MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN INFORMATIQUE DES ORGANISATIONS

Prédiction du comportement d'un attaquant à partir de métriques de distances sémantiques dérivées d'un grand darknet

Evrard, Laurent

Award date:
2018

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2017–2018

Prédiction du comportement d'un attaquant à partir de métriques de distances sémantiques dérivées d'un grand darknet

Laurent Evrard

Maître de stage : Jérôme François

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

L'analyse et la classification de données réseau est un domaine important de l'informatique et de la sécurisation des infrastructures informatiques d'entreprise. Ces méthodes permettent, aux frontières d'une infrastructere, d'opérer un premier filtrage des données ciblant les machines incluses dans cet architecture. Cette opération, en plus de d'aider à une bonne gestion de la charge sur les machines connectées, permet également de supprimer les tentatives d'attaques réseau les plus simples ou visibles. Au coeur de ces algorithmes de classification de données réseau se trouvent des notions de distances entre les flux ou paquets et c'est dernier sont également comparés via des distances calculées sur leurs caractéristiques. Dans ce travail, nous introduisons plusieurs notions de distance permettant de comparer de façon sémantique les ports réseau utilisés dans des protocoles réseau tels que TCP ou UDP. Ces notions de distances, construitent à l'aide d'algorithmes d'apprentissage automatique et d'algorithmes utilisant la notion de graphes, sont empiriquement comparées afin de vérifier leurs valeurs sémantiques. Deux cas d'utilisation liés à la sécurité sont finalement envisagés afin d'étudier l'avantage d'utiliser ces distances sémantiques dans des algorithmes de classification de données réseau.

Mots clefs

ports, distance, semantic, security, artificial intelligence, network port, semantic distance, network security, machine learning

Remerciements

Au début de ce travail, j'aimerais adresser quelques remerciements. Mes premiers vont vers mon maître de stage Monsieur Jérôme François et vers mon promoteur de mémoire Monsieur Jean-Noël Colin. Merci à tous deux pour vos conseils, votre expérience et le temps précieux que vous m'avez accordé. J'espère, dans l'avenir, pouvoir continuer à collaborer avec vous.

Ensuite, j'aimerais remercier ma compagne ainsi que ma famille pour leur aide et leurs encouragements dans les moments stressants de la phase d'écriture. Votre patience et votre soutient de tous les jours m'auront été très précieux.

Sommaire

Table des figures

Liste des tableaux

Première partie

Introduction

Chapitre 1

Introduction

1.1 De l'utilité d'une métrique de distance sémantique entre les ports réseau

Depuis quelques années maintenant, les experts relèvent une explosion dans le nombre de machines connectées à Internet ainsi que dans la puissance de ces dernières. Il n'est plus rare maintenant qu'un utilisateur ait quotidiennement en sa position de deux à quatres appareils [38] ayant la possiblité d'échanger des informations via ce réseau.

Cette augmentation du nombre de machines connectées a, en plus d'augmenter considérablement le nombre de services proposés sur Internet, engendré bon nombre de modifications aussi bien sociétales que techniques. Il est devenu commun d'utiliser publiquement, et un grand nombre de fois par jour, ces machines dans le but de consommer des services en ligne allant de la vente de produits, en passant par la communication, jusqu'au streaming de films.

Au niveau technique, l'augmentation du nombre de machines reliées à internet s'est d'abord fait de façon directement proportionnelle avec l'utilisation de la sphère d'adressage IP avant que, voyant le nombre d'adresses IP disponibles diminuer rapidement, les experts n'introduisent la technologie du NAT donnant une adresse IP publique par foyer. C'est maintenant finalement une nouvelle version du protocole IP qui est introduite avec l'arrivée d'IP version 6 qui théoriquement permettrait de connecter tous les atomes composant la planète (à supposer qu'ils puissent l'être).

Parallèlement à l'augmentation du trafic internet, se sont également développées les activités malicieuses en ligne. De la vente d'organes au vol de données personnelles, le type et le nombre de ces activités a énormément augmenté jusqu'à notre époque où l'on entend parfois dire que l'internet connu des utilisateurs lambda ne représente que 10% des activités d'internet et que le restant est dissimulé dans un "internet sombre ou caché".

Pour contrer cette face sombre de l'internet et protéger les systèmes qu'ils créent, déploient et maintiennent, les informaticiens doivent sans cesse se tenir informés des dernières attaques informatiques et des façons de les empêcher afin de protéger aux mieux les systèmes de tentatives d'intrusions provenants d'attaquants se situant, pour leur part, dans une réelle pente de professionalisation de leurs activités.

Il apparait des recherches et de l'expérience que, la meilleure sécurité possible n'est pas située à un seul et unique endroit de l'architecture technique ou logicielle d'une entreprise mais est composée de couches spécialisées permettant, de couche en couche, de réduire les risques d'incidents de sécurité.

Ainsi, l'une des premières de ces couches de sécurité a pour but de gérer les données arrivant aux machines par le trafic réseau d'internet. Ce trafic, composé de flux décom-

posables en paquets, doit être analysé afin de détecter, le plus rapidement possible, les tentatives d'intrusion contenues dans ce trafic et mettant en péril la sécurité des machines connectées.

Pour ce faire, le domaine de la recherche en informatique, en particulier la recherche en réseau et en sécurité informatique, a développé de très puissants algorithmes capables de comparer les flux ou paquets réseau afin de repérer lesquels sortent de la norme et sont donc probablement des attaques dirigées vers les machines à défendre. Or, pour que des flux ou paquets réseaux puissent être comparés, ces derniers doivent disposer de caractéristiques les rendant comparables. Ce sont par exemple les adresses IP ou encore la taille totale des données d'un flux ou encore l'intervalle de temps moyen entre deux paquets du flux. Si pour la plupart, ces données peuvent être comparées via des métriques de distance mathématiques habituelles, certaines autres, telles que les adresses IP ou encore les numéros de ports, ne peuvent l'être si facilement.

Si l'on imagine un exemple lié au numéro de port, on peut relever que le port 80 (permettant d'héberger un site web) est plus proche (distance euclidienne de 58) du port 22 (permettant de faire de l'accès à distance via le protocole SSH) que du port 443 (distance euclidienne de 363) permettant, lui aussi et via la même application, d'héberger des sites webs.

Des chercheurs [10] ont déjà proposé des notions de distances sémantiques pour ces entités liées au monde du réseau. Cependant, dans le cas des ports réseau (équivalent des portes d'une maison), la notion de distance sémantique proposée ne contient finalement qu'une sémantique assez faible calculée via les trois classes de ports réseau (well known, registered et dynamique). En effet, cette notion de distance considère que si deux ports sont dans la même classe, la distance entre ces derniers sera toujours identique. Ainsi, si l'on reprend notre exemple des ports 22, 80 et 443, ces ports sont, selon cette métrique, à la même distance les uns des autres alors que sémantiquement parlant, les port 80 et 443 devraient être beaucoup plus proches, étant donné qu'ils hébergent le même service.

Notre but dans ce travail est de chercher une méthode permettant de définir, à l'aide d'algorithme d'apprentissage automatique, une ou plusieurs autres notions de distance sémantique entre les ports réseau. Chacune des notions de distance sémantique définie devrait, si possible, être capable de représenter à la fois :

1. Que deux ports utilisés par deux applications différentes sont souvent présents conjointements sur une machine [1].
2. Que deux ports ouverts ensemble sur une même machine sont utilisés par la même application [2]

Suite à la création de ces notions de distance sémantique, notre but sera d'appliquer ces dernières dans un cas d'utilisation lié à la sécurité informatique pour démontrer la sémantique que ces distances portent ainsi que la réponse qu'elles founissent aux problèmatiques levées.

1.2 Contributions

Ce travail de recherche apporte les contributions suivantes :

1. La définition de plusieurs méthodes d'extraction des connaissances en réseau d'attaquants à partir de leurs activités malveillantes sur un grand Darknet ;

1. Les serveurs web sont, par exemple, souvent accompagnés d'applications dédiées aux mails
2. Les serveurs web utilisent, par exemple, les ports 80 et 443

2. La définition de notions de distance sémantique entre des ports réseau ;
3. La création de méthodes de comparaison de ces notions ;
4. L'utilisation et l'évaluation des dites notions dans deux cas d'utilisation liés à la sécurité informatique.

Ce document est accompagné en Annexe 2 de l'article scientifique (non encore publié) résultat de ce travail. Ce dernier contient une cinquième contribution sous la forme d'un ensemble de statistiques sur le grand Darknet utilisé comme jeu de données de base de ce travail de recherche.

1.3 Structure du document

Le reste de ce document est organisé comme suit : Après, une introduction quant à la motivation d'utiliser des distances réseau sémantique, il s'amorce dans le chapitre 2 avec un état de l'art commençant par un rappel des notions de réseau, d'apprentissage automatique et de la définition de distance nécessaire à ce travail. Il traite de la création et de l'utilisation actuellement faite de ces dernières. Cet état de l'art se compose également de deux autres parties correspondant à une étude des méthodes de classification des flux réseau (y compris à des fins de détection de trafic malicieux) et de la détection ainsi que du blocage des activités de scanning réseau.

Cet état de l'art est suivi dans le chapitre 2.3.4 par l'étude et l'amélioration théoriques de distances sémantiques de ports, non-encore publiées, développées à l'INRIA par d'autres étudiants dans le cadre d'un travail d'initiation à la recherche scientifique.

Le chapitre 3.2 introduit ensuite les distances sémantiques que nous avons développées dans notre travail de recherche. Ces métriques, ainsi que celles développées par les étudiants précédement mentionnés et les métriques issues de l'amélioration du travail de ces étudiants, sont alors comparées dans le chapitre 4 afin de s'assurer de la sémantique qu'elles portent et de leur utilisabilité dans des applications d'entreprise.

Cette phase de comparaison et de sélection parmi les distances définies s'ouvre ensuite dans le chapitre 5 sur un clustering et une représentation en nuage de points de la meilleure distance développée. Cette représentation a pour but d'exposer plus exhaustivement la structure et les regroupements opérés entre ports par la distance envisagée ainsi que d'exposer la sémantique présente dans cette dernière.

Deux cas d'utilisation sont ensuite abordés dans le but de démontrer la validité et la valeur sémantique des notions de distances créées. Le premier, présenté dans le chapitre 6.2 est une tentative de classification des flux réseau afin de détecter les flux malicieux de machines infectées par des botnets. Plusieurs algorithmes de classification sont utilisés sur les données des distances sémantiques définies afin les algorithmes et métriques soient comparés. Le second cas d'utilisation a pour but de, lors de la détection d'un scan réseau, bloquer préventivement les ports considérés comme étant les plus susceptibles d'être scannés ensuite.

Après l'étude de ce second cas d'utilisation, les résultats de ce travail de recherche seront synthétisés dans une conclusion reprenant les questions de recherche abordées et les réponses qui y ont été apportées. Cette conclusion est proposée dans le chapitre 7 et ouvre de plus le champs aux possibles amélioration de ce travail de recherche.

Chapitre 2

État de l'art

Comme présenté dans la section 1.3, ce document est composé de deux grandes parties. La première concerne notre travail de recherche sur la définition d'une distance sémantique entre les ports réseau. La seconde correspond à une application des distances créées à plusieurs cas d'utilisation. Du fait de cette découpe en parties, cet état de l'art sera également composé différentes parties dont la première consiste en un rapide rappel des protocoles TCP et IP, considérés comme des pré-requis pour une bonne compréhension de ce mémoire, vient ensuite un rappel des techniques d'apprentissage automatique suivi de deux autres parties dédiée aux deux étapes de notre travail de recherche.

2.1 Rappels réseau

Le monde du réseau est composé d'un ensemble de protocoles qui, habilement combinés selon les couches du modèle OSI[39] permettent la bonne transmission des données entre deux machines connectées en réseau. TCP et IP sont deux de ces protocoles.

2.1.1 Le protocole TCP

TCP est un protocole de niveau 4 dans le modèle OSI. Il définit une méthode de communication et d'adressage entre des applications en cours d'exécution sur deux machines distantes. Ainsi, dans le protocole TCP, à chaque application réseau fonctionnant sur une machine, est associé un "port" réseau auquel est lui même associé un numéro de port réseau.

Les ports réseau sont une entité logique indexée numériquement permettant, dans un système d'exploitation, de faire le lien entre les paquets réseau arrivant à une machine et l'application à laquelle ces paquets sont destinés.

Ainsi, une application réseau ouvrira un socket de communication sur un port donné et sera à l'écoute des paquets arrivant sur ce dernier. Le système d'exploitation ayant enregistré le lien entre le numéro de port et l'application. Il est ainsi capable de fournir les dits paquets à l'application une fois que ceux ci arrivent sur l'une des interfaces réseau de la machine.

Les ports réseaux sont séparés, en fonction de leur valeur numérique, en trois classes distinctes. La première classe est dénommée "*well-known*" et comprend les ports allant de 0 à 1023. La plupart des ports de cet ensemble sont liés de façon standardisée à une application particulière. De plus, l'accès à ces ports demande généralement des droits d'administration sur la machine considérée. La seconde classe de ports est nommée "*registered*". Elle reprend les ports numérotés de 1024 à 49.151. Si des conventions sont proposées sur ces ports, ces

derniers ne sont que peu standardisés. Il n'est ainsi pas rare de trouver des applications bien connues fonctionnant pourtant sur des ports exotiques de cet ensemble. La troisième et dernière classe de ports est la classe des ports "*dynamiques*" comprenant les ports entre 49.152 et 65.535. Cette classe est ainsi nommée car l'attribution d'une application à ces ports est souvent faite dynamiquement par le système d'exploitation, à l'ouverture d'un socket, par une application réseau ne spécifiant pas de ports d'ouverture. Aucun lien ne peut donc à priori être fait entre le port et l'application fonctionnant derrière ce dernier.

Il faut noter qu'un port réseau à plusieurs états possibles. Parmi ceux-ci, deux nous intéressent particulièrement dans le cadre de ce travail. Ces deux états sont les états *ouverts* et *fermés*. Un port réseau est en état *fermé* si aucune application n'est actuellement associée à ce dernier. À toute tentative de connexion à un port de ce type, le système d'exploitation répondra donc que le port est *fermé* c'est à dire non accessible. Le second état qui nous intéresse est à contrario l'état *ouvert*. Ce dernier signifie qu'une application est actuellement liée au port et que ce dernier est donc accessible et prêt à recevoir des paquets réseau.

Ces numéros de ports étant pour certains standardisés, certaines applications seront par défaut accrochées à ces ports (au moins d'une modification explicite). C'est de cette manière que le navigateur Web Firefox sait, que pour accéder au serveur Web de l'UNamur, il doit accéder soit au port 80 soit au port 443 de la machine distante. Finalement, pour que les deux applications puissent échanger à l'aide du protocole TCP, elles doivent établir une connexion via un *handshake* établi en 4 étapes mais assurant toutefois, par la suite, que les paquets arriveront tous à leur destination et qu'ils y arriveront dans le bon ordre.

2.1.2 Le protocole IP

IP est un protocole appartenant à l'ensemble de ceux utilisés dans la couche de niveau 3 du modèle OSI (ou couche réseau). Cette couche reçoit les données provenant de la couche transport et les encapsule avant de les fournir à la couche de liaison de données. La couche réseau du modèle OSI et donc en particulier le protocole IP a pour but de permettre aux machines de se reconnaitre selon un adressage spécifique. Dans le cas du protocole IP et dans sa version 4, le format de ces adresses est donné sous la forme de 4 nombres variants entre 0 et 255 séparés par des points (par exemple 127.0.0.1) représentant ainsi 32 bits de données.

2.1.3 La notion de flux réseau

En réseau, les informations échangées peuvent être vues selon différents types de granularité. La première et plus unitaire est le paquet. Un paquet est un ensemble de taille variable de bytes qui représente la plus petite entité pouvant circuler sur un réseau informatique. Il est composé de plusieurs couches correspondant aux couches du modèles OSI, chaque couche étant adaptée au protocole ou à l'application utilisée à cette couche pour le paquet considéré. Ces couches permettent aux différents ordinateurs que le paquet rencontre dans sa route jusqu'à la machine de destination, de savoir par où et comment router le paquet.

Le second et dernier niveau de granularité que nous abordons ici peut être découvert au regard des protocoles réseau établissant une connexion (comme le protocole TCP le fait par exemple) pour effectuer le transfert demandé. Ce second niveau de granularité est le flux réseau. Un flux est un ensemble de paquets échangés entre le début et la fin de la connexion. Les flux réseau peuvent être de deux types : unidirectionnels ou bidirectionnels. Un flux est unidirectionnel si il ne regarde, que les données envoyées d'une machine vers l'autre et

pas les réponses données à ces envois de données. Inversément, un flux est bidirectionnel si il prend en compte les deux sens de la connexion.

2.1.4 Le scan de ports réseau

Le phénomène de scan de ports réseau est une activité pendant laquelle un attaquant ou un administrateur système tente des connexions à une machine distante afin de découvrir quels ports réseau sont ouverts sur cette dernière. Cette activité mêlée à une analyse des entêtes des paquets réçus en réponse permet à l'opérateur de définir l'application fonctionnant derrière le port ouvert et parfois même la version de la dite application.

Ces informations permettent ainsi, à toute personne, de simplement cartographier les applications d'une machine qui sont exposées sur le réseau et de, le cas échéant, forger une attaque tentant de prendre le contrôle de la machine distante.

On distingue plusieurs catégorisations de scans réseau. Une première peut être faite sur la manière dont l'activité de scan cible les ports réseau sur les machines scannées.

Ainsi, on parlera de scan *horizontal* si l'attaquant cible le même port réseau sur une série d'ordinateurs. Inversément, on parlera de scan *vertical* si l'attaquant cible un ensemble de ports sur un machine donnée. Finalement, on parlera de scan par *block* si l'attaquant cible un certain nombre de ports sur un certain nombre de machines.

Un second type de caractérisation des scans réseau peut être faite sur le protocole réseau utilisé ainsi que sur les étapes suivies avec ce protocole pour savoir si le port est ouvert ou pas. Il existe, selon cette caractérisation, de nombreuses méthodes de scans réseau différents. Hors, dans le cas particulier de ce travail, le principal type de scan de ports utilisé est le scan de port TCP SYN.

Ce type de scan de port utilise le *handshake* TCP afin de définir si un port est ouvert. En effet, la première étape de l'établissement d'une connexion TCP correspond à l'envoi par le client et vers un certain port du serveur d'un paquet TCP SYN. Si le serveur accepte cette connexion (c'est à dire si une application est accessible derrière le port ciblé), il répond alors par un paquet TCP ACK. Si il la refuse, par exemple parce que ce port est fermé, la réponse retournée est un paquet TCP RST signalant que la connexion est refusée. Un attaquant voulant scanner une machine distante peut donc, par cette méthode, savoir si le port ciblé est effectivement ouvert ou pas.

2.2 Rappels sur les techniques d'apprentissage automatique

L'apprentissage automatique ou *Machine Learning* est un champs de l'intelligence artificielle dédié à la création d'algorithme améliorant leur capacité de prédiction ou de classification d'entités en fonction d'exemples appris par ces algorithmes.

Ces exemples sont regroupés dans un jeu de données d'instances, une instance représentant un exemple concret du jeu de données.

Chaque instance est, pour sa part, définie par ses caractéristiques propres également appelées *features* qui mesurent différents aspects de l'instance considérée.

Dans un jeu de données particulier, chaque instance devrait disposer des même caractéristiques même si les valeurs de ces caractéristiques peut évidement varier pour chaque instance.

Comme représenté dans la figure 2.1a, un jeu de données peut ainsi être représenté comme un grand tableau à deux entrées, les lignes représentant les différentes instances et les colonnes représentant les valeurs des caractéristiques pour les instances.

	Nom	Prénom	Âge	Taille	Poids
Instance 1	Evrard	Laurent	25	170	60
Instance 2	Deglas	Emilie	24	165	55
Instance 3	Doe	John	25	175	90
Instance 4	Jones	Jessica	55	160	70

(a) Précision de l'algorithme KNN

FIGURE 2.1 – Exemple de représentation d'un jeu de données

L'application d'un algorithme de Machine Learning fonctionne bien souvent en deux phases distinctes. La première est la phase d'*apprentissage* pendant laquelle l'algorithme apprend la structure et la forme des instances d'un petit sous ensemble du jeu de données afin de pouvoir par la suite, dans la seconde phase, *prédire* des valeurs sur des instances inconnues ou encore classifier ces dernières en différents groupes.

Il existe deux principaux types d'apprentissages. L'apprentissage *supervisé* et l'apprentissage *non supervisé*. Dans le cas de l'apprentissage *supervisé*, les données fournie à l'algorithme sont les instances du jeu de données accompagnée de la valeur résultat que l'algorithme devrait fournir pour chacune de ces instances. Dans le cadre d'un apprentissage *non supervisé*, seules les instances sont fournies à l'algorithme. Ce dernier doit alors, de lui apprendre comment classer et regrouper les données entre elles en fonction de leurs caractéristiques propres. Les valeurs de sortie de ce genre d'algorithme pour une instance inconnue sont donc plutôt un groupe auquel rattacher cette instance plutôt qu'une valeur prédite pour cette instance.

Un exemple concret de ces deux types d'apprentissage peut être tiré de la figure 2.1a. Ainsi, pour une tâche de prédiction du poids d'une personne, on fournira à un algorithme d'apprentissage *supervisé* son nom, son prénom, son âge et sa taille ainsi que le poids attendu comme prédiction pour la dite personne. L'algorithme apprendra alors comment à partir des caractéristiques fournies prédire le poids demandé. On fournira à une algorithme *non supervisé* toutes les caractéristiques de toutes les instances du sous ensemble du jeu de données utilisé pour l'entrainement l'algorithme apprendra alors à regrouper les instances entre elles, par exemple selon la corpulence de la personne et son age. La prédiction donnée par l'algorithme pour une personne non connue sera alors un groupe pour cette personne. Par exemple, les personnes jeunes et de corpulence moyenne.

La façon dont les algorithmes apprennent des instances du jeu de données est généralement propre aux algorithmes eux même. Cependant, cet apprentissage est bien souvent guidé par une fonction d'erreur représentant l'erreur faite par l'algorithme avec une valeur particulière de ses paramètres. L'observation de la valeur donnée par la fonction d'erreur permet ainsi à l'algorithme d'ajuster au mieux son fonctionnement afin de réduire la valeur de cette fonction.

Il faut toutefois noter que les algorithmes d'apprentissages ont pour la plupart des paramètres devant être fixés par les utilisateurs de ces algorithmes. Un réseau de neurones aura par exemple comme paramètres, le nombre de couches de neurones, le nombre de neurones dans chaque couche ainsi que la fonction d'activation à utiliser pour chaque couche. Ces paramètres varient généralement en fonction du cas d'utilisation et du jeu de données considérés. Il est donc de la charge de l'utilisateur de fixer ses paramètres afin que l'apprentissage de l'algorithme permettent ensuite la meilleure prédiction ou classification

possible.

Afin de pouvoir comparer les performances des algorithmes avec différentes valeurs de leurs paramètres, des métriques de performance doivent être définies. Pour ce faire, un second sous ensemble du jeu de départ (si possible disjoint du sous ensemble d'entrainement), nommé jeu de données de validation est utilisé afin que la classification ou la prédiction puisse être faite par l'algorithme (muni de paramètres fixés) sur ce jeu de données et que les métriques de performance puissent être appliquée sur ces résultats de classification ou de prédiction.

Dans le cadre d'une prédiction de valeur numérique, une simple différence entre la valeur attendue et la valeur retournée peut être faite pour chaque instance et aggrégée via, par exemple, une moyenne sur l'intégralité du jeu de données de validation.

Dans le cadre d'une tâche de classification en deux classes (l'une positive et l'autre négative), les instances classifiées par l'algorithme sont triées en 4 groupes.

1. Les vrai-positifs (tp) : l'algorithme a défini que cette instance a la classe positive et c'est en effet bien le cas
2. Les faux-positifs (fp) : l'algorithme a défini que cette instance a la classe positive alors qu'il est négatif
3. Les vrai-négatifs (tn) : l'algorithme a défini que cette instance a la classe négative et c'est en effet bien le cas
4. Les faux-négatifs (fn) : l'algorithme a défini que cette instance a la classe négative alors qu'il est positif.

Ces 4 groupes permettent définir deux métriques de performance qui nous intéresserons dans ce travail. La première est la *précision* définie comme suit :

$$precision = \frac{tp}{tp + fp}$$

La précision représente la capacité de l'algorithme a détecter les instances positives parmi l'intégralité des instances proposées. Cette métrique peut être combinée à celle de *rappel* définie de la sorte :

$$rappel = \frac{tp}{tp + fn}$$

Cette dernière donne l'aptitude de l'aptitude de l'algorithme a différencier les instances positives des instances négatives.

Un algorithme fonctionnant bien maximise au mieux à la fois la précision et le rappel. Ainsi, l'observation de ces deux valeurs pour chacune des combinaisons possible des paramètres de l'algorithme permet d'étudier quelle combinaison de valeurs des paramètres de l'algorithme donne les meilleures performance pour la tâche et l'algorithme envisagés.

Il existe de nombreux algorithmes d'appentissage automatique. Les expliquer tous serait une tâche vaine car de nouveaux algorithmes et méthodes d'apprentissage automatique sont introduits ou réintroduits au fil de mois par les chercheurs.

Toutefois, dans le cadre de notre travail, nous avons utilisé certains algorithmes standards que nous introduisons dans les sections suivantes.

2.2.1 L'algorithme KNN

L'algorithme des K nearest Neighbours (ou KNN) est un algorithme supervisé. Ce dernier prend en paramètre les données d'entrée et ne fait que les placer dans un espace à

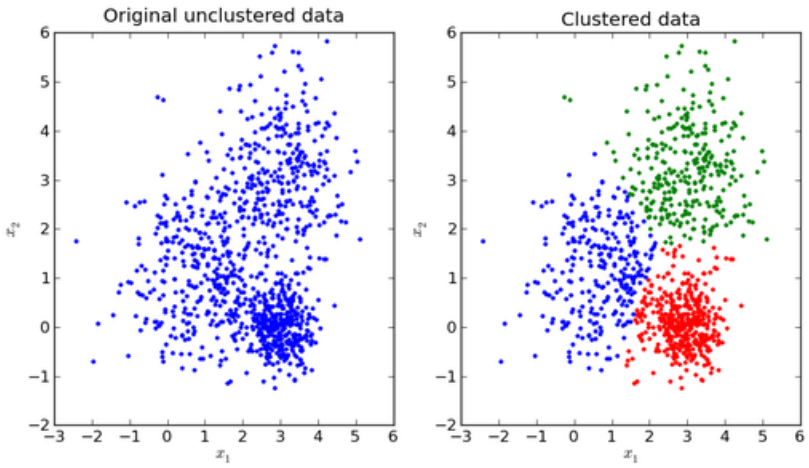


FIGURE 2.2 – Situations initiale et finale de l'algorithme KMeans

hautes dimensions. Ainsi, dans sa phase de prédiction, cet algorithme prend en entrée un point de donnée, le replace dans son espace à hautes dimensions et sélectionne les K autres points les plus proches de ce dernier. Le label du nouveau point de données (la classe à laquelle il appartient) est alors donné comme étant le label le plus représenté parmi les K plus proches voisins de la donnée reçue en paramètre de la classification.

2.2.2 L'algorithme du K-Means

Cet algorithme de clustering (non supervisé) a pour but de découvrir, de par lui même, un regroupement en K ensembles des données reçues en paramètres .

La figure 2.2 représente les situations initiale et finale de cet algorithme en supposant $K = 3$. On peut y voir que KMeans a effectivement trouvé un regroupement des données en trois groupes distincts.

Pour ce faire, cet algorithme commence par fixer aléatoirement K points portant le nom de *centroids*. Ensuite, chaque point dans les données est associé à son plus proche *centroid* formant ainsi trois groupes (*clusters*) dans ces données. Finalement, au sein de chaque groupe, un nouveau *centroid* est choisi avec pour but de placer ce dernier au centre du *cluster* avant que l'algorithme ne recommence une nouvelle itération. L'algorithme termine après un certain nombre d'itérations ou lorsque le déplacement des *centroids*, après chaque itération, est inférieur à un certain treshold.

2.2.3 L'algorithme DBScan

Comme pour l'algorithme KMeans défini plus haut, DBScan est un algorithme de clustering (non supervisé) ayant pour but de former de lui même des groupes (clusters) de points proches les uns des autres.

Dépendant, contrairement au KMeans, DBScan est un algorithme fonctionnant sur le principe de la densité entre les clusters de points. C'est à dire que pour DBScan, un cluster est un groupe de points dont la densité interne est haute séparé par un espace de densité plus faible.

Pour fonctionner, DBScan se base sur deux paramètres. Le premier ϵ représente la distance maximum séparant deux points pour que ceux-ci soient considérés comme liés. Le

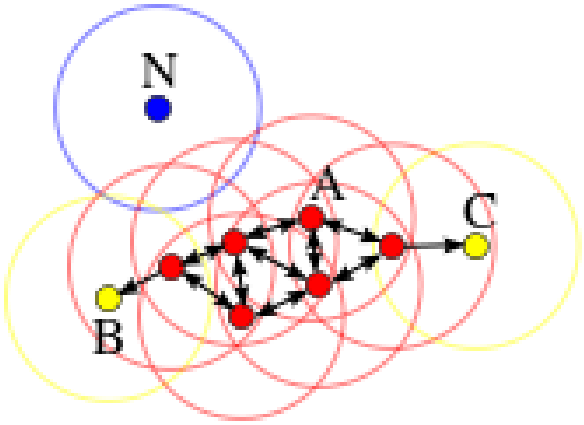


FIGURE 2.3 – Fonctionnement de l'algorithme KMeans

second $MinPoints$ est le nombre minimum de voisins pour qu'un point de données soit considéré commé étant un *CorePoint*. Avec ces deux paramètres, DBScan peut définir le nombre de voisins de chaque point et créer les clusters à l'aide de sa notion de *CorePoint*. Ainsi, un *CorePoint* est un point d'un cluster dont le nombre de voisin est supérieur ou égal à $MinPoints$. Un point lié à un *CorePoint* mais n'en étant pas un lui même, représentera le bord d'un cluster. Finalement, un point n'étant lié à aucun *CorePoint* sera considéré comme du "bruit" dans les données, c'est à dire à une donnée n'appartenant à aucun cluster.

La figure 2.3 représente le fonctionnement de cet algorithme. Dans cette dernière, les points en rouge montrés sont les *CorePoint* et ceux en jaune sont la limite du cluster. Vient ensuite le point bleu qui est considéré commé du "bruit" dans les données.

2.2.4 L'algorithme Merged DBScan

Cet algorithme nommé Merged DBScan et introduit dans [15] représente une méthode non supervisée de classification des données reçues en paramètre selon une méthode assez proche de celle utilisée dans l'algorithme DBScan présenté plus haut. La principale différence entre DBScan et le merged DBScan se situe dans l'ordre des étapes effectuées. Ainsi, le Merged DBScan, à la place d'introduire tous les points de données avant de calculer les clusters, calcule les clusters itérativement lors de l'introduction des points de données.

Pour ce faire, à chaque introduction d'un nouveau point de données, la distance entre ce dernier et tous les centroids de clusters est calculée. Si une ou plusieurs de ces distances est inférieure au paramètre ϵ (dont la définition est identique à celle utilisée pour l'algorithme DBScan), le point est ajouté au cluster avant que le centroid de ce cluster ne soit recalculé pour être le point central du cluster (c'est à dire le point minimisant la somme de sa distance à tous les autres points). Si la distance entre le point de données à ajouter et chacun des centroids de clusters est supérieure à ϵ, un nouveau cluster contenant ce point de données est créé.

Cet algorithme diminue le temps de calcul nécessaire pour l'opération de clustering, dans la mesure ou le nombre de distance à calculer à l'ajout d'un point est bien moindre.

2.2.5 L'algorithme t-SNE

L'algorithme t-SNE a été introduit dans [42] et a pour but d'opérer pour chaque élément i d'un espace de données à hautes dimenions m une réduction dans un espace de données n à dimension beaucoup plus basse.

Cet algorithme se base sur l'idée générale que la distribution probabiliste des données à haute dimension devrait être environ la même que celle des données à basse dimension afin que la structure régissant le placement des données soit respectée.

