4	0	-2.34	
14 :22	14 :23	AEYHBa-AEYaZb-AYaZb-Y	AEYHBa	83.05-79.92-60.06-6.93	9	3	**12**	4	1.66	2
14 :23	14 :24	YAbEaB-YAbaZ-YA-Y	YAbEaB	96.84-94.05-85.62-77.52	13	5	**18**	9	6.66	2
14 :24	14 :25	YAbEDG-YAbaZ-Y-A	YAbEDG	97.25-92.26-77.09-8.05	4	2	6	1	-1.34	2
14 :25	14 :26	YAEDGH-Y-AZb-A	YAEDGH	91.1-34.68-27.84-26.91	7	2	**9**	2.25	-0.09	2
14 :26	14 :27	AEDBHG-Y-b-Z	AEDBHG	86.39-1.93-1.37-1.03	6	3	**9**	2.25	-0.09	2
14 :27	14 :28	AEDHBG-A-Y-b	AEDHBG	87.92-35.54-1.76-1.25	4	1	5	0	-2.34	
14 :42	14 :43	AEDHGX-AD-A-A	AEDHGX	94.7-82.63-79.28-79.28	2	1	3	0	-2.34	
14 :43	14 :44	AENHDG-AENDBY-AD-A	AENHDG	92.9-89.89-75.47-73.08	8	3	**11**	3.36	1.02	3
14 :44	14 :45	AEYNHD-AEYN-A-A	AEYNHD	86.79-79.17-48.28-48.28	6	2	**8**	1.78	-0.56	3
14 :45	14 :46	AEYDBG-AEY-A-A	AEYDBG	89.81-78.37-50.92-50.92	4	3	**7**	1.36	-0.98	3
14 :46	14 :47	aAEDHB-a-A-A	aAEDHB	98.66-81.73-13.76-13.76	7	4	**11**	3.36	1.02	3
14 :47	14 :48	aAEDHG-a-A-A	aAEDHG	98.56-81.48-13.79-13.79	1	1	2	0	-2.34	
14 :48	14 :49	AEBDHG-A-A-B	AEBDHG	93.82-74.0-74.0-2.26	4	2	**6**	1	-1.34	4
14 :49	14 :50	AEBDHG-AB-A-A	AEBDHG	94.54-81.6-79.11-79.11	2	0	2	0	-2.34	
14 :50	14 :51	AEDBHG-A-A-A	AEDBHG	89.89-59.22-59.22-59.22	2	1	3	0	-2.34	
14 :51	14 :52	AEDHGB-AW-A-A	AEDHGB	95.67-84.14-83.83-83.83	3	2	5	0	-2.34	
14 :52	14 :53	AEDGHX-AW-A-A	AEDGHX	95.64-83.34-83.06-83.06	2	2	4	0	-2.34	
14 :53	14 :54	AEGHDB-A-A-A	AEGHDB	92.12-56.66-56.66-56.66	4	3	**7**	1.36	-0.98	5
14 :54	14 :55	AEGHBD-ABZC-A-A	AEGHBD	92.41-67.47-63.43-63.43	4	1	5	0	-2.34	
14 :55	14 :56	AEHGDB-ADBCZ-AY-E	AEHGDB	91.17-59.5-53.6-25.19	6	2	**8**	1.78	-0.56	6
14 :56	14 :57	WAaEHD-Wa-A-E	WAaEHD	94.05-53.77-24.49-11.61	10	4	**14**	5.44	3.1	7
14 :57	14 :58	aWAEDH-a-A-A	aWAEDH	95.46-45.0-21.15-21.15	5	3	8	1.78	-0.56	
14 :58	14 :59	aAYEBG-a-A-A	aAYEBG	94.22-37.72-29.67-29.67	4	4	8	1.78	-0.56	
14 :59	15 :00	AYEGHa-AYa-A-A	AYEGHa	92.23-82.75-48.71-48.71	6	4	**10**	2.78	0.44	8
15 :00	15 :01	ABaRCE-AaRYbZ-Aa-B	ABaRCE	86.64-54.58-46.26-27.62	11	5	**16**	7.11	4.77	9
15 :01	15 :02	ABaRHC-AB-AB-A	ABaRHC	90.18-78.19-78.19-63.12	9	2	**11**	3.36	1.02	9
15 :02	15 :03	aAYHEB-a-AY-A	aAYHEB	99.7-94.89-4.61-3.7	8	5	**13**	4.69	2.35	10
15 :03	15 :04	aAYbHG-a-AY-AbB	aAYbHG	99.64-95.59-3.42-3.01	5	3	8	1.78	-0.56	
15 :04	15 :05	AbBHEG-AbBEa-Ab-AB	AbBHEG	92.95-85.83-76.64-63.81	10	4	**14**	5.44	3.1	11
15 :05	15 :06	AbZBHE-AbBaY-AZ-b	AbZBHE	93.11-74.21-59.57-21.49	7	2	9	2.25	-0.09	

Capture : 2011-08-17 – $\mathcal{F}$ = 'DportSlice2048' – Activity = 'SrcBytes' – Precision = 6

$ANOD_{Max} = 20$ – $\overline{ANOD} = 8.35$ – $Severity_{Max} = 11.11$ – $\overline{Severity} = 2.34$ – $MeanGap_{Max} = 8.77$

TABLE 6.33
DNA & Résultats (agrégation de type A & masque en /24)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
12 :55	12 :56	AHBGCE-A-A-A	AHBGCE	98.92-93.22-93.22-93.22	0	0	0	0	-2.42	
12 :56	12 :57	AHBGCD-A-A-A	AHBGCD	98.59-89.05-89.05-89.05	1	1	2	0	-2.42	
12 :57	12 :58	AHBGEC-A-A-A	AHBGEC	96.88-84.71-84.71-84.71	2	2	4	0	-2.42	
12 :58	12 :59	AWBGHE-A-A-A	AWBGHE	96.9-86.35-86.35-86.35	3	3	6	1	-1.42	
12 :59	13 :00	ABWHGD-AB-A-A	ABWHGD	96.63-86.66-81.73-81.73	4	3	7	1.36	-1.06	
13 :00	13 :01	BaRACI-BaRCI-B-A	BaRACI	94.49-87.18-57.82-7.31	12	6	**18**	9	6.58	1
13 :01	13 :02	BAaRCI-BaRCI-B-A	BAaRCI	93.22-84.45-55.94-8.77	2	2	4	0	-2.42	
13 :02	13 :03	ABHGWY-AB-A-A	ABHGWY	88.77-66.56-48.99-48.99	11	5	**16**	7.11	4.69	1
13 :03	13 :04	ABHGWC-AB-A-BE	ABHGWC	95.04-81.54-66.49-15.2	3	1	4	0	-2.42	
13 :04	13 :05	AHBGCf-Af-A-BE	AHBGCf	96.47-85.25-84.86-3.57	4	3	**7**	1.36	-1.06	1
13 :05	13 :06	ABHGEC-AB-AB-A	ABHGEC	94.6-82.62-82.62-52.6	6	3	**9**	2.25	-0.17	1
13 :06	13 :07	ABHGEC-AB-A-A	ABHGEC	93.98-80.67-48.3-48.3	1	0	1	0	-2.42	
13 :07	13 :08	AHGBCS-A-A-A	AHGBCS	92.6-75.1-75.1-75.1	4	3	**7**	1.36	-1.06	1
13 :08	13 :09	ABHGXC-A-A-B	ABHGXC	95.66-45.31-45.31-39.93	4	3	**7**	1.36	-1.06	1
13 :09	13 :10	BAHGXC-B-AC-A	BAHGXC	95.78-47.46-38.53-36.89	4	1	5	0	-2.42	
Capture : 2011-08-17 – $\mathcal{F}$ = 'DportSlice2048' – Activity = 'SrcBytes' – Precision = 6										
$ANOD_{Max} = 21$ – $\overline{ANOD} = 8.46$ – $Severity_{Max} = 12.25$ – $\overline{Severity} = 2.42$ – $MeanGap_{Max} = 9.83$										

6.2.2.3 Conclusion

Du tableau 6.27 et des tableaux 6.31 à 6.33 nous pouvons conclure que notre méthodologie DiNATrA$\mathcal{X}$ fonctionne et permet de détecter des anomalies réseaux. Ces anomalies révèlent l'activité d'un botnet constitué de bots Neris, activités consistant à générer des attaques de type 'ClickFraude', 'Spam' ou 'PortScan'. En effet, ces attaques génèrent du trafic TCP que nous sommes à même de mettre en lumière.

Dans le tableau 6.31, une distance d'anormalité de '18' à 13 :00 marque le début d'une attaque TCP externe qui dure jusqu'à 13 :09 (ANOD = 7). Ces variations correspondent à l'événement '1' dans le tableau 6.27. Une distance d'anormalité de '14' à 14 :22 indique l'infection du premier bot et son réveil jusqu'à 14 :27 (ANOD = 10), événement '2'. Ce comportement est également détecté pour l'ensemble des bots (événements '3' à '11').

Des tableaux 6.31 et 6.32, nous pouvons déduire que l'utilisation d'un masque en '/16' ou '/24' donne quasiment les mêmes résultats ce qui peut s'expliquer par le tableau 6.30. En effet, le nombre de flux agrégés varie d'environ 10% entre un masque de 16 ou 24 bits. Des tableaux 6.31 et 6.33, il appert que le type d'agrégation ('A' ou 'B') n'influe pas sur les résultats de la détection d'anomalies, car le port source est choisi de façon aléatoire par le système d'exploitation.

Pour l'analyse des scénarios suivants, nous emploieront donc une agrégation de type 'B' pour s'affranchir des ports sources. De plus, nous utiliserons un masque de sous-réseau en '/16' pour diminuer les temps de calcul qui restent, dans tous les cas, très raisonnables pour cette étude.

6.2.3 Étude du scénario 10 (CTU-Malware-Capture-botnet-51)

Les données brutes relatives à ce scénario sont librement téléchargeables depuis le site d'hébergement du 'Stratosphere Research Laboratory' [3].

6.2.3.1 Caractéristiques & Déroulé

Il consiste en la capture de l'activité d'un botnet constitué de dix bots Windows XP infectés par le malware `Rbot`. Tout comme le scénario 9, les dix machines sont démarrées puis infectées tour à tour. La chronologie des événements [4] est détaillée dans le tableau 6.35. Il s'agit des mêmes bots utilisés pour générer le scénario 9. Ils ont donc tous les mêmes adresses IP appartenant aux mêmes sous-réseaux : 147.32.84.0/24, 147.32.80.0/20, 147.32.0.0/16, 147.32.0.0/12 et 147.0.0.0/8.

Comme pour le scénario 9, la caractéristique $\mathcal{F}$ définit la bande de ports correspondante au port destination. Ces bandes ont également une largeur de '2 048'. Le tableau 6.34 détaille le nombre de sous-réseaux impliqués dans ce scénario en fonction du masque.

L'analyse des données contenues dans le scénario 10 nous indique que les flux sont issus de 147 989 adresses IP et de 64 600 ports sources. Ils sont à destination de 67 424 adresses IP pour 41 653 ports. Les données issues du champ 'State' correspondent à 281 catégories.

Pour la détection des attaques, nous avons défini une période TP comprise entre le 18/08/2011 11 :00 et le 18/08/2011 15 :00. La fonction `computeTS` (algorithme 12) nous a permis de découper TP en TLs de 1 minute. Nous utilisons une agrégation de type 'B' avec un masque en '/16' et une précision de '5'. Du tableau 6.36, nous déduisons que des distances d'anormalité au voisinage de '10' marquant les débuts et fins des différentes attaques.

TABLE 6.34
NOMBRE DE SOUS-RÉSEAUX EN FONCTION DU MASQUE

	/8	**/12**	**/16**	**/20**	**/24**
SrcNetwork	182	2 142	18 192	77 371	128 122
DstNetwork	185	1 973	12 527	39 568	58 349
Scénario 10					

3. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-51/detailed-bidirectional-flow-labels/capture20110818.binetflow`
4. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-51/`

TABLE 6.35
DÉROULÉ DU SCÉNARIO 10

TimeSlot	Événement	Bot	EventId
10 :19	Début capture		
10 :46	Début démarrage VMs	SARUMAN à SARUMAN8	
11 :00	Fin démarrage VMs	SARUMAN9	
11 :03	Infection	SARUMAN9	
11 :03	Infection	SARUMAN8	
11 :04	Infection	SARUMAN7	
11 :04	Infection	SARUMAN6	
11 :05	Infection	SARUMAN4	
11 :05	Infection	SARUMAN2	
11 :06	Infection	SARUMAN	
11 :06	Infection	SARUMAN1 & SARUMAN3	
11 :07	Infection	SARUMAN5	
11 :52	Début attaque UDP	→ 147.32.96.69	1
12 :07	Fin attaque UDP	→ 147.32.96.69	1
12 :09	Début attaque UDP	→ 147.32.96.69	2
12 :11	Fin attaque UDP	→ 147.32.96.69	2
12 :13	Début attaque UDP	→ 147.32.96.69	3
12 :15	Fin attaque UDP	→ 147.32.96.69	3
12 :33	Début arrêt VMs	SARUMAN à SARUMAN8	
12 :36	Fin arrêt VMs	SARUMAN9	
13 :46	Début démarrage VMs	SARUMAN à SARUMAN8	
13 :52	Fin démarrage VMs	SARUMAN9	
13 :53	Début attaque UDP	→ 147.32.96.69	4
14 :05	Fin attaque UDP	→ 147.32.96.69	4
14 :07	Début attaque UDP	→ 147.32.96.69	5
14 :11	Fin attaque UDP	→ 147.32.96.69	5
14 :14	Début attaque UDP	→ 147.32.96.69	6
14 :16	Fin attaque UDP	→ 147.32.96.69	6
14 :40	Début attaque UDP	→ 147.32.96.69	7
14 :44-46	Fin attaque UDP	→ 147.32.96.69	7
15 :00	Début attaque UDP	→ 147.32.96.69	8
15 :03	Fin attaque UDP	→ 147.32.96.69	8
15 :04	Fin capture		
18/08/2011 – 5 heures de capture (175 Mo)			

TABLE 6.36
DNA & Résultats (agrégation de type B & masque en /16)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
11 :50	11 :51	AHDBC-ACE-A-D	AHDBC	88.04-74.16-66.52-6.23	5	3	8	2.56	-0.44	
11 :51	11 :52	EAHBD-E-AD-A	EAHBD	95.82-58.85-31.92-30.38	7	3	**10**	4	1	1
11 :52	11 :53	AEfaH-AEfa-AD-A	AEfaH	99.92-99.88-98.69-98.68	7	4	11	4.84	1.84	1
11 :53	11 :54	AfeaH-A-fe-f	AfeaH	99.94-73.09-26.67-15.51	8	2	10	4	1	1
11 :54	11 :55	feaEA-fe-f-f	feaEA	99.6-92.71-54.02-54.02	5	3	8	2.56	-0.44	1
11 :55	11 :56	faEAB-fa-f-E	faEAB	97.93-65.97-36.73-25.53	4	2	6	1.44	-1.56	1
11 :56	11 :57	fEAHB-f-E-AY	fEAHB	97.8-57.36-24.11-13.76	6	2	8	2.56	-0.44	1
11 :57	11 :58	ADHCB-ACY-D-D	ADHCB	89.13-69.87-9.29-9.29	10	4	14	7.84	4.84	1
11 :58	11 :59	ADHCX-A-D-D	ADHCX	88.16-48.49-23.07-23.07	3	1	4	0	-3	1
11 :59	12 :00	ZADHX-Z-A-A	ZADHX	96.05-71.8-16.46-16.46	5	2	7	1.96	-1.04	1
12 :00	12 :01	ZEAHB-Z-E-A	ZEAHB	97.77-53.54-27.41-14.36	4	3	7	1.96	-1.04	1
12 :01	12 :02	EAYHB-AY-E-A	EAYHB	96.09-47.45-44.8-23.84	4	2	6	1.44	-1.56	1
12 :02	12 :03	AYHBE-AY-AE-A	AYHBE	95.75-87.77-55.27-53.57	3	2	5	1	-2	1
12 :03	12 :04	bAaPH-bA-AY-a	bAaPH	96.31-72.36-33.67-16.18	9	5	14	7.84	4.84	1
12 :04	12 :05	beIAa-be-IAPY-a	beIAa	85.26-49.9-34.89-10.41	7	4	11	4.84	1.84	1
12 :05	12 :06	eIAPH-IA-e-AP	eIAPH	93.8-40.74-40.56-28.27	9	3	12	5.76	2.76	1
12 :06	12 :07	EAHBC-E-A-A	EAHBC	97.28-59.35-31.73-31.73	9	5	**14**	7.84	4.84	1
12 :07	12 :08	AEHBC-A-A-E	AEHBC	99.77-82.96-82.96-16.33	3	1	4	0	-3	
12 :08	12 :09	AEPHB-A-A-E	AEPHB	99.4-85.74-85.74-12.78	2	2	4	0	-3	
12 :09	12 :10	DAPBH-D-AB-A	DAPBH	98.66-88.7-6.91-6.04	6	3	**9**	3.24	0.24	2
12 :10	12 :11	DAPBH-D-AB-A	DAPBH	99.08-88.5-7.52-6.66	0	0	0	0	-3	2
12 :11	12 :12	APCHB-AP-A-C	APCHB	99.37-98.49-96.93-0.36	7	3	**10**	4	1	2
12 :12	12 :13	AEPCH-A-E-C	AEPCH	99.45-92.2-5.81-0.33	4	2	6	1.44	-1.56	
12 :13	12 :14	EADHB-E-AD-AR	EADHB	97.9-76.26-18.5-12.57	9	4	**13**	6.76	3.76	3
12 :14	12 :15	EAPDB-E-APD-AB	EAPDB	95.98-50.1-43.06-24.62	4	2	6	1.44	-1.56	3
12 :15	12 :16	AGPBH-AB-A-A	AGPBH	94.94-65.77-61.13-61.13	9	4	**13**	6.76	3.76	3
12 :16	12 :17	AUGBH-AB-A-U	AUGBH	95.06-48.94-45.74-30.77	3	2	5	1	-2	
13 :50	13 :51	EABHD-E-A-A	EABHD	97.77-45.65-45.27-45.27	5	3	8	2.56	-0.44	
13 :51	13 :52	AEHBD-AE-A-A	AEHBD	97.78-90.13-46.61-46.61	3	2	5	1	-2	
13 :52	13 :53	cEADH-cE-ADC-A	cEADH	97.11-56.6-38.67-27.63	7	4	**11**	4.84	1.84	4
13 :53	13 :54	fDcEA-DcEAC-f-A	fDcEA	97.25-53.26-44.71-11.65	8	4	12	5.76	2.76	4
13 :54	13 :55	fEDAH-fAC-f-ED	fEDAH	97.95-67.59-55.92-30.01	8	3	11	4.84	1.84	4
13 :55	13 :56	EAHCB-EACf-A-C	EAHCB	95.87-89.7-32.26-2.52	9	4	13	6.76	3.76	4
13 :56	13 :57	AEHCD-AEC-A-A	AEHCD	99.28-98.24-89.73-89.73	5	2	7	1.96	-1.04	4
13 :57	13 :58	AEHDB-AEC-A-A	AEHDB	99.38-98.61-83.61-83.61	2	2	4	0	-3	4
13 :58	13 :59	fEAHB-EAC-f-A	fEAHB	98.94-52.34-45.52-9.87	5	3	8	2.56	-0.44	4
13 :59	14 :00	AfEcH-AEcC-A-f	AfEcH	99.27-70.33-42.79-28.46	7	3	10	4	1	4
14 :00	14 :01	AcEDH-AcEC-AC-A	AcEDH	98.11-96.68-67.06-66.41	5	2	7	1.96	-1.04	4
14 :01	14 :02	EAGDP-E-A-C	EAGDP	87.97-48.27-23.78-3.37	9	4	13	6.76	3.76	4
14 :02	14 :03	AGHPB-AG-A-C	AGHPB	85.91-68.84-55.74-2.41	5	3	8	2.56	-0.44	4
14 :03	14 :04	AEaHD-AaC-A-E	AEaHD	98.7-86.18-80.12-11.42	7	4	**11**	4.84	1.84	4
14 :04	14 :05	AEaJH-AC-A-Ea	AEaJH	98.37-71.08-70.83-25.38	4	2	6	1.44	-1.56	
14 :05	14 :06	AEJHC-A-A-EC	AEJHC	99.01-67.53-67.53-29.14	4	2	6	1.44	-1.56	
14 :06	14 :07	AECHD-AC-A-E	AECHD	98.77-76.42-75.62-21.14	4	2	6	1.44	-1.56	
14 :07	14 :08	dAHCD-d-AC-A	dAHCD	92.79-50.73-35.4-32.81	7	3	**10**	4	1	5
14 :08	14 :09	AdEHD-AdEC-A-A	AdEHD	94.82-92.44-64.39-64.39	7	3	**10**	4	1	5
14 :09	14 :10	AEBJD-A-A-B	AEBJD	96.0-73.08-73.08-3.05	7	3	**10**	4	1	5
14 :10	14 :11	AEBJY-AEB-A-B	AEBJY	93.37-89.13-62.88-5.73	3	1	4	0	-3	5
14 :11	14 :12	EAJBH-E-AB-A	EAJBH	89.69-42.45-40.13-36.46	7	3	**10**	4	1	5
14 :12	14 :13	EAJHB-E-A-A	EAJHB	95.43-45.84-40.06-40.06	2	1	3	0	-3	
14 :13	14 :14	EADJY-E-AD-A	EADJY	91.78-45.31-39.31-33.61	4	3	**7**	1.96	-1.04	6
14 :14	14 :15	AEDYB-A-A-E	AEDYB	98.63-50.81-50.81-45.56	6	3	**9**	3.24	0.24	6
14 :15	14 :16	AEBDH-AE-A-B	AEBDH	99.25-95.93-50.41-2.08	5	3	**8**	2.56	-0.44	6
14 :16	14 :17	EABDH-EAB-A-B	EABDH	98.03-94.28-36.74-6.9	3	1	4	0	-3	
14 :39	14 :40	AbHRF-A-A-b	AbHRF	94.05-46.77-46.77-41.13	5	3	7	2.56	-0.44	
14 :40	14 :41	AEbFB-AEb-A-A	AEbFB	97.99-96.13-67.66-67.66	7	4	**11**	4.84	1.84	7
14 :41	14 :42	AEBHF-A-A-E	AEBHF	98.78-70.21-70.21-26.38	6	3	**9**	3.24	0.24	7
14 :42	14 :43	EAHBD-E-A-A	EAHBD	96.39-47.8-41.7-41.7	5	3	**8**	2.56	-0.44	7
14 :43	14 :44	ABHCE-ABE-AE-A	ABHCE	83.69-70.65-63.04-58.92	7	4	**11**	4.84	1.84	7
14 :44	14 :45	fABbC-f-AE-E	fABbC	91.05-76.09-7.56-0.92	7	3	10	4	1	
14 :45	14 :46	fBAEa-f-B-AEa	fBAEa	93.07-53.7-27.74-11.63	7	3	10	4	1	