Pour ce faire, et pour toute paire de points x_i et x_j, il commence par définir une distribution de probabilités p_{ij} que le point x_i soit proche du point x_j dans l'espace à haute dimension. Cette probabilité est définie de la sorte avec $\delta_{ij}^2 = ||x_i - x_j||^2$ la distance entre x_i et x_j :

$$p_{ij} = \frac{exp(-\delta_{ij}^2/\sigma)}{\sum_k \sum_{l \neq k} exp(-\delta_{kl}^2/\sigma)}, \forall i \forall j : i \neq j$$

L'algorithme définit également que la probabilité que les points x_i et x_j soient proches dans l'espace à basse dimensions est calculée de la sorte :

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||y_k - y_l||^2)^{-1}}, \forall i \forall j : i \neq j$$

Le but de cet algorithme étant que ces deux distributions soient les plus ressemblantes possible, il lui est nécessaire de pouvoir comparer ces deux distributions en hautes et en basses dimensions. Pour ce faire, il utilise la divergence de Kullback Leibler définie de la sorte :

$$KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

Ainsi, plus cette valeur de divergence est grande, plus les distributions sont différentes.

L'algorithme peut former la distribution q en basses dimensions par descente de gradient afin de minimiser la valeur de KL.

2.3 Les notions de distance

2.3.1 Définition de la notion de distance

Formellement, une distance est définie comme un *"intervalle qui sépare deux points dans l'espace ; longueur de l'espace à parcourir pour aller d'un point à un autre"*[26]. Cette notion est très communément utilisée dans la vie de tous les jours, par exemple, pour donner le nombre de kilomètres séparant deux villes ou encore une durée entre deux événements (intervalle temporel entre les deux événements).

Mathématiquement, une distance est une fonction $d : E.E \rightarrow R^+$ respectant les propriétés suivantes :

1. Symétrie : $\forall x, y \in E$, $d(x, y) = d(y, x)$
2. Identité de l'indiscernable : $\forall x, y \in E$, $d(x, y) = 0 \rightarrow x = y$
3. Inégalité triangulaire : $\forall x, y, z \in E$, $d(x, z) <= d(x, y) + d(y, z)$

Il existe un grand nombre de notions de distances différentes, chacune adaptée à une ou plusieurs situations particulières. La distance la plus couramment utilisée pour comparer

des éléments x et y de même dimensionalité n est la distance euclidienne calculée de la sorte :

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

Il existe cependant d'autres types de distances permettant d'effectuer ce genre d'opérations sur des éléments x et y de dimension n, tel que la distance de Manhattan calculée de cette façon :

$$d_m(x, y) = \sum_{i=1}^{n} |(x_i - y_i)|$$

De façon plus spécialisée à l'informatique et au traitement du signal, un dernier type de distance nommé distance de Hamming est fréquement utilisé. Cette distance prend en paramètres deux éléments x et y binaires et calcule le nombre de bits qui diffèrent entre l'un et l'autre. Cette opération est mathématiquement exprimée de la sorte en supposant x et y indexable et avec l le nombre de bits dans x :

$$d_h(x, y) = \#\{i | \forall i \in \mathbb{Q}, i < l, x[i] \neq y[i]\}$$

De plus, les notions de distances existantes peuvent également comparer des éléments non numériques. Ainsi, la distance de Levenshtein est par exemple capable de donner le nombre minimal d'opération d'ajout/suppression/remplacement nécessaire à transformer une chaine de caractère A en une seconde chaine B.

Finalement, la combinaison de ces différentes notions de distance peut permettre de comparer des entités à hautes dimensions composées de types de données différentes. Si l'on imagine une personne disposant d'un nom sous forme d'une chaine de caractère et d'un âge sous la forme d'un entier, un calcul possible de la distance entre deux personnes serait de :

1. Calculer la distance de Levenshtein entre les deux noms ;
2. Calculer la distance de Manhattan entre les deux âges ;
3. Combiner ces deux distances via une distance euclidienne entre deux entités à deux dimensions composées des distances précédentes.

2.3.2 Les distances sémantiques

Si les notions de distances expliquées dans la section précédente sont mathématiquement très formellement définies, certains cas d'utilisation demandent une vue plus large que les règles qui régissent les espaces métriques peuvent limiter.

Par exemple, si l'on s'intéresse à la comparaison de mots de la langue française, on voudrait probablement que le mot *cravate* soit défini comme ayant une distance faible au mot *costume*. Or, dans un tout autre contexte lié au Carnaval de Binche, on voudrait également que le mot *costume* (le costume de Gilles) ai également un distance assez faible au mot *orange* (les oranges lancées par les Gilles). Dans un espace métrique traditionnel, la règle de l'inégalité triangulaire force alors les mots *cravate* et *orange* à avoir entre eux une distance relativement faible non souhaitée.

De plus et toujours en s'intéressant à la sémantique du cas d'utilisation concerné, la distance via le réseau routier entre en ville A et une ville B n'est pas forcément identique à la distance entre la ville B et la ville A.

On remarque donc que les distances concernées sont dans ces deux cas mises à mal par l'ajout d'une notion de sémantique dans le cas d'application. En effet, afin de représenter correctement toutes les sémantiques possibles deux limitations des espaces métriques devraient être levées. La première est l'inégalité triangulaire et la seconde, la symétrie de la distance.

On définit ainsi une distance sémantique comme une notion d'interval sémantique dans un espace particulier entre deux éléments de cet espace. Cette distance se doit de respecter la règle d'identité de l'indiscernable mais lève le cas échéant l'une, l'autre ou les deux autres limitations normalement posées sur les espaces métriques permettant ainsi la représentation de sémantique entre ces éléments.

2.3.3 Distance sémantique de ports réseau

Les ports utilisés par les applications sont une information facilement accessible dans du monitoring de trafic réseau. Il faut cependant noter que certains chercheurs tels que dans [33] se questionnent à propos de l'utilisation des ports réseau dans les algorithmes de machines learning à cause de la versatilité de ceux-ci. En effet, la plupart des ports réseau sont affectés dynamiquement aux applications et si des organismes tels que l'IANA [19] ou l'IETF [20] proposent des assignations standards pour les autres ports, les administrateurs serveurs sont libres d'affecter les applications qu'ils déploient au port qu'ils souhaitent. Ainsi, en 2005, des auteurs ont montré [31] qu'une précision de classification de seulement 70% pouvait être atteinte en utilisant les assignations officielles des ports réseau.

De plus, de nos jours, la classification de trafic réseau rencontre de nouveaux challenges avec la recrudescence de trafic chiffré [43]. En effet, la norme actuelle sur internet est de chiffrer autant que faire se peut les connexions et transferts de données. Cette normalisation du chiffrement pose cependant problème dans le cas de méthodes de classification du trafic réseau reposant sur l'analyse en profondeur des paquets réseau tel que [2] qui n'ont de ce fait plus accès aux informations des paquets réseau. Il faut cependant noter que, comme relevé dans [43], les méthodes de classification utilisant des informations statistiques ou comportementales (qui sont de plus considérées comme plus respectueuses de la vie privée) sont moins impactées par cette normalisation du chiffrement et que, de plus, le processus d'établissement de la session permet également de définir, sans pourtant analyser le contenu du paquet, le protocole utilisé dans le flux réseau considéré.

Dans [25] les chercheurs ont étudié le trafic réseau lié aux scans effectués par des attaquants sur un large Darknet [1] et ont construit à l'aide de ces données un graphe de séquence de ports scannés à la suite les un des autres. Ce graphe a montré que des relations particulières existaient entre les séquences de ports scannés et que de, ce fait, cette information réseau pourrait être intéressante dans la classification de trafic.

Ainsi, la faiblesse d'utiliser des informations de ports réseau avec des méthodes avancées telles que des algorithmes de machine learning est le manque de métrique pour appréhender la similarité ou dissimilarité entre les numéros de ports. En effet, ces derniers n'étant pas situés dans un espace métrique, il n'est pas facilement possible de les comparer via des types de distances numériques. De ce fait, la plupart des analyses de trafic reposent sur d'autres caractéristiques [30] ou considèrent simplement le fait que les ports soient ou pas égaux [12, 48].

Quelques efforts, tels que les travaux présentés dans [18] et dans [10] ont été fait pour améliorer les possibilités de comparaison des ports réseau. Dans [18], le but des chercheurs

1. Un darknet est un réseau d'ordinateurs inactifs mais connectés à internet. Cette notion est définie plus largement dans la section 3.2.2.1

est de grouper les flux TCP afin d'identifier un port dominant dans le groupe, si celui-ci existe. Les auteurs dans [10] définissent pour leur part un ensemble de métriques sémantiques sur des entités réseau bien connues et souvent utilisées. Ils proposent ainsi une distance sémantique entre les adresses IP basée sur les classes d'adresse et font de même pour les ports réseau en utilisant les trois classes de ports présentées par les organismes de standardisation, à savoir, les ports *registered*, *well-known* et *dynamic*.

Au meilleur de nos connaissances sur le sujet, la métrique de distance de ports réseau présentée dans [10] est la plus utilisée et représente donc l'état de l'art en terme de distance sémantique entre les ports réseau. Cependant, la notion de distance proposée ne se basant que sur 3 classes de ports, il existe un risque que cette notion de distance manque de finesse dans la granularité utilisée pour représenter les distances entre les ports réseau. C'est pourquoi, dans ce document, nous proposons de construire une distance unique et capable de comparer chacune des paires possibles de numéros de ports réseau.

Notre motivation se trouvant principalement dans le monitoring de sécurité, notre distance est basée sur la connaissance extraite indirectement des activités de scanning (plus spécifiquement de scanning TCP) effectuées par les attaquants. En effet, certaines études telles que [5, 27] montrent que les activités de scans se montrent plus fréquentes et également de plus en plus capables de scanner l'intégralité d'internet en quelques minutes à l'aide d'outils tels que ZMAP[13].

Collecter ce genre d'activités de scan est donc une source d'information assez riche mais nécessite toutefois une approche de minage de données afin d'extraire les connaissances synthétiques telles que celles de notre distance sémantique. Il a été prouvé que les darknets sont efficaces pour monitorer les activités des attaquants à large échelle telle que les attaques DDOS [9]. De ce fait, beaucoup de travaux tels que [16], et [3] se concentrent sur l'analyse et la description d'observations faites au travers d'un darknet. Ce document se concentre pour sa part sur l'utilisation de ces observations dans le but de créer une fonction de distance applicable pour le monitoring de sécurité en temps réel.

Dans un premier temps, nous nous sommes intéressés aux différents travaux se rapportant à la sémantique dans les métriques réseau. Dans ce cadre, les distances les plus utilisées sont celles définies dans [10]. Cet article défini un ensemble de distances entre des entités souvent rencontrées dans le monde du réseau et sa contribution majeure (à notre sens) est une présentation de deux distances d'adresses IPs et de ports permettant un regroupement des valeurs de ces entités en classes pour lesquelles des distances sont alors données.

2.3.4 Travail étudiant précédent sur les distances sémantiques entre les ports réseau

Avant d'introduire nos propres notions de métriques sémantiques de distances de ports réseau, nous introduisons dans cette section trois métriques précédemment définies par des étudiants nous ayant précédé dans ce sujet à l'INRIA de Nancy.

Elles sont introduites à titre de comparaison avec notre propre travail et afin de vérifier de la sémantique que notre travail apporte dans les distances de port.

2.3.4.1 Idée générale

L'idée générale de définition de ces métriques est d'utiliser les pages Wikipedia donnant une explication textuelle de la sémantique des ports réseau afin de calculer une distance entre ces dernières. La supposition derrière ces distances sémantiques est donc que si deux ports sont sémantiquement proches, les documents décrivant ces ports le seront également.

Pour appliquer cette idée, les pages webs sont récupérées, une page par port, avant que trois types de distances ne soient calculées entre toutes les paires possibles de pages (et donc de ports). Les différentes méthodes de calcul des distances sont explicitées dans les sections suivantes.

2.3.4.2 Distance de Jaccard

La distance sémantique de Jaccard repose sur l'indice homonyme.
En supposant deux ensembles A et B, cet indice se calcule de la sorte :

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Appliqué aux pages Web décrivant l'utilisation des ports réseau, cet indice permet de calculer la distance entre deux ports réseau en prenant respectivement comme ensemble A et B l'ensemble des mots présents dans les pages de description des ports [2].

2.3.4.3 Distance Cosinus

2.3.4.3.1 Fonctionnement de la similarité cosinus Soient A et B deux vecteurs, le cosinus de l'angle θ entre ceux-ci peut être calculé de la sorte :

$$cos\_sim(A,B) = cos(\theta) = \frac{A.B}{||A||.||B||}$$

Ce cosinus est une valeur comprise dans l'interval $[-1;1]$ dont la valeur -1 indique des vecteurs résolument opposés, la valeur 0 représente des vecteurs indépendants (orthogonaux) et la valeur 1 représente des vecteurs identiques.

2.3.4.3.2 Vectorisation des documents récoltés Pour appliquer cette distance aux documents récoltés, ces derniers doivent avoir été, au préalable, transformés sous forme de vecteurs. Pour ce faire, la méthode de transformation utilisée est celle du Text Frequency Inverse Document Frequency ou TF-IDF. Cette méthode de transformation calcule l'importance d'un mot dans un corpus de document à l'aide de deux mesures, le tf et l'idf.

Pour un terme t et un document $d \in D$, avec D le corpus de documents on a :

$$tf(t,d) = f_{t,d}$$

Où $f_{t,d}$ représente la fréquence du terme t dans le document d, c'est à dire le nombre d'occurrences du mot dans le document.

Pour le même terme t dans le corpus de document D, l'idf se calcule de la façon suivante :

$$idf(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

À l'aide de ces deux mesures, la mesure $tfid(t,d,D)$ d'un terme t dans un document d pour un corpus D est calculée de la façon suivante :

$$tfidf(t,d,D) = tf(t,d).idf(t,D)$$

Ainsi, T, l'ensemble des mots dans tous le corpus D peut être défini de la sorte :

2. Il est à noter que dans ce cas, on ne prend pas en compte la fréquence de chaque mot dans le document

$$T = \cup_{d_i \in D} \{t : t \in d_i\}$$

On peut alors transformer chaque document d_i du corpus en un vecteur $v_i =< x_1, ..., x_N >$ de dimension $N = |T|$ tel que :

$$v_i =< tfidf(t1, d_i, D), ..., tfidf(t_N, d_i, D) >$$

2.3.4.3.3 Calcul des distances Une fois les documents vectorisés, les distances $d(p_i, p_j)$ entre les ports p_i et p_j dont les documents vectorisés respectifs sont v_i et v_j peuvent être calculées à l'aide de la similarité cosinus de la sorte :

$$d(p_i, p_j) = 1 - |cos\_sim(v_i, v_j)|$$

2.3.4.4 Distance Doc2Vec

L'algorithme Doc2vec est un algorithme de Machine Learning ayant pour but de définir une similarité entre tous les documents d'un corpus.

Pour ce faire, cet algorithme prend comme fondation l'algorithme Word2Vec ayant pour but d'étudier la similarité entre les mots d'un corpus de documents. Dans ce dernier, les mots sont vectorisés avant d'être donnés comme entrainement à un réseau de neurones ayant pour but de rapprocher les vecteurs représentant des mots syntaxiquement proches dans les documents du corpus d'entrée.

Une fois le réseau entrainé, ce dernier peut être interrogé pour donner les mots considérés comme étant sémantiquement les plus proches d'un mot donné ainsi que la distance entre le mot de l'interrogation et l'ensemble des autres mots du corpus.

Les modifications apportées par Doc2Vec à l'algorithme Word2Vec utilisent une méthode identique tout en rendant possible la comparaison de similitudes de documents dans le corpus.

Dans ce cas de figure, lorsque le réseau est entrainé, ce dernier peut alors être utilisé pour donner les documents les plus proches d'un document donné ainsi que la distance entre ce document et chacun des autres.

De ce fait, l'algorithme peut apprendre du corpus de pages web contenant les descriptions des ports réseau avant que chaque document (représentant un port) ne soit utilisé pour évaluer sa distance à tous les autres documents (et donc à tous les autres ports).

2.3.4.5 Résultats et possibilités d'amélioration

Dans leur article, les chercheurs présentent leurs résultats sous la forme de HeatMap de similarité entre les ports sans toutefois bien expliquer leur utilisation concrète ainsi que les résultats qu'elles représentent. Les chercheurs avancent cependant, que la similarité Cosinus est meilleure sans pour autant expliquer le raisonnement menant à cette conclusion.

De plus, selon nous, l'utilisation de Wikipedia comme corpus de document pourrait entrainer un certain manque de formalisme dans le modèle appris. En effet, les pages Wikipedia étant écrites par des utilisateurs non-experts, la syntaxe et le choix des mots utilisés pourrait fausser l'apprentissage correct des termes du domaine.

Une possibilité d'amélioration de cette distance sémantique serait d'utiliser des documents formellement plus définis tels que, par exemple, les pages web de description de l'utilisation des ports réseau faites par l'IANA.

2.4 Application de la distance sémantique

Afin de prouver l'efficacité de la distance sémantique créée dans la première partie de ce document, nous avons décidé d'appliquer cette dernière à un cas d'utilisation concret. Comme présenté dans la section 1.3, deux cas d'utilisation ont été envisagés. L'état de l'art de chacun de ces deux cas d'utilisation sera donc examiné dans cette section.

2.4.1 Classification de flux réseau de botnet

Un problème récurrent chez les fournisseurs d'accès internet ou ISP [3] ou encore au sein des entreprises connaissant une certaine croissance est la classification du trafic réseau en fonction des applications créant ce dernier. Cette classification peut être effectuée afin de vérifier des critères de qualité de service [33] ou encore aider les ISP à mieux bloquer le trafic réseau malicieux parcourant leurs lignes internet [23].

De nombreuses méthodes de classification de trafic réseau ont été développées par la recherche afin de proposer des solutions à cette problématique industrielle. La taxonomie proposée dans [44] divise ces méthodes en 4 grands groupes.

Le premier est composé des méthodes se basant sur un ensemble de règles appliquées aux numéros de port pour effectuer la classification. Cette méthode est décriée par la recherche dans la mesure ou elle n'atteint au mieux que 70% de classification [31] correcte en utilisant les assignations de ports proposées par l'IANA dû à l'aspect versatile de l'assignation des ports réseau.

Le second groupe est composé des méthodes inspectant en profondeur les paquets reçus afin d'extraire de ceux-ci des caractéristiques qui, à l'aide de règles bien définies, permettent de classer le trafic réseau. Comme le groupe précédent, cet ensemble de méthode est au fur et à mesure abandonné par la recherche à cause des aspects de performances et de respect de la vie privée liés. De plus, l'augmentation de la proportion de trafic chiffré sur internet met en péril cette méthode qui, de ce fait, ne sait pas facilement retrouver les caractéristiques cherchées dans les paquets en clair.

Les troisième et quatrième groupes proposés dans la taxonomie concernée utilisent des méthodes statistiques et comportementales afin de classifier le trafic reçu. Ils étudient ainsi, par exemple, des caractéristiques telles que la taille moyenne des paquets reçus dans un flux ou encore l'interval inter-paquets dans le flux. Ces deux ensembles de méthodes ont l'avantage de ne gérer de façon transparente que les paquets chiffrés aussi bien que les paquets en clair tout en réduisant la charge de calcul nécessaire (ces derniers n'étant fait que sur des statistiques sur les données) et en respectant la vie privée des utilisateurs envoyant les paquets (le contenu du paquet n'étant pas lui même directement inspecté). Finalement, ces deux groupes de méthodes ont également l'avantage de ne pas fonctionner sur des règles données par des utilisateurs mais d'utiliser des méthodes d'apprentissage automatique leur permettant d'apprendre par elles-mêmes la classification à apporter dans le trafic reçu.

Les méthodes utilisant des techniques d'apprentissage automatiques étant plus nombreuses, elles ont elles même été classées selon la méthode utilisée dans [33]. Dans les paragraphes suivants, nous reprendrons, pour l'exemple, une méthode de chacun des groupes proposés afin d'expliquer son fonctionnement ainsi que les idées de fonctionnement partagées par les éléments de ce groupe de méthodes.

3. Internet Service Provider

2.4.2 Méthodes non supervisées

Ces méthodes de classification du trafic réseau utilisent des méthodes d'apprentissage automatique ayant pour but d'elles même tirer une structure des données qu'elles étudient. Dans leur phase d'apprentissage, elles fonctionnent de façon générale en rassemblant les données qui se ressemblent en groupes nommés clusters, de telle sorte que lorsqu'un nouveau point de donnée est présenté à l'algorithme, ce dernier puisse voir si il fait ou non partie d'un cluster.

Dans [47], les auteurs proposent une classification des flux réseau en application via un algorithme Bayésien non supervisé nommé AutoClass lui même introduit dans [7]. Les auteurs de ce travail de recherche introduisent également une mesure H d'homogénéité des clusters qu'ils créent afin de définir à quel point un cluster est dédié à une application particulière. L'algorithme Autoclass peut ainsi apprendre à grouper au mieux les flux réseau complets utilisés en clusters afin de maximiser la valeur de H. Ils montrent alors que plus de 80% des flux peuvent être correctement classifiés via leur algorithme. Le principal problème de cette méthode est l'utilisation de flux réseau complets et donc terminés pour opérer la classification. En effet, l'algorithme doit utiliser une quantité importante de données afin de classifier celle-ci et doit, de plus, attendre que le flux soit complet et terminé pour y réagir. La réaction, dans le cas d'une attaque, ne pouvant donc se faire qu'après coup.

Une second travail, faisant usage d'un algorithme non supervisé et introduit dans [4] corrige ce problème en n'utilisant qu'un certain nombre de paquets se trouvant au départ du flux. L'algorithme non supervisé dont il est fait usage est le KMeans sur des caractéristiques extraites des 5 premiers paquets du flux réseau considéré. Le choix de ce groupe de paquets est fait car les premiers paquets d'une connexion représentent souvent l'établissement de cette dernière et la session qui lui est le cas échéant liée. Comme pour l'algorithme précédent, plus de 80% des flux peuvent ainsi être classifiés correctement mais en utilisant dans ce cas seulement les 5 premiers paquets du flux.

2.4.3 Méthodes supervisées

Les méthodes de classification supervisées portent ce nom car elles utilisent des données labellisées dans leur phase d'entrainement. Ainsi, si un flux réseau est lié à l'application Firefox de Mozilla, l'algorithme apprenant à classer les flux réseaux en application aura accès à cette information lors de sa phase d'entrainement. Le but étant, bien entendu, qu'il soit capable, après entrainement, de classifier un flux réseau de Firefox dans le groupe dédié sans pour autant avoir eu cette information.

De nombreux algorithmes de ce type ont été proposés dans le but de classifier des données réseau. L'un deux, introduit dans [40], propose un algorithme permettant d'atteindre une précision de 98% en ne reprenant que 25 paquets par flux. De plus, cet algorithme a l'avantage de ne pas devoir suivre un flux à partir de son début pour opérer une classification. Il peut ainsi commencer à étudier un flux à n'importe quel moment de celui-ci. Ces deux aspects liés permettent à cet algorithme d'atteindre des performances de calcul suffisantes que pour être utilisées dans des cas d'application réels.

Un second algorithme de cette catégorie a été introduit dans [36] et a la particularité d'utiliser un algorithme génétique afin de constuire deux arbres de décision se basant uniquement sur un certain nombre de paquets (10 semblant être le nombre optimal) pour classifier un flux réseau.

2.4.3.1 Autres méthodes et comparaisons

Il faut noter qu'un grand nombre d'autres méthodes et groupes de méthodes sont également proposées dans [33]. Ainsi, certains algorithmes tels que [37] utilisent à la fois des méthodes supervisées et non supervisées afin de pouvoir classer des applications aussi bien connues qu'inconnues du classifieur car non présentent dans le jeu de données d'entrainement.

De plus, de nombreux documents [14, 46]comparant des algorithmes et leurs implémentations ont été proposés. Cependant, certaines des méthodes et documents proposés sont anciens et reflètent ainsi moins l'état de l'art dans les techniques actuelles d'apprentissage automatique.

Cependant, [44] introduit une méthode de classification des malwares à partir de leur activité réseau en utilisant un réseau de noeud convolutionnel. Cette méthode, plus à l'ordre du jour, fonctionne en sélectionnant des blocs de flux réseau qui sont par la suite introduits dans un réseau de neurones profonds afin que l'application liée à cette partie de flux soit découverte. La précision de ce système est de plus de 99% et promet, après quelques améliorations proposées par les auteurs, une application industrielle possible.

2.4.3.2 De l'utilisation des distances de ports dans la classification de données réseau

Comme cité dans la section 2.4.1, l'utilisation des numéros de ports uniquement est décriée dans des cas de classification de flux réseau par application et donc également dans la détection de trafic malveillant. Cependant, les méthodes non-supervisées et supervisées utilisent cette caractéristique qui combinée à d'autres peut aider dans leur tâche de classification.

Dans cette utilisation, la distance entre des ports réseau est bien souvent calculée via des distances euclidiennes, qui au fond, ne correspondent pas à ce type de données qui n'évoluent pas dans un espace métrique.

La proposition de notre travail est d'utiliser la distance sémantique définie pour comparer les ports réseau avec une distance prenant en compte les liens sémantiques entre ces derniers et de comparer les performances des algorithmes avec et sans l'utilisation de sémantique dans la distance entre les ports réseau.

2.4.4 Blocage préventif de ports réseau lors de scans

Cette partie de l'état de l'art a été effectuée afin de rendre compte des techniques actuellement utilisées pour la détection et le blocage préventif de scan de ports réseau. Cette étude a permis de placer le cas d'application de notre distance sémantique de ports dans le blocage préventif par rapport aux autres travaux actuellement effectués dans ce milieu.

Une première partie de cette étude dédiée à la caractérisation des scans et des campagnes de scans est portée par [5], [6] et [27] et montre que les grandes campagnes de scans sont de plus en plus répandues et sembles de plus en plus capables de scanner l'intégralité des machines connectées à internet en quelques minutes à l'aide de grands scans distribués. Ces études notent également que les scans effectués sont en grande majorité des scans TCP Syn et que ces derniers sont répartis de façon uniforme sur la journée, prouvant que les activités malicieuses liées au scan sont une attaque constante contre la sécurité des machines connectées directement à internet.

Cependant, il apparait également dans [35] que les scans de ports ne semblent pas si fortement précurseurs d'attaques qu'il est coutume de le penser. Ainsi, seul 4% des

scans de ports sont suivis d'attaques. De plus, seuls 21% des scans de vulnérabilités sont effectivement précurseurs d'attaques. En revanche, si l'on étudie les combinaisons de scans de ports, il est notable de constater qu'une combinaison de scan de port et de vulnérabilité mène à une attaque dans plus de 71% des cas.

Il faut cependant noter que les résultats de [35] ont été récoltés en 2005. Or, comme cité précédement, les activités de scan de ports sont en augmentation d'années en années impliquant de ce fait que ces résultats mériteraient d'être recalculés afin de vérifier les tendances résultats de leur observation. Il en ressort cependant que la détection et le blocage de scan peuvent permettre de réduire significativement le nombre d'attaques subies par une machine en limitant ainsi dans le même temps la puissance machine allouée à cette activité indésirable.

La seconde partie de cette étude a ainsi été dédiée à une étude de l'état de l'art en terme de détection et de blocage de scans de ports réseau. Cette étude fait apparaitre que la méthode la plus simple [27] pour détecter le scan de ports réseau est une analyse de la fréquence de nouvelles connexions à des ports réseau différents.

Cette méthode de détection est toutefois aisément contournée par les attaquants en étalant la période de scans ou en distribuant les scans sur un grand nombre de machines attaquantes utilisées comme bots.

Pour circonvenir ce problème, la méthode présentée dans [21] étudie entre autres, le nombre de paquets contenus dans le flux réseau et montrent que des flux réseau contenant 4 ou 5 paquets sont souvent synonymes d'hôtes tentant d'effectuer des scans. Cette méthode donne de très bons résultats tout en promettant, en plus, la détection de scans distribués. Les chercheurs à l'origine de [29] présentent dans ce travail de recherche que cette méthode est toutefois sensible à une attaque permettant à l'attaquant de scanner sans être détecté par l'algorithme. Ce travail introduit alors plusieurs mitigations sur les attaques présentées tout en signalant que les dites mitigations ne corrigent pas pour autant tous les problèmes relevés.

D'autres méthodes encore telles que [32] sont proposées et utilisent une grande variété d'approches différentes telles que des approches statistiques, algorithmiques ou encore basées sur la construction de graphes. Ces travaux sont comparés de façon très efficace dans [33] selon l'approche utilisée et les résultats obtenus.

Il est bon de noter que si beaucoup de méthodes de détection de scans existent, aucune proposition n'est faite sur le comportement à tenir quand le scan est détecté. La solution la plus évidente semblerait être de bloquer l'attaquant uniquement sur le port scanné, cette solution n'empêchant pas ce dernier de continuer son scan sur d'autres ports. Une seconde solution serait de bloquer l'adresse IP de l'attaquant de façon globale. Cette solution serait toutefois fort incapacitante si la machine effectuant le scan fait partie d'un botnet et ne fait donc, par essence, pas que du trafic illégitime. De plus, une stratégie de blocage efficace devrait aussi répondre à la question du temps de blocage optimal à choisir. L'algorithme que nous développons dans ce document est une proposition de réponse à ces questions.

Deuxième partie

Distances sémantiques de ports réseau

Chapitre 3

Développement des distances sémantiques

3.1 Amélioration des distances sémantiques antérieures

Comme présenté dans la section 2.3.4.5, les distances sémantiques définies antérieurement n'étaient pas très bien testées et étaient basées sur Wikipedia dont la véracité documentaire est discutable.

De ce fait, ce chapitre a pour but de présenter l'extraction d'une distance sémantique utilisant le même principe que ces distances antérieures en utilisant toutefois les documents proposés par l'organisme de standardisation IANA.

3.1.1 Extraction des données

Sur son site web, à la page se trouvant en référence [19], l'IANA propose une liste de ports réseau liés au service et à la RFC associés ainsi que la possibilité de télécharger cette page au format XML.

Notre but étant de récupérer les RFCs liées aux ports TCP, nous avons filtrés ces derniers du fichier et avons alors remarqué que seul 125 ports TCP de ce fichier sont effectivement liés à une RFC et que, de plus, des ports assez courants comme le port 80 dédié aux serveurs web étaient pas représentés.

De ce fait, nous avons cherché un moyen complémentaire pour lier le port réseau à ses RFCs associées et avons découvert que la page de wikipedia disponible au lien [45] proposait une liste des RFCs avec le protocole lié.

Nous avons ainsi pu faire le lien entre le protocole des ports ne disposant pas de RFCs dans le fichier de l'IANA et le protocole présenté dans le lien Wikipedia et donc ainsi de lier une ou plusieurs RFCs de l'IANA à ces ports réseau nous permettant de passer de 125 à 138 ports disposant de RFCs. Cette augmentation bien que faible nous permet de récupérer des ports standards tels que le port 80.

La phase suivante de l'extraction des données a alors été de récupérer sur le site web de l'IETF les RFCs liées et décrivant les ports réseau cibles.

3.1.2 Calcul des distances

Une fois les documents de RFCs récupérés du site de l'IETF, le processus de calcul qui a suivi est fort semblable à celui utilisé dans le chapitre 2.3.4.5 lié aux distances antérieures.

Il faut cependant noter que certains ports pouvant être liés à plusieurs RFCs différentes, quand deux ports étaient comparés, plusieurs distances de RFCs à RFCs pouvaient être

calculées. La valeur de distance de port à port fixée a alors été la moyenne de l'ensemble des distances de RFC à RFC.

De plus, comparé à l'implémentation précédente de l'algorithme Doc2Vec présenté dans 2.3.4.5, l'implémentation que nous avons utilisée et qui est le standard de la librairie *Gensim* ne permettait pas de donner la similarité entre deux documents vectorisés après apprentissage. De fait, les vecteurs résultants de l'apprentissage ont alors été récupérés afin qu'une distance Cosinus soit calculée entre ces derniers. Ces méthodes ne doivent cependant pas être comparées ou confondues avec la distance cosinus utilisée précédement dans la mesure ou cette dernière utilisait une vectorisation sur prêt de 23.000 dimensions, via l'algorithme TF-IDF, alors que notre utilisation de l'algorithme Doc2Vec donne des vecteurs appris de 300 dimensions.

Ce calcul a donc créé les trois nouvelles notions de distance sémantique suivantes :

1. La distance *iana_jac* utilisant les données de l'IANA et une distance de Jaccard
2. La distance *iana_cos* utilisant les données de l'IANA et une distance cosinus
3. La distance *iana_doc* utilisant les données de l'IANA et une distance doc2vec

Ces trois notions de distance seront comparés conjointement avec celles introduites ci-après dans la section 4.

3.2 Distances sémantiques basées sur le comportement de l'attaquant

Après une introduction au besoin de créer une distance sémantique entre les ports réseau, suivie par une étude des distances précédement créées dans ce but et une tentative d'amélioration de ces dernières, ce chapitre a pour but de présenter les distances sémantiques développées dans ce travail.

3.2.1 Idée générale soutenant ces distances

Dès l'étape d'état de l'art de ce travail, il nous est apparu que l'aspect dynamiquement changeant de l'attribution des ports réseau serait un problème lors de l'apprentissage d'algorithmes ayant pour but de définir une distance entre ceux-ci.

En effet, si pour les ports "*well-known*" l'attribution des services aux ports est assez standardisée (par exemple, le port 80 pour les serveurs Web), les administrateurs systèmes ont toujours le choix de ne pas se conformer aux standards. Ainsi, il n'est, par exemple, pas rare de trouver des serveurs VPNs sur le port 443 afin de traverser les limitations posées par les firewalls. De plus, dans le cas des ports "*registered*", le cas se complique dans la mesure où, si des conseils sont donnés à propos de l'assignation des applications aux ports réseau, aucune restriction n'est fixée. Il n'est donc pas rare de voir plusieurs applications utiliser le même port. Finalement, dans le cas des ports "*dynamiques*", comme leur nom l'indique si bien, ils sont utilisés par les applications et le choix du lien entre une application et son port est bien souvent fait à l'exécution.

Il est donc très difficile de faire un lien évident entre un port et l'application qui s'y rapporte. Les connaissances des liens les plus connus sont souvent propres à l'expérience de la personne interrogée. C'est de cette connaissance propre aux personnes du métier que nous avons décidé d'extraire nos distances sémantiques.

Pour ce faire, nous avons basé notre travail sur des données de scans de ports réseau. En effet, si un attaquant connait une faille, par exemple, dans les serveurs webs, il sera

sans doute plus enclin à scanner les numéros de ports qu'il sait être liés aux serveurs web. Ces ports seront ainsi scannés ensemble dans un groupe à l'ordre cohérent.

Afin d'extraire ces groupes de ports scannés dans un ordre particulier, nous nous sommes basés sur une représentation de ces dernièrs sous forme d'un graphe qui sera défini plus loin.

3.2.2 Introduction aux données du Darknet

Afin d'extraire la distance entre des ports réseau contenue dans les scans atteignant ces derniers, il nous était nécessaire de trouver un très grand ensemble de données de scans réseau.

De plus, et dans un but de qualité dans l'apprentissage, ce jeu de données devrait n'être composé, au possible, que de scans réseau porteurs d'une possible sémantique. Tout trafic réseau légitime est à proscrire dans la mesure ou ce dernier ne serait pas porteur de la sémantique souhaitée, c'est à dire celle d'un attaquant ciblant les ports qu'il étudie afin de mieux les mettre en péril.

Cette définition de jeu de données correspond à celle du trafic réseau récolté par un Darknet.

3.2.2.1 Qu'est-ce qu'un Darknet ?

Un Darknet est un ensemble d'ordinateurs connectés à Internet et n'hébergeant pourtant aucun service.

Le but de ces ordinateurs est d'être à l'écoute et d'enregistrer le trafic réseau contenu dans l'Internet Background Radiation (ou IBR [34]). Ce dernier est composé de scans et de tentatives d'attaques à l'échelle d'Internet ainsi que, en quantité bien plus faible, de paquets réseau "perdus" provenant d'équipements réseau mal configurés.

Le Darknet est en ce sens comparable au concept de HoneyPot qui permet d'étudier le comportement des attaquants lors de leurs méfaits, hormis que dans le cas d'un Darknet, aucun service n'est actif sur les ordinateurs connectés. Contrairement au HoneyPots qui est lui actif, le trafic réseau entre donc dans le Darknet pour ne jamais en ressortir. Aucune réponse ne lui est ainsi jamais donnée.

Il faut noter que le principe de Darknet ne doit donc pas être confondu avec celui de DarkWeb qui représente lui la face "immergée" d'internet appartenant au réseau Tor et/ou n'étant pas indexée par les moteurs de recherches dits classiques.

3.2.2.2 Source

Afin d'extraire nos distances sémantiques de scans réseau, nous avons utilisé les données de captures réseau combinées de deux Darknets. En effet, ces derniers n'enregistrant que les paquets réseau liés à l'IBR, ils ne contiennent pas de trafic légitime et sont donc des candidats idéaux pour notre cas d'utilisation.

Nos deux Darknets sont deux réseaux /20 (4096 machines) implantés respectivement en France et au Japon.

Pour chacun des deux Darknets, les données à notre disposition sont réparties sur une longue période entre le 5/11/2014 et le 23/10/2017.

Des statistiques générales sur cet ensemble de données sont fournies dans l'article scientifique résultat de ce travail de recherche, en Annexe 2 de ce mémoire. Le tableau 3.1 en synthétise quelques caractéristiques.

TABLE 3.1 – Quelques caractéristiques des données du Darknet

Statistique	Valeur
Nombre de scans par jour	Entre 9 et 17 millions
Nombre de scans ciblant des ports déjà scannés sur la même journée	Entre 400.000 et 1.500.000
Nombre moyen de ports scannés par semaine et par adresse IP	Entre 100 et 900
Valeur médiane du temps interscan sur l'intégralité du darknet	0.02 seconde
Valeur médiane du temps interscan pour une adresse du darknet	100 secondes

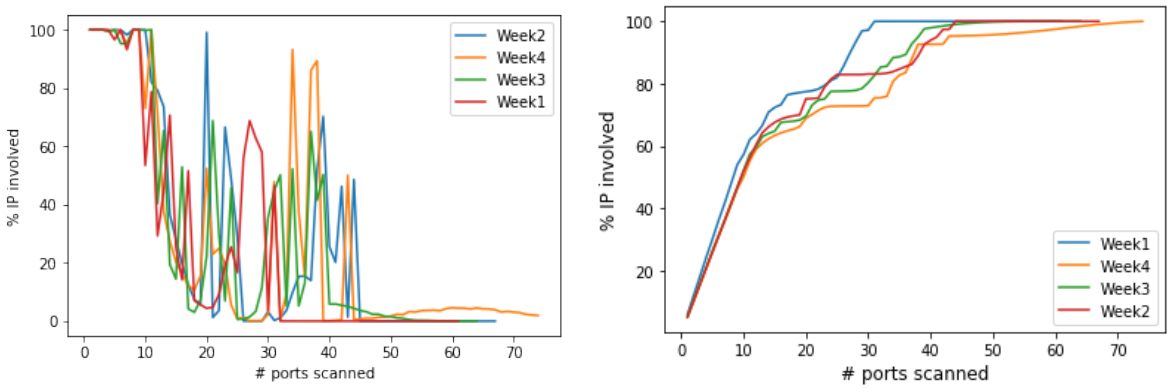
Il faut noter que dans le cadre de travail, nous considérerons les données conjointes de ces deux Darknets. Les deux jeux de données sont ainsi fusionnés en un seul en replaçant les paquets réseau en ordre temporel. En effet, nos besoins envers ce jeu de données n'étant aucunement influencés par des aspects géographiques, cette fusion ne pose à priori aucun problème et nous permet ainsi de disposer d'un jeu de données plus large et plus diversifié.

3.2.2.3 Structure

Les données de nos deux Darknets sont présentées sous la forme d'un ou plusieurs fichiers par jour de capture. Ces fichiers contiennent en particulier les champs suivants :

1. timestamp : La date de capture au format Timestamp
2. date : La date au format texte
3. bytes : Le nombre de bytes dans le paquet
4. src_ip : L'adresse IP source
5. dst_ip : L'adresse IP destination
6. protocol_name : Le nom du protocole réseau
7. tcp_srcport : Au cas où le protocole utilisé est TCP, ce champ contient le port TCP source
8. tcp_dstport : Au cas où le protocole utilisé est TCP, ce champ contient le port TCP destination
9. tcp_syn : Au cas où le protocole utilisé est TCP, ce champ contient le contenu du champ *syn* de ce protocole.
10. tcp_ack : Au cas où le protocole utilisé est TCP, ce champ contient le contenu du champ *ack* de ce protocole.

Les champs ici repris sont ceux ayant servi durant notre travail. Les fichiers en contiennent cependant bien plus afin de décrire toutes les informations réseau relevées par les détenteurs de ce jeu de données.



(a) Données brutes

(b) Somme cumulée

FIGURE 3.1 – Pourcentage d'adresses IP du darknet ayant un nombre fixé de port scanné

3.2.2.4 Sélection des données

Une distinction des scans réseau en vertu du nombre de ports scannés expose trois classes de scans :

Les scans verticaux : Une adresse IP est scannée sur un grand nombre de ports

Les scans horizontaux : Un grand nombre d'adresses IP sont scannées sur un seul port

Les scans par blocks : Une ou plusieurs IPs sont scannées sur un certain ensemble de ports

Il est évident que dépendant de leur classe, les scans sont porteurs de plus ou moins de sémantique. En effet, si un scan horizontal ne ciblant qu'un port, il ne permet pas de donner de sémantique entre les ports réseau. De plus, les très grands scans verticaux ou par blocs contiennent généralement peu de sémantique car ils listent les ports dans l'ordre ou dans le complet désordre mais sans jamais mettre une vraie sémantique dans l'ordre des ports scannés.

Il parait donc évident qu'il est nécessaire, dans notre cas, de filtrer les scans les plus porteurs de sémantique.

La figure 3.1a présente le pourcentage d'adresses IP du darknet ayant vu un certain nombre de ports (par attaquant) scanné par semaine. Elle est accompagnée de la figure 3.1b presentant une somme cumulée de cette valeur. On peut ainsi relever que plus de 40% des adresses IP du darknet ont moins de 10 ports scannés.

On peut ainsi relever dans cette figure que 20% des adresses IPs du darknet sont scannées sur moins de 5 ports et que seul 30% des IPs sont scannées sur plus de 40 ports.

Dans le cadre de l'extraction d'une distance sémantique, nous avons, au vu de ces mesures, décidé, dans ce travail, d'ignorer dans les fichiers des datasets les paquets liés à des scans ciblant moins de 3 ports et plus de 30. Ce choix nous permettant d'éviter les scans horizontaux, verticaux ainsi que les scans par blovs agissant sur un trop grand ou un trop petit nombre de ports que pour contenir une sémantique exploitable.

De plus, dans un but d'uniformité de sens dans la distance sémantique extraite, nous avons également décidé de n'utiliser que les paquets provenant de scans TCP SYN. En effet, combiner plusieurs protocoles et types de scan pourrait diminuer la qualité de notre sémantique dans la mesure ou, par nature, une connexion TCP demande plus de paquets

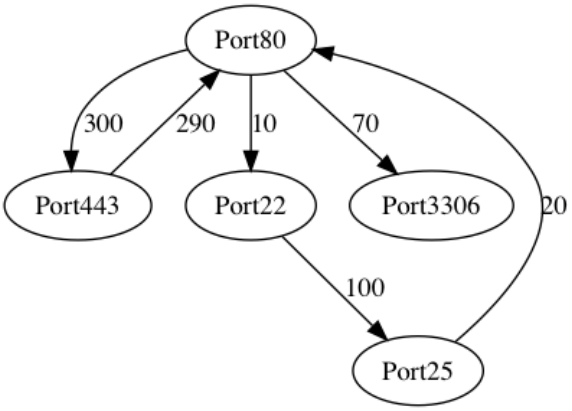


FIGURE 3.2 – Exemple de transformation de scans en un graphe

qu'une connexion UDP. La même méthode pourrait toutefois également être appliquée aux ports UDP.

3.2.2.5 Construction des graphes de ports

Afin de représenter au mieux le comportement des attaquants scannant les machines des deux darknets, nous avons utilisé la méthode proposée dans [25].

Celle-ci propose de modéliser le comportement des attaquants sous forme d'un graphe représentant le passage de ces derniers d'un port réseau à un autre lors de leurs scans.

Ce graphe est un triplet $G < N, E, \sigma >$ avec :

— N : Un ensemble de noeuds $n_1, \cdots, n_j$ représentant respectivement les ports réseau $p_1, \cdots, p_j$.

— E : Un ensemble d'arêtes tel que chaque arête $e_k = (n_i, n_j)$ liant le noeud n_i au noeud n_j représente le passage d'un attaquant du port p_i au port p_j sur un même hôte du Darknet lors d'un scan

— $\sigma(e) : E \rightarrow R$: Une fonction qui, à chaque arête $e_k = (n_i, n_j)$ associe un entier m représentant le nombre de fois que le scénario de passage d'un attaquant du port p_i au port p_j sur un même hôte s'est produit.

Un exemple de cette représentation est donné dans la figure 3.2. Dans ce dernier, sur l'ensemble du fichier de scan réseau, on peut voir que les attaquants ont réalises 10 fois le scénario de passage du port 80 au port 22 et 300 fois le scénario de passage du port 80 au port 443.

Dans le cas de ce graphe, et selon notre sémantique, le poids de l'arête indique donc une similarité sémantique entre deux ports réseau. Plus ce poids est grand entre deux ports plus la similarité est grande.

3.2.3 Distance sémantique basée sur les plus courts chemins

3.2.3.1 Introduction au concept

Le concept derrière cette méthode est d'utiliser le graphe de ports construit ci dessus et, à partir de celui, de calculer les distances deux à deux entre les ports.

En effet, les poids des arêtes du graphe précédemment définies entre les ports permettent de définir une similarité sémantique entre deux ports directement liés induisant que, si deux ports n'ont pas été scannés à la suite l'un de l'autre, ces derniers n'ont pas de similarité sémantique clairement définie.

Le calcul des plus courtes distances permet donc d'obtenir un algorithme capable de lier les ports deux à deux même si ces derniers ne sont pas directement connectés. De plus, cet algorithme assure que la distance sémantique trouvée entre deux ports sera toujours bien la plus petite distance possible.

Il est à noter que le graphe de scan de ports étant un graphe dirigé, cette notion de distance ne respectera donc pas la règle de symétrie de la distance des espaces métriques présentée dans la section 2.3.1. Ce choix delibéré permet la représentation d'une sémantique plus riche autorisant un couple de port à être sémantique proche dans un sens du couple mais éloigné dans le sens inverse.

3.2.3.2 Inversion et normalisation du poids des arêtes

Comment décrit ci dessus, le graphe définit dans la section 3.2.2.5 définit une similarité entre les ports réseau. Le principe de similarité implique que plus cette dernière est grande (plus les éléments se ressemblent) plus le nombre qui la définit est grand également.

Cependant, afin d'utiliser un algorithme de plus courts chemins, la notion nécessaire est une notion de distance dont la valeur diminue donc quand les éléments se ressemblent plus.

De ce fait, avant le calcul des plus courts chemins, une inversion du poids des arêtes du graphe est donc nécessaire.

Plusieurs types d'opérations mathématiques sont possibles pour inverser un nombre et de prime abord, notre choix s'est posé sur l'opération inverse qui, a un nombre x, associe son inverse $1/x$.

Cependant, après une inversion à l'aide de cette méthode, les distances variaient entre 1 et 0.000004177, avec 75% des valeurs inférieures à 0.33 entrainant la diminution de la valeur sémantique des petites distances.

De ce fait, nous avons étudié plus attentivement la distribution du poids des arêtes dans le graphe de port et en avons extrait le boxplot présenté dans la Figure 3.3

Ce dernier présente que plus de 75% des poids des arêtes sont petits alors qu'un faible nombre de poids sont eux très grands représentant donc un ensemble de données mal "distribuées".

De ce fait, une étape de normalisation des données est nécessaire afin de mieux redistribuer les données dans le but d'inverser leurs valeur.

Ainsi, pour éviter de perdre de notre sémantique, nous nous sommes penchés sur une possibilité d'inversion des similarités x en distances d selon l'équation suivante :

$$d = e^{-\alpha * x}$$

Dans cette équation, le paramètre α nous permet de contrôler la vitesse de descente de d lorsque x grandit tel que présenté dans la Figure 3.4.

Ainsi, il apparait que diminuer la valeur de α permet une chute plus lente des valeurs quand la valeur de x devient très grande. De ce fait, nous avons adapté la valeur de α à plusieurs reprises afin de faire correspondre respectivement au premier et au troisième quartile de la distribution la valeur 0.5 (la moitié de la similarité maximum) de la distribution d'entrée des poids des arêtes du graphe.

Pour ce faire, la valeur de α a alors été calculée de la façon suivante, avec p la valeur du percentile souhaité et W l'ensemble des poids des arêtes :

$$\alpha = -\frac{log(0.5)}{perc_p(W)}$$

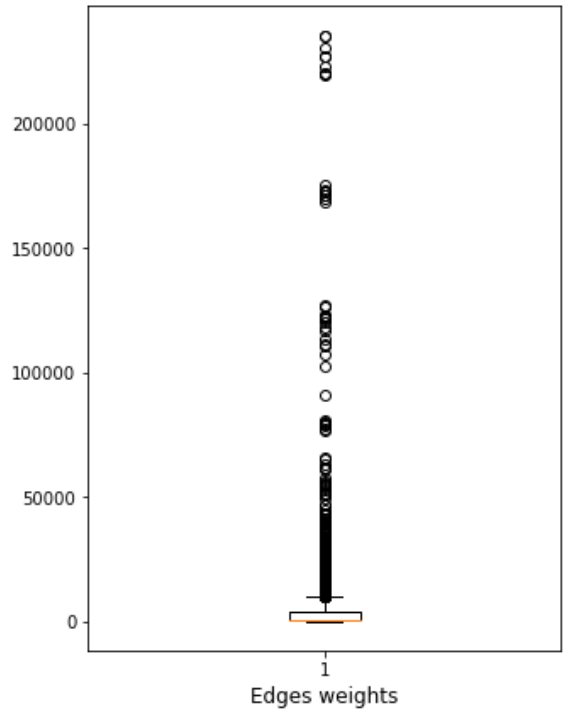


FIGURE 3.3 – Boxplot de répartition de répartition du poids des arêtes

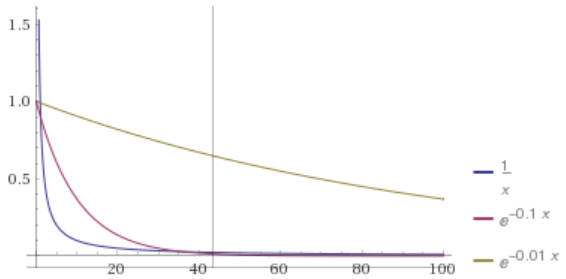


FIGURE 3.4 – Comparaison des fonctions d'inversion du poids des arêtes

Des statistiques de cette expérience peuvent être trouvées dans la Table 3.2.

Dans cette table, on peut voir qu'une inversion centrant sur le troisième quartile fixe un maximum à 0.79 et n'utilise donc pas tout l'espace disponible compris entre 0 et 1. Cependant, les inversions centrant sur le premier ou le second quartile utilisent tout l'espace disponible en laissant assez de place à tous les autres quartiles également. Ces deux méthodes pourraient donc être utilisées et gardées comme des inversions correctes.

Finalement, afin de tenter des méthodes plus standards pour régler ce problème de données mal distribuées, nous nous sommes également intéressés à une méthode d'inversion standard basée sur les quartiles souvent nommée RobustScaler. D'autres méthode d'inversion classiques telles que le MinMax scaler existent. Cependant, le RobustScaler semble plus approprié étant donné qu'il est prévu pour gérer ces problèmes de scaling comportant des outliers. Ainsi, pour un poids w appartenant à l'ensemble de poids W, cette méthode fonctionne de la sorte, avec Q_1 et Q_3 étant respectivement la valeur du premier et du troisième quartile d'une distribution :

TABLE 3.2 – Distrubtion des distances inversées

	Min	**1st quartile**	**2nd quartile**	**3rd quartile**	**Max**
Simple inversion	0.000004	0.018835	0.043478	0.333333	1.000000
1st quartile exponential	0	0.500000	0.740619	0.961592	0.987030
2nd quartile exponential	0	0.201881	0.500000	0.913556	0.970313
3rd quartile exponential	0	0.000005	0.004922	0.500000	0.793701

$$s = \frac{w - Q_1(W)}{Q_3(W) - Q_1(W)}$$

Cette méthode redistribue les valeurs des poids d'entrée en une nouvelle distribution mieux répartie. Toutefois et pour notre cas particulier, il faut noter que les valeurs, qui avant la redistribution étaient inférieures à la valeur du premier quartile, sont alors devenues négatives. De ce fait, avant inversion, toute la distribution doit être translatée vers les valeurs positives en ajoutant à chaque valeur à inverser le minimum de la distribution. Ainsi pour chaque valeur à inverser s appartant à l'ensemble de valeurs S redistribué, l'inversion et la translation peuvent se faire via l'équation suivante :

$$s' = (max(S) - min(S)) - (s - min(S))$$

Finalement, pour chacune des trois solutions proposées, les poids normalisés et inversés sont alors ré-assignés aux arêtes afin que les plus courts chemins puissent être calculés sur les noeuds liés par ces dernières.

3.2.3.3 Calcul des plus courts chemins

Une fois l'inversion du poids des arêtes effectuée, le graphe de scans de ports peut être réutilisé afin de calculer les plus courts chemins entre tous les noeuds du graphe.

Pour ce faire, nous avons décidé d'utiliser l'algorithme de Dijkstra. Cet algorithme, présenté dans [**?**], permet de calculer les plus courts chemins entre un noeud n_s d'un graphe pondéré G et tous les autres noeuds $n_i \in G$. Il peut ainsi être représenté par la fonction $sp : E \to R$ présentée dans l'équation ci dessous avec x_i la taille du plus court chemin entre n_i et n_s.

$$sp(n_s) = \{(n_i, x_i) | n_i \in G \text{ and } n_i \neq n_s\}$$

Dans cette équation et comme expliqué plus haut x_i représente la taille du plus court chemin entre les noeuds n_s et n_i et est défini comme la somme du poids des arêtes représentant le plus court chemin entre ces deux noeuds.

Ainsi, une fois les plus courts chemins calculés entre tous les noeuds du graphe, ces derniers ont pu être sauvegardés dans un format brut donnant vie à notre première distance sémantique.

3.2.4 Distance sémantique basée sur les communautés de ports

3.2.4.1 Introduction et différence avec la distance précédente

Comme pour la distance précédente, celle que nous allons présenté ici est basée sur le graphe de scans de ports défini plus haut.

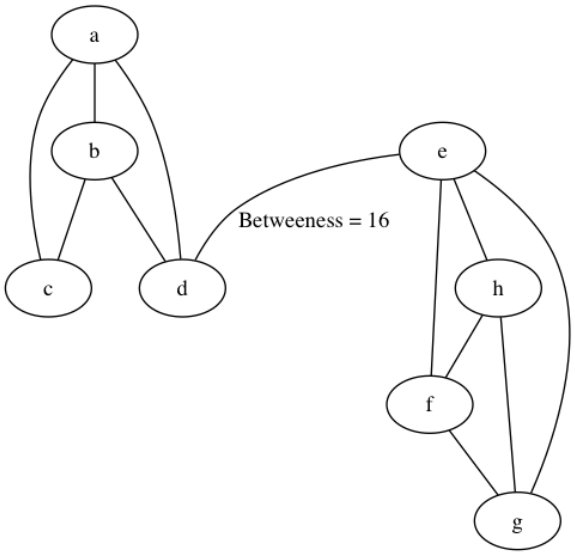


FIGURE 3.5 – Graphe d'exemple pour la distance de communauté

Cependant, à la place d'utiliser une notion de plus court chemin entre tous les noeuds du graphe, notre méthode sera d'extraire une distance des communautés présentes dans ce graphe.

Une communauté dans un graphe est un groupe de noeuds fortements liés entre eux et moins fortement liés avec les noeuds extérieurs à la communauté. Ainsi, un graphe est composé d'un ensemble de communautés de premier niveau qui, prisent séparément, représentent également des graphes eux aussi composés de communautés que nous qualifierons de niveau deux. Cette découpe hiérarchique peut être continuée indéfiniment jusqu'à obtenir un graphe complètement déconnecté.

L'idée de la notion de distance que nous allons créer ici est donc de trouver les communautés dans ce dernier avant d'extraire une notion de distance dans l'emboitement des communautés les unes dans les autres.

Au vu de cette notion d'emboitement entre les communautés, cette notion de distance devrait donc, contrairement à la précédente qui ne s'attaque qu'au lien entre deux ports donnés, être plus porteuse de la structure globale du graphe de scans de ports généré.

3.2.4.2 Présentation de l'algorithme GN

Un algorithme couramment utilisé pour découvrir les communautés dans un graphe est l'algorithme de Girvan Newman aussi appelé algorithme GN.

Cet algorithme, introduit dans [17], se base sur la notion de centralité (ou *betweeness*) des arêtes dans le graphe, c'est à dire le nombre de plus courts chemins passant par chaque arête, afin de définir quels sont les arêtes liant les communautés du graphe et supprimer ces dernières. Par exemple, comme présenté dans la figure 3.5, si l'on enlève l'arête pourtant marquée comme ayant la *betweeness* de 16, c'est à dire celle ayant la plus grande centralité, les deux communautés de gauche et de droite seront déconnectées et les communautés pourront ainsi être découvertes.

Dans cet algorithme, l'arête ayant la plus grande centralité est alors enlevée avant que les centralités ne soient recalculées et que l'algorithme ne réitère. Ces itérations terminent lorsqu'un certain treshold de centralité (défini par l'utilisateur) est atteint.

3.2.4.3 Présentation des modifications sur l'algorithme

Dans le cadre de notre utilisation, nous avons réimplémenté l'algorithme GN tout en le modifiant afin qu'il fonctionne sans treshold et donc jusqu'à une déconnexion complète du graphe.

Ainsi, dans la figure 3.5, les arêtes seront enlevées dans l'ordre de leur centralité. Il est à noter toutefois que lorsque l'algorithme a le choix entre plusieurs arêtes de même centralité à supprimer, il en sélectionnera une de manière aléatoire.

3.2.4.4 Calcul des distances à partir des communautés

Pour continuer notre explication sur l'exemple donné dans la figure 3.5, lorsque toutes les arêtes ont été supprimées selon l'ordre de leur numération, ces dernières sont réintroduites dans l'ordre inverse de leur suppression afin de recréer toutes les communautés possibles du graphe.

Ainsi, quand une arête est réintroduite, elle crée une nouvelle communauté contenant les entités qu'elle lie. Ces entités sont bien entendu des noeuds mais dépendent de l'appartenance de ceux-ci à une communauté déjà ré-introduite, plusieurs cas de calcul de la distance entre ceux-ci se présentent :

Aucun des deux noeuds ne fait partie d'une communauté : La distance entre les deux noeuds est de 1

L'un des noeuds fait partie d'une communauté mais pas l'autre : La distance entre les noeuds est la profondeur de cet arbre de communautés augmentée de 1

Les deux noeuds font partie d'une communauté : La distance entre les deux noeuds est la profondeur maximum des deux arbres de communautés augmentée de 1

Un exemple de ce calcul de distance, basé sur le graphe d'exemple donné en figure 3.6a, est sous forme d'un dendrogramme dans la figure 3.6b. Il est à noter que dans cette figure, les numéros placés sur les lignes horizontales sont les numéros des arêtes qui peuvent être enlevées à ce niveau.

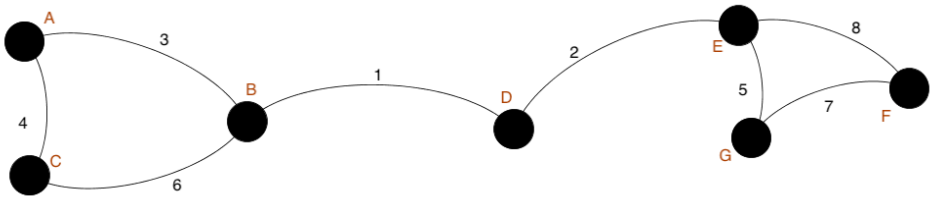
3.2.4.5 Implémentation

L'algorithme complet ayant été implémenté pour cette distance sémantique est défini en pseudo code dans l'algorithme 1.

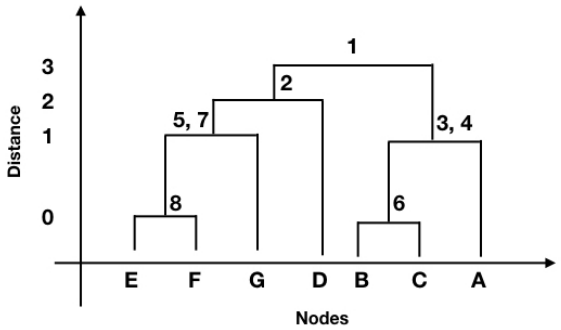
Dans ce dernier, les lignes de 3 à 7 représentent la première étape de l'algorithme c'est à dire la suppression répétitive des arêtes ayant l'edge betweeness la plus grande tout en retenant ces dernières. Les lignes 10 à 37 représentent elles la réintroduction des arêtes et le calcul de la distance entre les noeuds et donc les ports réseau à l'aide des communautés créées. En lignes 14 et 15, une fonction *node_community* est utilisée afin de définir à quelle communauté les noeuds de départ et de fin de l'arête appartiennent. Cette fonction renvoie la plus petite communauté contenant le noeud ou $NULL$ si le noeud n'est contenu dans aucune communauté. Ensuite, les lignes 16, 21, 26 et 31 représentent les différents cas d'appartenance, ou pas, des noeuds à une communauté. À partir de ces derniers, de nouvelles communautés sont créées et les distances sont calculées.

Le résultat de cet algorithme est donc un ensemble d'arbres de communautés contenant au minimum un arbre mais pouvant en contenir plusieurs au cas ou le graphe de départ serait composé de plusieurs composantes connexes.

La dernière étape pour le calcul de distance entre toutes les paires de noeuds est de soumettre cette paire de noeuds à chacun des arbres de communautés afin que ceux-ci



(a) Graphe d'exemple pour la distance de communautés



(b) Dendrogramme pour la distance de communauté

FIGURE 3.6 – Exemples pour le calcul de distance à partir des communautés

retournent la distance associée à la plus petite communauté contenue et contenant conjointement ces deux noeuds. En effet, si la communauté racine de l'arbre contient la paire de noeuds, cela signifie que cette paire peut aussi être contenue dans le noeud racine de l'un des sous arbres de communautés. La recherche de ce plus petit sous arbre assure que la distance retournée pour cette paire de noeuds sera bien la plus petite possible.

Algorithm 1 Compute communities distance

```
1: g ← graph
2: del_order ← Sequence()
3: while |g.edges| > 0 do
4:    edge ← g.max_betweenness_edge
5:    del_order.push(edge)
6:    g.remove_edge(edge)
7: end while
8: communities ← Set()
9: dists ← Set()
10: while |del_order| > 0 do
11:    edge ← del_order.pop()
12:    n_1 ← edge.start_node
13:    n_2 ← edge.end_node
14:    comm_n_1 ← node_community(communities, n_1)
15:    comm_n_2 ← node_community(communities, n_2)
16:    if comm_n_1 = NULL ∧ comm_n_2 = NULL then
17:       dist ← 0
18:       com ← Community(n_1, n_2, dist)
19:       communities.add(com)
20:       dists.add((n_1, n_2, dist))
21:    else if comm_n_1 ≠ NULL ∧ comm_n_2 = NULL then
22:       dist ← comm_n_1.dist + 1
23:       com ← Community(comm_n_1, n_2, dist)
24:       communities.add(com)
25:       dists.add((n_1, n_2, dist))
26:    else if comm_n_1 = NULL ∧ comm_n_2 ≠ NULL then
27:       dist ← comm_n_2.dist + 1
28:       com ← Community(n_1, comm_n_2, dist)
29:       communities.add(com)
30:       dists.add((n_1, n_2, dist))
31:    else
32:       dist ← max(comm_n_1.dist, comm_n_2.dist) + 1
33:       com ← Community(comm_n_1, comm_n_1, dist)
34:       communities.add(com)
35:       dists.add((n_1, n_2, dist))
36:    end if
37: end while
```

Chapitre 4

Comparaison des métriques de distance

Au sortir du chapitre précédent, nous disposons maintenant des distances sémantiques présentées dans le tableau 4.1 que nous voudrions comparer le plus exhaustivement possible.

En effet, si nous pouvons supposer que ces notions de distance disposent d'une sémantique particulière, il semble nécessaire de vérifier empiriquement que cette sémantique est bien présente et que les notions de distance définie sont de qualité suffisante que pour être utilisée dans des applications du monde réel.

Pour ce faire, nous allons commencer par définir un ensemble de ports TCP que nous savons sémantiquement liés avant de, pour chacune de ces distances sémantiques, observer sa représentation pour ces ports de deux façon différentes.

4.1 Définition des ports sémantiquement proches

Cette section a pour but de présenter un ensemble de ports réseau que nous savons avoir des liens sémantiques entre eux. Chaque port est ainsi présenté dans le tableau 4.2 avant que les ports sémantiquement proches ne soient liés et que la raison de ce lien sémantique ne soit donné dans l'Annexe 1.