Capture : 2011-08-18 – $\mathcal{F}$ = **'DportSlice2048'** – **Activity = 'SrcBytes'** – **Precision = 5**
$ANOD_{Max} = 17$ – $\overline{ANOD} = 8.03$ – $Severity_{Max} = 11.56$ – $\overline{Severity} = 3$ – $MeanGap_{Max} = 8.56$

6.2.3.2 Analyse & Résultats

A l'aide de la fonction `extract_data`, nous conservons *uniquement* les données issues du sous-réseau contenant les bots (srcNetwork16 = 147.32.0.0/16) puis nous calculons les ADN de chaque période TL et les distances correspondantes. Les résultats obtenus sont reportés dans le tableau 6.37. Nous constatons que les débuts et fin d'attaques sont bien détectés et matérialisés par de très importantes variations de la distance d'anormalité.

TABLE 6.37
DNA & Résultats (agrégation de type B avec sous-réseau 147.32.0.0/16)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
13 :51	13 :52	AEBDC-A-A-E	AEBDC	98.64-48.6-48.6-46.44	3	1	4	0	-2.11	
13 :52	13 :53	cEABC-cE-AC-A	cEABC	98.71-67.02-30.57-30.16	7	3	**10**	4	1.89	4
13 :59	14 :00	AEcBD-AEcC-A-A	AEcBD	99.65-99.18-60.36-60.36	7	3	**10**	4	1.89	4
14 :00	14 :01	AcEDC-AcEC-AC-A	AcEDC	98.91-97.71-67.76-67.11	5	3	**8**	2.56	0.45	4
14 :01	14 :02	EADCB-E-AC-C	EADCB	95.86-58.8-29.92-4.12	7	3	**10**	4	1.89	4
14 :02	14 :03	ABCMD-A-A-C	ABCMD	94.54-80.85-80.85-3.81	6	4	**10**	4	1.89	4
14 :03	14 :04	AEaBC-AC-A-Ea	AEaBC	99.44-81.68-81.39-17.26	7	4	**11**	4.84	2.73	4
14 :04	14 :05	AEaBC-AC-A-Ea	AEaBC	99.36-72.94-72.69-26.07	0	0	0	0	-2.11	
14 :05	14 :06	AECBD-A-A-EC	AECBD	99.51-69.31-69.31-29.87	4	2	6	1.44	-0.67	
14 :06	14 :07	AECBD-AC-A-E	AECBD	99.67-77.78-76.96-21.54	2	0	2	0	-2.11	
14 :07	14 :08	dACBD-d-AC-A	dACBD	97.95-58.67-36.36-33.39	6	2	**8**	2.56	0.45	6
14 :08	14 :09	AdEaC-AdC-A-A	AdEaC	97.88-89.65-66.65-66.65	7	4	**11**	4.84	2.73	6
14 :09	14 :10	AEBaY-AE-A-B	AEBaY	98.32-93.51-75.43-3.19	6	3	**9**	3.24	1.13	6
14 :10	14 :11	AEBYb-AEB-A-B	AEBYb	96.31-92.89-65.4-6.08	3	2	5	1	-1.11	
14 :11	14 :12	EABYZ-E-AB-A	EABYZ	93.28-48.16-41.03-36.87	6	2	8	2.56	0.45	
14 :12	14 :13	EABDV-E-A-A	EABDV	96.29-52.19-41.57-41.57	3	2	5	1	-1.11	
14 :13	14 :14	EAYBD-EA-A-A	EAYBD	96.38-89.86-35.96-35.96	3	2	5	1	-1.11	
14 :14	14 :15	AEYBD-A-A-E	AEYBD	99.57-51.47-51.47-46.88	3	1	4	0	-2.11	
14 :15	14 :16	AEBDZ-AE-A-B	AEBDZ	99.62-97.37-50.88-2.13	4	2	6	1.44	-0.67	
14 :16	14 :17	EABDZ-EAB-A-B	EABDZ	98.86-98.52-38.45-7.45	3	1	4	0	-2.11	
14 :38	14 :39	ARBCD-A-A-A	ARBCD	95.9-84.51-84.51-84.51	3	2	5	1	-1.11	
14 :39	14 :40	AbRBD-A-A-b	AbRBD	98.11-47.43-47.43-46.03	3	2	5	1	-1.11	
14 :40	14 :41	AEbBV-AEb-A-A	AEbBV	99.36-98.33-69.05-69.05	6	3	**9**	3.24	1.13	7
14 :41	14 :42	AEBVD-AE-A-A	AEBVD	99.49-98.26-71.49-71.49	3	2	**5**	1	-1.11	7
14 :42	14 :43	EABDG-E-A-A	EABDG	97.65-51.58-42.82-42.82	4	3	**7**	1.96	-0.15	7
14 :43	14 :44	ACBVD-ACB-A-A	ACBVD	94.49-90.43-76.74-76.74	7	4	**11**	4.84	2.73	7
14 :44	14 :45	ACBVT-AC-ABV-A	ACBVT	91.99-77.1-70.56-58.13	4	1	**5**	1	-1.11	7
14 :45	14 :46	AEBCY-A-A-E	AEBCY	92.35-43.34-43.34-37.61	7	3	**10**	4	1.89	7
14 :46	14 :47	AbEBe-AbE-A-A	AbEBe	94.29-89.94-43.22-43.22	6	3	**9**	3.24	1.13	7
14 :47	14 :48	AbWBe-Ab-A-A	AbWBe	95.26-86.21-65.02-65.02	2	1	3	0	-2.11	
14 :48	14 :49	ACWBa-A-A-A	ACWBa	97.22-80.09-80.09-80.09	3	2	5	1	-1.11	
14 :59	15 :00	EAdBD-E-AB-A	EAdBD	98.56-54.4-40.75-37.97	1	0	1	0	-2.11	
15 :00	15 :01	ABRDC-AB-A-A	ABRDC	97.01-95.36-87.62-87.62	7	4	**11**	4.84	2.73	
15 :01	15 :02	ABRDC-A-A-A	ABRDC	97.56-90.52-90.52-90.52	1	0	**1**	0	-2.11	
15 :02	15 :03	ABDeC-ABE-A-A	ABDeC	97.79-97.13-92.04-92.04	4	2	**6**	1.44	-0.67	
15 :03	15 :04	ABCaK-AC-A-A	ABCaK	97.04-88.51-85.79-85.79	5	3	**8**	2.56	0.45	
15 :04	15 :05	ABCaK-ABa-AC-A	ABCaK	96.92-90.98-85.23-79.88	3	0	3	0	-2.11	
Capture : 2011-08-18 — $\mathcal{F}$ = 'DportSlice2048' — Activity = 'SrcBytes' — Precision = 5										
$ANOD_{Max} = 17$ – $\overline{ANOD} = 8.03$ – $Severity_{Max} = 11.56$ – $\overline{Severity} = 3$ – $MeanGap_{Max} = 8.56$										

6.2.3.3 Conclusion

L'étude des tableaux 6.35 et 6.36 confirme que DiNATrA$\mathcal{X}$ permet de détecter les différentes attaques. Celles-ci ont été lancées au cours de deux périodes distinctes (11 :53 — 12 :15 & 13 :53 — 15 :00), et ce, à des intervalles de temps différents. Les débuts et fins de ces attaques sont, pour une majorité d'entre-elles, matérialisés par une distance d'anormalité (ANOD) élevée et un écart de sévérité (MeanGap) positif ou nul.

6.2.4 Étude du scénario 11 (CTU-Malware-Capture-botnet-52)

Les données brutes relatives à ce scénario du CTU-13 sont librement téléchargeables depuis le site d'hébergement du 'Stratosphere Research Laboratory' [5].

6.2.4.1 Caractéristiques & Déroulé

Il consiste en la capture de l'activité d'un botnet constitué de trois bots Windows XP infectés par le malware `Rbot`. Les trois VMs (Virtual Machines) sont démarrées simultanément. Il s'agit ici d'une attaque de type DDoS basée sur le protocole ICMP. La chronologie des événements [6] est détaillée dans le tableau 6.38. Les bots utilisés pour ce scénario sont '147.32.84.165' et '147.32.84.191 à .192', adresses appartenant aux sous-réseaux : 147.32.84.0/24, 147.32.80.0/20, 147.32.0.0/16, 147.32.0.0/12 et 147.0.0.0/8.

TABLE 6.38
Déroulé du scénario 11

TimeSlot	Événement	Bot	EventId
15 :40	Début capture		
15 :45	Démarrage VMs	SARUMAN à SARUMAN2	1
15 :47	Infection	SARUMAN à SARUMAN2	2
15 :52	Début attaque ICMP	SARUMAN à SARUMAN2	3
15 :54	Fin attaque ICMP	SARUMAN à SARUMAN2	3
15 :55	Fin capture		
18/08/2011 – 20 mn de capture (14 Mo)			

5. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-52/detailed-bidirectional-flow-labels/capture20110818.binetflow`
6. `https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-botnet-52/`

6.2.4.2 Analyse & Résultats

TABLE 6.39
DNA & Résultats (agrégation de type B & masque en /16)

Begin	End	DNA	Strand	Volumes	DNAD	STAD	ANOD	Severity	MeanGap	EventId
15 :40	15 :41	CZbcY-CZbcY-A-f	CZbcY	72.54-72.54-6.95-5.4	11	5	16	10.24	7.32	
15 :41	15 :42	CZYbc-dGQ-f-F	CZYbc	55.82-15.36-4.2-3.32	9	2	11	4.84	1.92	
15 :42	15 :43	dGEAQ-dG-Q-F	dGEAQ	77.02-34.8-13.07-8.85	7	5	12	5.76	2.84	
15 :43	15 :44	EARHf-E-AQ-R	EARHf	88.81-46.46-30.1-4.8	9	5	14	7.84	4.92	
15 :44	15 :45	AHCDB-AH-ACI-AQe	AHCDB	84.92-72.36-67.99-65.06	12	5	17	11.56	8.64	1
15 :45	15 :46	AHCDJ-AH-Ae-A	AHCDJ	87.43-76.66-68.97-67.69	5	1	6	1.44	-1.48	
15 :46	15 :47	ADHJE-AT-A-A	ADHJE	93.62-79.22-78.34-78.34	5	3	8	2.56	-0.36	2
15 :47	15 :48	ADHJE-AT-A-A	ADHJE	94.1-77.28-76.41-76.41	0	0	0	0	-2.92	
15 :48	15 :49	AHEDJ-Ae-A-A	AHEDJ	94.01-79.6-79.26-79.26	4	3	7	1.96	-0.96	2
15 :49	15 :50	AHDJE-AG-A-A	AHDJE	94.17-82.92-82.26-82.26	3	2	5	1	-1.92	
15 :50	15 :51	APHJD-AP-A-A	APHJD	95.79-88.9-64.08-64.08	4	3	7	1.96	-0.96	
15 :51	15 :52	APHEJ-AP-A-A	APHEJ	96.34-89.63-65.32-65.32	2	2	4	0	-2.92	
15 :52	15 :53	AHEJD-A-A-A	AHEJD	95.3-85.08-85.08-85.08	3	2	5	1	-1.92	
15 :53	15 :54	AHJDE-A-A-A	AHJDE	93.44-84.22-84.22-84.22	2	2	4	0	-2.92	
15 :54	15 :55	ADHJE-A-A-A	ADHJE	94.12-76.41-76.41-76.41	2	2	4	0	-2.92	
15 :55	15 :56	ADHJE-A-A-A	ADHJE	93.55-65.93-65.93-65.93	0	0	0	0	-2.92	
15 :56	15 :57	AEDHJ-AED-A-A	AEDHJ	92.52-84.9-58.83-58.83	4	2	6	1.44	-1.48	
15 :57	15 :58	AEHJD-AED-A-E	AEHJD	92.76-84.2-47.87-35.3	3	2	5	1	-1.92	
Capture : 2011-08-18 – $\mathcal{F}$ = 'DportSlice2048' – Activity = 'SrcBytes' – Precision = 5										
$ANOD_{Max} = 17$ – $\overline{ANOD} = 7.27$ – $Severity_{Max} = 11.56$ – $\overline{Severity} = 2.92$ – $MeanGap_{Max} = 8.64$										

6.2.4.3 Conclusion

Tout comme pour le scénario 9, le tableau 6.39 nous permet d'affirmer que nous sommes capable de détecter le trafic généré par les bots lors de leur démarrage. Celui-ci est dû aux tentatives de connexion au serveur de Commande & de Contrôle qui permet au botmaster de contrôler le botnet comme expliqué dans la section 3.1.2. Il est également intéressant de constater que les attaques ne sont pas détectées par DiNATrA$\mathcal{X}$. En effet, ces attaques sont basées sur le protocole 'ICMP', protocole qui a été exclu des scénarios comme expliqué dans la section 6.2.1.2.

6.3 Conclusion

6.3.1 Contributions

Notre "Framework de détection d'anomalies par signature numérique" nous a permis de détecter des anomalies réseaux grâce à l'apport des notions de *signature numérique* et de *secteurs* utilisées afin de caractériser l'activité réseau pour une zone particulière. Les difficultés liées à ces notions étaient : *(i)* le découpage ou la formalisation de ces secteurs *(ii)* la définition de la caractéristique permettant de mesurer ou matérialiser l'activité réseau *(iii)* l'automatisation de la comparaison des signatures numériques.

Ces difficultés ont été palliées grâce à notre méthodologie DiNATrA$\mathcal{X}$ qui a apporté les concepts de : *(i)* "Sector of Interest" ou SOI *(ii)* la caractérisation $\mathcal{F}$ *(iii)* DNA et de brin permettant de codifier les signatures numériques *(iv)* la distance d'anormalité ainsi que la mesure de sévérité permettant de quantifier les variations et confirmer les anomalies.

6.3.2 Résultats & Analyse

Des études menées sur les jeux de données CANCAN et CTU-13, nous pouvons conclure que notre méthodologie est fonctionnelle. En effet, elle nous a permis de mettre en lumière des événements ayant modifié la nature du trafic réseau comme ceux détaillés dans la section 6.1.4 ou encore de détecter les différentes étapes caractérisant l'activité malveillante de différents botnets comme expliqué dans les sections 6.2.2 et 6.2.3.

Une fois définies la logique de constitution des SOI, la caractéristique $\mathcal{F}$ et les périodes TP, TL et TS ; DiNATrA$\mathcal{X}$ permet de détecter les anomalies en *temps réel*. Bien évidemment, cette notion de temps réel dépend du contexte dans lequel est mise en œuvre notre méthodologie.

La principale difficulté pour déployer DiNATrA$\mathcal{X}$ est la configuration des paramètres d'analyse à savoir la période TL et la précision notamment. Ceci dépend des anomalies recherchées et nécessite de tester différentes valeurs.

Chapitre 7

Conclusion générale & Perspectives

7.1 Contexte

La plupart des objets que nous utilisons au quotidien sont connectés. Ils échangent régulièrement des données via différents réseaux informatiques et, de ce fait, génèrent de l'activité. La majeure partie de cette activité est maîtrisée, parfaitement connue et même souhaitée. Mais, dans certains cas, des échanges sortent de cet ordinaire rassurant. Il est souvent intéressant voire primordial de pouvoir détecter ces flux réseaux que nous pouvons qualifier d'anormaux car non désirés ou non standards. Ceux-ci sont le fait soit d'événements *intérieurs* soit de causes *extérieures* au réseau surveillé.