Il est à noter que dans les futures représentations, les distances évaluées le seront entre tous les ports définis dans le tableau 4.2. De plus, si un couple de ports n'appartient pas à la distance sémantique considérée, la distance sélectionnée pour ce couple sera la distance maximum normalisée possible, c'est à dire 1.

4.2 HeatMaps des distances entre les ports

L'idée principale de cette méthode de comparaison des distances sémantiques est de représenter les distances des ports sélectionnés visuellement sous forme de HeatMaps. Ces dernières permettent de comparer de façon assez simple les distances selon les critères suivants :

La diversité des distances : Les distances représentées ne sont-elles pas toutes identiques ou presques ? Les écarts entre petites et grandes distances sont-ils suffisement subtils ?

La cohérence des distances : Les ports sémantiquement proches sont-ils bien effectivement proches dans les représentations ?

TABLE 4.1 – Distances sémantiques créées

Nom	**Données**	**Distance**	**Information supplémentaire**
wiki_jac	Wikipedia	Jaccard	
wiki_cos	Wikipedia	Cosinus	
wiki_doc	Wikipedia	Doc2vec	
iana_jac	IANA + IETF	Jaccard	
iana_cos	IANA + IETF	Cosinus	
iana_doc	IANA + IETF	Doc2vec	
graph_first	Graphe de scans de ports		Normalisation par le premier quartile
graph_second	Graphe de scans de ports		Normalisation par le second quartile
graph_robust	Graphe de scans de ports		Normalisation par le RobustScaler
graph_commu	Graphe de scans de ports		Distance de communautés

TABLE 4.2 – Ports sélectionné

Ports	
N°	**But**
20	FTP
21	FTP
22	SSH
23	Telnet
25	SMTP
80	HTTP
109	POP
110	POP
115	SFTP
143	IMAP

Ports	
N°	**But**
220	IMAP
443	HTTPS
465	SMTPS
513	Rlogin
514	Remote Shell
556	Remote File System
989	FTPS
990	FTPS
992	Telnet sur SSL
993	IMAPS

Ports	
N°	**But**
1433	MSSQL
1434	MSSQL
1521	Oracle SQL
2049	NFS
2483	Oracle SQL
2484	Oracle SQL
3306	MySQL
3535	SMTP Alternatif
5432	Postegresql

Le nombre de pairs de ports dont la distance a pu être représentée : La distance couvre-t-elle suffisamment de ports ?

4.2.1 Distance wiki_jac

Dans cette notion de distance, visible dans la figure 4.1a, on peut remarquer que certains aspects sémaniques sont très bien représentés. Ainsi les ports FTPs (20,21 et 115) sont effectivement représentés proches. De plus, certaines similarités représentant du chiffrements sont également bien représentées entre les ports 443, 465 et les ports ports 989 à 993.

Cependant, il faut noter que dans cette notion de distance, les écarts entre ports proches et éloignés sont assez importantes menant à une faible utilisation des distances intermédiaires. Cela peut montrer un manque de sensibilité de la distance à de faibles différences.

4.2.2 Distance wiki_cos

Comme pour la distance précédente, celle-ci, présentée dans la figure 4.1b, représente assez bien les distances FTP et celles de chiffrement. Cependant, en opposition à la précédente, cette distance semble mieux utiliser l'ensemble de l'espace de distances possibles avec un plus grand jeu de couleurs utilisées.

Cet aspect entraine toutefois le fait que les distances faibles semblent rétrospectivement plus proches et peuvent donc sans doute être moins facilement être comparées.

4.2.3 Distance wiki_doc

La première chose à noter à propos de cette notion de distance, présentée dans la figure 4.1c, est qu'elle ne parait pas être symétrique. En effet, si certains ports tel que le port 220 ont une ligne à valeur constante de 1 (prouvant qu'aucune distance n'existe de ce port vers l'un des autres ports sélectionnés), ces lignes ne se retrouvent pas dans les colonnes. On peut de ce fait avancer que l'on ne peut via, cette distance pas aller du port 220 au port 3535 mais que l'on peut à l'inverse aller du port 3535 au port 220.

Hormis ce fait, on peut noter que les distances de ports FTPs sont assez bien représentées ainsi que les distances de chiffrement expliquées ci dessus. De plus, l'utilisation des distances et donc des couleurs possibles semble assez uniforme parmi l'ensemble des distances possibles.

4.2.4 Distances basées sur les données de l'IANA.

Pour chacune des trois distances suivantes présentées respectivement dans les figures 4.1d, 4.1e et 4.1f, on peut voir que fort peu de distances sont effectivement définies rendant donc ces trois notions de distances impraticables pour des réels cas d'utilisation. En effet, toute distance non représentée doit être considérée comme étant la distance maximum possible (c'est à dire 1). Donc, au vu du très faible nombre de distances effectivement représentées, toute autre sera fixée à la distance maximum et tout algorithme utilisant ces distances sera ainsi biaisé par cette distance arbitrairement et non sémantiquement fixée.

4.2.5 Distances du graphe de port normalisées par les quantiles

Ces deux notions de distances sont visibles dans les figures 4.0g et 4.0h et montrent une sensibilité quasiment nulle dans la façon dont les distances sont représentées en utilisant

des distances très faibles ou de très moyennes à très hautes, sans réelle gradation entre ces palliers.

4.2.6 Distances des plus courts chemins du graphe de port normalisées par le robust scaler

La heatmap représentant cette notion de distance est accessible dans la figure 4.0i.

Dans cette dernière, on peut relever une utilisation des différentes distances possibles relativement uniforme avec une utilisation des distances moyennes plus fréquente que les distances hautes et basses.

Cette distance semble assez bien représenter la sémantique d'utilisation conjointe des ports réseau en plaçant assez proches les ports 21 à 80 généralement utilisés conjointement dans le cadre de serveurs webs.

La sémantique représentant le but de l'application se cachant derrière un port semble également assez bien représentée au vu des distances utilisées entre les applications dédiées aux bases de données.

De plus, on peut également noter que fort peu de distances ne sont pas représentées et ont donc un valeur de 1. Ces dernières sont principalement situées auprès des ports 989 et 990 utilisés pour le FTPs.

Il faut cependant noter que cette notion de distance n'est pas symétrique. Cela s'expliquant par la non symétrie des distances des chemins liant les noeuds du graphe de scans de ports.

4.2.7 Distances de communautés du graphe de port

La heatmap représentant cette notion de distance est représentée dans la figure 4.0j.

De cette figure, on peut relever que les distances représentées sont le plus souvent très faibles et que le passage entre les distances très faibles et moyennes s'opère comme un "saut" brusque.

De plus, en dehors du port 220 pour lequel aucune distance n'a pu être calculée, on peut relever que cette notion de distance ne dépasse jamais une distance de 0,6 montrant que la distance n'occupe donc peut être pas tout l'espace possible qui lui est accordé.

Toutefois, on peut observer que les aspects sémantiques sont bien représentés au vu du fait que ceux précédement expliqués sont présents dans cette heatmap également.

4.3 Graphe des plus courts chemins pour chaque métrique

Afin de comparer les distances sémantiques, cette méthode reprend les K plus faibles au sein de l'ensemble des distances d'une métrique afin de les représenter sous la forme d'un graphe dont les noeuds représentent les ports impactés par ces distances et les arêtes représentent les distances entre ces ports.

Le but de cette représentation est de vérifier que les distances les plus courtes ont un réel intérêt sémantique.

Un second avantage de cette technique est qu'elle donne une représentation structurelle des noeuds impactés par les plus faibles distances. Ainsi, une analyse des composantes connexes des graphes de noeuds ainsi représentés donne une bonne vue de la "forme" de la sémantique créée.

Les sous-sections suivantes présentent cette représentation pour chacune des métriques de distance pré-citées.

4.3.1 Distance wiki_jac

Dans la figure 4.1a, on peut noter que les arêtes ayant les poids les plus faibles ont toutes un poids nul et que ces arêtes ne lient pas des ports porteurs de sémantique. Ainsi, bien que la forte connectée du graphe soit satisfaisante, les liens entre les ports ne sont pas porteurs de sémantique.

4.3.2 Distance wiki_cos

La figure 4.1b présente les 30 plus courtes arêtes de cette distance sémantique. Comme pour le cas précédent, la densité importante des liens dans le graphe montre une bonne connexion entre les ports et une "navigation" simple au sein du graphe mais ne relie pas des ports sémantiquement proches bien que les distances soient de 0 entre tous ces ports.

4.3.3 Distance wiki_doc

La figure 4.1c représente cette notion de distance. Celle-ci n'est cependant pas vraiment porteuse de sémantique car elle lie des ports qui ne sont pas sémantiquement proches tout en ne donnant pas un graphe connecté de façon intéressante. En effet, les ports proches de façon euclidienne sont liés entre eux dans un grand nombre de composantes connexes, elles même déconnectées des autres.

4.3.4 Les distances iana_jac, iana_cos et iana_doc

Bien que composées de beaucoup de composantes connexes dont les ports ne sont pas porteurs de sémantique, il est à noter que ces notions de distance, présentées dans les figures 4.1d, 4.0e et 4.0f lient de façon intéressantes les ports 465, 993 et 995, qui tous trois, utilisent le chiffrement proposé par la librairie SSL. Il faut cependant noter qu'aucune de ces trois distances n'exposent les 30 plus courts chemins demandés. Cela signifie donc qu'elles ne les contiennent pas et montrent une inefficacité de ces métriques à représenter exhaustivement les différences entres tous les ports réseau.

4.3.5 Les distances graph_first et graph_second

La première de ces deux distances est présentée dans la figure 4.0g. Celle-ci ne contient que peu ou pas de notion de sémantique et peu ou pas de composantes connexes intéressantes.

La seconde, représentée dans la figure 4.-1h donne des résultats sémantiquement très intéressant tout en fournissant une seule composante connexe à la connectivité élevée. Cette dernière représente assez bien les ports fréquemment utilisés ensemble dans une installation serveur en manquant toutefois de représenter les ports ayant le même but sémantique. Ainsi les ports 80, 443 et 8080 ne sont qu'indirectement liés alors qu'ils sont tout trois utilisés le plus souvent dans le cadre de serveurs webs.

4.3.6 Distance graph_robust

La figure 4.-1i représente cette notion de distance et montre d'assez bonnes performances. En effet, en plus de représenter les ports fréquemment utilisés ensemble dans une installation serveur, cette distance présente également les ports dans un but sémantique identique. On peut ainsi observer que les ports 80, 443 et 8080 sont liés ainsi que les ports 22, 23 et 3389 (utilisés dans le contrôle d'une machine distante).

4.3.7 Distance graph_commu

Cette métrique est présentée dans la figure 4.-1j et n'est au vu des résultats présentés pas du tout porteuse de sémantique. En effet, les composantes de ce graphe sont toutes composées de seulement deux ports qui ne sont en aucun cas sémantiquement liés.

4.4 Sélection des notions de distance les plus appropriées

Cette section a pour but de revenir sur les résultats précédents afin d'opérer une sélection parmi l'ensemble des distances sémantiques développées et de voir lesquelles sont les plus appropriées.

Ce choix, bien que subjectif sera fait selon les critères suivants :

— La qualité des sémantiques présentées dans chaque représentation. Dans le cas particulier des graphes de plus courtes distances, cette question revient à regarder si les arêtes présentes sont bien porteuses de sémantique mais également si ces dernières se combinent pour former un graphe d'un bon nombre de ports connexes.

— La bonne répartition des distances des sémantiques représentées dans les heatmaps. Pour les heatmaps, cette question revient à se demander si les distances sont réparties entre des valeurs basses et hautes et façon uniforme (sans grands sauts). Pour les graphes de distance, cette question revient à regarder si toutes les distances sont vraiment faibles et si elles ne sont pas cependant toutes identiques. En effet, des distances toutes identiques montre une difficulté de la métrique à différencier des distances fortement identiques.

— Le nombre de distances correctement représentées dans les différentes représentation. Ce critère permet l'observation des distances représentables via la métrique.

Dans ce but, chaque distance et chacune des observations prisent à propos de cette dernière vont être reprises afin de décider si cette notion de distance sera ou non sélectionnée pour la suite de notre travail de recherche.

4.4.1 Distance wiki_jac

Dans les pages précédentes, nous avons relevé que cette notion de distance représente assez bien certaines sémantiques telles que celles liant les ports de services FTP ou encore celle de ports utilisant le chiffrement proposé par SSL. Cependant, le graphe proposé, bien que fort connexe, n'expose pas une grande sémantique apparente et donne, de plus, à toutes ses arêtes, un poid de 0. De plus les écarts entre les distances sémantiques et non sémantiques dans la Heatmap sont forts importants.

Ces constatations mènent à la conclusion que cette notion de distance est porteuse d'une sémantique assez faible mais toutefois présente et que cette dernière n'est pas située dans les plus courtes distances que cette métrique propose.

Il faut toutefois noter que cette notion de distance semble capable de représenter la plupart des distances demandées (hormis pour les deux ports 110 et 1434).

Au vu des arguments précédents, cette distance est reprise à titre de comparaison comme distance de ports réseau sémantique "candidate".

4.4.2 Distance wiki_cos

Les résultats de cette notion de distance sont presques identiques à ceux de la distance précédente tout en étant cependant plus radicaux au niveau des distances représentées. En effet, les différences entre hautes et basses distances de la heatmaps sont plus grandes

rendant les distances faibles très proches de 0 alors que les distances plus importantes sont toutes très proches de 1.

Il faut cependant noter que cette notion de distance est, semble t'il, capable de représenter toutes les distances sémantiques demandées.

De ce fait, et comme pour la distance précédente, celle-ci est acceptée et reprise à titre de comparaison pour nos futurs résultats.

4.4.3 Distance wiki_doc

Cette notion de distance semble donner des résultats assez distribués au niveau des distances qu'elle propose, tout en proposant des liens sémantiques cohérents. On retrouve ainsi les ports FTPs sus-cités ainsi qu'un groupe de ports représentant assez bien les liens existants entre différents types de bases de données (ports de 1521 à 2483). Cependant le graphe de plus courtes distances dans la métrique expose un grand nombre de composantes constituées d'un faible nombre de ports qui ne semblent pas liés entre eux.

De plus, cette métrique ne permet pas de représenter les distances de certains ports.

Toutefois, au vu du nombre de types de sémantique gardées, cette distance sémantique sera gardée à titre de comparaison pour nos futurs travaux.

4.4.4 Les distances iana_jac, iana_cos et iana_doc

Comme cela a été le cas dans les sections précédentes, ces notions de distance vont être étudiées conjointement. En effet, même si la même sémantique est représentée dans chacune de ces distances de façon différente, ces dernières ne sont pas assez complètes que pour être utilisées efficacement dans un cas d'utilisation concret.

Ces trois notions de distance ne seront donc pas reprises dans la suite de ce document.

4.4.5 Les distances graph_first et graph_second

Les heatmaps de ces deux métriques montrent que ces distances ne sont que très peu capables de faire la différence entre les différents types de sémantique possibles. En effet, toutes les similarités sémantiques étudiées donnent lieux à des distances très faibles sans pour autant se différencier les une des autres.

Une étude des graphes de plus courtes distances montre, dans le cas de graph_first, une incapacité à représenter une sémantique intéressante avec un port lié à beaucoup d'autres avec lesquels les liens sémantique ne sont pas clairs. Dans le cas de graph_second, les liens représentés semblent intéressants mais un examen plus approfondi de ceux ci montre que, comme presque toutes les distances de cette métrique sont nulles, d'autres ports que ceux nous intéressant auraient pu être choisi pour construire cette représentation et que cette dernière n'aurait alors pas eu cet aspect sémantique intéressant. Cette représentation avantageuse n'est donc qu'un "coup de la chance" dans la génération de la figure.

Au vu de ces arguments, ces deux notions de distance ne seront pas reprises dans la suite de ce document.

4.4.6 La distance graph_robust

Les résultats de cette métrique montrent une capacité de la métrique à représenter une de la sémantique entre les ports réseau ayant une utilisation conjointe et/ou un but identique. De plus, l'utilisation de l'espace de distance dédié semble assez uniforme avec la plupart des distances ayant une valeur moyenne et un petit nombre ayant une valeur

importante. Ensuite, la heatmap de cette métrique semble prouver que cette distance est capable de représenter la distance entre tous les ports demandés.

Finalement, le graphe de plus courtes distances est également porteur de sémantique prouvant que cette métrique est capable de représenter aussi bien une sémantique d'utilisation conjointe que de but identique.

Cette distance sera donc gardée dans la suite des travaux de ce document.

4.4.7 La distance graph_commu

Cette dernière notion de distance montre une distance assez importante entre les distances basses et moyennes ou hautes. Certaines sémantiques sont ainsi visibles mais la heatmap aussi bien que le graphe de plus courtes distances montrent que les distances sémantiquement les plus intéressantes ne sont pas les plus faibles.

Cette distance étant quand même capable de représenter une certaine sémantique, elle sera reprise dans la suite de nos travaux.

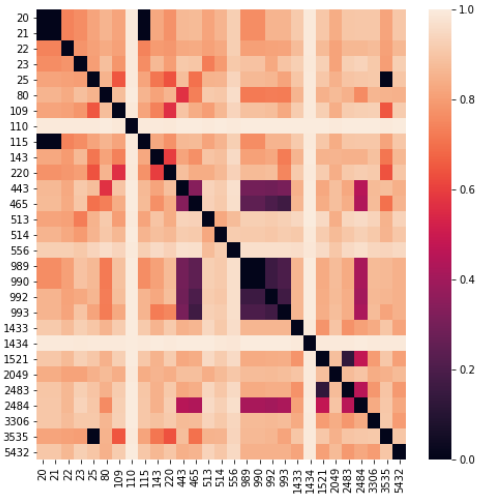
4.5 Résumé de la comparaison des distances sémantiques

Le tableau **??** résume les résultats des différentes notions de distances dans les deux tests de sémantique exprimés plus haut. La dernière colonne de ce tableau reprend la décision de garder ou pas la distance sémantique considérée.

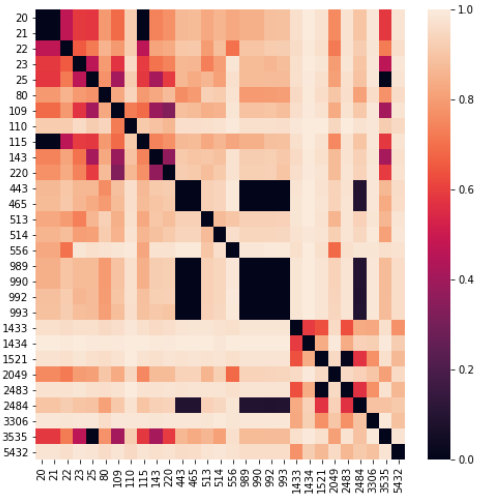
Certaines notions de distance porteuse d'une faible notion de sémantique sont reprises pour la suite de nos travaux à des fins de comparaison avec celles développées dans ce document.

TABLE 4.3 – My caption

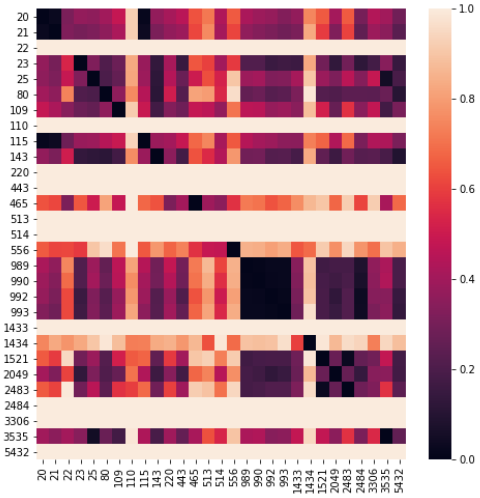
	HeatMaps de ports sémantiquement liés	**Graphe des plus courts chemins**	**Décision**
wiki_jac	Présence de sémantique et bonne distribution des distances	Pas de sémantique et toutes les distances sont nulles	Gardée à titre de comparaison
wiki_cos	Présence de sémantique et bonne distribution des distances	Pas de sémantique et toutes les distances sont nulles	Gardée à titre de comparaison
wiki_doc	Présence de sémantique et bonne distribution des distances	Pas de sémantique	Gardée à titre de comparaison
iana_jac	Trop fragmentaire	Intéressante mais manquant de données	Rejettée
iana_cos	Trop fragmentaire	Intéressante mais manquant de données	Rejettée
iana_doc	Trop fragmentaire	Intéressante mais manquant de données	Rejettée
graph_first	Manque de sensibilité	Pas de sémantique	Rejettée
graph_second	Manque de sensibilité	Intéressante mais toutes les distances sont nulles	Rejettée
graph_robust	Présence de sémantique et bonne distribution des distances	Haute valeur sémantique	Gardée
graph_commu	Présence de sémantique et distribution des distances moyenne	Pas de sémantique	Gardée



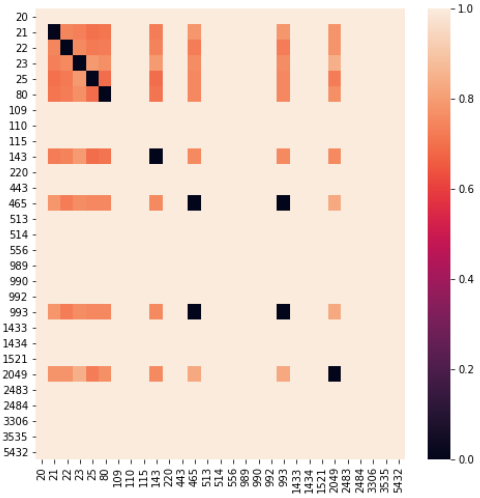
(a) HeatMap de la distance wiki_jac



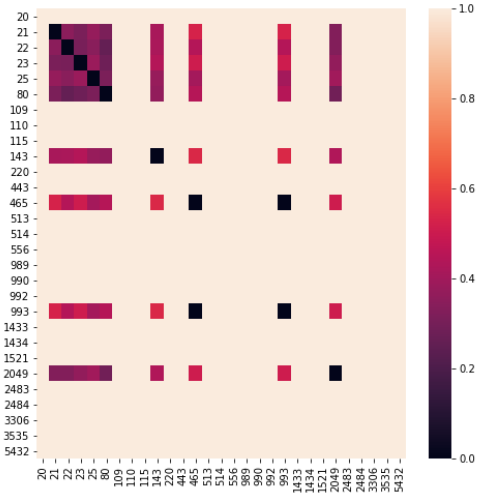
(b) HeatMap de la distance wiki_cos



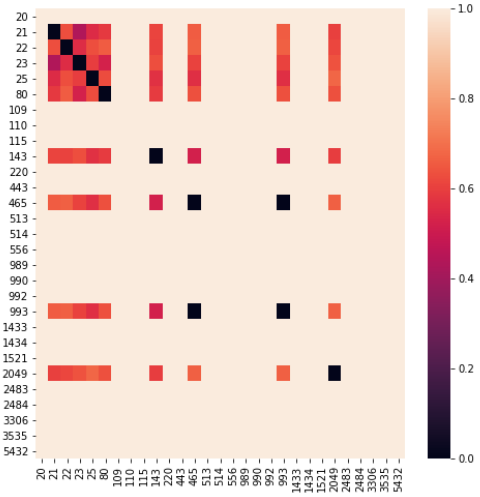
(c) HeatMap de la distance wiki_doc



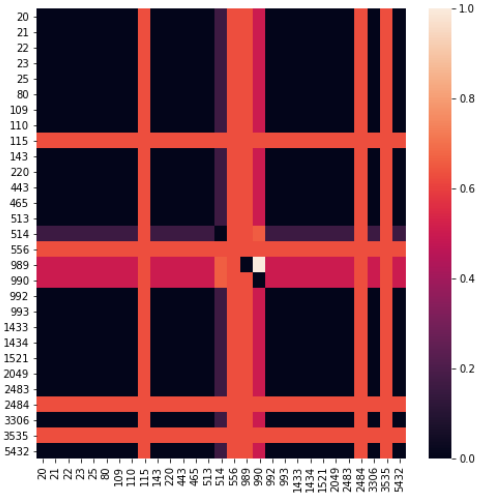
(d) HeatMap de la distance iana_jac



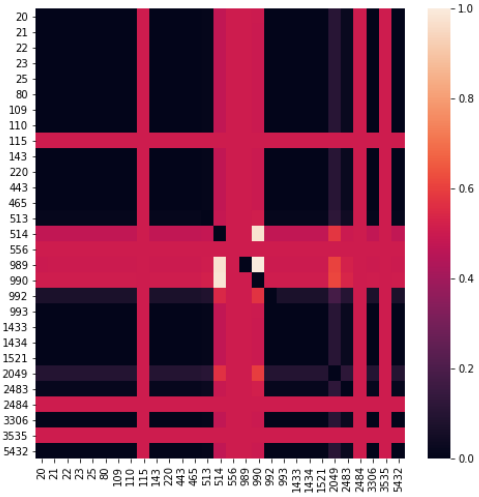
(e) HeatMap de la distance iana_cos



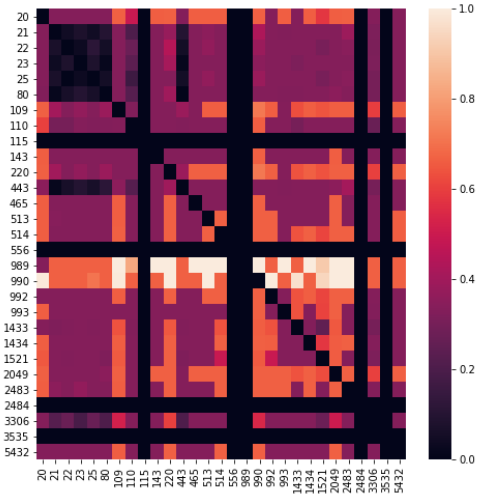
(f) HeatMap de la distance iana_doc



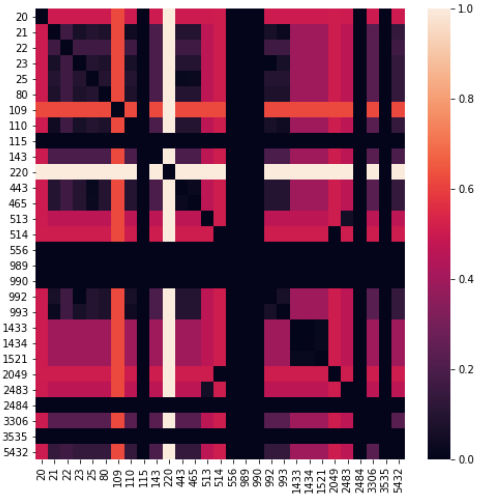
(g) HeatMap de la distance graph_first



(h) HeatMap de la distance graph_second

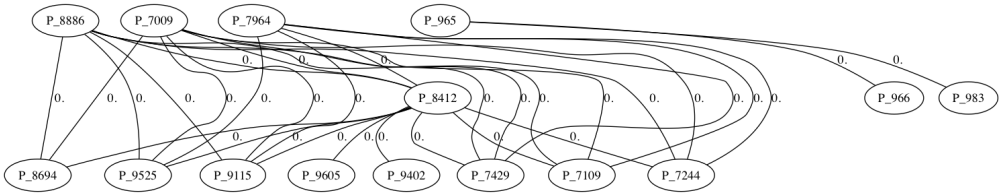


(i) HeatMap de la distance graph_robust

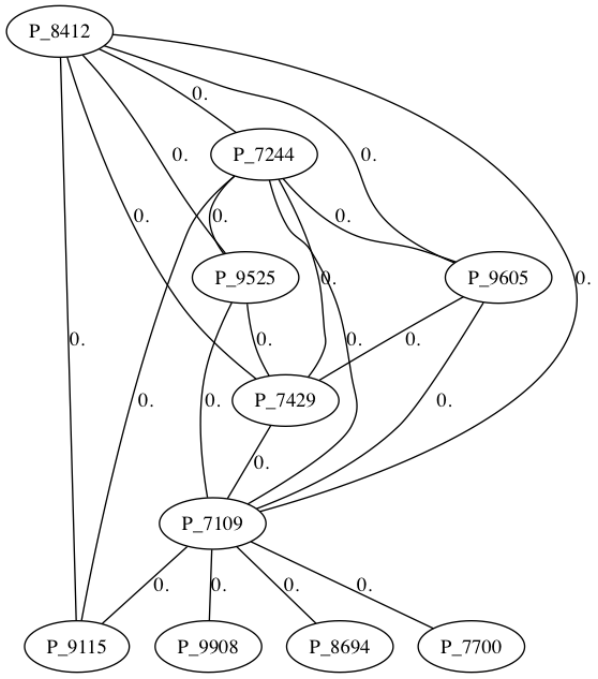


(j) HeatMap de la distance graph_commu

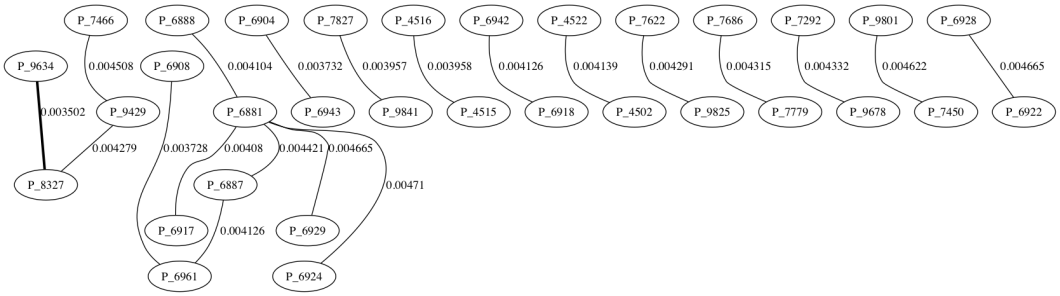
FIGURE 4.0 – Heatmaps de comparaison des distances sémantiques



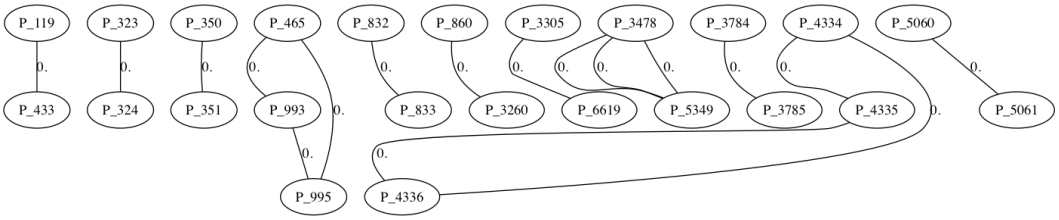
(a) Graphes des 30 plus courtes distances de la métrique wiki_jac



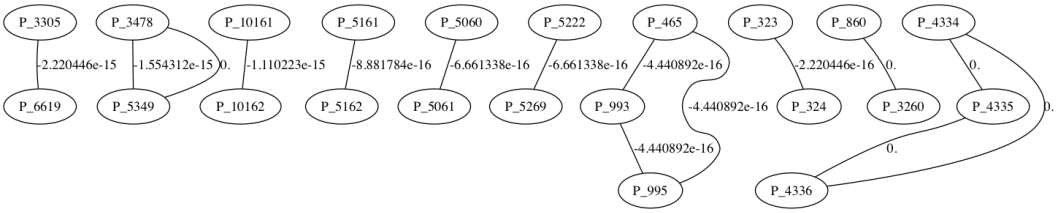
(b) Graphes des 30 plus courtes distances de la métrique wiki_cos



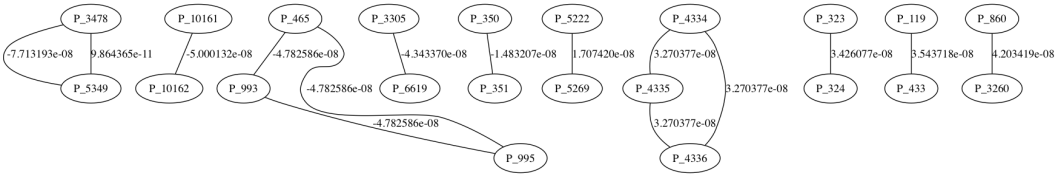
(c) Graphes des 30 plus courtes distances de la métrique wiki_doc



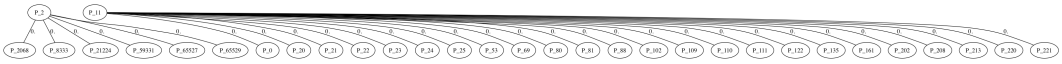
(d) Graphes des 30 plus courtes distances de la métrique iana_jac



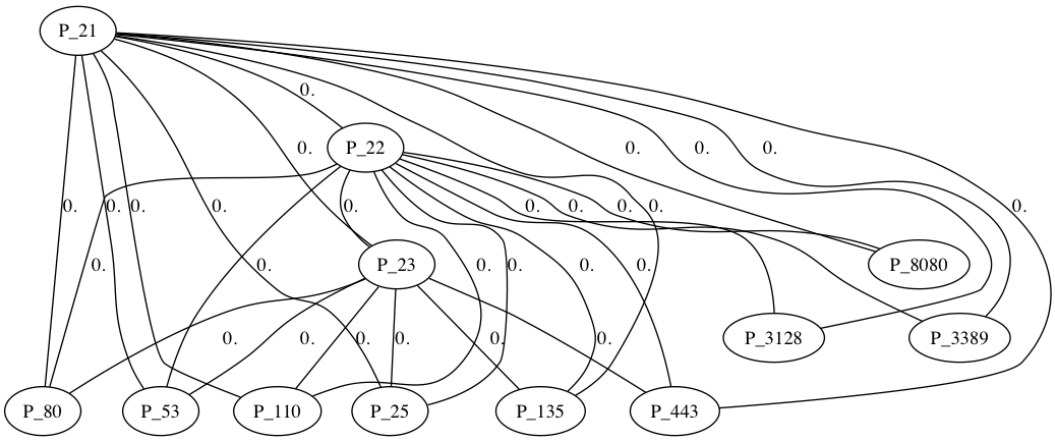
(e) Graphes des 30 plus courtes distances de la métrique iana_cos



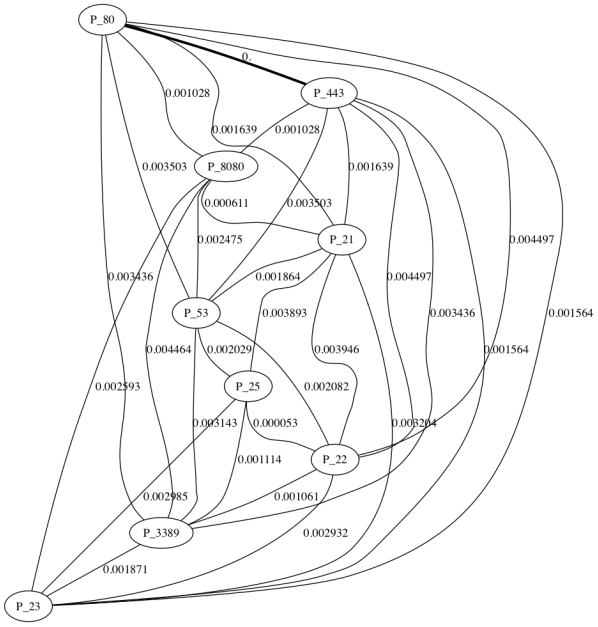
(f) Graphes des 30 plus courtes distances de la métrique iana_doc



(g) Graphes des 30 plus courtes distances de la métrique graph_first



(h) Graphes des 30 plus courtes distances de la métrique graph_second



(i) Graphes des 30 plus courtes distances de la métrique graph_robust

(j) Graphes des 30 plus courtes distances de la métrique graph_commu

FIGURE 4.-1 – Graphes des 30 plus courtes distances pour les métriques

Chapitre 5

Clustering des ports en vertu des distances sémantiques

Dans le chapitre 3, nous avons développé plusieurs notions de distances sémantiques qui ont ensuite été empiriquement évaluées et comparées dans le chapitre 4. Certaines des notions de distances développées ont alors été gardées pour de futurs travaux alors que d'autres ont été rejetées car elles ne portaient pas la sémantique désirée.

Dans ce chapitre, notre but sera d'opérer un clustering de ports réseau à l'aide des distances sémantiques développées. En effet, si nous avons sélectionné dans le chapitre 4 les distances sémantiques les plus porteuses, il est maintenant intéressant d'observer comment, de façon générale, ces distances regroupent les ports dont elles définissent les similarités.

Pour ce faire, la visualisation souhaitée est un nuage de points dont chaque point représente un port réseau particulier. Pour arriver à cette visualisation l'algorithme de clustering utilisé est l'algorithme t-SNE présenté dans [42] et expliqué dans le point 2.2.5 de l'état de l'art de ce document. Ce dernier s'est montré, dans de nombreux exemples, capable de représenter adéquatement des données de hautes dimensions dans des espaces de basses dimensions.

Pour notre cas d'utilisation de cet algorithme, les données fournies sont les matrices de distances sémantiques entre les différents ports réseau et la dimension attendue des données résultat est de 2 afin de permettre une visualisation en tant qu'image.

5.1 Symétrisation de la distance *graph_ robust*

Dans le cas de la distance sémantique *graph_ robust*, celle-ci étant basée sur une notion de plus courts chemins dans des graphes dirigés (voir la section 3.2.3), la matrice de distances n'est pas symétrique. C'est à dire que la distance entre un port x et un port y, n'est pas forcément la même que celle du port y au port x. Par exemple, la distance du port 80 au port 443 n'est pas obligatoirement la même que celle du port 443 au port 80.

Cette non-symétrie de la matrice de distance est problématique pour les algorithmes de clustering et de réduction de dimensions. En effet, si nos notions de distance sémantique peuvent ne pas respecter la symétrie de la distance, l'espace métrique à deux dimensions dans lequel nous voulons voir nos ports réseau représentés y est lui contraint.

Ainsi, une symétrisation de la matrice de distances est nécessaire avant que celle-ci ne soit fournie à l'algorithme t-SNE, pour représentation, dans un espace à basses dimensions.

Une étude de la distribution de la valeur des différences de distances devant normalement être symétrique montre, tel que présenté dans la figure, 5.1, que la différence moyenne

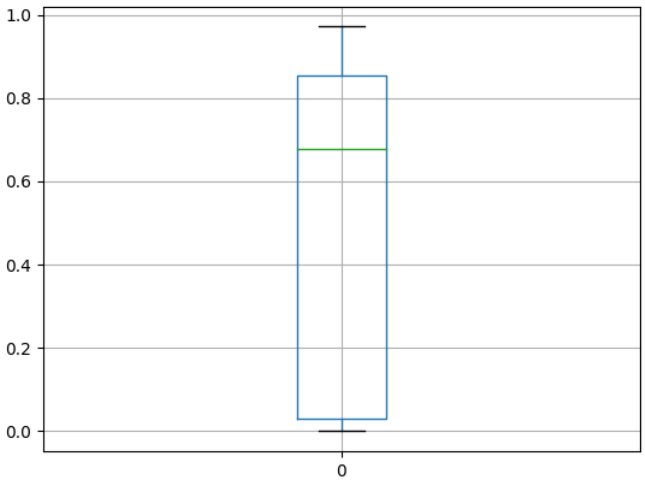


FIGURE 5.1 – Distribution des différences de distances dans la matrice

est de plus de 0.6 (les distances de la matrices étant toutes comprises entre 0 et 1).

Il apparait ainsi qu'une symétrisation de la matrice de distance par calcul de la moyenne entre les éléments de part et d'autre de la diagonale de cette dernière n'est pas envisageable au risque d'effacer les notions de sémantique portées par cette matrice.

Une solution à ce problème est d'utiliser pour deux distances $x = d(p_1, p_2)$ et $y = d(p_2, p_1)$ avec $x \neq y$ la fonction de normalisation suivante :

$$f(x, y) = \sqrt{\frac{x^2 + y^2}{2}}$$

En effet, lors du calcul, par t-SNE, de la distribution des données à hautes dimensions (comme présenté dans la section 2.2.5), cette fonction de normalisation sera alors englobée dans l'équation suivante :

$$p_{ij} = \frac{exp(-\delta_{ij}^2/\sigma)}{\sum_k \sum_{l \neq k} exp(-\delta_{kl}^2/\sigma)}, \forall i \forall j : i \neq j$$

Comme $\delta_{ij} = f(d(i,j), d(j,i))$, cette équation devient donc :

$$p_{ij} = \frac{exp(-\sqrt{\frac{d(i,j)^2}{2} + \frac{d(j,i)^2}{2}}^2/\sigma)}{\sum_k \sum_{l \neq k} exp(-\sqrt{\frac{d(l,k)^2}{2} + \frac{d(k,l)^2}{2}}^2/\sigma)}, \forall i \forall j : i \neq j$$

En simplifiant sur le numérateur on obtient :

$$p_{ij} = \frac{exp(\frac{-d(i,j)^2}{2}).exp(\frac{-d(j,i)^2}{2})}{\sum_k \sum_{l \neq k} exp(-\sqrt{\frac{d(l,k)^2}{2} + \frac{d(k,l)^2}{2}}^2/\sigma)}, \forall i \forall j : i \neq j$$

Ainsi, cette équation permet que la probabilité p_{ij} soit faible si les deux distances sont fort dissemblables tout en préservant une haute probabilité si les distances sont toutes les deux faibles et une basse probabilité si les deux distances sont toutes les deux plus importantes.

TABLE 5.1 – Exemple de matrice de distance

	Port 1	**Port 2**	**Port 3**	**Port 4**
Port 1	0	0.5	0.34	0.9
Port 2	0.5	0	0.7	0.1
Port 3	0.34	0.7	0	0.45
Port 4	0.9	0.1	0.45	0

La notion de distance sémantique *graph_ robust* peut ainsi être symétrisée à l'aide de la fonction f pour chacun de ses couples de distances $d(x, y)$, $d(y, x)$ avec $d(x, y) \neq d(y, x)$.

5.2 Application de t-SNE pour le clustering

Après la symmétrisation des distances devant l'être, ces données ont été fournies en entrée à l'algorithme t-SNE sous la forme d'une matrice de distance pair à pair entre les ports réseau. Un exemple d'une matrice de ce type est présenté dans le tableau 5.1.

Chaque matrice de distance (correspondant à l'une de nos notions de distance sémantique) est donnée à l'algorithme en réglant ce dernier sur son mode *pré-calculé*. Ce mode permet à l'algorithme de considérer que les données reçues ne sont pas un ensemble de points à hautes dimensions dont il doit calculer la matrice distance mais plutôt directement la dite matrice.

L'algorithme ne doit plus alors qu'utiliser la divergence de Kullback-Leibler pour faire converger la distribution de points entre les espaces à hautes et basses dimensions afin d'opérer le clustering (comme expliqué dans la section 2.2.5).

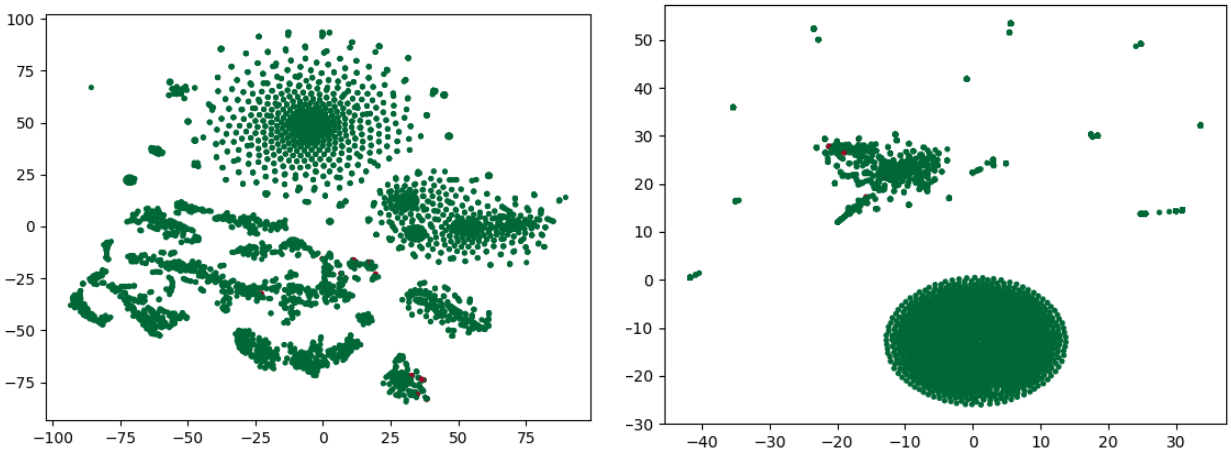
5.3 Résultat du clustering

Dans un but de brièveté, ce clustering n'est appliqué qu'à la meilleure des distances sémantiques que nous avons développées dans le chapitre 3, c'est à dire la distance *graph_ robust*. Toutefois, et à titre de comparaison, une seconde application a également été faite pour une distance à la sémantique plus faible, la distance *wiki_ cos*.

Les résultats de ces clustering sont présentés dans les figures 5.2a et 5.2b.

De ces dernières, on peut relever que la distance sémantique *graph_ robust* contient bien plus de clusters et que ceux-ci sont mieux séparés les un des autres. De plus, les figures 5.3a, 5.3b, 5.3c et 5.3d représentent des zooms sur des parties particulièrement significatives de la figure 5.2a.

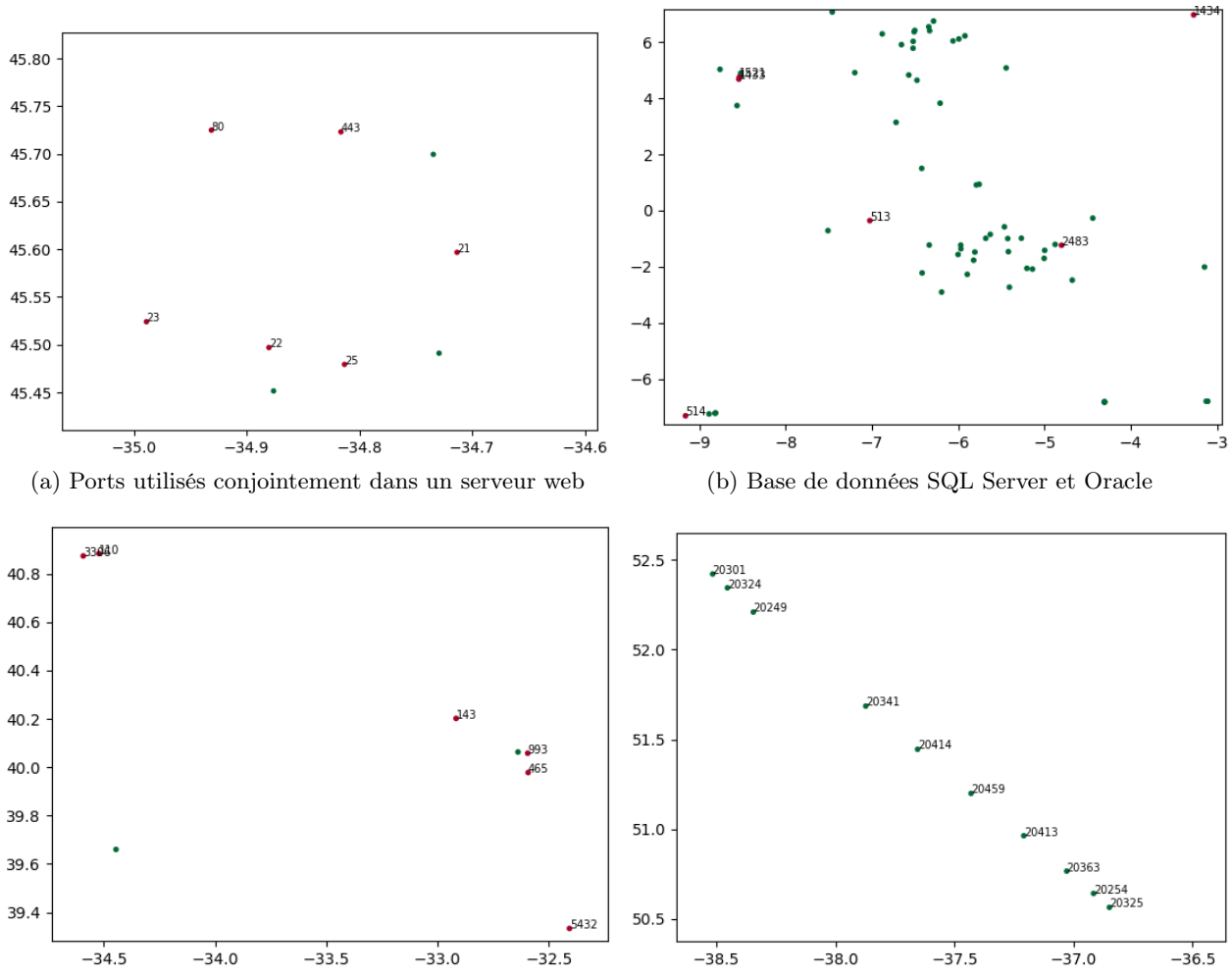
La figure 5.3a contient les ports SSH, web, FTP et mails souvent utilisés conjointement sur des serveurs hébergeant des applications web. La figure 5.3b représentent quant à elle les ports réseau des applications de base de données de Microsoft et de Oracle. La figure 5.3c reprend, pour sa part, des ports réseau liés aux applications de mail. Finalement, lorsqu'aucune sémantique n'est clairement définie entre des ports réseau, la distance *graph_ robust* semble simplement et, comme présenté dans la figure 5.3d, les classer selon un ordre proche de celui donné par une distance euclidienne entre les numéros de ports. précédemment



(a) Clustering pour la distance *graph_robust*

(b) Clustering pour la distance *wiki_cos*

FIGURE 5.2 – Vue générale du clustering pour les distances sémantiques



(a) Ports utilisés conjointement dans un serveur web

(b) Base de données SQL Server et Oracle

(c) Ports de mail et de base de données

(d) Sans sémantique particulière

FIGURE 5.3 – Sémantique dans le clustering de la distance *graph_robust*

Troisième partie

Cas d'utilisation de la distance sémantique

Chapitre 6

Développement des cas d'utilisation

6.1 Présentation du but des cas d'utilisation

Dans les parties précédentes, nous avons démontré l'utilité de distances sémantiques entre les ports réseau avant d'aborder les distances existantes et d'en développer de nouvelles soit sur base d'une méthode existante soit sur une base nouvelle.

Dans cette partie, nous nous concentrerons sur l'application des distances sémantiques définies dans le cadre d'un cas d'utilisation concret et prouvant empiriquement l'utilité des distances sémantiques définies.

Deux cas d'utilisation représentant les deux champs de recherche que nous avons abordés vont être introduits et présentés. Pour chacun, des distances de ports sémantique et non sémantique vont être utilisées dans le cadre de plusieurs algorithmes et ce afin de comparer les performances de ces deux groupes de distances.

Le premier cas d'utilisation s'intéresse à l'application des distances sémantiques à la tâche de classification de flux réseau en flux provenant ou non d'un botnet. Différentes méthodes de classification utilisant diverses distances sémantiques et non sémantiques seront utilisées sur les données avant qu'une conclusion générale ne soit donnée sur cette méthode.

Le second cas d'utilisation est une application des distances de ports réseau au monde du blocage préventif de ports. Dans ce cadre, différents algorithmes seront développés, entrainés et testés sur un premier dataset ne contenant que des scans de ports avant que les algorithmes ne soient appliqués à un dataset contenant également du trafic légitime. Ceci étant fait dans le but de vérifier le taux de faux positif des algorithmes. L'algorithme optimal et ses paramètres seront alors donnés comme conclusion de cette partie de la recherche.

6.2 Classification de flux réseau de botnet

6.2.1 Présentation du but de la classification

Ce cas d'application a pour but de définir, à la lecture d'un flux réseau, si ce dernier provient ou non d'un Botnet.

Un Botnet est un ensemble de machines infectées et contrôlées à distance par un attaquant malveillant. Les machines faisant partie du botnet sont infectées par un malware afin que l'attaquant puisse en prendre le contrôle et les utiliser à des fins malveillantes. Ainsi, le Botnet caméras connectées infectées "Mirai" [8] a déjà fait trembler certains géant du web dans des attaques de très grandes envergures.

La classification rapide et intelligente de flux réseau comme provenant ou non d'un Botnet serait donc d'utilité afin de protéger un réseau d'entreprise d'une attaque de ce dernier. En effet, si des géants du web tel qu'OVH sont capable de soutenir et contrer des attaques massives menées par des Botnets, ce n'est pas le cas de tous les réseaux d'entreprise. De plus, les Botnets ne sont pas utilisés exclusivement que pour des activités de DDOS mais également dans le cadre de grande campagnes de scans ou de spams à l'échelle d'internet.

Le principe général de ce cas d'utilisation est ainsi, selon des critères réseau aisément utilisable et sans lire le contenu du paquet de définir si ce dernier provient ou pas d'une machine faisant partie intégrante d'un Botnet.

6.3 Présentation du dataset

Ce cas d'utilisation nécessite du trafic réseau contenant des données réseau provenant de machines infectées par des Botnet aussi bien que de trafic légitime. De plus, dans un but d'exhaustivité, plusieurs types de botnet devraient, si possible, être envisagés.

Le dataset remplissant le mieux ces besoins est le jeu de données CTU13 [11]. Ce dernier contient en effet 13 parties chacune composée de données réseau provenant d'une université dont une ou plusieurs machines avaient été infectées par un botnet d'un type particulier, chaque partie du dataset contenant ainsi un ou plusieurs types de botnet.

Au vu de la taille importante de ce jeu de données, un sous ensemble des parties de ce dernier a été sélectionné tout en garantissant que tous les types de botnet présents dans ces parties soient représentés dans le sous ensemble de données sélectionnées.

La répartition des botnets parmi les parties du dataset est présentée dans les figures 6.1a et 6.1b et amène à penser que, pour tenir compte de toutes les particularités de ce dataset, aussi bien au niveau du type de botnet qu'au niveau du nombre de bots présents, les scénarios suivants sont à sélectionner :

Scénario 1 : Présence du botnet *Neris* et présence de trafic de spam

Scénario 3 : Présence du botnet *Rbot* et longueur de la partie du jeu de données

Scénario 4 : Présence de trafic provenant d'attaques DDOS

Scénario 5 : Présence du botnet *Virut* et présence de trafic HTTP.

Scénario 9 : Présence de 10 bots.

Scénario 10 : Attaque DDOS de 10 bots.

Scénario 12 : Présence du botnet *NSIS.ay* et présence de trafic en P2P.

Scénario 13 : Présence de trafic de Webmail

Ces scénarios sélectionnés ont été liés aux fichiers correspondant du jeu de données. En effet, les numéros des parties du jeu de données ne correspondant pas aux scénarios proposés, un appariement des scénarios aux parties du jeu de données a été fait à partir de l'information de nombre de paquets contenus dans la partie du jeu de données et dans le scénario lié.

Ensuite, parmi les caractéristiques proposées par ce jeu de données, ces dernières ont été sélectionnées :

— La durée totale du flux réseau
— Le protocole utilisé
— L'adresse IP source
— L'adresse IP destination
— Le port de départ

Id	Duration(hrs)	# Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Table 3. Amount of data on each botnet scenario

(a) Caractéristiques de la partie du jeu de données

Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	√	√	√							
2	√	√	√							
3	√			√				√		
4	√				√			√		UDP and ICMP DDoS.
5		√		√					√	Scan web proxies.
6				√						Proprietary C&C. RDP.
7									√	Chinese hosts.
8				√						Proprietary C&C. Net-BIOS, STUN.
9	√	√	√	√						
10	√				√			√		UDP DDoS.
11	√				√			√		ICMP DDoS.
12							√			Synchronization.
13		√		√					√	Captcha. Web mail.

Table 2. Characteristics of botnet scenarios

(b) Type de trafic présent dans le jeu de données

FIGURE 6.1 – Caractéristiques des différentes parties du dataset

— Le port de destination
— Le nombre total de bytes dans le flux
— Le nombre total de paquets dans le flux
— Les labels donnés au flux (par exemple, botnet ou pas)

6.3.1 Obtention de distances à partir des flux réseau

Pour fonctionner, les algorithmes d'apprentissage automatique décrits et utilisés dans les sections suivantes utilisent tous, d'une manière ou d'une autre, une notion de distance entre les points qu'ils comparent.

Or, dans notre cas spécifique, les notions de distance que nous voudrions utiliser sont des distances sémantiques.

Il nous est alors nécessaire de pouvoir comparer deux flux réseau tels que définis ci dessus pour donner une seule et unique distance sémantique entre ces deux flux.

Pour chacune des caractéristiques reprisent ci dessus, le tableau 6.1 explique, en supposant deux flux f_1 et f_2 comment la distance de cette caractéristique pour les deux flux est calculée.

Dans ce tableau, la fonction *ipdist* est celle de distance sémantique proposée dans [10] pour comparer deux adresses IP. Le fonctionnement de cette distance sémantique est présenté dans la figure 6.2 [1].

Ainsi, si l'on compare les deux adresses IP *37.187.61.47* et *192.168.19.77* la distance sémantique retournée sera de 64. En effet, ces deux adresses sont des adresses Unicast

1. Cette figure est issue de [10]

TABLE 6.1 – Calcul des distances entre deux flux f_1 et f_2 pour les différentes caractéristiques

Caractéristique	**Dénomination**	**Calcul de la distance**
Durée	$f.dur$	$dur = \|f_1.dur - f_2.dur\|$
Protocol	$f.prot$	$prot$ =0 si $f_1.prot = f_2.prot$, 1 sinon
IP	$f.sip$, $f.dip$	$ip = \frac{ipdist(f_1.sip,f_2.sip)+ipdist(f_1.dip,f_2.dip)}{256}$
Port	$f.sport$, $f.dport$	$port = \frac{pdist(f_1.sport,f_2.sport)+pdist(f_1.dport,f_2.dport)}{2}$
Nombre de bytes	$f.bytes$	$byt = \|f_1.bytes - f_2.bytes\|$
Nombre de paquets	$f.pkts$	$pkts = \|f_1.pkts - f_2.pkts\|$

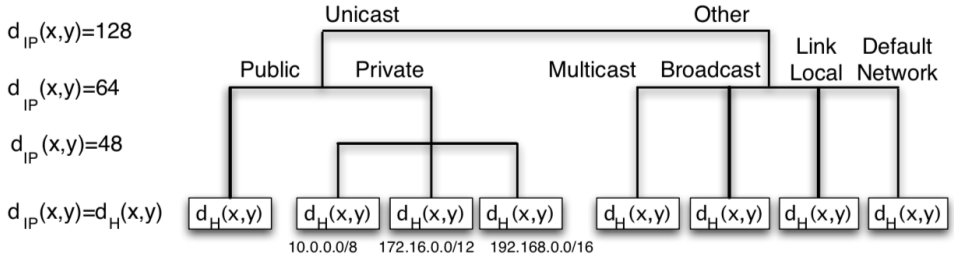


FIGURE 6.2 – Distance sémantique entre des adresses IP

mais l'une d'entre elle est publique (*37.187.61.47*) et la seconde est privée (*192.168.19.77*). Comme second exemple, si l'on compare les deux adresses IP *37.187.61.47* et *164.132.100.87*, ces deux adresses étant publiques, c'est la distance de Hamming (expliquée dans 2.3.1) entre ces dernières qui sera calculée et la distance retournée sera donc de 16.

La seconde fonction actuellement non définie est la fonction *pdist*. Celle-ci représente la distance entre deux ports réseau limitée entre 0 et 1. Dépendant de la notion de distance utilisée, sémantique ou pas, cette fonction prendra différentes formes. En effet, dans le cas de l'utilisation de l'une de nos métriques de distance sémantique, cette fonction prendra comme valeur, la distance sémantique définie entre ces deux ports par la métrique considérée. La distance par défaut pour les ports réseau est celle employée dans [10]. Cette dernière est expliquée dans la figure 6.3 [2] et sa valeur est divisée par 4 pour être comprise entre 0 et 1.

L'ensemble des distances pour chaque caractéristique est alors finalement calculé de la façon suivante :

$$dist(f_1, f_2) = \frac{dur + port + ip + port + byt + pkts}{6}$$

Il est donc à noter que contrairement au fonctionnement habituel des algorithmes qui d'eux même calculent des distances euclidiennes entre les points de données à hautes dimensions, les algorithmes que nous utiliserons dans les sections suivantes devront être capables de prendre en entrée une matrice de distances contenant les distances pair à pair entre tous les flux réseau considérés.

2. Cette figure est issue de [10]

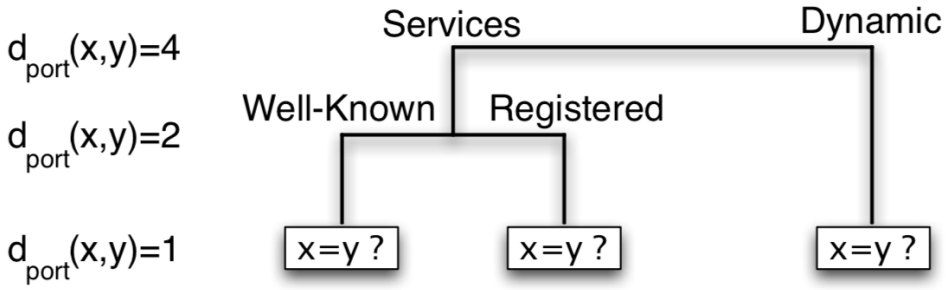


FIGURE 6.3 – Distance entre des ports

6.3.2 Méthodes de classification employées

Dans cette section, nous étudions, dans l'ordre, les différents moyens de classification ayant été abordés pour résoudre cette tâche.

Chaque algorithme sera expliqué avant que la phase de sélection des paramètres ne soit effectuée sur un des treize fichiers du jeu de données présenté. Les autre fichiers de ce jeu de données sont quant à eux utilisés pour la validation des paramètres sélectionnés pour les algorithmes et les tests de performances de ceux-ci. Cette phase de validation et de tests des paramètres ne sera effectuée que si les valeurs fixées à ceux-ci donnent, sur les données d'entrainement, des résultats suffisants que pour continuer vers la dite étape de validation.

6.3.2.1 KMeans

Cet algorithme a été choisi pour notre cas d'utilisation afin d'observer si il serait capable de découvrir un cluster représentant de façon efficace les paquets réseau appartenant au trafic généré par le botnet considéré.

Utiliser un algorithme non supervisé pour effectuer cette tâche est une approche intéressante car ces derniers n'utilisant pas les labels donnés aux flux réseau, il est possible de découvrir si l'algorithme est de lui même capable de regrouper ensemble les flux provenant des machines infectées. Une fois le clustering effectué, il est ensuite possible de définir quel cluster est composé de quel type de flux grace aux labels contenus dans les flux eux même et donc de savoir si un cluster est majoritairement composé de flux appartenant au botnet.

Ayant abordé cet algorithme en début de travail de recherche, nous nous sommes rapidement rendu compte que son implémentation dans la librairie utilisée ne permettait pas de recevoir une matrice de distance entre les différents points de données telle que définie plus haut. Il n'est de ce fait pas possible avec cet algorithme d'utiliser l'une ou l'autre de nos distances sémantique. Cet algorithme du KMeans a donc été abandonné au profit de l'algorithme DBScan expliqué dans la section suivante.

6.3.2.2 DBScan

Les deux paramètres à optimiser dans notre algorithme sont donc ϵ et $MinPoints$. De façon naïve, nous avons commencé par faire varier ces deux derniers et les appliquons sur nos différentes parties sélectionnées dans le jeu de données. Les résultats de cette sélection empirique des paramètres ne dépassant pas les 20% de précision, et après de plus amples renseignements, nous avons découverts que les paramètres de cet algorithme peuvent être défini de façon automatique (comme défini dans [1]).

Cette méthode fonctionne selon deux heuristique permettant de sélectionner une valeur pour ϵ par rapport

Cependant, après plusieurs jours de fonctionnement, il est apparu que cet algorithme de détection automatique des paramètres n'était pas envisageable sur un jeu de donnée de la taille de celui envisagé.

Ainsi, cet algorithme ne donnant pas de bons résultats avec des paramètres empirique et demandant un temps démesuré pour fixer formellement ses paramètres, nous avons réorienté notre choix d'algorithme pour utiliser celui présenté dans la section suivante.

6.3.2.3 KNN

Dans notre cas d'utilisation, à la place d'appliquer une méthode de clustering sur les données et de labelliser les clusters obtenus, il est également possible d'utiliser des algorithmes supervisés de classification utilisant lors de leur apprentissage les labels fixés sur les données afin de pouvoir, pour chaque nouvelle donnée, définir le label que cette dernière devrait avoir.

En tant que tel, cet algorithme ne nécessite pas de phase d'apprentissage. Cependant, dans notre cas particulier et comme pour chacun des algorithmes précédents, les données d'entrée de l'algorithme ne sont pas constituées des points de données eux même mais des distances entre ces derniers. De ce fait, il est nécessaire de calculer la distance entre toutes les paires de points parmi les données d'entrée ce qui entraine un temps de calcul fort conséquent.