Les événements intérieurs possibles sont par exemples des pannes, des événement particuliers comme des déploiements de mises à jour ou encore des incidents de sécurité telles que des cyberattaques orchestrées par des équipements compromis.

Les causes extérieures pouvant influer sur la nature ou le volume des échanges réseaux sont nécessairement des événements ayant un impact direct sur le trafic ou sur le comportement des usagers. Dans le cas contraire, ces événements ne se matérialiseraient pas au niveau du réseau supervisé. Nous pouvons citer par exemple une forte augmentation ou diminution du nombre d'utilisateurs connectés, un changement dans le type d'activité ou de méthode d'accès aux ressources.

Dans le cadre de cette thèse, nous avons proposé différentes approches dans le but de détecter des anomalies réseaux liées tout d'abord à des événements internes de type activité de botnets et des événements externes comme des mouvements de foules ou des attroupements.

7.2 Contributions

Ces dernières années, les botnets sont devenus une menace majeure pour l'ensemble des systèmes d'information comme l'ont encore malheureusement démontré les cyberattaques récentes subies par les deux sociétés intermédiaires de santé (voir section 2.2.9) et le RIE (voir section 2.2.10). Notre "Forêt combinée" [4], présentée dans le chapitre 3, est un métamodèle capable de détecter différentes familles de ces botnets. Il est efficient en termes de performance, temps de construction et de détection. De plus, contrairement à d'autres approches, celui-ci est très facilement évolutif. Si une autre famille de botnets apparaît et qu'un jeu de données matérialisant son type d'activité réseau existe, il suffit d'en créer un modèle pour la détecter et de l'amender à notre "Forêt combinée". Par contre, les modèles à combiner sont construits à l'aide d'algorithmes d'apprentissage automatique *supervisés* ce qui nécessite des données labellisées et mises à jour. L'idée à l'origine de notre "Forêt combinée" est l'algorithme supervisé "Random Forest" (voir section 2.8.3. Celui-ci consiste à générer puis fusionner des arbres de décision créés de façon *aléatoire*. Notre idée a été de fusionner non pas des arbres construits aléatoirement, mais des arbres capables, chacun pris isolément, de détecter l'activité intrinsèque (échanges internes au botnet ou les communications avec le C&C serveur) d'une famille bien particulière de botnets. Pour ce faire, nous avons sélectionné et utilisé différents scénarios du jeu de données CTU-13, décrit dans la section 2.6, pour construire nos arbres de décision.

Notre deuxième contribution, expliquée dans le chapitre 4, nous a permis d'introduire notre concept de *signature numérique* [2]. Contrairement à d'autres travaux basés sur l'étude statistique de quelques critères caractérisant les trames circulant dans un segment/lien réseau bien précis, notre *signature numérique* est une "photographie" à un instant t des *flux réseaux agrégés* pour un secteur réseau particulier. Les signatures numériques qui sont les images de l'activité réseau de différents secteurs sont générées à l'aide d'un algorithme d'apprentissage automatique *non supervisé*. Sur la base de ces photographies, nous avons proposé notre "Framework de détection d'anomalies par signature numérique" [1]. Celui-ci permet de mettre en lumière des anomalies réseaux liées au changement d'activité des utilisateurs et de corréler ces variations d'activité à des événements extérieurs. Les principaux avantages de notre cadre de travail sont qu'il est *générique, cyclique et fractal*. Notre approche est générique, car les secteurs à « photographier » peuvent être définis librement soit techniquement, soit fonctionnellement tout comme la caractéristique réseau décrivant l'activité, cyclique, car le processus de détection est

répété périodiquement afin de détecter les variations de l'activité et fractal, car la couverture/taille d'un secteur est également personnalisable. L'idée sous-jacente à notre framework et ses caractéristiques est tirée du concept de la "roue de DEMING" [142], sur lequel sont basées les normes ISO 9001 (SMQ) [143] et 27001 (SMSI) [144]. Nos travaux de recherche ont été menés dans le cadre du projet CANCAN (voir section 2.7) avec les données éponymes non labellisées et anonymisées, décrites dans la section 2.7. Notre cadre de travail nous a permis de détecter des événements de grande ampleur comme des matchs de football ou encore l'incendie qui a ravagé Notre-Dame de Paris en avril 2019. D'un autre coté, bien qu'efficace, celui-ci nécessite une analyse visuelle des signatures numériques afin de détecter les variations d'activité et confirmer les anomalies.

Notre dernière contribution, la méthodologie "DiNATrA$\mathcal{X}$" détaillée dans le chapitre 5, est une évolution de notre framework. Dans le but de l'automatiser, nous avons introduit le concept de *DNA*. Pour chaque signature, nous calculons ce que nous avons appelé son DNA équivalent et son brin associé. Il s'agit de la signature numérique encodée en une chaîne de caractères et de la liste ordonnée des nucléotides. Le DNA est construit en concaténant les séquences caractérisant chacun des clusters constituant la signature numérique. Chaque séquence est constituée de nucléotides codifiant l'activité. Pour chacun des SOI (secteur d'intérêt), nous générons périodiquement sa signature numérique et calculons son DNA et le brin. Pour détecter une variation de l'activité et donc une potentielle anomalie, nous mesurons la distance de DAMERAU-LEVENSTEIN ainsi que la *sévérité* entre chaque DNA et brin adjacent. À partir de ces mesures, nous déterminons le niveau de variation et en déduisons si une alerte doit être levée ou pas. L'idée des DNAs nous est venue non pas suite à une chute dans les toilettes comme pour 'Doc' et son convecteur temporel [145], [146] mais suite à la diffusion du film "Bienvenue à Gattaca" [147], [148] dont le nom du centre de recherches spatiales est construit à partir des différents nucléotides composant l'ADN humain. DiNATrA$\mathcal{X}$ a été mis en œuvre et testé avec les jeux de données CANCAN et CTU-13. L'ensemble de nos résultats est présenté dans le chapitre 6. Notre méthodologie nous a permis de détecter les grands événements cités précédemment, des cérémonies funéraires en analysant des SOI de plus petites dimensions et une panne impactant l'ensemble des équipements réseaux d'`Orange`. Nous avons également été à même de détecter les anomalies réseaux générées par l'activité intrinsèque de différents botnets ainsi que les prémices de leurs différentes attaques.

7.3 Perspectives

Notre approche par "Forêt combinée" est efficace à condition de disposer de modèles capables de détecter les différentes familles de botnets. Pour ce faire, il faut disposer d'un jeu de données labellisées pour pouvoir entraîner un algorithme supervisé et générer le modèle. Il pourrait être intéressant d'amender le CTU-13, conçu en 2011, avec des scénarios créés à partir de botnets récents. Ces botnets pourraient être constitués d'équipements plus modernes comme des téléphones ou bien issus de l'IoT. Une autre solution pourrait être d'essayer de créer ce *jeu de données idéal* dont nous avons proposé une description dans la section 2.8.2. Nous avons également expliqué que notre framework pouvait permettre de classifier les secteurs en fonction de leur signature numérique. Nous avons par exemple été capable de catégoriser des signatures et donc des secteurs comme étant caractéristiques d'une gare. Cette idée de catégorisation des secteurs en fonction de l'activité qui y est générée et par conséquent de leur signature numérique pourrait être étendue à d'autres catégories. Cette idée pourrait être testée sur le jeu de donnée MAWI référencé dans le tableau 2.3.

Nous sommes convaincus par le principe mis en œuvre par DiNATrA$\mathcal{X}$ consistant à comparer périodiquement les DNAs de différents SOI pour détecter des changements soit de l'activité réseau soit du comportement des utilisateurs. Cette idée pourrait être transposée à d'autres jeux de données pour détecter d'autres types d'attaques ou d'événements. Nous avions envisagé par exemple d'employer notre méthodologie pour matérialiser ou suivre des mouvements de foules. Pour ce faire, la formule de calcul de la *Sévérité* nécessiterait d'autres recherches, car sa définition ne donne pas entière satisfaction. En effet, les écarts qu'elle quantifie ne sont pas toujours très représentatifs. Par conséquent, cette notion mériterait d'être mieux définie.

Enfin, une autre piste d'étude envisageable est l'application de notre méthodologie DiNATrA$\mathcal{X}$ à la détection des événements rares. Nous avons abordé et présenté la théorie autour de ce concept dans la section 1.4. Pour ce faire, lors de la construction des brins, les nucléotides pourraient être ordonnés de façon croissante, c'est-à-dire de celui représentant le type d'activité le plus faible vers le plus important. Puis, les brins seraient assemblés également dans un ordre croissant lors de la constitution des DNAs faisant ressortir de la sorte les activités avec la plus faible représentation.

Bibliographie

[1] C. Maudoux et S. Boumerdassi, « Unsupervised Anomaly Knowledge Flow : A Digital Signatures Extraction Approach », in *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, oct. 2023, p. 1-6. doi : 10.1109/WINCOM59760.2023.10323022. adresse : https://ieeexplore.ieee.org/abstract/document/10323022 (visité le 24/12/2023).

[2] C. Maudoux et S. Boumerdassi, « Network Anomalies Detection by Unsupervised Activity Deviations Extraction », in *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, sept. 2022, p. 1-5. doi : 10.1109/GIIS56506.2022.9937022. adresse : https://ieeexplore.ieee.org/document/9937022 (visité le 24/12/2023).

[3] C. Maudoux et S. Boumerdassi, « LemonLDAP : :NG A Full AAA Free Open Source WebSSO Solution », in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, nov. 2022, p. 277-281. doi : 10.1109/CloudNet55617.2022.9978777. adresse : https://ieeexplore.ieee.org/abstract/document/9978777.

[4] C. Maudoux, S. Boumerdassi, A. Barcello et E. Renault, « Combined Forest : A New Supervised Approach for a Machine-Learning-based Botnets Detection », in *2021 IEEE Global Communications Conference (GLOBECOM)*, déc. 2021, p. 01-06. doi : 10.1109/GLOBECOM46510.2021.9685261.

[5] C. Maudoux et S. Boumerdassi, « Smart and Sustainable Agriculture », in *Smart and Sustainable Agriculture*, S. Boumerdassi, M. Ghogho et É. Renault, éd., sér. Communications in Computer and Information Science, Cham : Springer International Publishing, 2021, p. 103-121, isbn : 978-3-030-88259-4. doi : 10.1007/978-3-030-88259-4_8.

[6] C. Maudoux et S. Boumerdassi, *LemonLDAP : :NG - A Full AAA Free Open Source WebSSO Solution*, nov. 2022. doi : 10.1109/CloudNet55617.2022.9978777. adresse : https://hal.science/hal-03949890 (visité le 20/01/2024).

[7] X. Guimard, C. Oudot, C. Maudoux et M. Besson, « LemonLDAP : :NG », déc. 2010. adresse : https://hal.inria.fr/hal-03776592.

[8] B. Bodnar, *Une cyberattaque de hackers pro-russes perturbe le réseau de ministères français*, mars 2024. adresse : https://www.numerama.com/cyberguerre/1650508-une-cyberattaque-de-hackers-pro-russes-perturbe-le-reseau-de-ministeres-francais.html (visité le 13/03/2024).

[9] N. Goix, A. Sabourin et S. Clémençon, « Sparse Representation of Multivariate Extremes with Applications to Anomaly Detection », *Journal of Multivariate Analysis*, t. 161, p. 12-31, sept. 2017, issn : 0047-259X. doi : 10.1016/j.jmva.2017.06.010. adresse : https://www.sciencedirect.com/science/article/pii/S0047259X17304062 (visité le 31/03/2024).

[10] *What is a DDoS Attack ? DDoS Meaning, Definition & Types*. adresse : https://www.fortinet.com/resources/cyberglossary/ddos-attack (visité le 31/03/2024).

[11] *What is Click Fraud ? How it Works, Examples, and Red Flags | CHEQ*, jan. 2024. adresse : https://cheq.ai/blog/what-is-click-fraud/ (visité le 31/03/2024).

[12] *What is Ransomware ? How It Works and How to Remove It*. adresse : https://www.techtarget.com/searchsecurity/definition/ransomware (visité le 31/03/2024).

[13] *Spam | What is Spam ? | Definition & Types of Spam*. adresse : https://www.malwarebytes.com/spam (visité le 31/03/2024).

[14] *Secuser.Com - Virus ILoveYou*. adresse : http://www.secuser.com/alertes/2000/iloveyou.htm (visité le 27/03/2024).

[15] C. Boone, *Le "Pump and Dump" : Explication d'une technique de spamming rentable*, août 2006. adresse : https://www.snipemail.com/spamming/le-pump-and-dump-explication-technique-de-spamming-rentable.html (visité le 27/03/2024).

[16] J.-L. Goudet, *Spam pharmaceutique : l'internaute mord 1 fois sur 12,4 millions*. adresse : https://www.futura-sciences.com/tech/actualites/internet-spam-pharmaceutique-internaute-mord-1-fois-124-millions-17327/ (visité le 27/03/2024).

[17] *FBI Warns of Storm Worm Virus*. adresse : https://www.fbi.gov/news/pressrel/press-releases/fbi-warns-of-storm-worm-virus (visité le 27/03/2024).

[18] *Scam ou Nigérian-419 : définition du scamming*, fév. 2009. adresse : https://www.altospam.com/glossaire/scam-nigerian419/ (visité le 27/03/2024).

[19] *Cross Site Request Forgery (CSRF) | OWASP Foundation*. adresse : https://owasp.org/www-community/attacks/csrf (visité le 31/03/2024).

[20] *What is a Phishing Attack ? | IBM*. adresse : https://www.ibm.com/topics/phishing (visité le 31/03/2024).

[21] *Qu'est-ce que le malware MyDoom ? Genèse, fonctionnement et mesures de protection | Okta*. adresse : https://www.okta.com/fr/identity-101/mydoom/ (visité le 09/02/2024).

[22] *Stuxnet : qu'est-ce que c'est et comment agit-il ?* Adresse : https://www.avast.com/fr-fr/c-stuxnet (visité le 27/03/2024).

[23] *Qu'est-Ce Que Le Botnet Mirai ? | Avast*. adresse : https://www.avast.com/fr-fr/c-mirai (visité le 27/03/2024).

[24] *Cyberattaque "WannaCry"*, mai 2017. adresse : https://www.lexpress.fr/monde/vague-internationale-de-cyberattaques_1907798.html (visité le 12/01/2024).

[25] *TICsanté - Articles*. adresse : https://www.ticsante.com/story?ID=3521 (visité le 09/02/2024).

[26] A. Greenberg, « The Untold Story of NotPetya, the Most Devastating Cyberattack in History », *Wired*, issn : 1059-1028. adresse : https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/ (visité le 27/03/2024).

[27] *Equifax Data Breach Settlement*, juill. 2019. adresse : https://www.ftc.gov/enforcement/refunds/equifax-data-breach-settlement (visité le 12/01/2024).

[28] Y. Sahin, *How GitHub Survived the Biggest DDoS Attack Ever Recorded ?*, nov. 2021. adresse : https://medium.com/technology-hits/how-github-survived-the-biggest-ddos-attack-ever-recorded-6907ba6f5c98 (visité le 27/03/2024).

[29] *SolarWinds Hack Explained : Everything You Need to Know*. adresse : https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know (visité le 27/03/2024).

[30] *Cyberattaque chez Viamedis et Almerys : ce que l'on sait du vol de données de plus de 33 millions d'assurés en France*, fév. 2024. adresse : https://www.francetvinfo.fr/sante/cyberattaque-chez-viamedis-et-almerys-ce-que-l-on-sait-du-vol-de-donnees-de-plus-33-millions-d-assures-en-france_6352741.html (visité le 21/02/2024).

[31] *Numéro de sécu, mutuelle : 33 millions de Français victimes d'une cyberattaque au tiers payant*, fév. 2024. adresse : https://www.lesechos.fr/industrie-services/pharmacie-sante/numero-de-secu-mutuelle-33-millions-de-francais-victimes-dune-cyberattaque-au-tiers-payant-2074842 (visité le 21/02/2024).

[32] *Vol de données : plus de 33 millions de personnes concernées par des cyberattaques contre deux spécialistes du tiers payant, annonce la Cnil*, fév. 2024. adresse : https://www.francetvinfo.fr/sante/vol-de-donnees-plus-de-33-millions-de-personnes-concernees-par-des-cyberattaques-contre-deux-specialistes-du-tiers-payant-annonce-la-cnil.html (visité le 21/02/2024).

[33] *State of the Internet Reports | Akamai*. adresse : https://www.akamai.com/security-research/the-state-of-the-internet (visité le 31/03/2024).

[34] *2023 Ransomware Report : Sophos State of Ransomware*. adresse : https://www.sophos.com/en-us/content/state-of-ransomware (visité le 31/03/2024).

[35] *Malwarebytes report : ransomware attacks increased 68% in 2023 | Digital Watch Observatory*, fév. 2024. adresse : https://dig.watch/updates/malwarebytes-report-ransomware-attacks-increased-68-in-2023 (visité le 31/03/2024).

[36] *IoT Cyberattacks Escalate in 2021, According to Kaspersky*. adresse : https://www.iotworldtoday.com/security/iot-cyberattacks-escalate-in-2021-according-to-kaspersky (visité le 31/03/2024).

[37] *Erratic Phishing Volume Increases 28% in 2021*. adresse : https://www.phishlabs.com/blog/erratic-phishing-volume-increases-28-in-2021 (visité le 31/03/2024).

[38] J. Brownlee, *Master Machine Learning Algorithms : Discover How They Work and Implement Them*. Machine Learning Mastery, mars 2016.

[39] R. Bapat, A. Mandya, X. Liu et al., « Identifying Malicious Botnet Traffic Using Logistic Regression », in *2018 Systems and Information Engineering Design Symposium (SIEDS)*, Charlottesville, VA : IEEE, avr. 2018, p. 266-271, isbn : 978-1-5386-6343-1. doi : 10.1109/SIEDS.2018.8374749. adresse : https://ieeexplore.ieee.org/document/8374749/ (visité le 25/02/2021).

[40] Y. Zhang, D. Xiao et Y. Liu, « Automatic Identification Algorithm of the Rice Tiller Period Based on PCA and SVM », *IEEE Access*, t. 9, p. 86 843-86 854, 2021, ISSN : 2169-3536. DOI : 10.1109/ACCESS.2021.3089670.

[41] D. N. Bhargava, G. Sharma, D. R. Bhargava et M. Mathuria, « Decision Tree Analysis on J48 Algorithm for Data Mining », *International Journal of Advanced Research in Computer Science and Software Engineering*, p. 6, 2013, ISSN : 22776451, 2277128X.

[42] K. Singh, S. C. Guntuku, A. Thakur et C. Hota, « Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests », *Information Sciences*, t. 278, p. 488-497, sept. 2014, ISSN : 0020-0255. DOI : 10.1016/j.ins.2014.03.066. adresse : http://www.sciencedirect.com/science/article/pii/S0020025514003570 (visité le 18/03/2020).

[43] A. Feizollah, N. B. Anuar, R. Salleh, F. Amalina, R. R. Ma'arof et S. Shamshirband, « A Study Of Machine Learning Classifiers for Anomaly-Based Mobile Botnet Detection », *Malaysian Journal of Computer Science*, t. 26, n° 4, p. 251-265, déc. 2013, ISSN : 0127-9084. adresse : https://ejournal.um.edu.my/index.php/MJCS/article/view/6785 (visité le 12/04/2020).