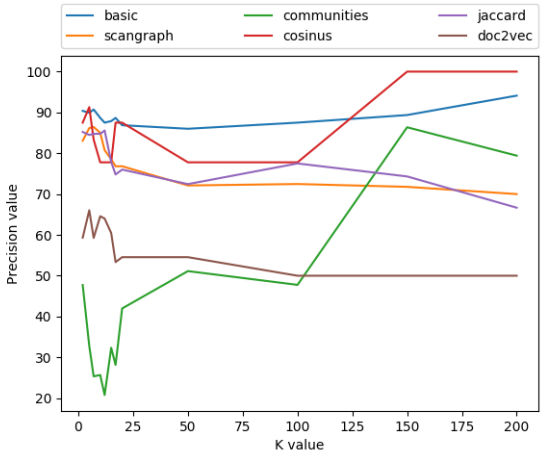
De ce fait et pour limiter ce temps de calcul, nous sélectionnons 3000 points parmi les données d'entrainement en tentant de respecter les proportions présente dans le jeu de données en entrée. Cependant, au cas où la proportion de données labelisée "Botnet" dans le jeu de données est trop faible, cette dernière est gonflée artificiellement pour atteindre 10% du jeu de données d'entrainement. Ainsi, ce jeu de données contient au minimum 300 points de données de botnet et le reste de données "Non-botnet". Toujours pour limiter le temps de calcul nécessaire, le jeu de données de test formé d'un ensemble de 5 groupes de 300 points eux aussi sélectionné aléatoirement et toujours en respectant les proportions des labels dans ce dernier. La sélection de 5 groupes permettant d'opérer une cross-validation des résultats observés.

Les résultats de cet algorithme sont présentés dans les figures 6.4a et 6.4b pour le jeu de données du scénario 1 et représentent respectivement la précision et le recall de l'algorithme considéré en fonction de la valeur de K pour chacune des distances sémantiques.

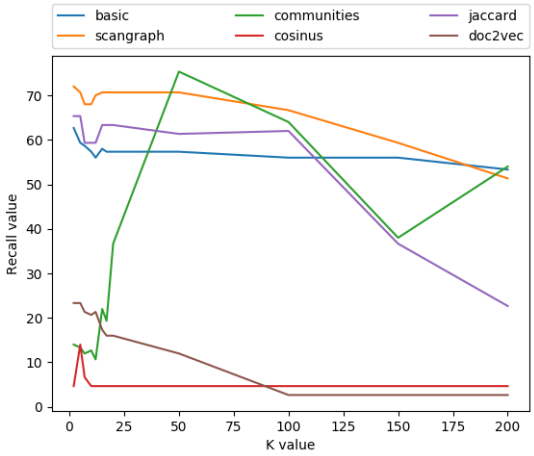
Ces résultats sont obtenus en calculant le nombre de vrai-positifs, vrai-négatifs, faux-positifs et faux-négatifs à partir de la labellisation faite par l'algorithme des données de validation fournies.

De façon générale, on peut relever de ces deux figures que la valeur de la précision ne semble pas évoluer de façon proportionnelle avec K. Inversément, la valeur du recall diminue lorsque K augmente signifiant donc qu'augmenter le nombre de voisins observés augmente le nombre de faux négatifs montrant ainsi que les points de données correspondant à des instances positives sont probablement disséminés parmi les données positives. On remarque également que l'algorithme cosinus présente une très bonne précision pour de grandes valeurs de K mais un mauvais recall. Finalement l'algorithme utilisant la distance doc2vec ne donne de bon résultat ni en précision ni en recall.

Si l'on compare les distances sémantiques et non sémantiques, on remarque qu'au niveau de la précision, les algorithmes utilisant l'une des sémantique évoluée présentés ne font pas mieux que l'algorithme basique utilisant les méthodes utilisée dans l'état de l'art des distances de port. Ces algorithmes ne sont donc pas plus capable de détecter les flux



(a) Précision de l'algorithme KNN



(b) Recall de l'algorithme KNN

FIGURE 6.4 – Performance de classification de l'algorithme KNN

botnet que les algorithmes de l'état de l'art. En terme de recall, les distances sémantiques scangraph et Jaccard donnent de meilleur résultat que la distance basique pour la plupart des valeurs de K.

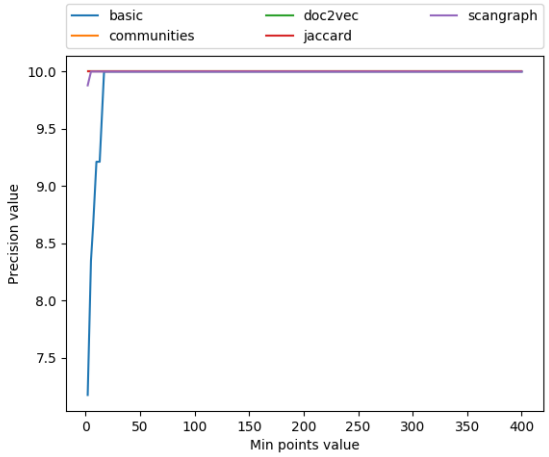
6.3.2.4 Merged Clustering

L'une des limites des algorithmes précédemment utilisés est le temps d'exécution très long qu'ils ont demandé pour opérer sur les données fournies. Cette section a pour but de présenter un algorithme ayant pour but de résoudre ce problème de temps d'exécution.

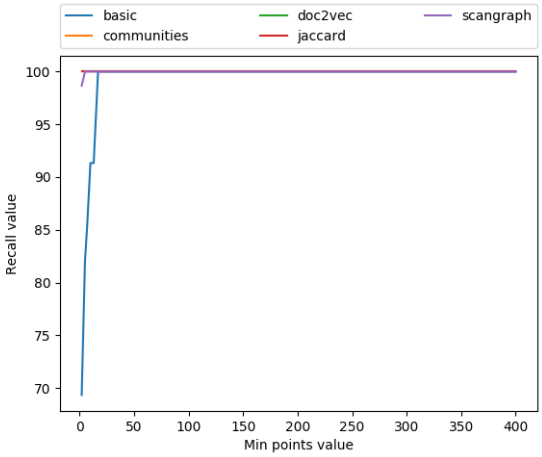
Cependant, nos tests préalables sur cet algorithme ont montré que son temps d'exécution était malgré tout assez important pour une grande masse de données dû au temps de calcul nécessaire pour trouver le meilleur centroid possible au sein d'un cluster. De ce fait, une heuristique de résolution a été implémentée dans cette étape. L'idée de cette dernière est, qu'à la place de choisir un centroid qui minimise sa distance à tous les points du cluster, il devrait être possible à partir d'une certaine taille de cluster de choisir un centroid minimisant sa distance à un certain nombre de paires de points judicieusement choisies au sein du cluster.

Le choix de ces paires de points est fait à la sélection d'un nouveau centroid et opère de sorte que si le point a de la paire se trouve d'un côté du centroid, le point b, second élément de la paire, soit situé à son opposé. L'ensemble des pairs de points "entoure" donc le centroid du cluster de telle sorte qu'à l'ajout d'un nouveau point, ce dernier ne doive calculer sa distance qu'à l'ensemble des points formant ces paires. Si il s'avère que le point ajouté est un meilleur centre (qu'il minimise mieux sa distance aux paires) de cluster que le centroid, il devient alors le nouveau centre du cluster et un nouvel ensemble de paires de points est choisi.

Lors de la phase de sélection des paramètres de cet algorithme, 1500 points de données ainsi qu'un ensemble de valeurs pour min_points et min_dist lui ont été fournis. La limite du treshold pour appliquer l'heuristique d'optimisation de performance de l'algorithme a été fixée à 100 points dans le cluster et le nombre de paires de points significatifs entourant le barycentre (ou centroid) a été fixé à 20. Ces paramètres signifient donc que si un point doit être ajouté à un cluster en contenant plus de 100, ce dernier ne sera pas comparé à



(a) Précision



(b) Recall

FIGURE 6.5 – Performance de classification de l'algorithme MC Clustering $min\_dist = 0.0001$

l'intégralité des 100 points du cluster dans la phase de sélection du nouveau barycentre mais plutôt au 40 points formant les 20 paires significatives. De plus, cet algorithme étant sensible à l'ordre d'ajout des points de données, deux phases d'entrainement de l'algorithme ont été faites en mélangeant le jeu de données d'entrainement entre chaque phase.

Finalement, pour calculer les performances de classification de cet algorithme, le nombre de vrai-positifs, vrai-négatifs, faux-positifs et faux-négatifs est calculé en considérant que les flux de données appartenant aux données échangées avec le botnet devraient faire partie intégrante des données considérées comme "bruit" par l'algorithme.

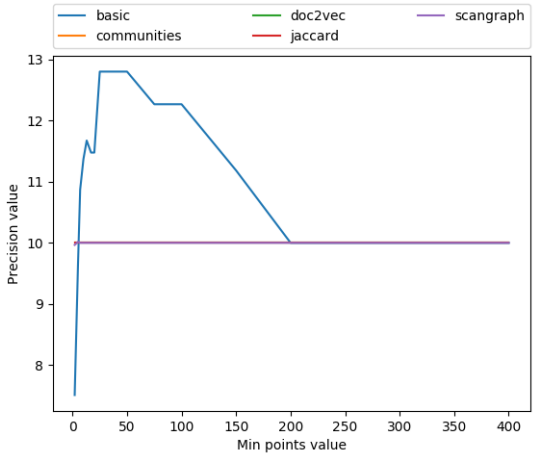
Les résultats de cet algorithme pour la tâche de classification considérée ici sont présentés dans les figures 6.5a, 6.5b pour une valeur de min_dist de 0.0001, dans les figures 6.6a, 6.6b pour une valeur de min_dist de 0.015, dans les figures 6.7a, 6.7b pour une valeur de min_dist de 0.1 et dans les figures 6.8a, 6.8b pour une valeur de min_dist de 0.5.

Dans ceux-ci, on peut relever que pour de petites valeurs du paramètre min_dist, la précision ne dépasse pas les 10% alors que le recall est bien supérieur et atteint les 100% pour des valeurs de min_points supérieures à 50. Lorsque min_dist grandit à des valeurs supérieures ou égales à 0.015, la précision de l'algorithme *basic* augmente jusqu'à plus de 12% pour des valeurs de min_points inférieures à 50 puis redescend vers les 10% quand ce paramètre augmente.

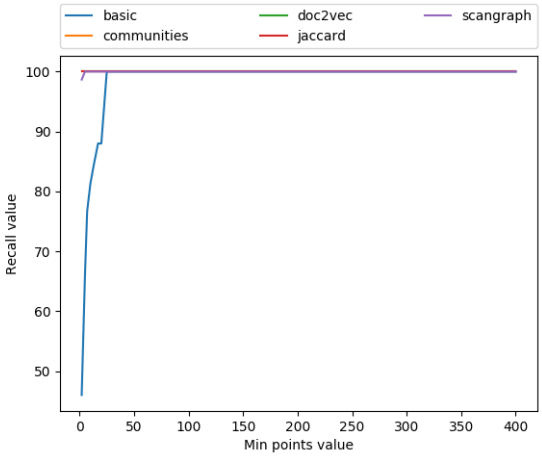
En fixant min_dist à 0.1, les valeurs des métriques sémantiques scangraph et jaccard surpassent alors la métrique basique aussi bien en terme de précision que de recall. La précision reste cependant fort faible montrant que cet algorithme ne donne pas une bonne qualité de classification.

La dernière valeur testée pour le paramètre min_dist est 0.5. En effet, à cette valeur, la métrique basique redevient définitivement meilleure que les métriques à sémantique évoluée.

En analysant l'ensemble de ces constatations, on peut tirer la conclusion que l'algorithme est dans la plupart des cas meilleurs si il utilise une distance sémantique basique telle qu'actuellement utilisée dans l'état de l'art qu'en utilisant l'une de nos distances sémantiques évoluées. De plus, et quelle que soit la distance sémantique ou non sémantique utilisée, la précision de l'algorithme reste assez mauvaise et ne permet pas une application

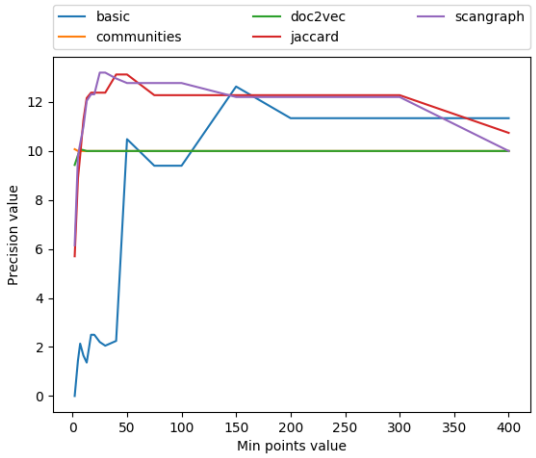


(a) Précision

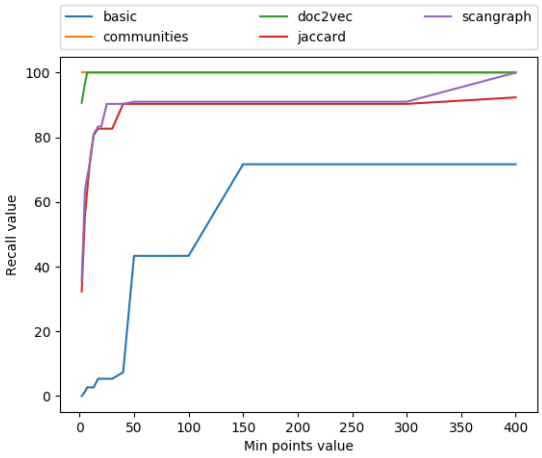


(b) Recall

FIGURE 6.6 – Performance de classification de l'algorithme MC Clustering avec $min\_dist = 0.015$



(a) Précision

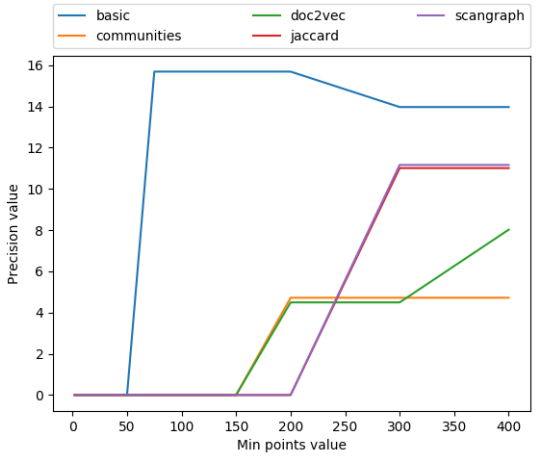


(b) Recall

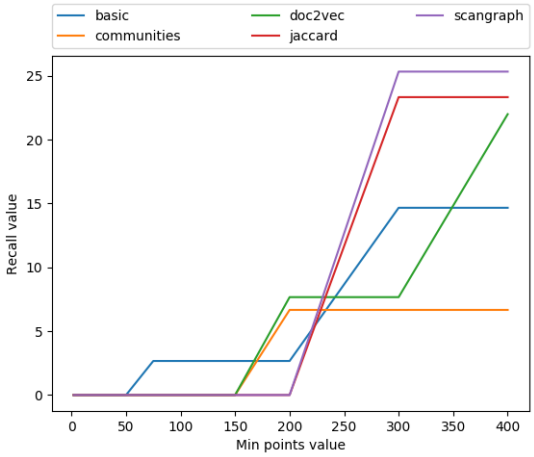
FIGURE 6.7 – Performance de classification de l'algorithme MC Clustering avec $min\_dist = 0.1$

concrète de ce dernier. De ce fait, et la sélection des paramètres n'étant pas concluante, nous ne poursuivons pas les investigations avec cet algorithme et n'étudions donc pas le comportement de cet algorithme dans une phase de validation sur le reste du jeu de données.

Il est à noter, de plus, que cet algorithme utilisant les deux même paramètres que l'algorithme DBScan, il souffre du même problème de besoin de choix très fin des ses paramètres.



(a) Précision



(b) Recall

FIGURE 6.8 – Performance de classification de l'algorithme MC Clustering avec $min\_dist = 0.5$

6.3.3 Discussion des résultats

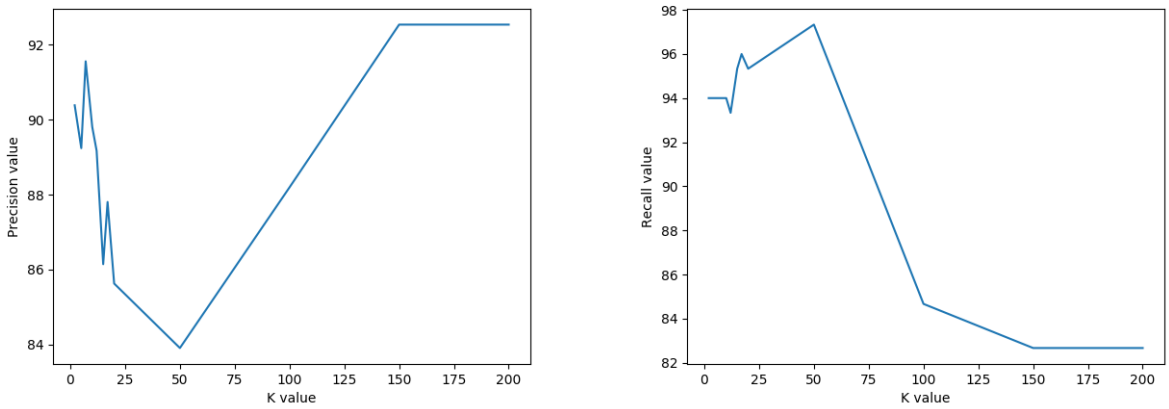
Le premier algorithme abordé dans cette partie de la recherche a été l'algorithme DBSCAN pour lequel un choix empirique des paramètres a été tenté. N'obtenant pas de résultats supérieurs à 20% de précision et une recherche formelle des paramètres de cet algorithme ne pouvant être réalisée en un temps raisonnable, cet algorithme a été abandonné au profit de l'algorithme supervisé KNN. Ce dernier a montré d'assez bons résultats en terme de recall mais expose également que l'utilisation d'une distance sémantiquement porteuse n'améliore pas les performances de classification de cet algorithme. Le dernier algorithme considéré correspond à un clustering basé sur une opération de merging des points du clusters en un barycentre ayant pour but d'améliorer la rapidité de calcul de cet algorithme. Cependant et malgré les choix de paramètres assez large, cet algorithme ne dépasse pas les 20% de précision et n'est donc pas envisageable dans un cadre d'utilisation réel.

Au vu de ces données, il semble clair que quel que soit l'algorithme et quel que soit la distance sémantique utilisée, les algorithmes utilisant une distance sémantique ne semblent pas faire mieux que ceux usant d'une distance non sémantique. Cela peut être dû à une trop grande importance des autres caractéristiques dans la tâche de classification considérée. Cette supposition est confirmée dans les figures 6.9a et 6.9b présentant les résultats de l'algorithme KNN dans lequel le concept de distance de ports a été retiré dans le calcul de la distance totale. Ces résultats étant presque aussi bons que les résultats précédents, il semble clair que l'utilisation d'une distance sémantique n'est pas bénéfique ou du moins pas prépondérante à un cas d'utilisation de classification flux réseau en clusters Botnet ou pas.

6.4 Prédiction de scans de ports en vue de blocage préventif

6.4.1 Présentation du besoin et du but

Sur ces dernières années, une augmentation significative de l'activité cyber-criminelle a été remarquée. Ainsi, si 2016 a été l'année des crypto-ransomwares, 2017 a vu grand



(a) Précision (b) Recall

FIGURE 6.9 – Performance de classification de l'algorithme KNN sans utilisation de distance de port

nombre d'attaque par déni de service telles que celles opérées par le botnet d'appareils connectés Miraï. Si des attaques de cette ampleur font trembler des géants du web tel que OVH ou Cloudflare, il est certain que de plus petites entreprises sont vulnérables de cette ampleur.

En parallèle à ces attaques se développent également les campagnes de scans qui y sont liées et apparentées. En effet, une bonne part des attaques informatiques sont précédées de scans plus ou moins exhaustifs des infrastructures réseau de l'entreprise. Chaque machine directement connectée à internet est ainsi scannée à longueur de journée et tous les jours de l'année.

Ces deux faits mènent à la conclusion que de nouvelles méthodes de détection et de blocage de ce genre de scans et d'attaques doivent être développées afin de maintenir le niveau de sécurité des infrastructures informatiques d'entreprise.

Comme le précise l'état de l'art en chapitre 2 de ce document, le domaine de la recherche s'est actuellement beaucoup intéressé à la détection de scans de ports réseau. De nombreuses méthodes basées ou non sur l'intelligence artificielle ont été mise au point afin de classifier le trafic arrivant à un ordinateur en trafic malicieux ou pas. Cependant peu de travaux ont été menés sur les meilleurs stratégies à appliquer afin de réagir à un scan en cours.

Une multitude de réactions sont possibles. Ainsi, nous ne souhaitons probablement pas que l'attaquant termine son scan et découvre de potentielles vulnérabilités. Quels ports devrions nous alors fermer et pour quelle durée (ce que nous définissons comme le *bantime*) ? Devons nous bloquer l'attaquant sur tous les ports réseau de l'ordinateur cible ou sur certains seulement ? Devons nous bloquer uniquement le port utilisé pour opérer le scan ou d'autres également ? Devons nous également fermer le port qui vient d'être ciblé sur l'ordinateur victime ?

Toutes ces questions font parties de celles à laquelle une bonne stratégie de blocage de scans réseau devrait répondre. En effet, ne pas bloquer le port utilisé par l'attaquant sur le port qui vient d'être scanné permet à ce dernier d'y revenir pour un futur scan ou pas une attaque. De surcroit, bloquer l'intégralité des ports de l'attaquant sur le port venant d'être scanné pourrait empêcher du trafic légitime fait par une machine infectée. Finalement si

il est vrai que bloquer indéfiniment le port attaquant sur la machine subissant l'attaque permet d'en augmenter la sécurité, l'aspect dynamiquement changeant de l'attribution des ports réseau pourrait transformer cette défense en limitation pour une application réutilisant un port ayant été précément utilisé à des fins illégitimes.

Dans cette partie de ce document, nous utiliserons les distances sémantiques définies précédement afin de définir une stratégie de ce type ayant pour but d'opérer, lors d'un scan, un blocage préventif des ports réseau qui seront le plus probablement scannés après ce scan donné. Nous observerons ainsi l'influence d'utiliser une distance sémantique ou non au sein de plusieurs algorithmes de blocages et compareront ces différents algorithmes sur plusieurs datasets.

Dans un premier temps, nous utiliserons nos algorithmes sur des données ne comprennant que des scans réseau. Cette phase d'entrainement nous permettra de définir quelle est la meilleure notion de distance à utiliser et dans quel algorithme cette dernière est la plus efficace. Nous appliquerons alors finalement l'algorithme et la notion de distance sélectionnés sur un second jeu de données contenant des flux réseaux frauduleux ainsi que des flux réseau légitimes et observerons le taux de faux positifs opérés par notre algorithme de blocage de ports intelligent.

6.4.2 Présentation du dataset

La première moitié du travail de cette partie est de définir quel algorithme et quelle distance sémantique sont les plus appropriés pour le blocage préventif de ports réseau. Dans ce but, nous utiliserons le jeu de données déjà défini dans la section 3.2.2 de ce document. Ce jeu de données ne contient, par définition, que du trafic anormal qui sera donc considéré comme malicieux. Les données utilisées ici sont celles de la semaine du premier au 7 juillet 2015 représentant 4.421.737 paquets réseau. Comme pour la définition des distances sémantiques, nous n'utiliserons de ce dataset que les données issues de scans TCP SYN mais ne nous limiterons, dans ce cas, pas qu'aux scans ayant leur nombre de ports cibles limités.

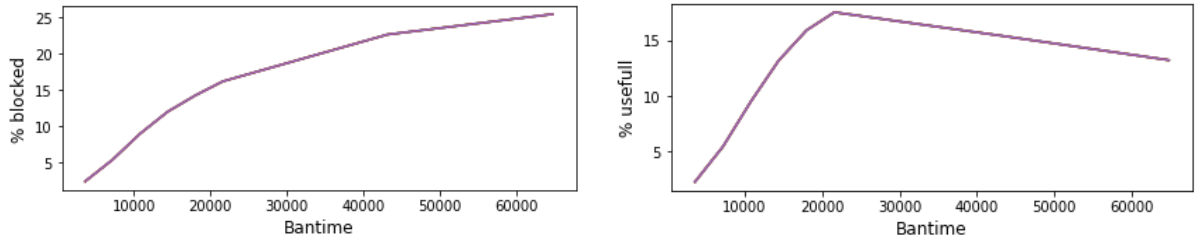
Le seconde moitié de cette partie se concentrera sur l'utilisation de l'algorithme et de la distance sémantique choisis afin de bloquer les scans contenus dans un jeu de données reprennant du trafic malicieux aussi bien que légitime. Ce choix est fait afin d'étudier la proportion de trafic légitime bloqué par notre algorithme (c'est à dire le nombre de faux positifs de celui-ci).

Le jeu de données choisi pour servir cette seconde partie est le dataset du MAWI Labs[24]. Ce dernier propose ainsi pour chaque jour 15 minutes de trafic réseau provenant d'une ligne de communication océanique reliant les États Unis et le Japon. Chaque fichier réseau est accompagné d'un fichier labelisant les paquets qui s'y trouvent en paquets malicieux (issus de scans ou d'attaque) ou pas.

Dans notre cas, ces labels seront utilisés pour simuler un algorithme de détection des scans réseau, notre algorithme ne se concentrant que sur la manière de bloquer au mieux les scans détectés.

6.4.3 Implémentation et tests sur le cas d'utilisation

Dans les sections suivantes, nous présentons les différents algorithmes de blocage imaginés. Chacun sera présenté avec ses résultats avant qu'un autre soit abordé. Cette présentation suit la démarche de recherche utilisée et justifie de plus le choix de l'algorithme en lui même ainsi que de la distance sémantique ou non sémantique choisie.



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.10 – Performances de l'algorithme *myself*

Pour chaque algorithme considéré, deux résultats seront observés. Le premier est le pourcentage de tentatives de scans qui ont été bloqués par l'algorithme. Le second est le pourcentage d'utilité des blocages mis en place. Ainsi, un blocage est considéré comme ayant été utile si il a bloqué au moins une tentative de scan. Ces résultats seront observés par rapport au paramètre commun à tous nos algorithmes, le temps de blocage que ceux-ci mettent en place sur les ports qu'ils bloquent.

6.4.4 Blocage uniquement des ports scannés

Un premier algorithme possible pour bloquer les ports lors d'un scan est de ne bloquer que le port qui vient de subir un scan et uniquement pour l'adresse IP effectuant le scan. Cet algorithme n'a pas pour but, comme ceux à venir, de mettre en place une sécurité préventive mais simplement de dissuader l'attaquant de recommencer son scan en bloquant l'adresse IP de ce dernier sur le port qu'il vient de scanner sur la machine victime.

Les résultats de cet algorithme pour les données provenant du darknet sont présentés mis en place dans les figures 6.10a et 6.10b en fonction du temps de blocage (*bantime*) de ce port de l'attaquant sur le port cible.

Dans ces dernières, il semble que le pourcentage de scans bloqués par cet algorithme croit avec le temps de blocage mis en place mais qu'à contrario, l'utilité des blocages augmente jusqu'à un certain niveau, se situant au environ de 21.000 secondes de blocage, avant de commencer à redescendre.

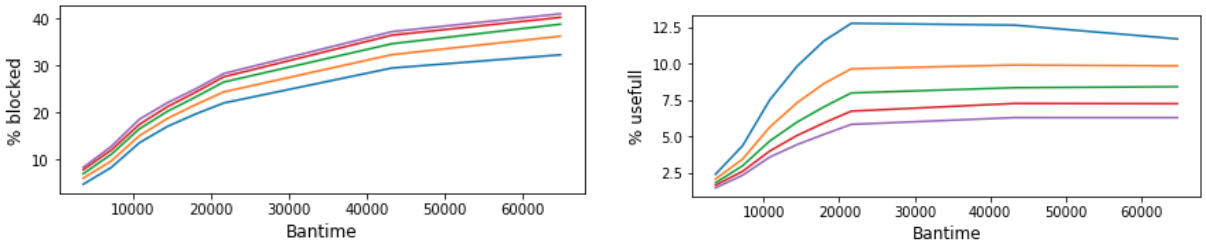
Ainsi, il est clair qu'augmenter le temps de blocage de l'attaquant sur le port augmente la probabilité que ce dernier ne revienne scanner le port dans l'interval de blocage.

On peut de plus noter que le pourcentage de paquets bloqués atteint, avec cette méthode, 25% mais que l'utilité des blocages mis en place ne culmine qu'à 15% environ.

De ce fait et au vu du pourcentage de scans que nous avons déjà pu bloquer de cette manière, nous pouvons définir que quel que soit le cas et l'algorithme de blocage sémantique utilisé, bloquer le port qui vient d'être scanné semble être nécessaire pour obtenir un blocage optimal. De ce fait, tous les algorithmes suivants sont créés afin de bloquer en particulier le port venant de subir un scan donné.

6.4.5 Blocage des K plus proches voisins

Dans cet algorithme, le but défini est d'utiliser une notion de distance entre les ports réseau afin d'opérer un blocage préventifde l'adresse IP effectuant l'attaque sur les ports réseau qui pourraient être ciblés après celui qui est actuellement en train de subir un scan.



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.11 – Performances de l'algorithme avec distance *euclidienne*

Pour ce faire, cette méthode utilise la notion de K plus proches ports liés au ports qui est actuellement scanné afin de sélectionner les ports qui devraient être bloqués de façon préventive.

Ainsi, si le port scanné est le port 80, que le paramètre K de notre algorithme est fixé à $K = 2$ et que la distance choisie est une distance euclidienne, ce sont les ports 79, 80 et 81 qui seront bloqués. Les ports 79 et 81 le seront à titre préventif et le port 80 à titre défensif (pour empêcher l'attaquant de revenir).

Si K est fixé à un nombre impair, l'algorithme choisira de façon subjective que le denier port devant être bloqué est le port le plus haut parmi ses choix possibles. Ainsi, dans le cas précédent et avec $K = 3$, les ports bloqués seront les ports 79, 80, 81 et 82.

Finalement, dans le cas d'une distance sémantique, l'algorithme définira les ports devant être bloqués préventivement en utilisant la distance sémantique considérée. Avec $K = 2$, et les paramètres précédents, les ports bloqués seraient alors par exemple les ports 80, 443 et 8080.

Les sections suivantes présentent les résultats de cet algorithmes tout en faisant varier la notion de distance de port que ce dernier utilise.

6.4.5.1 Distance non sémantique

Comme expliqué ci dessus, notre algorithme de blocage préventif de ports peut être utilisé aussi bien avec une distance sémantique que non sémantique. Dans cette section, et comme expliqué dans la section 6.4.5, nous utilisons une distance euclidienne non sémantique.

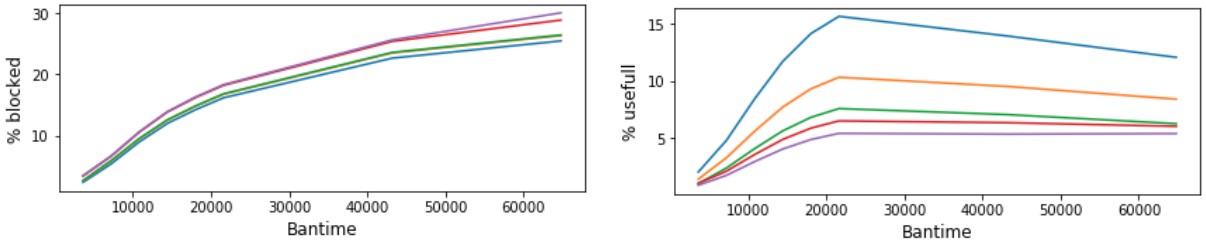
Dans ce cas, les ports considérés les plus probalbes sont donc ceux qui sont numériquement les plus proches. Cette distance ne prenant pas ici compte d'une quelconque sémantique pouvant exister entre les ports réseau.

Les résultats de *blocage* et d'*utilité* sont présenté respectivement dans les figures 6.11a et 6.11b.

Un aspect de ces résultats qui n'était précédemment pas présent est l'ensemble de courbes. Ce dernier représente l'ensemble des valeurs de K testées. Ce nombre représentant le nombre de ports bloqués de façon préventive lors du scan d'un autre port.

Si l'on observe le pourcentage de scans bloqués par cet algorithme, on peut voir que quelle que soit la valeur de K l'algorithme avec cette notion de distance est au moins aussi bon que l'algorithme de blocage du port scanné uniquement.

On peut cependant relever qu'augmenter la valeur de K ou le temps de blocage fait également augmenter le pourcentage de scans bloqués. Ce fait s'explique assez simplement car augmenter le nombre de ports bloqués ou le temps de blocage sur les ports augmente



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.12 – Performances de l'algorithme avec distance *wiki_jac*

la probabilité de voir un nouveau scan arriver sur le dit port.