[44] M. Stevanovic et J. M. Pedersen, « An Efficient Flow-Based Botnet Detection Using Supervised Machine Learning », in *ICIN*, Honolulu, HI, USA : IEEE, fév. 2014, p. 797-801, ISBN : 978-1-4799-2358-8. DOI : 10.1109/ICCNC.2014.6785439. adresse : http://ieeexplore.ieee.org/document/6785439/ (visité le 25/02/2021).

[45] S. Ghosh et S. Kumar, « Comparative Analysis of K-Means and Fuzzy C-Means Algorithms », *International Journal of Advanced Computer Science and Applications*, t. 4, n° 4, mai 2013, ISSN : 2158107X, 21565570. DOI : 10.14569/IJACSA.2013.040406. (visité le 25/02/2021).

[46] S. Wan et Y.-P. Wang, « The Comparison of Density-Based Clustering Approach among Different Machine Learning Models on Paddy Rice Image Classification of Multispectral and Hyperspectral Image Data », *Agriculture*, t. 10, n° 10, p. 465, oct. 2020. DOI : 10.3390/agriculture10100465. adresse : https://www.mdpi.com/2077-0472/10/10/465 (visité le 29/08/2021).

[47] B. Abraham, A. Mandya, R. Bapat, F. Alali, D. E. Brown et M. Veeraraghavan, « A Comparison of Machine Learning Approaches to Detect Botnet Traffic », in *IJCNN*, Rio de Janeiro : IEEE, juill. 2018, p. 1-8. DOI : 10.1109/IJCNN.2018.8489096. adresse : https://ieeexplore.ieee.org/document/8489096/ (visité le 25/02/2021).

[48] E. S. C. Vilaça, T. P. B. Vieira, R. T. de Sousa et J. P. C. L. da Costa, « Botnet Traffic Detection Using RPCA and Mahalanobis Distance », in *WCNPS*, Brasilia, Brazil : IEEE, oct. 2019, p. 1-6. DOI : 10.1109/WCNPS.2019.8896228. adresse : https://ieeexplore.ieee.org/document/8896228/.

[49] *The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic.* adresse : https://www.stratosphereips.org/datasets-ctu13 (visité le 14/06/2020).

[50] D. C. Le, A. Nur Zincir-Heywood et M. I. Heywood, « Data Analytics on Network Traffic Flows for Botnet Behaviour Detection », in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, Greece : IEEE, déc. 2016, p. 1-7, ISBN : 978-1-5090-4240-1. DOI : 10.1109/SSCI.2016.7850078. adresse : http://ieeexplore.ieee.org/document/7850078/.

[51] J. Kim, A. Sim, J. Kim et K. Wu, « Botnet Detection Using Recurrent Variational Autoencoder », in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, déc. 2020, p. 1-6. DOI : 10.1109/GLOBECOM42002.2020.9348169. adresse : https://ieeexplore.ieee.org/document/9348169 (visité le 29/03/2024).

[52] A. Blaise, M. Bouet, V. Conan et S. Secci, « BotFP : FingerPrints Clustering for Bot Detection », in *IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary : IEEE, avr. 2020. DOI : 10.1109/NOMS47738.2020.9110420. adresse : https://hal.archives-ouvertes.fr/hal-02501912 (visité le 10/04/2021).

[53] B. M. Rahal, A. Santos et M. Nogueira, « A Distributed Architecture for DDoS Prediction and Bot Detection », *IEEE Access*, t. 8, p. 159 756-159 772, 2020, ISSN : 2169-3536. DOI : 10.1109/ACCESS.2020.3020507.

[54] *Content and Context based Adaptation in Mobile Networks*. adresse : https://anr.fr/Project-ANR-18-CE25-0011 (visité le 09/04/2022).

[55] *RGPD : de quoi parle-t-on ?* Adresse : https://www.cnil.fr/fr/rgpd-de-quoi-parle-t-on (visité le 30/08/2023).

[56] « Code de conduite 2020 de la qualité de service d'internet 14/09/2020 »,

[57] *KDD Cup 1999 Data*. adresse : http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (visité le 29/03/2024).

[58] N. Araújo, R. de Oliveira, E. Ferreira, A. A. Shinoda et B. Bhargava, « Identifying Important Characteristics in the KDD99 Intrusion Detection Dataset by Feature Selection Using a Hybrid Approach », in *2010 17th International Conference on Telecommunications*, avr. 2010, p. 552-558. DOI : 10.1109/ICTel.2010.5478852. adresse : https://ieeexplore.ieee.org/document/5478852 (visité le 29/03/2024).

[59] *NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. adresse : https://www.unb.ca/cic/datasets/nsl.html (visité le 29/03/2024).

[60] *A Detailed Analysis of the KDD CUP 99 Data Set | IEEE Conference Publication | IEEE Xplore*. adresse : https://ieeexplore.ieee.org/document/5356528 (visité le 29/03/2024).

[61] *CRAWDAD : A Community Resource for Archiving Wireless Data at Dartmouth | Request PDF*. adresse : https://www.researchgate.net/publication/3437123_CRAWDAD_A_Community_Resource_for_Archiving_Wireless_Data_at_Dartmouth (visité le 29/03/2024).

[62] J. Soto, M. Nogueira et K. R. Chowdhury, « Resilient and Multi-Dimensional Cooperative Spectrum Sensing on Cognitive Radio Networks », in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, sept. 2013, p. 2533-2538. DOI : 10.1109/PIMRC.2013.6666573. adresse : https://ieeexplore.ieee.org/document/6666573 (visité le 29/03/2024).

[63] V. Shirsath, *CAIDA UCSD DDoS 2007 Attack Dataset*, mai 2023. adresse : https://ieee-dataport.org/documents/caida-ucsd-ddos-2007-attack-dataset (visité le 20/03/2024).

[64] D. Zhao, I. Traore, B. Sayed et al., « Botnet detection based on traffic behavior analysis and flow intervals », *Computers & Security*, 27th IFIP International Information Security Conference, t. 39, p. 2-16, nov. 2013, ISSN : 0167-4048. DOI : 10.1016/j.cose.2013.04.007. adresse : http://www.sciencedirect.com/science/article/pii/S0167404813000837 (visité le 18/03/2020).

[65] *Android Malware 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. adresse : https://www.unb.ca/cic/datasets/andmal2017.html (visité le 20/03/2024).

[66] E. Karbab, M. Debbabi, A. Derhab et D. Mouheb, « MalDozer : Automatic Framework for Android Malware Detection Using Deep Learning », *Digital Investigation*, t. 24, S48-S59, mars 2018. doi : 10.1016/j.diin.2018.01.007.

[67] N.-E.-H. Yellas, S. Boumerdassi, A. Ceselli, B. Maaz et S. Secci, « Robust Access Point Clustering in Edge Computing Resource Optimization », *IEEE Transactions on Network and Service Management*, t. 19, n° 3, p. 2738-2750, sept. 2022, issn : 1932-4537, 2373-7379. doi : 10.1109/TNSM.2022.3186856. adresse : https://ieeexplore.ieee.org/document/9810020/ (visité le 29/03/2024).

[68] *Mawi Working Group Traffic Archive*. adresse : https://mawi.wide.ad.jp/mawi/ (visité le 14/06/2020).

[69] O. Salem, A. Makke, J. Tajer et A. Mehaoua, « Flooding Attacks Detection in Traffic of Backbone Networks », in *2011 IEEE 36th Conference on Local Computer Networks*, Bonn, Germany : IEEE, oct. 2011, p. 441-449, isbn : 978-1-61284-928-7 978-1-61284-926-3 978-1-61284-927-0. doi : 10.1109/LCN.2011.6115504. adresse : http://ieeexplore.ieee.org/document/6115504/.

[70] K. Allix, T. F. Bissyandé, J. Klein et Y. Le Traon, « AndroZoo : collecting millions of Android apps for the research community », in *Proceedings of the 13th International Conference on Mining Software Repositories*, Austin Texas : ACM, mai 2016, p. 468-471, isbn : 978-1-4503-4186-8. doi : 10.1145/2901739.2903508. adresse : https://dl.acm.org/doi/10.1145/2901739.2903508 (visité le 29/03/2024).

[71] L. Li, A. Bartel, T. F. Bissyande et al., « IccTA : Detecting Inter-Component Privacy Leaks in Android Apps », in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy : IEEE, mai 2015, p. 280-291, isbn : 978-1-4799-1934-5. doi : 10.1109/ICSE.2015.48. adresse : http://ieeexplore.ieee.org/document/7194581/ (visité le 29/03/2024).

[72] *IDS 2012 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. adresse : https://www.unb.ca/cic/datasets/ids.html (visité le 20/03/2024).

[73] *Botnet 2014 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. adresse : https://www.unb.ca/cic/datasets/botnet.html (visité le 20/03/2024).

[74] A. Guirakhoo, *FBI IC3 2019 : Cybercrime results in over $3.5 billion in reported losses | Digital Shadows*, mars 2020. adresse : https://www.digitalshadows.com/blog-and-research/fbi-ic3-2019-cybercrime-results-in-over-3-5-billion-in-reported-losses/ (visité le 12/11/2020).

[75] M. Kumar, *French Police Remotely Removed RETADUP Malware from 850,000 Infected PCs*, Article, août 2019. adresse : https://thehackernews.com/2019/08/retadup-botnet-malware.html (visité le 15/11/2020).

[76] E. C. Ogu, O. A. Ojesanmi, O. Awodele et 'S. Kuyoro, « A Botnets Circumspection : The Current Threat Landscape, and What We Know So Far », *Information*, t. 10, n° 11, p. 337, oct. 2019, issn : 2078-2489. doi : 10.3390/info10110337. adresse : https://www.mdpi.com/2078-2489/10/11/337 (visité le 08/11/2020).

[77] S. García, M. Grill, J. Stiborek et A. Zunino, « An empirical comparison of botnet detection methods », *Computers & Security*, t. 45, p. 100-123, sept. 2014, ISSN : 0167-4048. DOI : 10.1016/j.cose.2014.05.011. adresse : http://www.sciencedirect.com/science/article/pii/S0167404814000923 (visité le 18/03/2020).

[78] *Machine Learning Project at the University of Waikato in New Zealand*. adresse : http://old-www.cms.waikato.ac.nz/ml/index.html (visité le 05/05/2022).

[79] *Cmaudoux / Parsers / Basic.Pl — Bitbucket*. adresse : https://bitbucket.org/cmaudoux/parsers/src/master/basic.pl (visité le 04/03/2024).

[80] *Index of /publicDatasets/CTU-Malware-Capture-Botnet-42*. adresse : https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-42/ (visité le 04/03/2024).

[81] *Stratosphere Testing Framework*. adresse : https://www.stratosphereips.org/stratosphere-testing-framework (visité le 17/01/2021).

[82] *Example of Using STF for Detecting C&C channels. An analysis of Pushdo malware.* adresse : https://www.stratosphereips.org/blog/2014/03/9/example-of-using-stf-for-detecting-cc-channels (visité le 15/01/2021).

[83] *Cmaudoux / Parsers / Advanced.Pl — Bitbucket*. adresse : https://bitbucket.org/cmaudoux/parsers/src/master/advanced.pl (visité le 06/03/2024).

[84] *AdaBoostM1*. adresse : https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/AdaBoostM1.html (visité le 04/03/2024).

[85] *Bagging*. adresse : https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/Bagging.html (visité le 03/03/2024).

[86] *IterativeClassifierOptimizer*. adresse : https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/IterativeClassifierOptimizer.html (visité le 04/03/2024).

[87] Y. Freund et R. Schapire, « Experiments with a New Boosting Algorithm », in *International Conference on Machine Learning*, juill. 1996. adresse : https://www.semanticscholar.org/paper/Experiments-with-a-New-Boosting-Algorithm-Freund-Schapire (visité le 10/01/2024).

[88] *Vote*. adresse : https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/Vote.html (visité le 04/03/2024).

[89] *Stacking*. adresse : https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/Stacking.html (visité le 04/03/2024).

[90] D. Wolpert, « Stacked Generalization », *Neural Networks*, t. 5, n° 2, p. 241-259, déc. 1992, ISSN : 08936080. DOI : 10.1016/S0893-6080(05)80023-1. adresse : https://linkinghub.elsevier.com/retrieve/pii/S0893608005800231.

[91] J. Russell, *The world's largest DDoS attack took GitHub offline for fewer than 10 minutes*, mars 2018. adresse : https://social.techcrunch.com/2018/03/02/the-worlds-largest-ddos-attack-took-github-offline-for-less-than-tens-mn (visité le 15/11/2020).

[92] *MyDoom, Un Virus Qui Fait Très Mail. – Libération*. adresse : https://www.liberation.fr/futurs/2004/02/03/mydoom-un-virus-qui-fait-tres-mail_467532/ (visité le 12/01/2024).

[93] W. Song, M. Beshley, K. Przystupa, H. Beshley, O. Kochan et A. Pryslupskyi, « Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection », *Sensors*, t. 20, p. 1637, mars 2020. doi : 10.3390/s20061637.

[94] S. B. Wankhede, « Anomaly Detection Using Machine Learning Techniques », in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, mars 2019, p. 1-3. doi : 10.1109/I2CT45611.2019.9033532.

[95] M. Szmit, R. Wezyk et M. Skowronski, « Traffic Anomaly Detection with Snort », jan. 2007.

[96] O. Podzins et A. Romanovs, « Why SIEM Is Irreplaceable in a Secure IT Environment ? », avr. 2019, p. 1-5. doi : 10.1109/eStream.2019.8732173.

[97] A. Guezzaz, Y. Asimi, M. Azrour et A. Asimi, « Mathematical Validation of Proposed Machine Learning Classifier for Heterogeneous Traffic and Anomaly Detection », *Big Data Mining and Analytics*, t. 4, n° 1, p. 18-24, mars 2021, issn : 2096-0654. doi : 10.26599/BDMA.2020.9020019.

[98] W. Wang, Z. Wang, Z. Zhuo, H. Deng, W. Zhao et C. Wang, « Anomaly Detection of Industrial Control Systems Based on Transfer Learning », *Tsinghua Science and Technology*, t. 26, n° 6, oct. 2021, issn : 1007-0214. doi : 10.26599/TST.2020.9010041.

[99] N. Ghosh, K. Maity, R. Paul et S. Maity, « Outlier Detection in Sensor Data Using Machine Learning Techniques for IoT Framework and Wireless Sensor Networks », in *2019 International Conference on Applied Machine Learning (ICAML)*, mai 2019, p. 187-190. doi : 10.1109/ICAML48257.2019.00043.

[100] A. Dawoud, S. Shahristani et C. Raun, « Deep Learning for Network Anomalies Detection », in *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, déc. 2018, p. 149-153. doi : 10.1109/iCMLDE.2018.00035.

[101] M. A. Kabir et X. Luo, « Unsupervised Learning for Network Flow Based Anomaly Detection in the Era of Deep Learning », in *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, août 2020, p. 165-168. doi : 10.1109/BigDataService49289.2020.00032.

[102] *Geohash Intro | Big Fast Blog*, jan. 2012. adresse : https://web.archive.org/web/20120112004608 (visité le 06/05/2022).

[103] D. Pelleg et A. Moore, « X-Means : Extending K-means with Efficient Estimation of the Number of Clusters », *Machine Learning*, jan. 2002.

[104] *Detect Outlier (LOF) - RapidMiner Documentation*. adresse : https://docs.rapidminer.com/latest/studio/operators/cleansing/outliers/detect_outlier_lof.html (visité le 29/03/2022).

[105] *Cmaudoux / Digital-Signatures-Extraction*. adresse : https://bitbucket.org/cmaudoux/digital-signatures-extraction/src/master/ (visité le 06/05/2022).

[106] *Geohash - Great All in One Geohash Library - Metacpan.Org*. adresse : https://metacpan.org/pod/Geohash (visité le 05/05/2022).

[107] I. H. Witten, E. Frank, M. A. Hall et C. J. Pal, *Data Mining : Practical Machine Learning Tools* (MK Series in Data Management Systems). San Diego, CA, USA : Morgan Kaufmann, oct. 2016, isbn : 978-0-12-804357-8.

[108] *Retour sur l'incendie de Notre-Dame de Paris*. adresse : https://revivre-notre-dame.fr/incendie-cathedrale-notre-dame/incendie-notre-dame-de-paris/ (visité le 25/08/2023).

[109] A. Jung et I. Baranov, *Basic Principles of Clustering Methods*, déc. 2019. doi : 10.48550/arXiv.1911.07891. arXiv : 1911.07891 [cs, stat]. adresse : http://arxiv.org/abs/1911.07891 (visité le 18/08/2023).

[110] F. Iglesias et T. Zseby, « Analysis of network traffic features for anomaly detection », *Machine Learning*, t. 101, n° 1, p. 59-84, oct. 2015, issn : 1573-0565. doi : 10.1007/s10994-014-5473-9. adresse : https://doi.org/10.1007/s10994-014-5473-9 (visité le 01/08/2023).

[111] J. Brownlee, *Machine Learning Mastery With Python : Understand Your Data, Create Accurate Models, and Work Projects End-to-End*. Machine Learning Mastery, avr. 2016.

[112] A. Zheng et A. Casari, *Feature Engineering for Machine Learning : Principles and Techniques for Data Scientists*. O'Reilly, 2018, isbn : 978-1-4919-5324-2.

[113] M. N. Lima, A. L. dos Santos et G. Pujolle, « A Survey of Survivability in Mobile Ad Hoc Networks », *IEEE Communications Surveys & Tutorials*, t. 11, n° 1, p. 66-77, 2009, issn : 1553-877X. doi : 10.1109/SURV.2009.090106.

[114] A. Lakhina, M. Crovella et C. Diot, « Characterization of Network-Wide Anomalies in Traffic Flows », Boston University Computer Science Department, Technical Report, mai 2004. adresse : https://open.bu.edu/handle/2144/1546 (visité le 02/08/2023).

[115] M. L. Proença, C. Coppelmans, M. Bottoli, A. Alberti et L. S. Mendes, « The Hurst Parameter for Digital Signature of Network Segment », in *Telecommunications and Networking - ICT 2004*, J. N. de Souza, P. Dini et P. Lorenz, éd., sér. Lecture Notes in Computer Science, Berlin, Heidelberg : Springer, 2004, p. 772-781, isbn : 978-3-540-27824-5. doi : 10.1007/978-3-540-27824-5_103.

[116] M. Proenca, B. Zarpelao et L. Mendes, « Anomaly Detection for Network Servers Using Digital Signature of Network Segment », in *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*, juill. 2005, p. 290-295. doi : 10.1109/AICT.2005.26.

[117] P. Siyari, B. Dilkina et C. Dovrolis, « Lexis : An Optimization Framework for Discovering the Hierarchical Structure of Sequential Data », fév. 2016.

[118] S. kulkarni, *Jaro Winkler vs Levenshtein Distance*, mars 2021. adresse : https://srinivas-kulkarni.medium.com/jaro-winkler-vs-levenshtein-distance-2eab21832fd6 (visité le 26/01/2024).

[119] F. J. Damerau, « A Technique for Computer Detection and Correction of Spelling Errors », *Communications of the ACM*, t. 7, n° 3, p. 171-176, mars 1964, issn : 0001-0782. doi : 10.1145/363958.363994. adresse : https://dl.acm.org/doi/10.1145/363958.363994 (visité le 02/08/2023).

[120] C. Zhao et S. Sahni, « String Correction Using the Damerau-Levenshtein Distance », *BMC Bioinformatics*, t. 20, n° 11, p. 277, juin 2019, issn : 1471-2105. doi : 10.1186/s12859-019-2819-0. adresse : https://doi.org/10.1186/s12859-019-2819-0 (visité le 02/08/2023).

[121] C. Maudoux et S. Boumerdassi, *DiNATraX_CANCAN*. adresse : https://kaggle.com/code/christophemaudoux/dinatrax-cancan (visité le 20/01/2024).

[122] U. Angkhawey et V. Muangsin, « Detecting Points of Interest in a City from Taxi GPS with Adaptive DBSCAN », in *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, juill. 2018, p. 1-6. doi : 10.1109/ICT-ISPC.2018.8523966.

[123] G. Boeing, *Clustering to Reduce Spatial Data Set Size*, mars 2018. doi : 10.31235/osf.io/nzhdc. adresse : https://osf.io/preprints/socarxiv/nzhdc/ (visité le 04/08/2023).

[124] *Sklearn.cluster.DBSCAN*. adresse : https://scikit-learn/stable/modules/generated/sklearn.cluster.DBSCAN.html (visité le 04/08/2023).

[125] N. Bhatia et Vandana, *Survey of Nearest Neighbor Techniques*, juill. 2010. doi : 10.48550/arXiv.1007.0085. arXiv : 1007.0085 [cs]. adresse : http://arxiv.org/abs/1007.0085 (visité le 04/08/2023).

[126] Z. Guba, *An Introduction to Python List Comprehension with Pandas*, juin 2022. adresse : https://medium.com/geekculture/python-list-comprehension-with-pandas-6514557babc6 (visité le 05/08/2023).

[127] E. H. M. Pena, M. V. O. de Assis et M. L. Proença, « Anomaly Detection Using Forecasting Methods ARIMA and HWDS », in *2013 32nd International Conference of the Chilean Computer Science Society (SCCC)*, nov. 2013, p. 63-66. doi : 10.1109/SCCC.2013.18.

[128] E. H. M. Pena, S. Barbon, J. J. P. C. Rodrigues et M. L. Proença, « Anomaly Detection Using Digital Signature of Network Segment with Adaptive ARIMA Model and Paraconsistent Logic », in *2014 IEEE Symposium on Computers and Communications (ISCC)*, juin 2014. doi : 10.1109/ISCC.2014.6912503.

[129] J. Blakes, O. Raz, U. Feige et al., « Heuristic for Maximizing DNA Reuse in Synthetic DNA Library Assembly », *ACS Synthetic Biology*, t. 3, n° 8, p. 529-542, août 2014. doi : 10.1021/sb400161v. adresse : https://doi.org/10.1021/sb400161v (visité le 06/08/2023).

[130] P. Siyari, B. Dilkina et C. Dovrolis, *Emergence and Evolution of Hierarchical Structure in Complex Systems*, août 2018. doi : 10.48550/arXiv.1805.04924. arXiv : 1805.04924 [cs, stat]. adresse : http://arxiv.org/abs/1805.04924 (visité le 06/08/2023).

[131] P. Siyari, B. Dilkina et C. Dovrolis, *Evolution of Hierarchical Structure & Reuse in iGEM Synthetic DNA Sequences*, juin 2019. doi : 10.48550/arXiv.1906.02446. arXiv : 1906.02446 [cs, stat]. adresse : http://arxiv.org/abs/1906.02446 (visité le 06/08/2023).

[132] C. Varol et H. M. T. Abdulhadi, « Comparision of String Matching Algorithms on Spam Email Detection », in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, déc. 2018, p. 6-11. doi : 10.1109/IBIGDELFT.2018.8625317.

[133] J. Auskalnis, N. Paulauskas et A. Baskys, « Application of Local Outlier Factor Algorithm to Detect Anomalies in Computer Network », *Elektronika ir Elektrotechnika*, t. 24, juin 2018. doi : 10.5755/j01.eie.24.3.20972.

[134] N. Paulauskas, « Local outlier factor use for the network flow anomaly detection », *Security and Communication Networks*, t. 8, n° 18, p. 4203-4212, 2015, issn : 1939-0122. doi : 10.1002/sec.1335. adresse : https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1335 (visité le 06/08/2023).

[135] *Sklearn.neighbors.LocalOutlierFactor*. adresse : https://scikit-learn/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html (visité le 14/08/2023).

[136] A. Pathak, *11 meilleurs outils SIEM pour sécuriser votre organisation contre les cyberattaques*, juill. 2021. adresse : https://geekflare.com/fr/best-siem-solutions/ (visité le 25/08/2023).

[137] *Evènements passés*. adresse : https://www.stadefrance.com/fr/billetterie/archives (visité le 25/08/2023).

[138] *Obsèques de Dick Rivers*, mai 2019. adresse : https://www.leparisien.fr/culture-loisirs/musique/obseques-de-dick-rivers-c-etait-le-plus-rockeur-des-rockeurs.php (visité le 25/08/2023).

[139] M. Turcan, *Panne chez Orange : des problèmes de connexion Internet et 4G sur toute la France*, avr. 2019. adresse : https://www.numerama.com/tech/panne-4g-orange.html (visité le 25/08/2023).

[140] *CIDR : what is classless inter-domain routing ?*, août 2019. adresse : https://www.ionos.com/digitalguide/server/know-how/cidr-classless-inter-domain-routing/ (visité le 26/12/2023).

[141] C. Maudoux et S. Boumerdassi, *DiNATraX_CTU13*. adresse : https://kaggle.com/code/christophemaudoux/dinatrax-ctu13 (visité le 20/01/2024).

[142] *La roue de deming ou méthode PDCA : définition et étapes*, oct. 2018. adresse : https://www.supplychaininfo.eu/la-roue-de-deming/ (visité le 19/03/2024).

[143] 14 :00-17 :00, *ISO 9001 :2015*. adresse : https://www.iso.org/fr/standard/62085.html (visité le 02/04/2024).

[144] 14 :00-17 :00, *ISO/IEC 27001 :2022*. adresse : https://www.iso.org/fr/standard/27001 (visité le 02/04/2024).

[145] R. Zemeckis, *Back to the Future*, Aventure, Comédie, Science-fiction, oct. 1985.

[146] *Back to the Future - Activision (1986)*. adresse : https://www.cpc-power.com/index.php?page=detail&num=343 (visité le 02/04/2024).

[147] A. Niccol, *GATTACA*, Drame, Science-fiction, Thriller, avr. 1998.

[148] « 'Bienvenue à Gattaca', sur France 5 : une dystopie eugéniste devenue un classique du cinéma d'anticipation », *Le Monde.fr*, oct. 2022. adresse : https://www.lemonde.fr/culture/article/2022/10/07/bienvenue-a-gattaca-sur-france-5-une-dystopie-eugeniste-devenue-un-classique-du-cinema-d-anticipation_6144900_3246.html (visité le 19/03/2024).

[149] *TensorFlow*. adresse : https://www.tensorflow.org/?hl=fr (visité le 27/03/2024).

[150] *Keras : Deep Learning for Humans*. adresse : https://keras.io/ (visité le 27/03/2024).

[151] *PyTorch*. adresse : https://pytorch.org/ (visité le 27/03/2024).

[152] *XGBoost Documentation — Xgboost 2.0.3 Documentation*. adresse : https://xgboost.readthedocs.io/en/stable/ (visité le 27/03/2024).

[153] *Welcome to LightGBM's Documentation ! — LightGBM 4.0.0 Documentation*. adresse : https://lightgbm.readthedocs.io/en/stable/ (visité le 27/03/2024).

[154] *MLlib | Apache Spark*. adresse : `https://spark.apache.org/mllib/` (visité le 27/03/2024).

[155] *Fastai - Welcome to fastai*. adresse : `https://docs.fast.ai/` (visité le 27/03/2024).

[156] *H2O.Ai | The Fastest, Most Accurate AI Cloud Platform*. adresse : `https://h2o.ai/` (visité le 27/03/2024).

[157] *Onnx/Onnx*, Open Neural Network Exchange, mars 2024. adresse : `https://github.com/onnx/onnx` (visité le 27/03/2024).

[158] *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. adresse : `https://www.cs.waikato.ac.nz/ml/weka/` (visité le 05/05/2022).

[159] *Machine Learning in R - Next Generation*. adresse : `https://mlr3.mlr-org.com/` (visité le 27/03/2024).

[160] *Scikit-Learn : Machine Learning in Python — Scikit-Learn 1.4.1 Documentation*. adresse : `https://scikit-learn.org/stable/` (visité le 27/03/2024).

Annexe A

Outils pour l'apprentissage automatique

A.1 Tour d'horizon

Le domaine de l'apprentissage automatique ou "Machine Learning" (ML) dispose d'une large gamme d'outils logiciels, chacun ayant ses spécificités et ses domaines d'application. Ci-dessous, nous listons les principaux outils et bibliothèques couramment utilisés :

`TensorFlow` [**149**] Développé par GOOGLE, il s'agit d'un des frameworks les plus populaires pour la création de réseaux de neurones profonds. Il offre une flexibilité et permet un passage à l'échelle pour la recherche expérimentale et la production.

`Keras` [**150**] Interface de haut niveau pour les réseaux de neurones, fonctionnant sur `TensorFlow`, `CNTK` et `Theano`. `Keras` est conçu pour être simple et intuitif pour les prototypes rapides de réseaux de neurones profonds.

`PyTorch` [**151**] Proposé par FACEBOOK, il est réputé pour sa facilité d'utilisation et sa flexibilité, surtout en matière de recherche et développement. Il est particulièrement apprécié pour ses capacités dynamiques de graphes de calcul.

`XGBoost` [**152**] Optimisation de l'algorithme de boosting de gradient, `XGBoost` est très puissant pour les applications où la performance de prédiction est critique.

`LightGBM` [**153**] Un autre outil de boosting de gradient, développé par MICROSOFT, qui se distingue par sa vitesse et son efficacité, surtout sur des ensembles de données de grande taille.

`Apache Spark MLlib` [**154**] Bibliothèque de machine learning pour `Apache Spark`, elle est destinée au traitement parallèle de grandes données et s'adapte bien aux environnements distribués.

`FastAI` [**155**] Construite sur `PyTorch, FastAI` vise à rendre l'apprentissage profond plus accessible

en simplifiant le code nécessaire pour construire des modèles complexes.

H2O [**156**] Plateforme open source pour le machine learning automatisé, H2O est particulièrement forte dans l'analyse prédictive et permet de travailler avec des données de grande dimension.

ONNX (Open Neural Network Exchange) [**157**] Format ouvert conçu pour représenter les modèles de machine learning, permettant leur échange entre différents frameworks pour faciliter le déploiement.

Weka [158] (Waikato Environment for Knowledge Analysis) Plateforme logicielle populaire pour le machine learning et l'analyse de données, elle est développée par l'Université de Waikato en Nouvelle-Zélande. Écrite en Java, elle est disponible gratuitement sous licence GNU General Public License.

MLR3 [**159**] Il s'agit d'un framework pour le machine learning en R, successeur du package MLR. Il vise à fournir un ensemble cohérent et facile à utiliser d'outils pour les tâches de machine learning, en améliorant la performance et l'expérience utilisateur par rapport à son prédécesseur.

Scikit-learn [**160**] Bibliothèque Python axée sur le machine learning classique. Idéale pour les tâches de classification, de régression, de regroupement et de réduction de dimensionnalité, elle est très appréciée pour sa simplicité et son efficacité pour les projets de machine learning.

Ces outils diffèrent par leur niveau d'abstraction, leur performance, leur facilité d'utilisation et les types de modèles qu'ils supportent. Le choix de l'outil dépend donc des besoins spécifiques de chacun, que ce soit pour la recherche, le développement de prototypes ou la mise en production de solutions de machine learning. Dans le cadre de nos travaux, nous avons choisi d'utiliser dans un premier Weka pour sa facilité de mise en œuvre. Ensuite, nous avons testé le langage R et sa bibliothèque dédiée à l'apprentissage automatique MLR3. Mais, malgré nos contributions à sa documentation et correction, nous sommes passés rapidement à Python et sa bibliothèque Scikit-learn pour sa puissance et ses performances sans commune mesure avec MLR3 qui, de surcroît, ne donnait pas entière satisfaction.

A.2 Java & Weka

A.2.1 Présentation

Weka est un logiciel libre développé par l'Université de Waikato en Nouvelle-Zélande qui fournit une collection complète d'outils de visualisation et d'algorithmes pour l'analyse des données et la

modélisation prédictive. `Weka` est conçu pour être accessible aux utilisateurs non spécialistes en informatique, tout en offrant de puissantes fonctionnalités de machine learning. En effet, ce logiciel propose une interface graphique intuitive qui permet aux utilisateurs de facilement charger des ensembles de données, d'appliquer des algorithmes de machine learning et de visualiser les résultats sans nécessiter de programmation. Il fournit une large collection d'algorithmes pour la classification, la régression, le regroupement (clustering) et l'association, permettant ainsi de traiter divers types de problèmes de machine learning. De plus, de nombreux outils pour le filtrage et la transformation des données sont disponibles facilitant leur préparation avant l'application des algorithmes. Enfin, `Weka` intègre des fonctionnalités d'évaluation de la performance des modèles générés, y compris des méthodes de validation croisée et des matrices de confusion.

Grâce à son architecture modulaire, il peut être étendu en ajoutant de nouveaux algorithmes ou méthodes de traitement des données et permet d'importer des données à partir de fichiers au format CSV, ARFF (format spécifique à Weka), et d'autres bases de données via des connecteurs JDBC.

Weka est utilisée dans l'éducation, la recherche, et dans des applications industrielles pour diverses tâches d'apprentissage automatique comme la classification (identifier à quelle catégorie appartient une instance donnée), la régression (réduire une valeur numérique continue), le "clustering" (regrouper des instances similaires en clusters) ou encore la recherche de motifs fréquents (identifier des associations ou des corrélations intéressantes entre les attributs des données).

A.2.2 Avantages & Inconvénients

Bien que la suite logicielle `Weka` soit très puissante pour des tâches spécifiques de machine learning, elle présente certaines limitations, notamment à cause de sa performance qui peut être limitée sur de très grands ensembles de données, en raison de sa conception et de sa consommation mémoire élevée.

Étant une interface principalement graphique et écrite en Java, elle peut ne pas être la solution la plus efficace pour des pipelines de traitement de données hautement personnalisés ou pour des applications nécessitant une intégration étroite avec d'autres technologies de développement logiciel.

En résumé, Weka est un excellent outil pour débuter en machine learning ou pour des petits projets de recherche et d'éducation, grâce à sa facilité d'utilisation et à sa large gamme d'algorithmes intégrés, mais assez vite limité pour l'analyse de gros volume de données. Par conséquent, nous avons employé Weka pour nos premiers travaux puis le langage R et MLR3 pour les suivants.

A.2.3 Éco-système

Weka étant une suite logicielle à part entière, elle dispose de sa propre interface graphique permettant d'interagir avec l'ensemble de ses fonctionnalités via différents outils comme 'Explorer' qui permet d'étudier rapidement un ensemble de données ou bien 'Experimenter' qui est une interface facilitant la création de pipelines de traitements. Weka expose également une API très complète afin de piloter ou d'appeler ses fonctions ou méthodes depuis des programmes externes.

A.3 R & MLR3

A.3.1 Présentation

Le langage R est un environnement de programmation et un langage de script conçu spécifiquement pour les statistiques, la visualisation de données puis le machine learning. Il est très apprécié dans la communauté académique et parmi les professionnels de la data science pour sa flexibilité, sa puissance, et l'énorme quantité de paquets disponibles qui étendent ses fonctionnalités dans presque tous les domaines de la recherche quantitative et l'analyse de données. Contrairement à MatLab, R est un logiciel sous licence GNU GPL qui être utilisé, modifier et redistribuer librement. Le dépôt "Comprehensive R Archive Network" (CRAN) offre des milliers de paquets développés par la communauté pour diverses applications, y compris le machine learning, l'analyse statistique, la visualisation graphique, et bien plus. Il existe plusieurs environnements de développement pour R, tels que RStudio, qui offrent une interface utilisateur conviviale pour la programmation, la gestion de données et la visualisation. R est réputé pour ses capacités avancées de visualisation de données, avec des paquets comme 'ggplot2' permettant de créer des graphiques complexes et esthétiquement plaisants. De plus, il inclut de

nombreuses fonctions pour des analyses statistiques allant de la statistique descriptive aux modèles linéaires et non linéaires, tests statistiques et techniques de clustering. Enfin, pour pouvoir faciliter l'emploi de R en apprentissage automatique, a été développé la bibliothèque MLR3.

MLR3 est un framework dédié au machine learning en R, successeur du package MLR. Il vise à fournir un ensemble cohérent et facile à utiliser d'outils pour les tâches de machine learning, en améliorant la performance et l'expérience utilisateur par rapport à son prédécesseur. MLR3 simplifie la mise en œuvre du machine learning en R en fournissant une interface de plus haut niveau pour les différentes étapes d'un projet, telles que la préparation des données, la sélection du modèle, l'entraînement, la prédiction et l'évaluation. Il est construit autour d'une architecture modulaire, permettant aux utilisateurs de combiner facilement différents composants comme les prétraitements, les modèles, et les méthodes d'évaluation. MLR3 peut être étendu avec des paquets supplémentaires (comme 'mlr3learners', 'mlr3tuning' et 'mlr3pipelines') pour ajouter de nouveaux algorithmes de ML, des techniques d'optimisation des hyper-paramètres ou encore des méthodes pour construire des pipelines de traitement de données complexes. Il est optimisé pour la performance et gère efficacement les grandes données.

Ce framework est particulièrement utile pour l'expérimentation, car permettant de facilement comparer différents modèles de machine learning et techniques de prétraitement sur un ensemble de données, l'optimisation des performances des modèles en ajustant leurs hyper-paramètres ou la construction de séquences complexes de prétraitement et de modélisation qui peuvent être réutilisées et partagées.

MLR3 combine la flexibilité et la puissance du langage R avec une interface de haut niveau pour le machine learning, rendant l'expérimentation et le développement de modèles de machine learning plus accessibles et efficaces.

A.3.2 Avantages & Inconvénients

Le langage R et son framework MLR3 offrent plusieurs avantages pour les tâches de machine learning, mais ils présentent également certains inconvénients.

R bénéficie d'une large et active communauté d'utilisateurs et de développeurs. Il y a une abondance de ressources d'apprentissage, de forums, et de documentation. MLR3, en tant que partie de cet

écosystème, hérite de cette richesse de connaissances et de support. Avec des milliers de paquets disponibles sur le CRAN et d'autres dépôts, R offre une vaste gamme d'outils pour le prétraitement des données, l'analyse statistique, la visualisation et le machine learning, y compris MLR3 qui est spécifiquement conçu pour faciliter les tâches de machine learning. R est exceptionnel pour la visualisation de données grâce à des paquets comme 'ggplot2', ce qui est crucial pour l'analyse exploratoire des données, une étape importante dans les projets de machine learning. Avec R et son framework `MLR3` qui est très modulaire et extensible, il est possible de créer des "workflows" de machine learning complexes avec un effort relativement faible. Ils sont très populaires dans les milieux académiques pour l'enseignement et la recherche en statistiques et machine learning, en grande partie grâce à leur accessibilité et leur vaste fonctionnalité.