De plus et contrairement à l'algorithme précédent, on peut relever une diminution de l'utilité des blocages mis en place. Ainsi, ce pourcentage descend de 15% à environ 12.5%.

Ces remarques mènent à la conclusion que bloquer préventivement les ports réseau semble intéressant mais que toutefois les ports choisis dans ce but ne le sont pas au mieux et qu'un choix plus intelligent pourrait être opéré. La possibilité que nous étudions dans les paragraphes ci dessous est d'utiliser nos notions de distance sémantique afin d'opérer cette sélection intelligente des ports à bloquer.

6.4.5.2 Distance sémantique

Comme les résultats de l'utilisation de notre algorithme de blocage préventif de ports réseau utilisant une distance euclidienne le prouve, bloquer les ports préventivement semble augmenter le nombre de scans bloqué de façon significative. Cependant, au vu de la diminution de l'utilité des blocages mis en place, il semble sain de conclure que ces ports pourraient être choisis de façon plus maligne.

De ce fait, nous étudierons dans cette section la possiblité d'utiliser l'une de nos distances sémantique afin d'opérer ce choix intelligent des ports à bloquer.

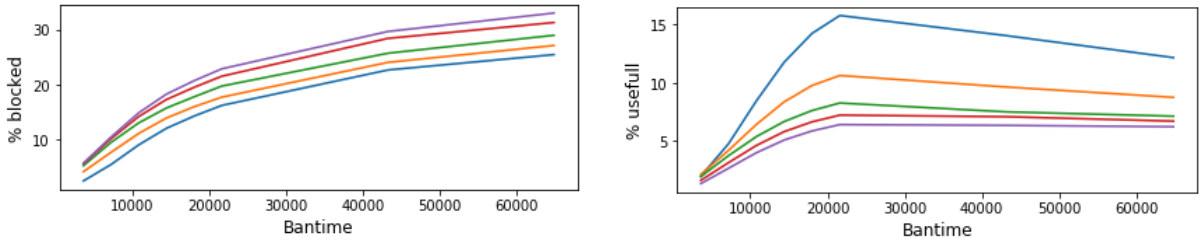
Pour rappel, les distances sémantiques choisies à la fin du chapitre 4 pour continuer nos expérimentations sont les distances suivantes :

— La distance *wiki_jac*
— La distance *wiki_cos*
— La distance *wiki_doc*
— La distance *graph_robust*
— La distance *graph_commu*

6.4.5.2.1 La distance sémantique *wiki_jac* Les résultats de cette notion de distance sont présentés dans les figures 6.12a et 6.12b et montrent un pourcentage de scans bloqués moins important que dans le cas de l'utilisation. L'utilité des blocages passe quant-à elle à un maximum de 15% en gardant toutefois une valeur moyenne proche de celle utilisant un blocage euclidien.

Loin d'être meilleure qu'un blocage utilisant une distance euclidienne, cette distance semble au contraire moins bonne à prédire, efficacement et à l'avance, les ports qui seront scannés.

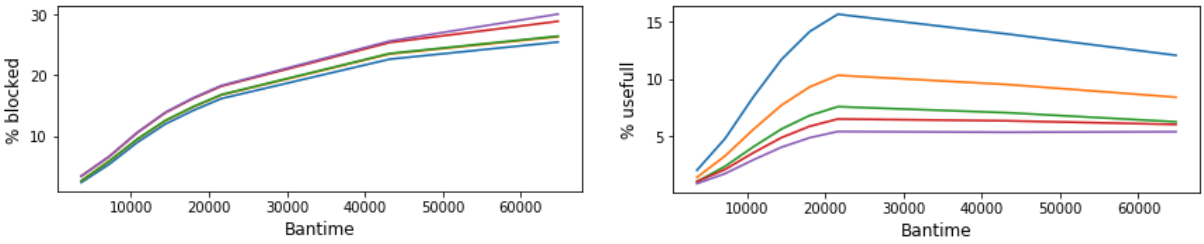
6.4.5.2.2 La distance sémantique *wiki_cos* Bien que le pourcentage de scans bloqués par rapport à K soient un peu plus dispersés, les performances de cet algorithme,



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.13 – Performances de l'algorithme avec distance *wiki_cos*



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.14 – Performances de l'algorithme avec distance *wiki_doc*

présentées dans les figures 6.13a et 6.13b sont identiques au précédent et les conclusions à en tirer sont donc les mêmes.

6.4.5.2.3 La distance sémantique *wiki_doc* Les performances de cet algorithme sont présentées dans les figures 6.14a et 6.14b et sont les même que pour les deux précédents.

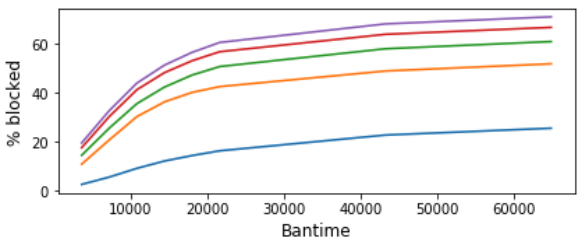
Au vu de ces trois importantes ressemblances, on peut supposer que les limitations de ces algorithmes sont imposées par la source de données dont les distances ont été extraites, Wikipedia dans ce cas donc. Cela prouve donc que toute distance sémantique extraite de cette source de données n'est pas suffisement apte à opérer ce type de cas d'utilisation.

6.4.5.2.4 La distance sémantique *graph_robust* Le pourcentage de scans bloqués par cet algorithme et l'utilité de ces derniers présentés dans les figures 6.15a et 6.15b présentent des résultats deux fois meilleurs que les algorithmes précédents.

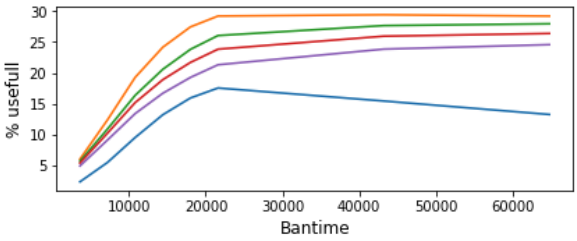
Ainsi, plus de 60% de scans sont bloqués par cet algorithme et plus de 30% des blocages mis en place se révèlent utiles. De plus, contrairement aux algorithmes précédents, la valeur de K pour le pourcentage de scans bloqués n'évolue pas de façon inversément proportionnelle avec cette dernière dans l'utilité des blocages mis en place.

En effet, dans les cas précédents, augmenter la valeur de K augmentait le pourcentage de scans bloqués mais diminuait toutefois l'utilité. Cette réaction est normale dans la mesure où augmenter le nombre de ports à bloquer tout en sélectionnant les mauvais diminue évidement l'utilité des blocages.

Cependant, dans le cas présent, et jusqu'à un certain point, augmenter la valeur de K augmente aussi bien le pourcentage de blocage que celui d'utilité impliquant que les blocages mis en place sont effectivement utiles et que les ports sélectionnés pour le blocage

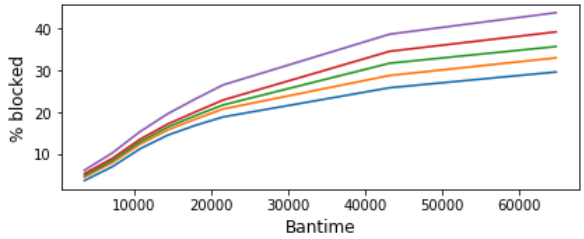


(a) Pourcentage de scans bloqués

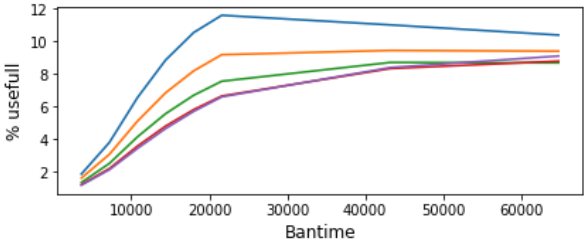


(b) Pourcentage de blocages utiles

FIGURE 6.15 – Performances de l'algorithme avec distance *graph_ robust*



(a) Pourcentage de scans bloqués



(b) Pourcentage de blocages utiles

FIGURE 6.16 – Performances de l'algorithme avec distance *graph_ commu*

préventif le sont de façon appropriée.

6.4.5.2.5 La distance sémantique *graph_ commu*

L'étude des performances de cet algorithme est visible dans les figures 6.16a et 6.16b. Dans ces dernières, on peut relever que cet algorithme arrive à bloquer environ 40% des scans qui lui sont soumis mais avec une utilité de maximum 12% dans les blocages.

Ainsi, si la valeur du pourcentage de blocage est meilleure que pour certains autres algorithmes, en particulier ceux basés sur les données de Wikipédia, l'utilité des blocages mis en place est plus faible que pour ces derniers.

6.4.5.3 Distance sémantique dans une séquence temporelle

L'idée principale de cet algorithme est de façon générale identique à l'algorithme précédent se basant sur les K plus proches voisins du port scanné. La seule modification envisagée dans le présent algorithme est de prendre en compte la séquence des ports scannés sur la machine attaquée afin de garder une "mémoire" de la série de scans en cours et donc d'être potentiellement plus apte à prédire le port suivant.

Pour ce faire, pour chaque scan d'un attaquant sur un port d'une machine victime, l'algorithme ajoute ce port à une séquence des ports scannés par le dit attaquant.

Ensuite, dans l'étape de sélection des ports à bloquer, l'algorithme reprend en ordre inverse l'historique des ports scannés par cet attaquant (du plus récent au plus ancien donc) afin de, pour chacun de ces ports historiques, calculer une série de ports candidats au blocage.

Algorithm 2 Blocage par séquence

1: $seq \leftarrow history\_for(ip_{at}, ip_{vi})$
2: $seq\_len \leftarrow |seq|$
3: $i \leftarrow 0$
4: $all\_candidates \leftarrow Sequence()$
5: **while** $|seq| > 0$ **do**
6: $p = seq.pop()$
7: $c = \frac{k}{\frac{seq\_len}{seq\_len - i}}$
8: $sorted\_p\_distances \leftarrow distances\_to(p)$
9: $candidates \leftarrow Sequence()$
10: $j = 0$
11: **while** $j < c$ **do**
12: $p_i, d_i \leftarrow p\_distances[j]$
13: $candidates.push(p_i)$
14: $j \leftarrow j + 1$
15: **end while**
16: $all\_candidates.push(candidates)$
17: $i \leftarrow i + 1$
18: **end while**

Dans ce but, et en imaginant le $i$ème port p d'une séquence historique de taille t un coefficient c est calculé de la sorte avant d'être transformé en entier :

$$c = \frac{k}{\frac{t}{t-i}}$$

Une fois ce coefficient calculé, la distance du port historique p à tous les autres ports connus est récupérée sous forme d'une liste ordonnées de tuples (p_i, d_i).

Finalement, les c premiers éléments de cette séquence sont sélectionnés comme ports candidats pour le blocage.

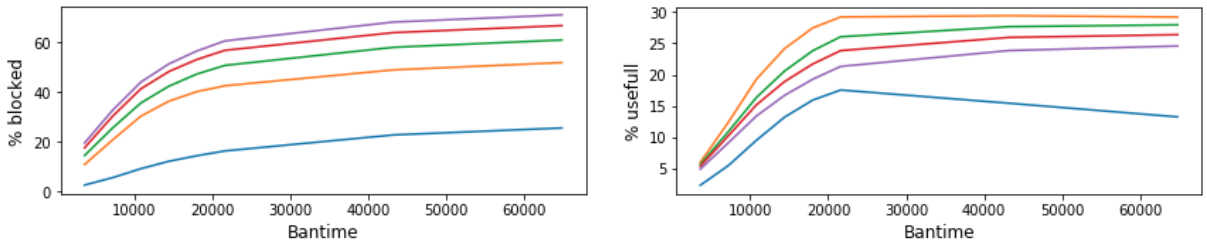
Une fois tous les ports historiques étudiés, les t séquences de ports candidats sont récupérées et les ports effectivement sélectionnés pour être bloqués sont les k ports les plus représentés parmi ces t séquences.

Dans cet algorithme, on peut remarquer que c est utilisé comme facteur de vieillissement des ports dans la séquence historique. Ainsi, si un port a été scanné il y a longtemps, la valeur de c sera plus basse que si ce port était récent et le nombre de ports candidats sélectionnés est donc plus bas afin de favoriser les ports scannés récemment.

Une version de cet algorithme en pseudo-code supposant les quatre paramètres ip_{at}, ip_{vi}, p_{vi} et k est disponible dans l'algorithme 2. Ces paramètres représentent respectivement, l'adresse IP de l'attaquant, celle de la victime, ainsi que le port ciblé sur la machine victime et finalement le nombre de ports à bloquer préventivement. À la fin de cet algorithme, la variable *all_candidates* contient une séquence de séquences chacune contenant une série de ports candidats pour le blocage. Les ports finalement sélectionnés pour le blocage étant les plus représentés parmi tous les ports présents dans cette structure de données.

Les résultats de pourcentage de blocage et d'utilité de cet algorithme utilisant la notion de distance sémantique *graph_robust* sont présentés dans les figures 6.17a et 6.17b.

Comme on peut le voir, et à l'identique que pour l'algorithme de blocage par K plus proches voisins avec la distance *graph_robust*, les résultats de blocages dépassent les 60% avec une utilité de plus de 30% dans les blocages mis en place.



(a) Pourcentage de scans bloqués

(b) Pourcentage de blocages utiles

FIGURE 6.17 – Performances de l'algorithme de séquence avec distance *graph_ robust*

Ces résultats ne sont donc pas meilleur que le meilleur alogorithme décrit jusqu'alors. Il faut cependant noter qu'au vu de la complexité algorithmique ajoutée pour la gestion des séquences, l'algorithme de blocage par séquence des ports scannés est beaucoup plus lent que son "rival".

Finalement, nous remarquons que les résultats identiques relevés le sont dû à la notion de distance utilisée et non à la pertinence d'utiliser la séquence historique des ports. De ce fait, il est clair que ne pas utiliser la notion de séquence ne désavantage pas les performances de classification tout en réduisant toutefois drastiquement le temps d'exécution du code.

6.4.6 Discussion des résultats

Dans les sections précédentes, nous avons appliqué nos algorithmes de blocage de ports à un jeu de données provenant d'un grand darknet et donc uniquement composé de scans réseau. Si cette configuration nous a permis de choisir nos paramètres au mieux et de définir quels sont les meilleurs algorithmes, le trafic réseau dans le "monde réel" n'est pas composé que de scans réseau. Il contient, bien heureusement, du contenu légitime également. Or si notre cas d'utilisation fonctionne bien pour le blocage de scans réseau, il n'est pas dit qu'il ne bloque pas de façon trop restrictive le trafic légitime du réseau.

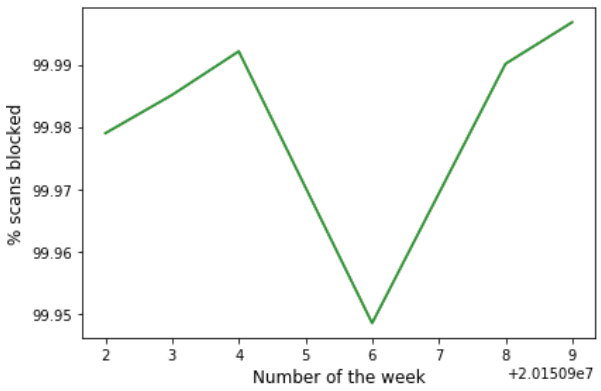
De ce fait, notre cas d'utilisation est, dans cette section, appliqué à un jeu de données contenant du trafic réseau légitime aussi bien que du trafic réseau malicieux composé de scans. Ce jeu de données est celui proposé par le MAWI Labs. Ce dernier est composé de 15 minutes par jour de paquets réseau labellisé et capturés sur une ligne de communication trans-océanique entre les États Unis et le japon. Le labelling de ce jeu de données nous servira ainsi d'oracle afin de définir quels paquets sont des scans et lesquels sont légitimes.

Ainsi, à l'arrivée de chaque paquet réseau défini comme légitime par notre oracle mais bloqué par notre bloqueur de ports sera considéré comme un faux positif. Ce nombre devrait si possible être maintenu le plus bas possible tout en gardant toutefois de bonnes performances de blocage.

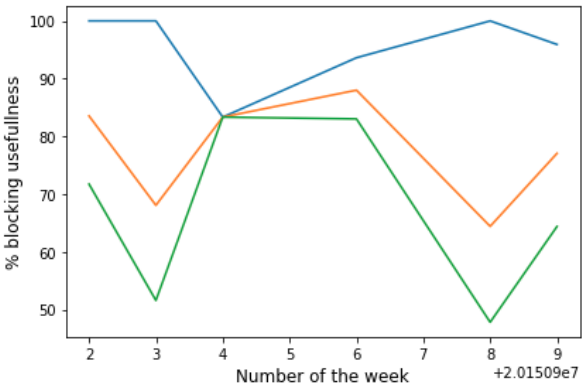
Ce cas est testé en utilisant l'algorithme ayant eu les meilleurs résultats dans les sections précédentes. Deux d'entre eux ayant eu des résultats similaires (les k plus proches voisins sur la distance *graph_ robust* avec ou sans séquences), l'algorithme sélectionné est celui donnant le meilleur temps d'exécution (les k plus proches voisins sans utilisation de séquences).

Les résultats de cet algorithme dans le cas d'utilisation défini plus haut sont visibles dans les figures 6.18a, 6.18b et 6.18c.

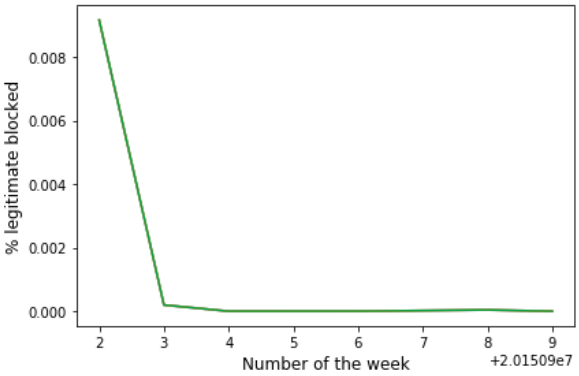
De ces dernières, on peut relever que quelle que soit la valeur de K, plus de 99.9% des scans réseau sont bloqués avec une utilité variant entre 85 à 100% pour la meilleure valeur



(a) Pourcentage de scans bloqués



(b) Pourcentage de blocages utiles



(c) Pourcentage de trafic légitime bloqué

FIGURE 6.18 – Performances de l'algorithme avec distance *graph_robust* sur du trafic réseau habituel

de K (dans ce cas, cette valeur est de 1). Finalement, le moins de 0.01% de paquets réseau légitimes sont bloqués par l'algorithme.

Ces très bons résultats montrent une grande capacité de notre algorithme à bloquer efficacement et préventivement les scans réseau qui lui sont fournis.

Quatrième partie

Conclusion

Chapitre 7

Conclusion

Dans l'introduction de ce document de recherche, les buts du présent travail ont été fixés comme étant la création d'une ou plusieurs notions de distances sémantiques entre les numéros de port réseau à des fins d'amélioration de notre capacité à comparer des flux ou paquets réseau et ce afin d'améliorer, en particulier, la sécurité des systèmes informatiques connectés à internet. Ces notions de sémantiques devaient le plus possible être capables de représenter deux types de sémantique. La première est une sémantique d'utilisation conjointe sur une même machine là où la seconde sémantique représente une utilisation de deux ports dans une même "famille" d'application, par exemple, deux serveurs web.

Pour servir ce but, 4 questions de recherche ont été soulevées. La première était la définition de méthodes permettant l'extraction de la connaissance en réseau d'attaquant à partir de leurs activités malicieuses afin de construire les notions de distance sémantiques qui représentent la seconde question de recherche posée. En troisième lieu, le but était de définir des méthodes de comparaison et d'évaluation des notions de distance afin de vérifier la sémantique portée et l'utilisabilité dans des cas d'utilisation répondant à des problématiques réelles représentant la quatrième et dernière question de recherche posée.

L'état de l'art des métriques réseau sémantique et leur utilisation dans diverses tâches liées à la classification de données réseau a alors été présenté dans le chapitre 2. Ce dernier présente conjointement la complexité de la tâche de classification des données réseau et les méthodes actuelles et passées tentant de résoudre cette problématique. Il avance alors la conclusion que chaque méthode comparant des flux ou paquets réseau utilise dans son fonctionnement un ensemble de caractéristiques de ces données et qu'une comparaison via une distance euclidienne des numéros de ports réseau ne porte pas de réelle sémantique sur les applications qu'ils portent. Cet état de l'art introduit également les dernières recherches en terme de classification de flux provenant de botnet et de détection ainsi que de blocage de flux réseau de scans de ports. Ces deux derniers sujets y sont expliqués afin de mieux préparer le terrain à nos deux cas d'utilisation présentés plus loin.

Suite à cet état de l'art, est présenté dans la section 2.3.4 un ensemble de trois distances sémantiques développées par des étudiants faisant un stage d'initiation à la recherche à l'INRIA de Nancy. Ces trois notions de distance utilisant des techniques de traitement du langage naturel sur un jeu de données extrait de Wikipedia et lié aux ports réseau sont présentées et étudiées avant qu'une proposition d'amélioration ne leur soit apportée. Cette proposition d'amélioration utilise la même méthode d'extraction de données tout en choisissant toutefois l'utilisation de documents écrits par l'IANA et décrivant les même ports réseau tout en étant toutefois plus formellement défini. Les 6 distances sémantiques résultats de ce chapitre sont alors reprises pour une évaluation exhaustive.

Les métriques sémantiques développées dans ce document sont alors introduites dans

le chapitre 3.2. La première métrique utilise une notion de plus courts chemins dans un graphe de ports réseau scannés afin de définir les plus courtes distances entre ceux-ci. Différentes déclinaison de cette notion de distance existent dû à la fonction du poids des arêtes considérées dans le graphe de ports scannés. La seconde métrique présentée déconstruit pour sa part le graphe de port scannés afin d'en reformer les communautés tout en calculant la distance entre les ports lors de cette phase de reconstruction. 4 nouvelles notions de distances sémantiques sont ainsi extraites de ce chapitre. Ce chapitre répond ainsi à la première et la seconde question de recherche en définissant que les connaissances des attaquants peuvent être extraite d'un graphe de scans de ports réseau dont est ensuite extrait plusieurs notions de distance sémantique.

Le chapitre 4 prend pour mission de comparer et classifier les 10 notions de distances sémantiques définies dans les chapitres précédents. Pour ce faire, deux types de comparaisons, répondant à notre troisième question de recherche, sont envisagées. La première sélectionne des ports que nous savons être sémantiquement liés et regarde si la distance considérée est capable de représenter ce lien sémantique connu. La seconde représente les plus courtes distances représentées par chaque notion et les ports qui leurs sont liés pour représenter ces derniers sous forme de graphe afin qu'un opérateur humain puisse vérifier que les liens représentés sont bien porteurs de la sémantique souhaitée. Ce chapitre compare ainsi les notions de distance sémantique pour finalement conclure que 3 des notions ne sont pas porteuses d'une sémantique adéquate. Ces dernières ne sont donc pas reprises dans les travaux suivants dédiés à l'application de ces distances sémantiques.

Le premier cas d'utilisation de ces notions de distance est présenté dans le chapitre 6.2. Le but de cette application est de classifier les flux réseau qu'elle reçoit en flux provenant ou pas d'une machine infectée et faisant partie d'un botnet. Pour ce faire, plusieurs algorithmes ont été envisagés sur les différentes notions de distance. Les résultats de ces algorithmes sur les données montrent que la présence d'une sémantique dans la distance entre les ports réseau n'aide pas les algorithmes de classification pour ce cas d'utilisation. La supposition faite sur ce cas d'utilisation est que les autres caractéristiques choisies sur les données sont prépondérantes et que la caractéristique formée par la paire des adresses IP source et destination est presque suffisante à elle seule pour donner les performance de cet algorithme de classification.

Le second cas d'utilisation présenté est introduit dans le chapitre 6.4. Il présente une application permettant lors d'un scan réseau en cours de bloquer les ports qui seront le plus vraisemblablement scannés ensuite. Dans ce but, plusieurs algorithmes utilisant ou pas les distances sémantiques sont développés et évalués aussi bien sur du trafic contenant uniquement des scans réseau que du trafic légitime. Ce dernier cas de figure étant envisagé pour vérifier que les blocages mis en place par l'algorithme ne bloquent pas une trop grande proportion de trafic légitime. Les résultats de ce cas d'utilisation sont très bons et montrent une réelle habilité de l'algorithme à bloquer préventivement les ports réseau empêchant ainsi de possibles futures attaques sur ces ports. Ce cas d'utilisation montre également l'avantage d'utiliser une distance sémantique pour résoudre ce problème, ce type de distance donnant de bien meilleures performances que les distances non sémantiques envisagées.

Ces deux utilisations répondent ainsi à notre dernière question de recherche en montrant que si l'usage de sémantique n'est pas bénéfique dans tous les cas, il peut cependant se réléver particulièrement prépondérant pour certaines tâches spécifiques.

À la fin de ce travail, plusieurs vecteurs de travaux futurs nous semblent possibles. En premier lieu, la proposition ayant été faite d'améliorer les notions de distance proposées par les étudiants précédents s'étant révélée un échec par un réel manque de complétude

dans la liste de l'IANA faisant le lien entre les ports réseau et les RFCs qui leur sont liées, un possible futur travail serait d'opérer un lien manuel entre les ports et leurs RFCs liées afin que cette notion de distance puisse trouver tous les documents qu'elle nécessite.

Une seconde voie de travail possible serait de transformer le travail actuel pour permettre un apprentissage et un calcul en temps réel des distances pouvant être assignées entre deux ports réseau. En effet, le système actuel est entrainé sur des données historiques afin que les distances soient extraites. Cependant, les habitudes des attaquants scannant les ports étant changeantes en vertu de nouvelles menaces et vulnérabilités voyant le jour, une construction en temps réel de graphes de ports scannés permettant la construction de nos distances sémantiques ainsi qu'une extraction fréquente des distances contenues permettrait d'obtenir une notion de distance sémantique toujours à jour quant aux dernières évolutions des sémantiques liant les ports réseau.

Bibliographie

[1] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, D. Giordano, M. Mellia, and L. Venturini. Selina : A self-learning insightful network analyzer. *IEEE Transactions on Network and Service Management*, 13(3) :696–710, Sept 2016.

[2] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 18(1) :223–239, Jan 2007.

[3] E. Balkanli, J. Alves, and A. N. Zincir-Heywood. Supervised learning to detect ddos attacks. In *Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on*, pages 1–8, Dec 2014.

[4] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2) :23–26, April 2006.

[5] Monowar H Bhuyan, K Bhattacharyya, and J K Kalita. Surveying port scans and their detection methodologies. In *The Computer Journal*, volume 54, pages 1565–1581, 10 2011.

[6] Elias Bou-Hard, Mourad Debbadi, and Chadi Assi. Cyber scanning : A comprehensive survey. *IEEE Communication Surveys and Tutorials*, 1 2014.

[7] Peter Cheeseman and John Stutz. Advances in knowledge discovery and data mining. chapter Bayesian Classification (AutoClass) : Theory and Results, pages 153–180. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[8] Cloudflare. Inside the infamous mirai iot botnet : A retrospective analysis. `https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/`, 12 2017.

[9] Marc Coudriau, Abdelkader Lahmadi, and Jerome Francois. Topological Analysis and Visualisation of Network Monitoring Data : Darknet case study. In *International Workshop on Information Forensics and Security (WIFS)*, Abu Dhabi, United Arab Emirates, 2016. IEEE.

[10] Scott E. Coull, Fabian Monrose, and Michael Bailey. On measuring the similarity of network hosts : Pitfalls, new metrics, and empirical analyses. In *Network and Distributed System Security Symposium*, 01 2011.

[11] Czech Republic CTU University. The ctu13 dataset. `https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html`, 5 2018.

[12] W. De Donato, A. Pescape, and A. Dainotti. Traffic identification engine : an open platform for traffic classification. *Network*, 28(2) :56–64, March 2014.

[13] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Zmap : Fast internet-wide scanning and its security applications. In *Conference on Security*. USENIX Association, 2013.

[14] J. Erman, A. Mahanti, and M. Arlitt. Qrp05-4 : Internet traffic identification using machine learning. In *IEEE Globecom 2006*, pages 1–6, Nov 2006.

[15] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection : Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer, 2002.

[16] C. Fachkha and M. Debbabi. Darknet as a source of cyber intelligence : Survey, taxonomy, and characterization. *Communications Surveys Tutorials*, 18(2) :1197–1227, 2016.

[17] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *PNAS*, 2002.

[18] L. Grimaudo, M. Mellia, E. Baralis, and R. Keralapura. Select : Self-learning classifier for internet traffic. *Transactions on Network and Service Management*, 11(2) :144–157, June 2014.

[19] Public Technical Identifiers. Internet assigned numbers authority. `www.iana.org`, 4 2018.

[20] IETF. Internet engineering task force. `www.ietf.org`, 4 2018.

[21] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy*, pages 211 – 225. IEEE, 5 2004.

[22] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy*, pages 211 – 225. iEEE, 5 2004.

[23] Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[24] MAWI Labs. Mawi labs - data set. `http://www.fukuda-lab.org/mawilab/v1.1/index.html`, 03 2018.

[25] S. Lagraa and J. François. Knowledge discovery of port scans from darknet. In *Symposium on Integrated Network and Service Management (IM)*. IFIP/IEEE, 2017.

[26] Éditions Larousse. Distance. `http://www.larousse.fr/dictionnaires/francais/distance/26042`, 4 2018.

[27] Cynthia Bailey Lee, Chris Roedel, and Elena Silenok. Detection and characterization of port scan attacks.

[28] Peter Mell and Richard Harang. Limitations to threshold random walk scan detection and mitigating enhancements. In *Communications and Network Security (CNS)*, pages 332 – 340. IEEE, 10 2013.

[29] Peter Mell and Richard Harang. Limitations to treshold random walk scan detection and mitigating enhancements. In *Communications and Network Security (CNS)*, pages 332 – 340. IEEE, 10 2013.

[30] Ang Kun Joo Michael, Emma Valla, Natinael Solomon Neggatu, and Andrew W. Moore. Network traffic classification via neural networks. Technical Report UCAM-CL-TR-912, University of Cambridge, Computer Laboratory, September 2017.

[31] Andrew W. Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In Constantinos Dovrolis, editor, *Passive and Active Network Measurement*. Springer, 2005.

[32] Gerhard Münz and Georg Carle. Application of forecasting techniques and control charts for traffic anomaly detection.

[33] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials*, 10(4) :56–76, 2008.

[34] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. In *SIGCOMM Conference on Internet Measurement (IMC)*, pages 27 – 40. ACM, 10 2004.

[35] Susmit Panjwani, Stephanie Tan, Keith M. Jarrin, and Michel Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *Dependable Systems and Networks*, Yokohama, Japan, Japan, 2005. IEEE.

[36] J. Park, H. r. Tyan, and C. c. J. Kuo. Ga-based internet traffic classification technique for qos provisioning. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 251–254, Dec 2006.

[37] A. Shrivastav and A. Tiwari. Network traffic classification using semi-supervised approach. In *2010 Second International Conference on Machine Learning and Computing*, pages 345–349, Feb 2010.

[38] Statista. Number of network connected devices per person around the world from 2003 to 2020. `https://www.statista.com/statistics/678739/forecast-on-connected-devices-per-person/)`, 5 2018.

[39] SUPINFO. Modèle osi et modèle tcp/ip. `https://www.supinfo.com/articles/single/365-modele-osi-modele-tcp-ip`, 5 2018.

[40] T. T. t. Nguyen and G. Armitage. Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 369–376, Nov 2006.

[41] NMAP Team. Nmap : the network mapper. `nmap.org`, 04 2018.

[42] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne, 2008.

[43] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5) :355–374, 2015.

[44] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*, pages 712–717, Jan 2017.

[45] Wikipedia. List of rfcs. `https://en.wikipedia.org/wiki/List_of_RFCs`, 5 18.

[46] Nigel Williams, Sebastian Zander, and Grenville Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5) :5–16, October 2006.

[47] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, pages 250–257, Nov 2005.

[48] Jun Zhang, Xiao Chen, Yang Xiang, Wanlei Zhou, and Jie Wu. Robust network traffic classification. *Transactions on Networking*, 23(4), August 2015.