Par contre, bien que R ait fait des progrès significatifs en termes de performance, il peut encore être plus lent que d'autres langages, surtout pour les gros volumes de données ou les tâches de calcul intensif, bien que des solutions comme l'utilisation de paquets optimisés et le calcul parallèle puissent aider. Ceci est dû au fait que R charge les données en mémoire, ce qui peut limiter sa capacité à travailler avec de très grands ensembles de données sur des machines avec des ressources limitées. En outre, malgré une large communauté et de nombreuses ressources, la courbe d'apprentissage de R et de `MLR3` peut être abrupte pour les débutants, particulièrement pour ceux qui ne sont pas familiers avec la programmation ou les concepts statistiques. L'intégration de solutions développées en R dans des environnements de production peut être plus complexe comparée à d'autres langages plus orientés vers les applications, comme `Python`. Cela peut inclure des défis autour de la performance, de la gestion de la mémoire et de l'interopérabilité avec d'autres systèmes. En comparaison avec d'autres frameworks de machine learning, alors que MLR3 est puissant, des frameworks dans d'autres langages comme `TensorFlow`, `PyTorch` ou `Scikit-learn` en Python) peuvent offrir de meilleures performances et plus de flexibilité pour le machine learning, le deep learning et les réseaux de neurones complexes.

En conclusion, R et MLR3 sont des outils puissants pour le machine learning, particulièrement bien adaptés pour l'analyse statistique, la visualisation de données et la recherche. Cependant, selon le contexte d'utilisation, notamment pour le traitement de très grandes données ou pour des applications en production à grande échelle, d'autres outils pourraient être plus appropriés. Après une période d'essai avec R et MLR3, nous avons choisi d'utiliser `Python` et sa bibliothèque `Scikit-learn`.

A.3.3 Éco-système

Le langage R peut-être utilisé avec un IDE [1] open source comme `RStudio`. Il facilite la création de scripts R, l'exécution de commandes, l'observation des résultats graphiques et textuels, la gestion des fichiers de données et la visualisation de l'historique des commandes. Il est conçu pour rendre le travail avec R plus intuitif, surtout pour les utilisateurs moins familiers avec la ligne de commande.

A.4 Python & Scikit-learn

A.4.1 Présentation

`Python` est un langage de programmation de haut niveau, interprété, et polyvalent, qui a gagné une immense popularité dans divers domaines, particulièrement pour le développement web, l'automatisation, l'analyse de données, et le machine learning. Sa syntaxe claire et sa puissante en font un choix idéal pour les débutants en programmation, tandis que sa flexibilité et son large éventail de bibliothèques tierces attirent les développeurs expérimentés et les professionnels de la data science.

`Python` favorise une syntaxe claire et concise, rendant le code plus lisible et réduisant le coût de maintenance. Il peut être utilisé pour le développement de scripts simples, applications web complexes, analyse de données, visualisation et machine learning. Ensuite et surtout, `Python` dispose d'une communauté massive et active, offrant un vaste éventail de ressources d'apprentissage, de forums, et de documentation. Enfin, il offre une énorme collection de bibliothèques pour presque tous les besoins en programmation, particulièrement dans les domaines de l'analyse de données et du machine learning (par exemple, `NumPy, Pandas, Matplotlib, TensorFlow` et `Scikit-learn`).

`Scikit-learn` est une bibliothèque `Python` open source qui fournit une gamme d'outils simples et efficaces pour l'analyse de données et le machine learning. Elle est construite sur NumPy, SciPy, et Matplotlib, offrant une cohérence dans la manipulation des données et la visualisation. Elle inclut un large éventail d'algorithmes pour la classification, la régression, le clustering, la réduction de dimension, la sélection de modèle et le prétraitement des données. Avec une API cohérente, `Scikit-learn` rend l'application de techniques de machine learning aussi simple que possible. De plus, elle est réputée pour sa documentation détaillée et ses exemples de code, facilitant l'apprentissage et l'utilisation de ses fonctionnalités. Elle peut être utilisée avec d'autres bibliothèques de data science et de machine learning,

1. Environnement de Développement Intégré

permettant de créer des pipelines de traitement de données complexes et efficaces. Pour terminer, une vaste communauté de développeurs contribue régulièrement à l'amélioration de `Scikit-learn`, assurant le support pour les dernières techniques d'apprentissage automatique.

Scikit-learn est particulièrement adapté pour les projets de machine learning impliquant des ensembles de données de taille moyenne. Elle permet de prétraiter les données (normalisation, encodage, imputation,...), d'appliquer différents algorithmes de ML, d'évaluer les modèles à l'aide de divers métriques, d'optimiser les hyper-paramètres des modèles ou encore de construire des pipelines de traitement de données pour automatiser le workflow.

En conclusion, `Python` avec `Scikit-learn` forme une combinaison puissante pour le développement rapide et efficace de projets de machine learning. La simplicité et la flexibilité de Python, combinées à la richesse fonctionnelle et à la facilité d'utilisation de Scikit-learn, rendent cette combinaison particulièrement attrayante pour expérimenter des modèles de machine learning.

A.4.2 Avantages & Inconvénients

`Python` et sa bibliothèque `Scikit-learn` sont largement utilisés dans le domaine de la science des données et du ML pour leur simplicité, flexibilité et puissance. Cependant, comme tout ensemble d'outils, ils présentent à la fois des avantages et des inconvénients.

`Python` est connu pour sa syntaxe claire et sa lisibilité, ce qui facilite l'apprentissage et la compréhension du code pour les débutants et les professionnels. Langage polyvalent, il peut être utilisé pour une large gamme d'applications, des scripts simples aux systèmes complexes de back-end, en passant par l'analyse de données et le machine learning. En effet, `Python` bénéficie d'une vaste communauté de développeurs qui contribuent régulièrement à un large éventail de bibliothèques et de frameworks, offrant un support et des ressources précieuses. Utilisé dans de nombreux domaines scientifiques et techniques, il facilite la collaboration interdisciplinaire. Par contre, étant un langage interprété, `Python` peut être plus lent que des langages compilés comme C++ ou Java pour certaines tâches de calcul intensif, bien que cela puisse être atténué par l'utilisation de bibliothèques optimisées ou de `CPython`. En outre, `Python` gère la mémoire automatiquement, ce qui est pratique, mais peut entraîner une utilisation inefficace de la mémoire dans certains cas.

Scikit-learn offre une vaste collection d'algorithmes de ML, allant de la classification à la régression au clustering ou à la réduction de dimensionnalité. Son API est bien conçue, ce qui permet une intégration facile et une utilisation homogène de différents modèles et techniques. Scikit-learn dispose d'une documentation complète et de nombreux tutoriels et exemples, ce qui rend l'apprentissage et l'utilisation de la bibliothèque plus accessibles. Conçu pour fonctionner sans effort avec les bibliothèques NumPy et Pandas, Scikit-learn simplifie le traitement et la manipulation des données. Bien que très complet pour le machine learning traditionnel, Scikit-learn ne dispose pas de fonctionnalités intégrées pour le deep learning ou des architectures de réseaux de neurones complexes, pour lesquelles des bibliothèques comme TensorFlow ou PyTorch sont préférées. De plus, il peut être lent à intégrer les dernières avancées en machine learning, en partie à cause de son processus d'examen rigoureux et de son accent sur la stabilité et la facilité d'utilisation. Enfin, Scikit-learn peut ne pas être le meilleur choix pour travailler avec des ensembles de données très volumineux ou pour des calculs distribués à grande échelle, comparé à d'autres solutions comme Spark MLlib.

Python et Scikit-learn constituent un excellent point de départ pour la data science et le ML, offrant une combinaison équilibrée de facilité d'utilisation, de flexibilité, et de puissance. Cependant, pour des applications spécifiques nécessitant une performance maximale, une gestion fine de la mémoire ou des techniques de deep learning avancées, d'autres outils peuvent être plus appropriés. S'agissant de nos besoins, nous avons choisi d'utiliser ces derniers pour la réalisation de nos travaux.

A.4.3 Éco-système

Anaconda est une distribution gratuite et open source des langages Python et R pour la science des données et le machine learning. Elle vise à simplifier la gestion des paquets et le déploiement. Anaconda est construit autour de Conda (système de gestion de paquets) et comprend une collection de plus de 1 500 paquets scientifiques. Il est spécifiquement conçu pour la data science, offrant un accès facile à une large gamme d'outils spécialisés pour l'analyse de données, la visualisation, et le machine learning. De plus, il fournit un environnement cohérent pour le développement et l'exploration de projets de data science et une interface graphique utilisateur (Anaconda Navigator) qui permet de gérer les environnements Conda et de lancer des applications installées sans utiliser de commandes en ligne. Enfin, Anaconda simplifie l'installation de Python, des nombreux paquets scientifiques ou de machine learning, éliminant les difficultés liées à la gestion des dépendances. Il donne accès à

une riche bibliothèque de paquets pré-compilés pour la science des données, réduisant le besoin de compilation manuelle et facilite la création d'environnements isolés pour différents projets, aidant à gérer les dépendances et à éviter les conflits entre les paquets.

Pour résumer, `Conda` est un outil puissant pour la gestion des paquets et des environnements, offrant des fonctionnalités avancées qui dépassent la simple gestion des paquets. `Anaconda`, basé sur `Conda`, est spécifiquement conçu pour répondre aux besoins de la science des données, offrant un écosystème complet pour le développement, le test et le déploiement. Ensemble, ils forment une solution robuste pour gérer des projets complexes en science des données.

Le seul vrai bémol qui peut être reproché à `Conda` est sa lenteur. C'en est même frustrant tellement il est lent, surtout dans la résolution de dépendances complexes ou lors de l'installation de paquets à partir de canaux avec un grand nombre de paquets. Dans ce cas, `Mamba` offre une alternative beaucoup plus rapide. `Mamba` est un gestionnaire de paquets *open source et rapide* conçu pour améliorer l'expérience utilisateur de `Conda`, en particulier en termes de vitesse et de performance.

Tout comme `Conda, Mamba` est multi-plateformes et prend en charge plusieurs langages de programmation, permettant ainsi la gestion des environnements et des paquet. `Mamba` utilise le même format de paquets que `Conda` et peut directement interagir avec les dépôts `Conda` tels que `Anaconda`. L'un des principaux avantages de `Mamba` par rapport à `Conda` est sa vitesse de résolution de dépendances et d'installation de paquets, grâce à son implémentation en C++ pour les parties critiques du processus de résolution de paquets. Cela rend l'installation de paquets et la création d'environnements considérablement plus rapides. `Mamba` peut être installé dans un environnement `Conda` existant en utilisant `Conda` lui-même. Une fois installé, `Mamba` peut être utilisé à la place de `Conda` pour la plupart des opérations de gestion de paquets et d'environnements. Il est également le gestionnaire de paquets par défaut dans certaines distributions de `Conda` axées sur la performance, telles que `Mambaforge` ou `Miniforge`.

Pour ce qui est de l'utilisation de ces différents outils ; il existe `JupyterLab`, une interface utilisateur web interactive pour le projet `Jupyter`, permettant de créer et de gérer des documents de type NoteBook `Jupyter` ou autres comme des scripts `Python`, des fichiers 'Markdown', des images et des données. `JupyterLab` offre une expérience utilisateur extensible et modulaire, qui inclut des fonctionnalités de codage, de visualisation de données, de calcul numérique et de narration scientifique. C'est l'évolution de l'interface classique des NoteBooks `Jupyter`, avec une interface plus flexible et plus riche.

Pour conclure, `JupyterLab` est largement utilisé dans les domaines de la data science, de l'analyse de données, de la recherche scientifique ou de l'enseignement. Il représente une avancée significative par rapport à l'interface traditionnelle des NoteBooks Jupyter, offrant une expérience utilisateur plus riche, plus interactive et plus productive.

A.5 Conclusion

La comparaison entre `Weka` (`Java`), `MLR3` (pour R) et `Scikit-learn` (pour `Python`) peut être structurée autour de plusieurs critères clefs comme la performance, la facilité d'utilisation, la popularité, le nombre de contributeurs ou encore la facilité de déploiement. Chacun de ces outils a ses propres forces et faiblesses dans le domaine du ML, reflétant leurs langages de programmation sous-jacents, leurs philosophies de conception et leurs écosystèmes. Nous proposons, dans cette section, une étude comparative basée sur ces cinq critères.

A.5.1 Performance

Conçu principalement pour l'enseignement et la recherche, `Weka` fonctionne bien pour les ensembles de données de petite et moyenne taille. Sa performance peut être limitée par sa conception basée sur la mémoire, ce qui le rend moins adapté aux très grands ensembles de données.

`MLR3` offre des performances solides pour un large éventail de tâches de ML, avec des optimisations pour le traitement efficace des données en R. Cependant, comme `Weka`, il peut être confronté à des limitations de performance pour les très grandes données en raison des contraintes de mémoire de R.

Connu pour sa performance et son efficacité sur des ensembles de données de taille moyenne, `Scikit-learn` est optimisé pour `Python`, qui peut être accéléré avec des bibliothèques spécifiques. Pour des ensembles de données très volumineux, des stratégies comme le traitement par lots ou l'utilisation de bibliothèques complémentaires peuvent être nécessaires.

A.5.2 Facilité d'utilisation

Avec son interface graphique, `Weka` est particulièrement accessible aux débutants et à ceux qui préfèrent une approche visuelle pour le machine learning. Il ne nécessite pas de compétences de programmation pour effectuer de nombreuses tâches.

`MLR3`, tout en étant puissant, a une courbe d'apprentissage plus abrupte, surtout pour ceux qui ne sont pas déjà familiers avec R. Cependant, il offre une flexibilité considérable pour la personnalisation des workflows de machine learning.

`Scikit-learn` est réputé pour sa documentation exhaustive et sa conception intuitive, ce qui le rend facile à apprendre et à utiliser, y compris pour les débutants en machine learning.

A.5.3 Popularité

Bien qu'il soit très populaire dans le milieu académique, surtout pour l'enseignement, `Weka` est moins utilisé dans l'industrie par rapport à `Scikit-learn` ou aux outils basés sur R.

`MLR3` est bien respecté dans la communauté R pour l'apprentissage automatique, mais sa popularité générale est surpassée par celle de `Scikit-learn`, en partie à cause de la dominance globale de `Python` dans le machine learning et la science des données.

`Scikit-learn` est l'un des frameworks de machine learning les plus populaires et largement utilisés, tant dans la recherche que dans l'industrie, grâce à la popularité de `Python` dans ces domaines.

A.5.4 Communauté

Le nombre de contributeurs est un indicateur de l'activité de développement et de la diversité des contributions, ce qui peut affecter la rapidité avec laquelle les bugs sont corrigés, la variété des fonctionnalités offertes, et la capacité du projet à innover et à s'adapter aux nouveaux défis. Un grand nombre de contributeurs peut aussi indiquer une bonne documentation et un soutien communautaire solide, ce qui est crucial pour les utilisateurs finaux pour résoudre les problèmes et apprendre à utiliser l'outil efficacement. D'un autre côté, il est important de noter que le nombre de contributeurs ne reflète pas nécessairement la qualité ou la pertinence d'un outil pour un projet spécifique. Des outils avec moins de contributeurs peuvent offrir des fonctionnalités uniques ou mieux adaptées. Le nombre de contributeurs pour chacun de ces projets de machine learning – `Weka, MLR3, Scikit-learn` – peut varier considérablement et reflète souvent la popularité, l'engagement de la communauté et l'étendue de l'écosystème autour de chaque outil.

`Weka` est un projet développé à l'Université de Waikato en Nouvelle-Zélande. Bien qu'il ait une base solide d'utilisateurs, surtout dans le monde académique, le nombre de contributeurs directs peut être relativement faible comparé aux projets open source gérés par une large communauté. La majorité des contributions viennent de l'équipe universitaire originale et de quelques collaborateurs externes.

`MLR3` est spécifique à la communauté R et bien que moins connu que `Scikit-learn`. Il bénéficie du soutien d'une communauté active de développeurs et de chercheurs en statistique et en machine learning. Le nombre de contributeurs est limité, mais actif, avec des contributions provenant principalement de la communauté de recherche en statistique et de développeurs spécialisés en R.

`Scikit-learn` bénéficie d'une large popularité et d'une communauté très active, ce qui se reflète dans le nombre de contributeurs. En tant que l'un des projets open source les plus influents dans le domaine du machine learning, Scikit-learn compte des centaines de contributeurs qui ont participé à son développement au fil des ans, allant des contributions mineures aux développements majeurs.

A.5.5 Déploiement

Le déploiement de modèles entraînés avec `Weka` dans des applications de production peut être compliqué, notamment à cause de sa nature essentiellement GUI et de la nécessité de fonctionner dans un environnement `Java`.

Le déploiement de modèles R, y compris ceux développés avec `MLR3`, peut nécessiter des efforts supplémentaires pour l'intégration dans des applications de production, surtout si l'environnement cible n'est pas principalement basé sur R.

Les modèles développés avec `Scikit-learn` sont relativement plus faciles à déployer dans des environnements de production, en partie grâce à la compatibilité de `Python` avec de nombreux systèmes et à sa large utilisation dans l'industrie pour le développement d'applications de machine learning.

Annexe B

Bibliothèques `Python` utilisées pour DiNATrA$\mathcal{X}$

`Glob` pour rechercher tous les chemins correspondant à un motif particulier selon les règles utilisées par le shell `Unix`. Les résultats sont renvoyés dans un ordre arbitraire.

`Numpy` pour introduire une gestion facilitée des tableaux de nombres et des calculs associés.

`Pandas` pour l'analyse et la manipulation de tableaux de données mixtes ou 'dataframes'.

`Folium` pour la visualisation et la conception de cartes interactives.

`Seaborn` pour visualiser des données grâce à `Matplotlib` en fournissant une interface de haut niveau permettant de dessiner des graphiques statistiques.

`Geopandas` qui est un projet open source visant à faciliter le travail avec des données géospatiales. Il étend les types de données utilisés par `Pandas` pour permettre des opérations spatiales sur les types géométriques.

`Geopy` un client pour plusieurs services web de géocodage. Il permet de localiser facilement des coordonnées à travers le monde.

`Shapely` pour l'analyse théorique des ensembles et la manipulation des caractéristiques planaires.

`Matplotlib` permettant de produire des graphes de qualité.

`Scikit-learn` une bibliothèque spécialisée dans la science des données et l'apprentissage automatique.

`sklearnex` extension permettant d'optimiser `Scikit-learn` et d'en améliorer les performances.

`JellyFish` permet l'appariement approximatif et phonétique de chaînes de caractères. Cette bibliothèque implémente différentes mesures de distance.

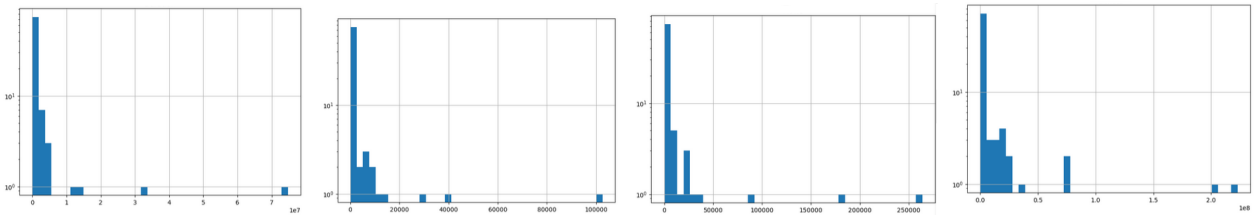
Annexe C

Résultats supplémentaires pour DiNATrA$\mathcal{X}$

SOI *Stade de France*

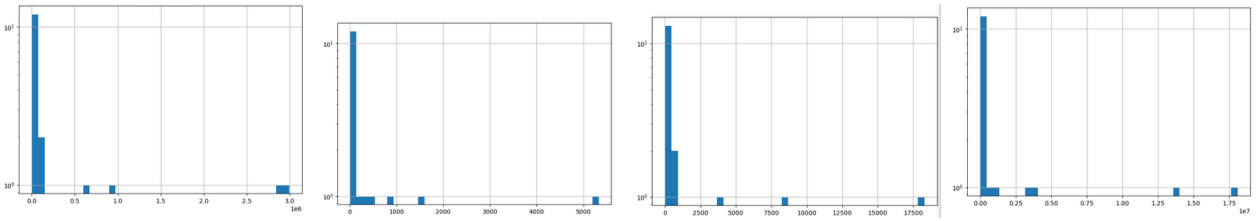
TABLE C.1
OUTLIERS SOI *Stade de France*

TL	index	TimeSlot	grpDesc	Duration	Users	Flows	Packets	Distance		
								EUC.	MAN.	MIN.
TL_1	987801	2019-04-13	Web	3 475 207	6 960	20 748	18 495 047	X	X	X
	988443	2019-04-13	Download	111 939	551	1 041	3 416 703	X	X	X
	988595	2019-04-13	CloudStorage	67 916	457	1 057	1 432 045	X	X	X
	988649	2019-04-13	Chat	641 289	977	3 523	3 321 743	X	X	X
TL_2	988592	2019-04-20	CloudStorage	37 738	323	746	1 190 851		X	
	987795	2019-04-20	Web	2 992 722	5 331	18 291	13 952 109	X	X	X
	988444	2019-04-20	Download	95 650	436	846	3 473 516	X	X	X
	988092	2019-04-20	StreamAVSP	2 809	10	274	786 489	X		X
	988652	2019-04-20	Chat	647 531	861	3 941	3 684 464	X	X	X
TL_3	988125	2019-04-27	StreamAVSP	101 464	281	4 056	22 195 864		X	
	988484	2019-04-27	Download	1 958 526	10 004	30 648	35 207 925	X	X	X
	988633	2019-04-27	CloudStorage	4 651 668	11 979	34 931	**75 650 121**	X	X	X
	988684	2019-04-27	Chat	**14 633 840**	**29 217**	**88 774**	**76 058 144**	X	X	X
	987836	2019-04-27	Web	**33 536 020**	**102 666**	**263 866**	**201 007 729**	X		X
TL_4	988034	2019-05-04	Streaming	768 986	1 251	6 965	15 261 447		X	
	988447	2019-05-04	Download	65 602	339	727	4 317 036	X	X	X
	988615	2019-05-04	CloudStorage	32 688	264	692	3 413 144	X	X	X
	988643	2019-05-04	Chat	450 775	559	2 173	2 509 926	X	X	X
	988094	2019-05-04	StreamAVSP	4 584	6	53	835 353	X		X
TL_5	988078	2019-05-12	Streaming	**15 212 790**	**46 537**	**152 797**	**163 582 903**	X	X	X
	988105	2019-05-12	StreamAVSP	8 653	10	201	1 700 986	X		X
	988333	2019-05-12	Mail	2 030 219	3 945	5 722	5 247 200	X	X	X
	988433	2019-05-12	Games	351 864	841	1 881	1 000 813	X	X	X
	988634	2019-05-12	CloudStorage	9 495 566	13 934	48 376	89 158 252	X	X	X
	988683	2019-05-12	Chat	**15 155 680**	24 176	62 242	68 175 067	X	X	X
Contamination=0.2 & TL=1 jour										



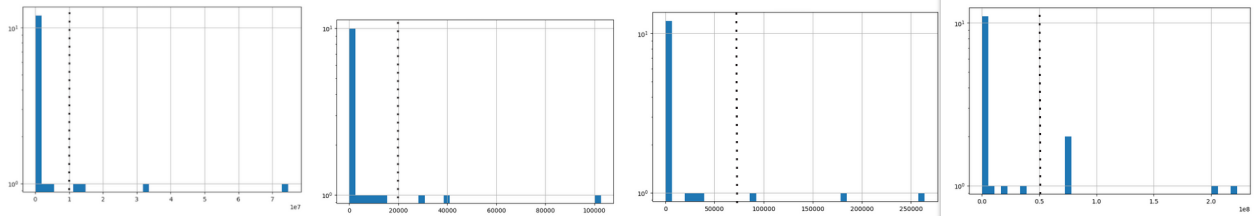
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.1. Distribution SOI *Stade de France* – ALL



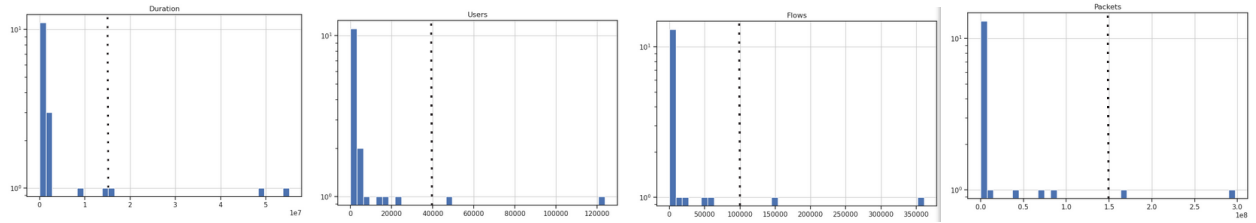
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.2. Distribution SOI *Stade de France* – TL_2



(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.3. Distribution SOI *Stade de France* – TL_3



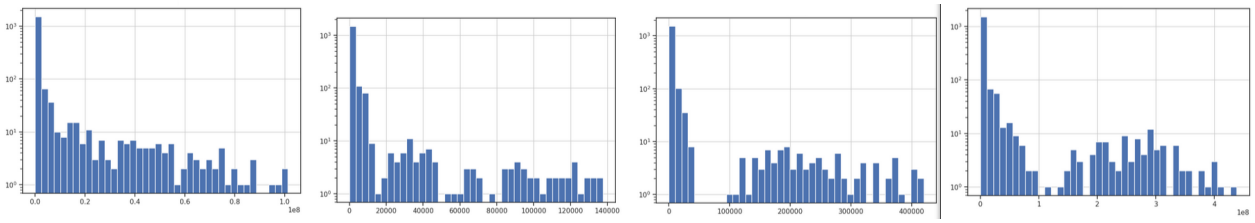
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.4. Distribution SOI *Stade de France* – TL_5

SOI *Notre-Dame de Paris*

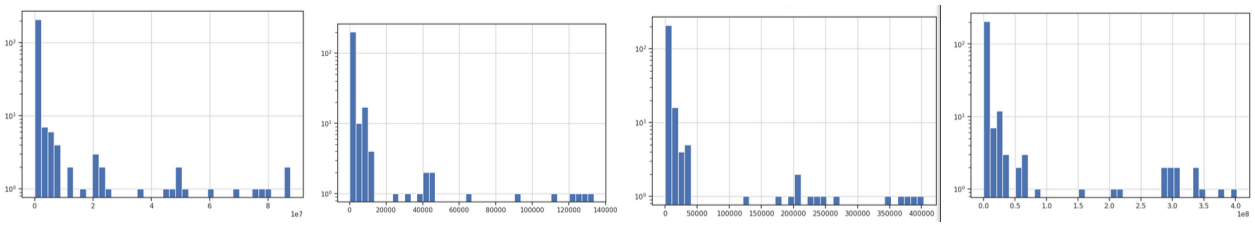
TABLE C.2
OUTLIERS SOI *Notre-Dame de Paris*

TL	index	TimeSlot	grpDesc	Duration	Users	Flows	Packets	Distance		
								EUC.	MAN.	MIN.
TL_1	894914	2019-04-07	Web	27 727 508	**67 066**	**207 332**	171 461 755	X	X	X
	894927	2019-04-06	Web	36 047 068	**95 977**	**277 914**	217 154 861	X	X	X
	894938	2019-04-01	Web	**43 356 480**	**110 498**	**337 023**	**264 558 154**	X	X	X
	894941	2019-04-03	Web	**47 023 640**	**118 656**	**360 294**	**285 592 117**	X	X	X
	895166	2019-04-07	Streaming	12 960 874	24 326	137 343	**248 946 768**	X	X	
	895176	2019-04-06	Streaming	16 346 533	33 454	176 144	**279 335 726**			X
TL_2	894909	2019-04-14	Web	25 993 708	**65 450**	**198 345**	157 170 720	X	X	X
	894923	2019-04-13	Web	35 383 580	**90 512**	**265 054**	209 051 405	X	X	X
	894942	2019-04-10	Web	**47 967 432**	**123 831**	**370 488**	**286 209 471**	X	X	X
	894943	2019-04-08	Web	**44 343 816**	**112 553**	**343 610**	**286 283 344**	X	X	X
	895640	2019-04-10	Download	1 698 301	8 140	16 597	81 743 656	X	X	X
	895161	2019-04-14	Streaming	12 343 773	23 464	127 812	220 044 140	X	X	X
TL_3	894911	2019-04-21	Web	27 040 040	64 687	194 373	159 654 143	X	X	X
	894917	2019-04-20	Web	33 436 130	**83 276**	**244 146**	188 579 873	X	X	X
	894933	2019-04-18	Web	**41 762 180**	**109 529**	**320 434**	**242 061 832**	X	X	X
	895158	2019-04-21	Streaming	11 864 740	22 253	118 787	207 063 300		X	X
	894934	2019-04-19	Web	**41 546 240**	**106 197**	**316 999**	**247 091 569**	X		X
	894936	2019-04-15	Web	**46 185 200**	**115 152**	**337 418**	**252 700 053**	X	X	
	894940	2019-04-16	Web	**44 416 500**	**115 355**	**341 442**	**265 633 224**	X		X
	895628	2019-04-16	Download	1 289 390	6 661	13 012	60 850 566	X	X	
TL_4	894906	2019-04-28	Web	22 362 484	52 239	164 771	138 930 671	X	X	X
	894907	2019-04-22	Web	23 632 574	58 320	180 597	146 852 060	X	X	X
	894913	2019-04-27	Web	27 922 120	**70 764**	**205 205**	164 212 248	X	X	X
	894922	2019-04-23	Web	34 123 084	**86 736**	**261 787**	205 975 697	X	X	X
	894929	2019-04-25	Web	37 792 704	**94 413**	**283 677**	223 836 264	X	X	X
	895155	2019-04-28	Streaming	10 087 536	18 454	112 143	196 386 104	X	X	X
	895155	2019-04-28	Streaming	10 087 536	18 454	112 143	196 386 104	X	X	X
Contamination=0.03 & TL=1 semaine										



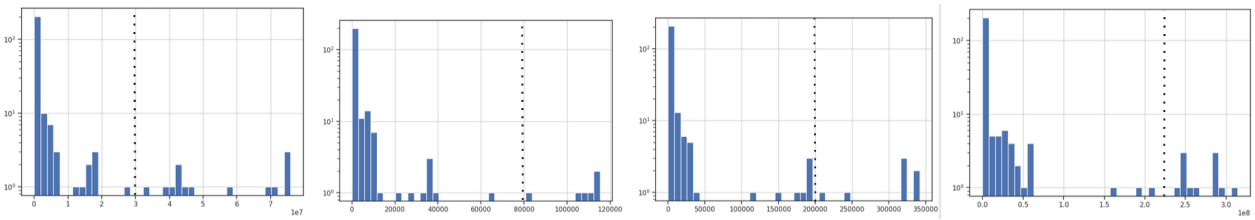
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.5. Distribution SOI *Notre-Dame de Paris* – ALL



(a) Duration (b) Users (c) Flows (d) Packets

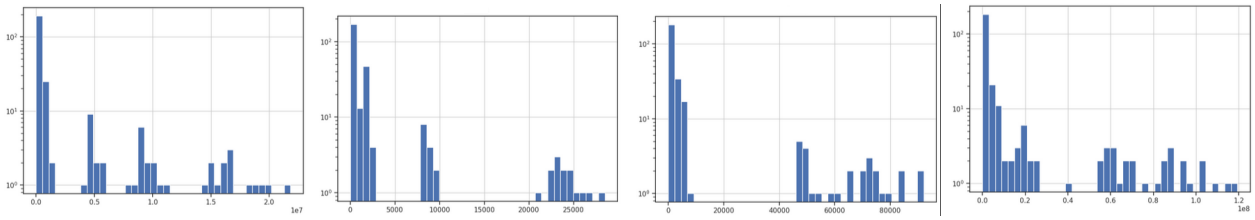
FIG. C.6. Distribution SOI *Notre-Dame de Paris* – TL_2



(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.7. Distribution SOI *Notre-Dame de Paris* – TL_3

SOI *Cimetière de Montmartre*

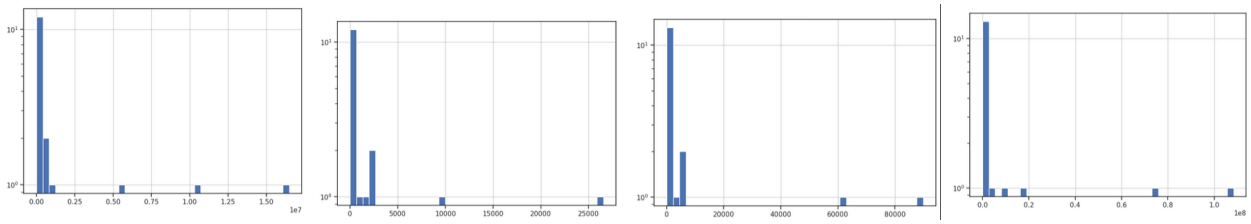


(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.8. Distribution SOI *Cimetière de Montmartre* – ALL

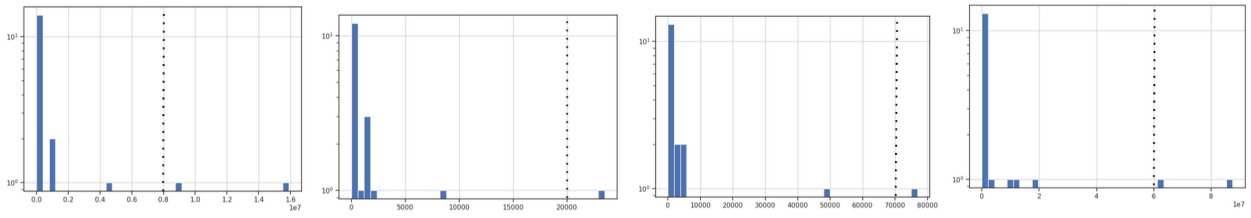
TABLE C.3
OUTLIERS SOI *Cimetière de Montmartre*

TL	index	TimeSlot	grpDesc	Duration	Users	Flows	Packets	Distance		
								EUC.	MAN.	MIN.
TL_7	811198	2019-04-07	Web	**10 543 200**	**26 557**	**89 964**	**75 101 977**	X	X	X
	811489	2019-04-07	StreamAVSP	8 268	**12**	**127**	75 101 977	X	X	X
	811864	2019-04-07	Download	450 856	1 525	3 401	17 838 783	X	X	X
	812019	2019-04-07	CloudStorage	266 635	2 316	5 576	8 484 850	X	X	X
	812077	2019-04-07	Chat	1 043 722	2 095	5 932	5 104 387	X	X	X
TL_8	811482	2019-04-08	StreamAVSP	7 683	**9**	**238**	1 562 760	X	X	X
	811528	2019-04-08	P2P	2 796	**2**	**217**	270 250	X	X	X
	811181	2019-04-08	Web	**8 876 122**	**23 492**	**76 859**	**62 716 170**	X	X	X
	811866	2019-04-08	Download	313 580	1 597	3 791	18 312 680	X	X	X
	811743	2019-04-08	Mail	803 487	1 702	3 004	8 889 304	X	X	X
	812036	2019-04-08	CloudStorage	289 003	1 980	5 431	12 761 494	X	X	X
TL_9	811351	2019-04-09	VPN	9712	**16**	**17**	199 508	X	X	X
	811476	2019-04-09	StreamAVSP	5097	**8**	**200**	1 210 930	X	X	X
	811179	2019-04-09	Web	896 435	**23 383**	**74 419**	**61 320 469**	X	X	X
	811858	2019-04-09	Download	328 126	1 849	3 583	16 264 395	X	X	X
	811736	2019-04-09	Mail	961 881	1 758	2 916	5 357 073	X	X	X
	812005	2019-04-09	CloudStorage	311 608	1 938	4 054	6 771 900	X	X	X
TL_{10}	811191	2019-04-10	Web	**10 280 430**	**25 023**	**79 486**	**69 025 551**	X	X	X
	811478	2019-04-10	StreamAVSP	5447	**3**	**162**	1 296 320	X	X	X
	811884	2019-04-10	Download	372 092	1 889	3 977	24 788 578	X	X	X
	812035	2019-04-10	CloudStorage	344 015	2 097	5 290	11 281 918	X	X	X
	812073	2019-04-10	Chat	946 183	1 866	5 427	4 687 324	X	X	X
TL_{12}	811185	2019-05-02	Web	**9 955 558**	**24 329**	**74 958**	**65 126 905**	X	X	X
	811497	2019-05-02	StreamAVSP	20 386	**16**	**195**	3 045 909	X	X	X
	811671	2019-05-02	MailOrange	72 404	**280**	**528**	322 138	X	X	X
	811728	2019-05-02	Mail	773 555	1 774	2 746	3 582 860	X	X	X
	811860	2019-05-02	Download	216 796	1 491	2 975	16 721 202	X	X	X
	812011	2019-05-02	CloudStorage	290 659	2 069	4 611	7 540 459	X	X	X
	812085	2019-05-02	Chat	862 105	1 715	5 322	5 599 638	X	X	X
Contamination=0.5 & TL=1 jour										



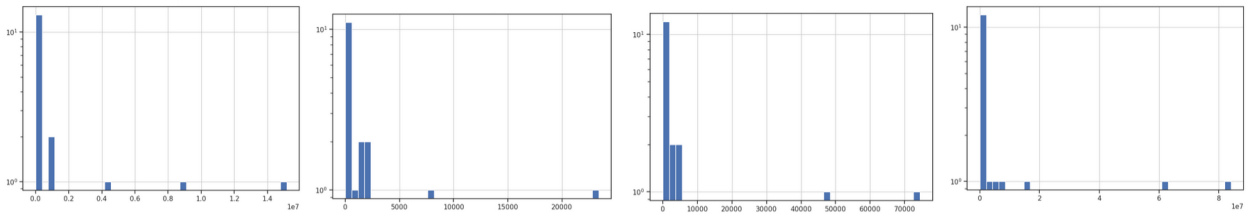
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.9. Distribution SOI *Cimetière de Montmartre* – TL_7



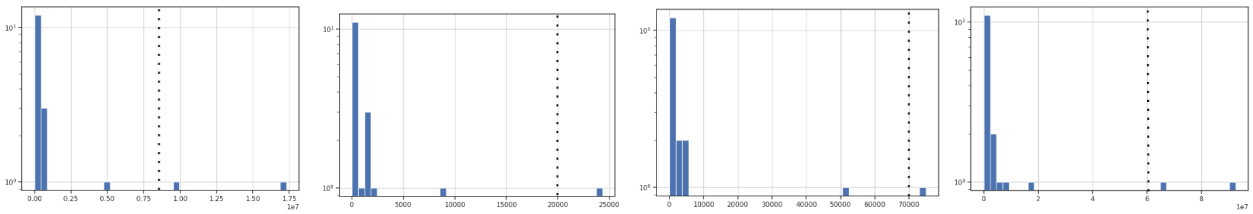
(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.10. Distribution SOI *Cimetière de Montmartre* – TL_8



(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.11. Distribution SOI *Cimetière de Montmartre* – TL_9



(a) Duration (b) Users (c) Flows (d) Packets

FIG. C.12. Distribution SOI *Cimetière de Montmartre* – TL_{12}

Annexe D

Algorithmes

Algorithm 1 basic.pl

```
1:  while line ← scenario do
2:      if lineIsWellFormated(line) then        ▷ IP address, protocol, number of fields & not empty
3:          line[ ] ← split(line, ',')
4:      else
5:          next line
6:      line.dropFields([0, 4, 7, 8, 9])        ▷ Drop 'StartTime', 'Dir' & 'State', 'sTos', 'dTos'
7:      if line[9].match('Botnet') then         ▷ Label as 'true' if botnet traffic
8:          line[9] ← 'true'
9:      else
10:         line[9] ← 'false'
11:     ▷ 0 :Dur 1 :Proto 2 :SrcAddr 3 :Sport 4 :DstAddr ◁
12:     ▷ 5 :Dport 6 :TotPkts 7 :TotBytes 8 :SrcBytes 9 :Bot ◁
13:
14:     print join(line[ ], ',')                ▷ Print line into ARFF file
```

Algorithm 2 advanced.pl

```
1:  hash ← { }                                              ▷ Init. empty dictionary

2:  while line ← scenario do
3:      if lineIsWellFormated(line) then                    ▷ IP address, protocol, number of fields & not empty
4:          line[ ] ← split(line, ',')
5:      else
6:          next line
7:      line.dropFields([0, 4, 8, 9])                       ▷ Drop 'StartTime', 'Dir' & 'sTos', 'dTos'
8:      if line[10].match('Botnet') then                    ▷ Label as '1' if botnet traffic
9:          line[10] ← '1'
10:     else
11:         line[10] ← '0'

12:     AgK ← "line[2] − line[5] : line[6]_line[1]"         ▷ Build aggregation key
13:     hash.AgK['TotReq']++
14:     hash.AgK['TotDur'] += line[0]                       ▷ Sum up 'duration'
15:     hash.AgK['Proto'] = line[1]
16:     hash.AgK['Dport'] = line[6]
17:     hash.AgK['TotPkts'] += line[8]                      ▷ Sum up 'TotPkts'
18:     hash.AgK['TotBytes'] += line[9]                     ▷ Sum up 'TotBytes'
19:     hash.AgK['TotSrcBytes'] += line[10]                 ▷ Sum up 'TotSrcBytes'
20:     hash.AgK['TotBots'] += line[11]                     ▷ Sum up 'TotBots'
21:     for all flag ∈ split(", line[7]) do                 ▷ Count TCP flags
22:         hash.AgK['TotAPRSF'][flag]++

23:     for all key ∈ hash do                               ▷ Print ARFF file
24:         print hash[key]['Proto', 'TotAPRSF', 'Dport', 'TotReq',
            'TotDur', 'TotPkts', 'TotBytes', 'TotSrcBytes', 'TotBots']
```

Algorithm 3 bySiteAggregation.pl

```
1: file ← { }                                              ▷ Init. file dictionary
2: appGroups ← { }                                         ▷ Init. appGroups dictionary
3: appGroupsDesc ← { }                                     ▷ Init. appGroupsiDesc dictionary

4: procedure readAppGroups(appGroupsList)
5:     while line ← appGroupsList do
6:         if wellFormedAppGroupLine(line) then
7:             line[ ] ← split(line, ',')
8:         else
9:             next line
10:        appGroups[line[0]] ← line[1]
11:        appGroupsDesc[line[1]] ← line[2]

12: procedure printARFFfile(file)
13:    printARFFheader(precision)
14:    for all time ∈ sort keys file do
15:        print(file[ time ][ 'SiteName' ][ 'CoordX' ][ 'CoordY' ][ 'AppGroup' ][ [ 'nPktUp' ] +
           [ 'nPktDn' ] ]
16:        [ 'Duration' ][ 'Users' ][ 'Flows' ] . appGroupsDesc[ file[ 'AppGroup' ] ])

17: while line ← CANCANcsvFile do
18:    if wellFormedcsvFile(line) then
19:        line[ ] ← split(line, ',')
20:    else
21:        next line
22:    ▷ 0 :PortApp 1 :LocInfo 2 :COORD_X 3 :COORD_Y 4 :SiteName 5 :TimeSlot 6 :nPacketUp
       7 :nPacketDn 8 :Duration 9 :Users 10 :Flows ◁
23:                                  ▷ Aggregate CANCAN flows by TimeSlot -> Site -> AppGroup
24:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'CoordX' ] ← line[2]
25:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'CoordY' ] ← line[3]
26:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'nPktUp' ] += line[6]
27:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'nPktDn' ] += line[7]
28:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'Duration' ] += line[8]
29:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'Users' ] += line[9]
30:    file[ line[5] ][ line[4] ][ appGroups[ line[0] or '0' ][ 'Flows' ] += line[10]
```

Algorithm 4 bySectorAggregation.pl

```
1: file ← { }                                                    ▷ Init. file dictionary
2: sites ← { }                          ▷ Init. sites dictionary with LambertII & GPS coordinates

3: procedure readSitesList(sitesList)
4:     while line ← sitesList do
5:         if wellFormedSitesListLine(line) then
6:             line[ ] ← split(line, ',')
7:         else
8:             next line
9:         sites[ line[0] ][ 'CoordX' ] ← line[1]
10:        sites[ line[0] ][ 'CoordY' ] ← line[2]
11:        sites[ line[0] ][ 'Lon' ] ← line[3]
12:        sites[ line[0] ][ 'Lat' ] ← line[4]

13: procedure printARFFfile(file)
14:    printARFFheader(precision)
15:    for all time ∈ sort keys file do
16:        print(file[ time ][ gh ][ 'AppGroup' ][ 'Description' ]
           [ 'Packets' ][ 'Duration' ][ 'Users' ][ 'Flows' ])

17: while line ← aggDataFile do
18:    if wellFormedcsvFile(line) then
19:        line[ ] ← split(line, ',')
20:    else
21:        next line
22:                                  ▷ 0 :TimeSlot 1 :SiteName 2 :COORD_X 3 :COORD_Y 4 :AppGroup
                                     ▷ 5 :Description 6 :Packets 7 :Duration 8 :Users 9 :Flows
23:    lat ← site[ line[1] ]['Lat']
24:    lon ← site[ line[1] ]['lon']
25:    geohash ← Geohash→new()
26:    gh ← geohash→encode(lat, lon, precision)
27:                                  ▷ Aggregate data by TimeSlot -> Geohash -> AppGroup
28:    file[ line[0] ][ gh ][ line[4] ][ 'Description' ] ← line[5]
29:    file[ line[0] ][ gh ][ line[4] ][ 'Packets' ] += line[6]
30:    file[ line[0] ][ gh ][ line[4] ][ 'Duration' ] += line[7]
31:    file[ line[0] ][ gh ][ line[4] ][ 'Users' ] += line[8]
32:    file[ line[0] ][ gh ][ line[4] ][ 'Flows' ] += line[9]
```

Algorithm 5 printARFFheader(precision)

```
print ”
@RELATION CANCAN-{precision}-SECTORS
@attribute TimeSlot date "yyyy-MM-dd HH:mm:ss"
@attribute Geohash string
@attribute AppGroup numeric
@attribute AppDesc string
@attribute Packets numeric
@attribute Duration numeric
@attribute Users numeric
@attribute Flows numeric
@data"
```

Algorithm 6 computeSectors(df, radius=0.15)

geo_df ← geopandas.GeoDataFrame(*df*, *geometry* ←geopandas.points_from_xy(*df*.CoordX, *df*.CoordY), *crs*=27572) ▷ *Coordonnées dans le système LambertII*

geo_df ← *geo_df*.to_crs(4326) ▷ *World Geodetic System utilisé par le système GPS*

geo_df['location'] ← [(*point*.xy[1][0], *point*.xy[0][0]) **for** *point* **in** *geo_df*['geometry']]

geo_df.drop(['CoordX', 'CoordY', 'geometry'], *axis* ← 1, *inplace* ← True) ▷ *Suppression des colonnes inutiles*

$kms\_per\_radian \leftarrow 6371$ ▷ *Rayon de la Terre*

$\epsilon \leftarrow \frac{radius}{kms\_per\_radian}$

$clusters \leftarrow \textbf{DBSCAN}(eps = \epsilon, min\_samples = 1, algorithm =$ 'ball_tree', *metric* ← 'haversine').fit(numpy.radians(*geo_df*['location'].tolist()))

geo_df['ClusterId'] ← *clusters*.labels_

nbr_clusters ← length(numpy.unique(*clusters*.labels_))

return (*geo_df*, *nbr_clusters*)

Algorithm 7 computeCentroids(geo_df, nclusters)

```
1: function GETCENTERMOSTPOINT(cluster)
2:     centroid ← (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
3:     centermostPoint ← min(cluster, key ←lambda point : great_circle(point, centroid).m)
4:     return centermostPoint
5:
6: for c ∈ {0...nclusters} do
7:     cluster ← geo_df[geo_df['ClusterId'] == c]['location'].tolist()
8:     center ← getCentermostPoint(cluster)
9:     centers.append(center)
10: return centers
```

Algorithm 8 Implémentations de compréhension de listes en `Python` avec `Pandas`

```
# clusters -> liste des secteurs dont il faut extraire les donnees
# adf -> dataframe contenant toutes les donnees agregees
def extract_data(adf, clusters):
        dfs = []
        for c in clusters:
                df = adf[adf['cluster'] == c].drop('cluster', axis=1)
                dfs.append(df)
        return pandas.concat(dfs)

# ts -> liste des TL a generer [(b1,e1),(b2,e2),...,(bn,en)]
def split_data(df, ts):
        dfs = [] # df slices
        for b,e in ts:
                dfs.append(df[df['TimeSlot'].between(b,e)]
                    .drop(['TimeSlot','grpDesc'], axis=1))
        return dfs
```

Algorithm 9 computeDNA(dfs, ts, labels, nclusters, precision, activity='Users')

```
function ROTATE(list, n)
    return list[-n :] + list[ :-n]                          ▷ Fin liste concaténé avec début

function APPGROUPTOLETTER(df)                               ▷ Convertit caractère en chiffre
    df['appGroupL'] ← df['AppGroup'].apply(lambda x : character(ord('A') + x))
    return df

df ← DataFrame(['Begin', 'End', 'DNA', 'Strand', 'Volumes']).new        ▷ Init. dataframe
i ← 0
for all dfts ∈ dfs do
    dfts ← appGroupToLetter(dfts)                           ▷ AppGroup : 0→A, 1→B, ...
    dfts['ClusterId'] ← labels[i]                           ▷ Affecte les clusters aux données
    (D, N, A) ←DNA(dfts, nclusters, precision, activity)    ▷ (DNA, straNd, ActivityVolumes)
    df.append({ts[i][0], ts[i][1], D, N, A})
    i ← i + 1

df['_DNA'] ← rotate(df['DNA'], 1)                           ▷ chaîne : 'ABCD' → 'DABC'
df['_Strand'] ← rotate(df['Strand'], 1)
df['DNAD'] ← df.apply(lambda x : DLdistance(x['DNA'], x['_DNA']))   ▷ Damerau-Levenshtein
df['STAD'] ← df.apply(lambda x : DLdistance(x['Strand'], x['_Strand']))

df['ANOD'] ← df['DNAD'] + df['STAD']                        ▷ Distance d'anormalité & Sévérité
df['Severity'] ← df['ANOD'].apply(lambda x : 0 if x < precision else round((x/precision)^2, 2))
return df
```

Algorithm 10 DNA(df, nclusters, precision, activity)

```
list sequences
list volumes
activities ← quickSort(df.groupby('appGroupL')[activity].sum())
activities['percent'] ← round(100.activities[activity] / activities[activity].sum(), 2)
list activity ← activities['appGroupL']

for i ∈ {0 ... nclusters} do                                   ▷ Construction du DNA
    seq ← DataFrame(df[df['ClusterId'] == i]['appGroupL'].unique(),
    columns=['appGroupL']).new
    seq['Rank'] ← seq['appGroupL'].apply(lambda x : activity.index(x))   ▷ Recherche le rang
    seq ← ''.join(seq.sort('Rank')['appGroupL']) ▷ Construit la séquence correspondant au cluster
    seq ← seq[ :precision]                          ▷ Tronque la séquence à la longueur precision
    sequences.append(seq)

    vol ← 0
    for all a ∈ seq do                          ▷ Volume de données représenté par la séquence
        vol ← round(vol + activities[activities['appGroupL'] == a]['percent'],2)
    volumes.append(vol)

dfSeq ← DataFrame(sequences, columns=['Seq']).new
dfSeq['Volumes'] ← volumes
D ← '-'.join(dfSeq.sort('Volumes')['Seq'])      ▷ Concatène les séquences dans l'ordre 'Volumes'

N ← D.replace('-','')                                          ▷ Construit le brin à partir du DNA
N ← Series([*N]).unique()
dfN ← DataFrame(N, columns=['N']).new
dfN['Rank'] ← dfN['N'].apply(lambda x : activity.index(x))
N ← ''.join(dfN.sort('Rank')['N'])                          ▷ Construit le brin dans l'ordre activity
N ← N[ :precision]                                          ▷ Tronque le brin à la longueur precision

A ← '-'.join(dfSeq.sort('Volumes')['Volumes'].apply(lambda x : str(round(x,2))))
return (D,N,A)
```

Algorithm 11 featureToLetter(df, feature)

```
letters ← list(upperCase) + list(lowerCase) + list(numbers)        ▷ Init. letter list
df["{feature}L"] ← df["{feature}L"].apply(lambda f : letters[f])
return df
```

Algorithm 12 computeTS(b=0, e=22, d=10, debug=False)

```
ts ← [ ]                                                           ▷ Init. empty list

for h ∈ [b, e[ do
    for m ∈ [0, 60[ do
        if m == 59 then
            ts.append(("2011-08-{d} {h} :{m}", "2011-08-{d} {h+1} :00"))
        else
            ts.append(("2011-08-{d} {h} :{m}", "2011-08-{d} {h} :{m+1}"))
return ts
```

le cnam

Christophe MAUDOUX

Vers l'automatisation de la détection d'anomalies réseaux

HESAM UNIVERSITÉ

Résumé : Nous vivons dans un monde hyperconnecté. À présent, la majorité des objets qui nous entourent échangent des données soit entre-eux soit avec un serveur. Ces échanges produisent alors de l'activité réseau. C'est l'étude de cette activité réseau qui nous intéresse ici et sur laquelle porte ce document. En effet, tous les messages et donc le trafic réseau généré par ces équipements est voulu et par conséquent légitime. Il est de ce fait parfaitement formaté et connu. Parallèlement à ce trafic qui peut être qualifié de "normal", il peut exister du trafic qui ne respecte pas les critères attendus. Ces échanges non conformes aux attendus peuvent être catégorisés comme étant du trafic "anormal". Ce trafic illégitime peut être dû à plusieurs causes tant internes qu'externes. Tout d'abord, pour des raisons bassement mercantiles, la plus part de ces équipements connectés (téléphones, montres, serrures, caméras,...) est peu, mal, voire pas protégée du tout. De ce fait, ils sont devenus les cibles privilégiées des cybercriminels. Une fois compromis, ces matériels communiquant constituent des réseaux capables de lancer des attaques coordonnées : des *botnets*. Le trafic induit par ces attaques ou les communications de synchronisation internes à ces botnets génèrent alors du trafic illégitime qu'il faut pouvoir détecter. Notre première contribution a pour objectif de mettre en lumière ces échanges internes, spécifiques aux botnets. Du trafic anormal peut également être généré lorsque surviennent des événements externes non prévus ou extra-ordinaires tels des incidents ou des changements de comportement des utilisateurs. Ces événements peuvent impacter les caractéristiques des flux de trafic échangés comme leur volume, leurs sources, destinations ou encore les paramètres réseaux qui les caractérisent. La détection de ces variations de l'activité réseau ou de la fluctuation de ces caractéristiques est l'objet de nos contributions suivantes. Il s'agit d'un framework puis d'une méthodologie qui en découle permettant d'automatiser la détection de ces anomalies réseaux et éventuellement de lever des alertes.

Mots clés : cybersécurité, apprentissage automatique, détection d'anomalies, réseaux, botnets, ADN, signatures numériques, distance de Damerau-Levenstein, corrélation d'événements

Abstract: We live in a hyperconnected world. Currently, the majority of the objects surrounding us exchange data either among themselves or with a server. These exchanges consequently generate network activity. It is the study of this network activity that interests us here and forms the focus of this thesis. Indeed, all messages and thus the network traffic generated by these devices are intentional and therefore legitimate. Consequently, it is perfectly formatted and known. Alongside this traffic, which can be termed "normal," there may exist traffic that does not adhere to expected criteria. These non-conforming exchanges can be categorized as "abnormal" traffic. This illegitimate traffic can be due to several internal and external causes. Firstly, for purely commercial reasons, most of these connected devices (phones, watches, locks, cameras, etc.) are poorly, inadequately, or not protected at all. Consequently, they have become prime targets for cybercriminals. Once compromised, these communicating devices form networks capable of launching coordinated attacks: botnets. The traffic induced by these attacks or the internal synchronization communications within these botnets then generates illegitimate traffic that needs to be detected. Our first contribution aims to highlight these internal exchanges, specific to botnets. Abnormal traffic can also be generated when unforeseen or extraordinary external events occur, such as incidents or changes in user behavior. These events can impact the characteristics of the exchanged traffic flows, such as their volume, sources, destinations, or the network parameters that characterize them. Detecting these variations in network activity or the fluctuation of these characteristics is the focus of our subsequent contributions. This involves a framework and resulting methodology that automates the detection of these network anomalies and potentially raises alerts.

Keywords: Cybersecuriry, Machine Learning, Anomalies Detection, Networks, Botnets, DNA, Digital Signatures, Damerau-Levenstein Distance, Events Correlation