Annexes

Annexe 1 : Raison de proximité des ports réseau

Port 1	Port 2	Reason
20 (FTP)	20 (FTP)	Same port
20 (FTP)	80 (HTTP)	Used togheter
20 (FTP)	115 (SFTP)	Same service
20 (FTP)	443 (HTTPS)	Used togheter
20 (FTP)	556 (Remote file system)	Same purpose
20 (FTP)	989 (FTPS)	Same service
20 (FTP)	2049 (NFS)	Same purpose
21 (FTP)	21 (FTP)	Same port
22 (SSH)	22 (SSH)	Same port
22 (SSH)	23 (Telnet)	Same purpose
22 (SSH)	513 (rlogin)	Same purpose
22 (SSH)	514 (Remote Shell)	Same purpose
22 (SSH)	992 (Telnet sur SSL)	Same purpose
23 (Telnet)	23 (Telnet)	Same port
23 (Telnet)	513 (rlogin)	Same purpose
23 (Telnet)	514 (Remote Shell)	Same purpose
23 (Telnet)	992 (Telnet sur SSL)	Same service
25 (SMTP)	25 (SMTP)	Same port
25 (SMTP)	109 (POP)	Same purpose
25 (SMTP)	143 (IMAP)	Same purpose
25 (SMTP)	465 (SMTPS)	Same service
25 (SMTP)	993 (IMAPS)	Same purpose
25 (SMTP)	3535 (SMTP alternatif)	Same service
80 (HTTP)	80 (HTTP)	Same port
80 (HTTP)	115 (SFTP)	Used togheter
80 (HTTP)	443 (HTTPS)	Same service

80 (HTTP)	989 (FTPS)	Used togheter
80 (HTTP)	1433 (MSSQL)	Used togheter
80 (HTTP)	1521 (Oracle SQL)	Used togheter
80 (HTTP)	3306 (MySQL)	Used togheter
80 (HTTP)	5432 (Postegres)	Used togheter
109 (POP)	109 (POP)	Same port
109 (POP)	143 (IMAP)	Same purpose
109 (POP)	465 (SMTPS)	Same purpose
109 (POP)	993 (IMAPS)	Same purpose
109 (POP)	3535 (SMTP alternatif)	Same purpose
110 (POP)	110 (POP)	Same port
115 (SFTP)	115 (SFTP)	Same port
115 (SFTP)	443 (HTTPS)	Used togheter
115 (SFTP)	465 (SMTPS)	Crypto
115 (SFTP)	556 (Remote file system)	Same purpose
115 (SFTP)	989 (FTPS)	Same service
115 (SFTP)	992 (Telnet sur SSL)	Crypto
115 (SFTP)	993 (IMAPS)	Crypto
115 (SFTP)	2049 (NFS)	Same purpose
143 (IMAP)	143 (IMAP)	Same port
143 (IMAP)	465 (SMTPS)	Crypto
143 (IMAP)	993 (IMAPS)	Same service
143 (IMAP)	3535 (SMTP alternatif)	Same purpose
220 (IMAP)	220 (IMAP)	Same port
443 (HTTPS)	443 (HTTPS)	Same port
443 (HTTPS)	465 (SMTPS)	Crypto
443 (HTTPS)	989 (FTPS)	Used togheter
443 (HTTPS)	992 (Telnet sur SSL)	Crypto
443 (HTTPS)	993 (IMAPS)	Crypto
443 (HTTPS)	1433 (MSSQL)	Used togheter
443 (HTTPS)	1521 (Oracle SQL)	Used togheter
443 (HTTPS)	3306 (MySQL)	Used togheter
443 (HTTPS)	5432 (Postegres)	Used togheter
465 (SMTPS)	465 (SMTPS)	Same port
465 (SMTPS)	989 (FTPS)	Crypto
465 (SMTPS)	992 (Telnet sur SSL)	Crypto

465 (SMTPS)	993 (IMAPS)	Same purpose
465 (SMTPS)	3535 (SMTP alternatif)	Same service
513 (rlogin)	513 (rlogin)	Same port
513 (rlogin)	514 (Remote Shell)	Same purpose
513 (rlogin)	992 (Telnet sur SSL)	Same purpose
514 (Remote Shell)	514 (Remote Shell)	Same port
514 (Remote Shell)	992 (Telnet sur SSL)	Same purpose
556 (Remote file system)	556 (Remote file system)	Same port
556 (Remote file system)	989 (FTPS)	Same purpose
556 (Remote file system)	2049 (NFS)	Same purpose
989 (FTPS)	989 (FTPS)	Same port
989 (FTPS)	992 (Telnet sur SSL)	Crypto
989 (FTPS)	993 (IMAPS)	Crypto
989 (FTPS)	2049 (NFS)	Same purpose
990 (FTPS)	990 (FTPS)	Same port
992 (Telnet sur SSL)	992 (Telnet sur SSL)	Same port
992 (Telnet sur SSL)	993 (IMAPS)	Crypto
993 (IMAPS)	993 (IMAPS)	Same port
993 (IMAPS)	3535 (SMTP alternatif)	Same purpose
1433 (MSSQL)	1433 (MSSQL)	Same port
1433 (MSSQL)	1521 (Oracle SQL)	Same purpose
1433 (MSSQL)	3306 (MySQL)	Same purpose
1433 (MSSQL)	5432 (Postegres)	Same purpose
1434 (MSSQL)	1434 (MSSQL)	Same port
1521 (Oracle SQL)	1521 (Oracle SQL)	Same port
1521 (Oracle SQL)	3306 (MySQL)	Same purpose
1521 (Oracle SQL)	5432 (Postegres)	Same purpose
2483 (Oracle SQL)	2483 (Oracle SQL)	Same port
2484 (Oracle SQL)	2484 (Oracle SQL)	Same port
2049 (NFS)	2049 (NFS)	Same port
3306 (MySQL)	3306 (MySQL)	Same port
3306 (MySQL)	5432 (Postegres)	Same purpose
3535 (SMTP alternatif)	3535 (SMTP alternatif)	Same port
5432 (Postegres)	5432 (Postegres)	Same port

TABLE 7.1 – Raison de la proximité sémantique des ports

Annexe 2 : Article scientifique résultant de ce travail de recherche

Attacker Behavior-Based Metric for Security Monitoring Applied to Darknet Analysis

ABSTRACT

Network traffic monitoring is a primordial task for network operations and management. It supports many purposes such as Quality-of-Service or security. In the context of security, monitoring and analyzing network flows are widely used for detecting and mitigating attacks such as denial-of-service or botnets. Indeed, deep packet inspection loses in efficiency with the wide adoption of encryption and raises privacy and scalability issues. One major difficulty when dealing with flow-based information is the poor semantic of data (number of bytes, packets, IP addresses, protocol, TCP/UDP port number...). As a result, advanced methods like machine learning became popular to mine similarities and dependencies among flows or group of flows, which supposes first to compare flows. Many attributes extracted from flows are or can be represented as numerical values but cannot be mapped to a meaningful metric space. Most notably are application port numbers. They are numeric values but comparing them as integer is meaningless, for instance using the Euclidian distance. In this paper, we propose a fine grained attacker-based network ports similarity metric allowing traffic analysis to take into account semantic relations between network port numbers. The semantic is automatically derived from attacker behaviors who probe particular sequences of ports when performing scanning. The behavior of attacker is thus derived from passive observation of a Darknet or telescope, aggregated in a graph model, from which a semantic similarity function between port number is defined. We demonstrate the veracity of this function in order to proactively block scans.

1. INTRODUCTION

TCP and UDP are major transport protocols in Internet. Port numbers allow the end-hosts to differentiate flows and forward them to the right sockets and so services. Being encoded in 16 bits, there are 65,536 possible ports for both TCP and UDP. There are different segments: system or well-known ports (0-1023), reserved ports for specific applications or vendors (1024-49151) and dynamic ports (49152-65535). Although the dynamic ports are mainly used as ephemeral port, such as source ports when establishing a connection, other ports are associated to a special use, *i.e.* service. Their numbering is managed by the Internet Assigned Numbers Authority (IANA). Even if this does not prevent any user to use a registered port for any usage, using assigned port numbers eases access to the service. In addition, there are some ports which are usually diverted from their normal usage, such as 443 originally reserved for HTTPS but often used by VPN services to avoid filtering.

Therefore, port numbers are representative of the provided services but there is no one-to-one mapping between a port number and a service. They are valuable source of information for managing and operating a network as for instance to perform traffic engineering for QoS purposes [CITATION] or to detect anomalies[CITATION]. In many cases, in particular for security purposes, packets or flows need to be compared for supporting machine learning or data-ming algorithms. For example, Netflow records can be analyzed to detect anomalies [CITATION] but all data to handle cannot be represented in a metric space and so easily compared. While using longest common prefixes can solve the problem with IP addresses [CITATION], the problem remains for port numbers. Considering only the three possible ranges is an option [CITATION] but results in a large granularity.

In this paper, we propose a fine-grained approach which catches and quantifies two types of similarities between port numbers:

- Service-semantic similarity: this represents port numbers supporting services from the same type. For instance, TCP ports 80 and 443 are semantically close to each other (Web). However, TCP ports 443 and 22 are also close semantically because they provide a secure connections.

- Context-semantic similarity: this abstracts the relations between ports which are often present together (on the same machine or more generally in a close vicinity, e.g. same subnetwork). As an example, an medium-scale enterprise network often provides a web and email server.

It is worth to mention that two ports can be similar on both perspective, *e.g.* 443 and 80, both for web services and usually co-located on the same server. In a preamble of an attack, scan is often performed to find open ports. In order to remain undetected, attackers may prefer to select particular ports rather than using massive scans. The selection of this port is not random and follow a certain logic from which the semantic between port numbers could be thus derived. Three contributions are presented in this paper. Firstly, major trends on port scanning are highlighted from a 40 weeks long darknet network dataset. It results in the clear observations of relationships among targeted ports. Secondly, this actually motivates and guides the definition of a single metric able to catch both types of similarities (service- and context- semantic). Using graph-based formalism proposed in a previous work [1], we propose and evaluate a definition of the metric. Finally, our third contribution leverages our metric to preventively block an attacker scanning ports by predicting the next targeted ports.

The remaining of the paper is structured as follows : Section 2 presents related works. Section 3 introduces our attacker-based semantic port distance that is evaluated in Section 5. Section 4 makes an introduction to darknets and presents statistics on our darknet dataset. Our semantic distance is then used in a concrete use case in Section 6. Section 7 finally gives the conclusion of our work together with possible future work.

2. RELATED WORK

Ports used by applications is a very accessible information when performing traffic monitoring for various purposes. It's worth noting that some researchers like in [2] question the use of network ports informations in machine learning methods because of the versatility of this information. In 2005, the authors already show that only 70% can be properly classified based on official port numbering [3]. Nowadays, traffic classification is even facing new challenges with encrypted traffic [4]. However in many cases, port numbers brings valuable information about type of potential services use or targeted by attackers. We show that there are particular relationships between sequence of ports probed by attackers [1]. Actually, the weakness of using port information with advanced methods such as machine learning algorithms is the lack of proper metric to apprehend the similarity or dissimilarity between port numbers since they are not embedded in a metric space. As a result, many traffic analysis relies on other features [5] or simply consider if port numbers are equal or not [6, 7]. Few efforts have been made to enhance port number comparison [8, 9]. In [8], the aim is to group TCP flows in order to identify a dominant port per group if it exists in comparison with random ports. The authors in [9] defines a set of metric between well known computer network entities. Especially, network port number are compared accordingly to the ranges they belong to (registered, well-known or dynamic). This draws the path to a higher comprehensive metric to include port in analysis. We propose to go further by deriving a single distance returning a result for any couple of TCP ports.

As our main motivation is security monitoring, our distance relies on knowledge indirectly embedded in attacker activities, especially TCP scans. Some surveys like [10, 11] show that big scanning campaigns become more frequent and can be based on very efficient tools like ZMAP[12]. Collecting scanning activities is thus a rich source information but necessitates a mining approach to extract synthetic knowledge like our proposed distances. Darknets have been proved to be efficient to monitor large scale attacker activities such as scans or DDoS (Distributed Denial-of-Service) attacks [13]. Many of existing works focuses on analyzing and describing observations made through the darknet [14, 15]. This paper proposes to build a distance function based on those observations to be applied for real time security monitoring. In particular, we show the viability of our technique to proactively block future TCP scans once a first attempt is made by the attacker. It is complementary to many existing techniques dealing with detection of scans [16, 17].

3. ATTACKER BEHAVIOR-BASED INTER-PORT DISTANCES

3.1 Rationale

When performing an attack, the first stage usually consist in fingerprinting the potential target and all the more with Advanced Persistent Threats[CITATION]. It is a critical step for crafting an attack as most as possible specific. Discovering accessible machines and services often relies on IP sweeping and/or scanning TCP and UDP ports[CITATION]. Naive approach testing all ports numbers and all IP addresses of a targeted subnetwork is time-consuming and has a large footprint which can be easily detected. However, smarter attackers would look for specific ports to search for particular services with potential vulnerabilities. For example, if she looks for web servers, TCP/443, TCP/80, TCP/8080 will be targeted in priority and can reveal a service-semantic similarity. Similarly, attackers may target a particular type of environment with various services close from a context-semantic point of view. For example, a web service relies usually on a web server and on database. So both of them are regularly co-located in a close network vicinity, even in the same host. In that sense, we provide some observations in a previous work [1].

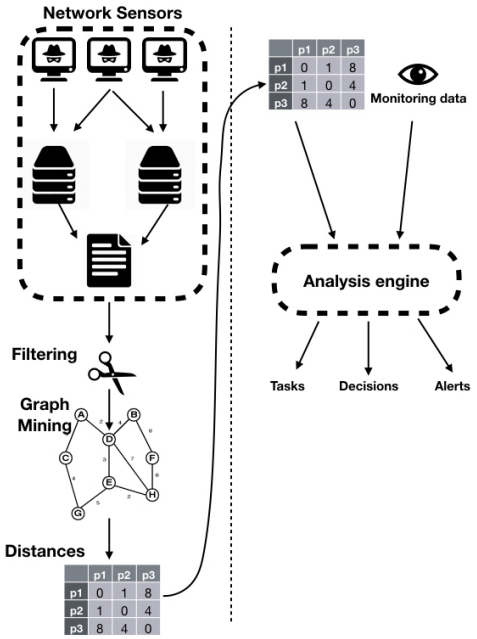


Figure 1: Semantic distance creation method

We propose to aggregate, from the massive observation of a darknet, such a knowledge into two distance metrics which by design are able to catch both behaviors: the attackers looking for a particular type of service (service-semantic) and the attackers looking for services which are usually co-located together (context-semantic). We therefore assume the co-existence of these behaviors in scanning strategy of the attackers. Whereas this could be intuitively admitted, the detailed analysis of the darknet data in section 4 confirms it.

3.2 Methodology

Our rationale is to extract similarities between ports by observing attacker behaviors, *i.e.* ports targeted by the same attacker in vantage point. More especially, we use a darknet or network telescope which allows us to silently collect undesired trafic including TCP scans on a non active subnetwork. It is possible to derive a distance from the distance between two ports in a sequence but this does not lead to good results in preliminary experiments. Intuitively, a set of semantically close port numbers (either by context or by services) may not have been massively observed integrally (in individual sequences) but rather through multiple overlapping sequences.

All sequences must be aggregated together in a unique representation. In addition, even ports which are supposed to be similar are not targeted in the same order each time by the different attackers. No global order or sequence should be constructed but a graph represents all of them actually in a compact format. We thus transform successively probed ports by attackers into a graph prior to extract the network ports semantic knowledge.

Figure 1 illustrates the whole process to infer a distance metric between port numbers:

1. Multiple attacker behaviors, e.g. targeted ports, are collected. In order to avoid a bias, it is required to collect such behavior in a massive scale. In our case, we use a darknet or telescope (see section 4).

2. Scan extraction: since collected data can embed some noise, filtering is necessary and directly dependent on the collecting process. The goal is to only extract the attacker behaviors. For example, big vertical scans running on all ports do not contain an extractable semantic and should then be omitted in graph construction.

3. Graph: the graph of scans is created with the filtered data gathered from the previous step. In this graph, nodes represent network ports and the directed edge between two port means that they have probed sequentially at least once.

4. Distances: thanks to the network port scans graph created, several algorithms have been defined and assessed to extract an attacker behavior based semantic distance between pair of ports.

It is worth to mention that the methodology can be helpful to derive a distance function for both TCP and UDP ports. However, the remaining of the paper is focused on TCP ports. The main difference would be the scan extraction, *i.e.* characterizing unique and meaningful scan observations.

3.3 Extraction of network scans

To be able to extract an appropriate semantic, we would use network scan data that were themselves semantically consistent. However automated scan tools make network scans on a very wide range of ports (sometimes all) that are not especially semantically connected (horizontal or block scans). Other tools or scan campaigns make scans on the same port for a big set of computers (vertical scans).

Such scans are out of the interest of our research since the goal is to catch a supposed smart strategy in selecting ports by attackers in order to establish measurable relations between distinct port numbers. Our dataset is precisely described in section 4 but Figure 2 represents a cumulative sum of IP addresses having a less than a fixed number of ports scanned during a week. Moreover, our analysis is restricted to TCP SYN scans. Most IP addresses (72%) are scanned with a limited number (less than 30) selected port numbers during a week and the cumulative sum is then more stable. The long tail of the curve, here limited up to 75 (99.5%), represents so the vertical scan.

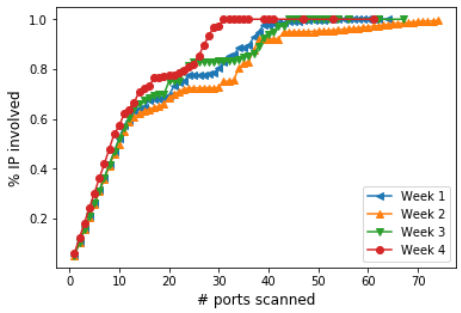


Figure 2: Cumulative sum of number of ports scanned per destination IP address per week (in a 4-weeks period)

Figure 2 also highlights that a very wide number of IP addresses have less than 3 ports scanned, which is more a representative of a probing technique targeting few ports, *i.e.* vertical/block scans.

Therefore, data used for learning is filtered according to these observations by discarding network traffic related vertical (same IP address probed with more than 30 ports) and horizontal/block scans (an IP addressed probed with less than 3 ports).

3.4 Graph-based port sequence model

The second step of our method is to create a graph model from observed scanned ports. It is derived from the method described in [1], which has highlighted semantic relationships between port numbers.

A scan graph is a weighted graph $G = (N, E, \omega)$ with:

N The set of nodes of the graph. Each of them represents a unique TCP port.

E The set of edges of the graph. The existence of an edge $e_{i,j}$ between two ports p_i and p_j shows that port p_j has been probed (by the same scanner) just after p_i on the same destination IP address.

ω Weight function that, for an edge, returns its weight. The weight $\omega(e_{i,j})$ of $e_{i,j}$ is the number of times that p_j have followed p_i in all scan sequences.

This graph representation allows us to represent a very large dataset of scans thanks to a well defined an summarized structure.

3.5 Shortest path based inter-port distance

The defined graph intuitively contains the desired semantic. If two ports are connected by an edge with a high weight, they have been probed a lot of time one successively. By generalization, the graph also contains ports that are near each other thanks to transitivity. For example, if scans go repeatedly from 80 to 443 and from 443 to 3306, the graph contains a transitive link between ports 80 and 3306 which reveals a semantic similarity between these ports, but lower than between 80 and 443 (connected by a direct edge).

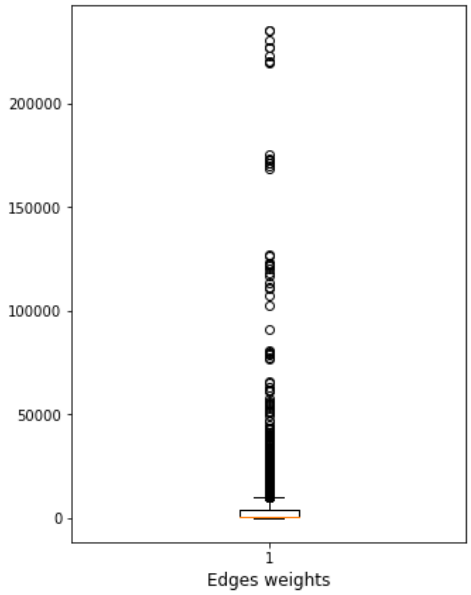


Figure 3: Edges weights boxplot

The intuition of this semantic is to swap (or invert) the weight of edges in the graph to reduce the shortest path length between ports which are regularly scanned in a same sequence (*i.e.* those connected together with heavy weights). Then, shortest paths $sp(n_i, n_j)$ between pair of nodes n_i and n_j (port numbers) are computed. $sp(n_i, n_j)$ is the smallest sequence of edges from source n_i to destination n_j according to the inverted weights. The length $l(sp(n_i, n_j))$ of this shortest path is then used as the distance between the two ports, i and j, represented by the nodes n_i and n_j respectively. This distance is denoted as d_{sp}:

$$d_{sp}(i,j) = l(s) = \sum_{\forall e_{i,j} \in s} \omega'(e_{i,j}), s = sp(n_i, n_j) \quad (1)$$

Finding shortest paths in a graph is a common problem. Methods, like the Dijkstra algorithm, are well defined. The main challenge resides in defining a correct rescaling and swapping method for edge weight, *i.e.* deriving $\omega'(e_{i,j})$ from $\omega(e_{i,j})$ to make closer nodes linked with heavy weighted edges. In the original graph, the weight of an edge represents the number of times a transition occurs between two ports. For the sake of simplicity and brevity, we will simplify $\omega(e_{i,j})$ weights notation to $\omega_{i,j}$.

The distribution of edges weight, θ, in the graph is given in Figure 3 based on our dataset described in section 4. In this figure, we can observe an unbalanced distribution with most of the values concentrated between

$Q_1 = 299$ and $Q_3 = 4082$ (θ inter-quartiles range). Therefore, data needs to be rescaled. Due to the occurrence of outliers, we have chosen to use the IQR as a basis for rescaling as follows:

$$\omega_{i,j}^{iqr} = \frac{\omega_{i,j} - Q_1(\theta)}{Q_3(\theta) - Q_1(\theta)}, \theta = \{\omega_{i,j} \forall i, j\}$$

Actually, the IQR-based rescaling method looks at data distribution instead of data value and allows then a higher distance range for most represented values. It is worth to mention that rescaled values originally below Q_1 become negative. Prior to swap weights, we shift rescaled data to positive values by deducing the minimal value, $\omega_{i,j}^{iqr} - \lambda$ with $\lambda = \min_{i,j}(\{\omega_{i,j}^{iqr}\})$.

Finally, weights can be swapped regarding the maximum value (which is also shifted):

$$\omega'_{i,j} = (\max_{i,j}(\{\omega_{i,j}^{iqr}\}) - \lambda) - (\omega_{i,j}^{iqr} - \lambda)$$

This data-driven scaling and swapping technique avoids to use arbitrary factor when inverting the edges weights.

3.6 Communities-structure based inter-port distance

Whereas the shortest path approach is focused on analyzing connections between two nodes, a community approach is more global by looking into groups of groups of nodes that are all highly linked together. As our graph is created with ports scanned together by attackers, our aim is to find groups of ports that are linked together in scans and are then supposed to be semantically related. In this section, we define a second distance between ports i and j, $d_{com}(i, j)$.

Girvan Newman (GN) algorithm [18] can be used to group together well interconnected nodes (communities), according to a stopping criteria which guarantees a minimal level of interconnectivity between nodes of the same community. The GN algorithm relies on the edge betweenness for each edge which is measured as the number of shortest paths going through that edge (assuming weight rescaling and swapping similarly to the previous section). GN algorithm remove the edge with the maximum betweenness and iteratively continues this process (edge betweenness calculation and edge removal) until the maximal betweenness is lower than a user-defined threshold. Partitions of the graphs are then the communities.

Partitions are thus created iteratively. At each iteration step, a connected component of the graph may be divided by two creating thus two sub-communities. From that perspective, a hierarchy among communities can be established. In our case, the algorithm stops until we reach single node communities (the most specific type of community) but consider all the ones built at any intermediate levels in order to establish this hierarchy from which distances between nodes are derived.

An example is given in Figure 4. All weights are equals one and the actual number associated to edges represented the order of the removal. Traversing the sequence of edges removed in the reverse order allows to reconstruct a parent-child relationship between partitions. This leads to create a dendrogram as illustrated in Figure 5. In this Figure, leafs represent nodes of the Figure 4 and numbers on splits represent edges numbers concerned at this level. As shown, removing a single link is not always sufficient to create a new partition. In this figure, the community (E,F) is included in a larger community (E,F,G) so $d_{com}(E, F) < d_{com}(E, G)$ and $d_{com}(E, F) = d_{com}(F, G)$.

Assuming the maximal depth of the tree, max_depth and the $depth(ancestor(i, j))$ as the depth of the common ancestors of port i and j, the distance $d_{com}(i, j)$ is defined as:

$$d_{com}(i, j) = max\_depth - depth(ancestor(i, j) - 1 \quad (2)$$

4. DARKNET OVERVIEW

A darknet also known network telescope or Internet blackhole is an entire reachable subnetwork collecting all incoming traffic with no active hosts and so not sending any packets. Such traffic which can be considered as the noise from Internet or the Internet Background Radiation (IBR) [19] which have been proved to contain valuable information to understand major security threats like DDoS attacks and scanning activities [14]. Therefore, our goal is to analyze the vast quantity of network traffic targeting a darknet, modeling the attacker behavior when the latter performs a port scan, and then summarize this information into a unique port distance.

4.1 Datasets and pre-processing

Two darknets are used in this paper over 40 weeks of observations. The first one (FR), in France with a dedicated /20 subnetwork. The second one (JP) is a /20 subnetwork in Japan. Using both datasets will strengthen our evaluation, especially to assess if there are dependencies between locations. General statistics are provided in Table 1. It shows that Japanese darknet attracts more traffic than French one but from less attackers meaning that people attacking Japanese darknet use significantly more packets in their scan probes.

In next sections, detailed statistics about observed port scans are provided to understand the behavior of attackers during scan campaigns. Hence, data related to scans is first isolated in the datasets selecting TCP SYN scans only (based on the method presented in 3.3). Except when mentioned, all statistics given in the next sections are presented over a joint datasets of both the JP and FR scans.

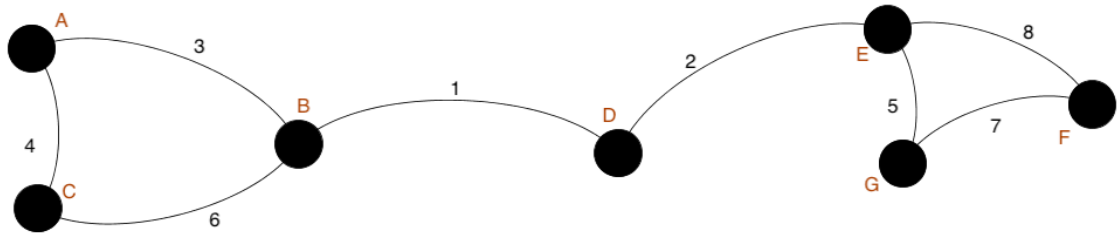


Figure 4: Example graph

Table 1: General darknet statistics (attackers are identified by unique source IP addresses)

	France	**Japan**
Begin date	1st January 2015	1st January 2015
End date	30th September 2015	30th September 2015
Total # of attackers	3,771,092	3,712,209
Average # of attackers per day	19,776.66	19,621.43
Total # of packets	399,344,813	415,642,444
Average # of packets per day	1,426,231.47	1,484,437.3

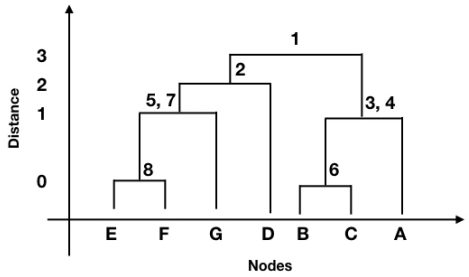


Figure 5: Distance dendrogram

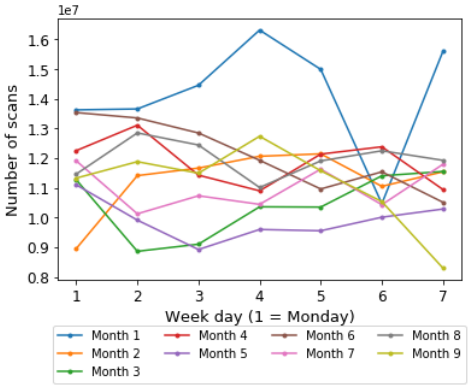


Figure 6: Number of re-scan in a day by week.

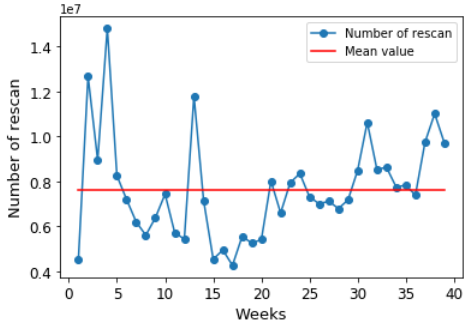


Figure 7: Number of re-scan in a day by week.

4.2 Number of scans

Assuming the number of observed scans, Figure 6 shows no explicit correlation between the day of the week and the number of scans. However, we can notice that the number of scans a day is always between around 9 million and 17 million. Moreover, on Saturday, less variations are observed.

In Figure 7, we evaluate the number of times an attacker (identified by the source IP address) targets the same port on the same destination IP address within the same day. Such a value is actually very high. Once a scan detected, an efficient preemptive blocking technique should always block the associated port as it will be undoubtedly targeted again by the attacker. This would significantly reduce the number of scans reaching the aimed computers or services.

4.3 Number of distinct targeted ports in scans

Our datasets reveal that all TCP ports are targeted within a week. In fact, even a single vertical scans can lead to such a situation. With a more fine-grained focus, in Figure 8, the curve entitled *distinct destination ports* depicts the average number of targeted ports by destination IP address of the darknet.

The average number of ports scanned in a week for

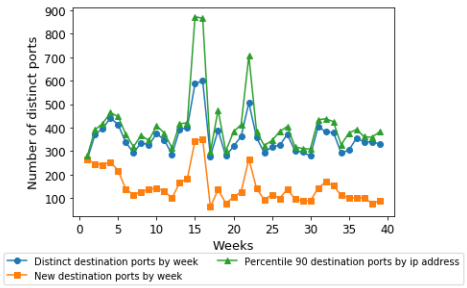


Figure 8: Average number of ports scanned by IP address

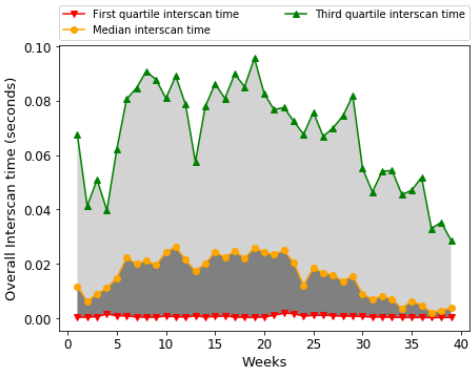


Figure 9: IQR distribution for overall inter-scan time

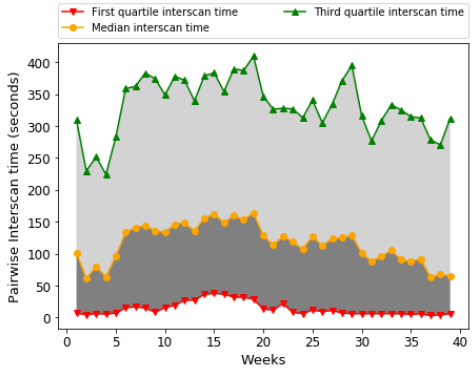


Figure 10: IQR distribution for pairwise inter-scan time

an IP address is around 350. Comparatively to the 65.536 available TCP ports, this number is relatively small. More precisely, 90% of IP addresses are scanned on less than 900 ports a week (with a mean around 400 ports) as highlighted in the same figure. It confirms that scans are targeted towards selected ports and the probing strategy is not random. Moreover, Figure 8 also presents the average number of new ports scanned per destination IP address each week. Compared to the previous curve, the dynamic is the same with a very similar shift along the weeks. There is so a similar number of new port number targeted every week.

Intuitively, the targets of attacks are motivated by the apparition of discovered vulnerabilities in devices and services. Our observation confirms this intuition and quantifies it. Besides, modeling the attacker behavior has to be done over long periods and need to be reassessed regularly in order to properly set a meaningful and condensed metric between ports, which is not biased by ephemeral behaviors.

4.4 Inter-scan time

Another question regarding the scanning behavior is the vivacity of a port scan denoted as the inter-scan time. It is the average elapsed time between consecutive probes monitored in the darknet. This gives an insight revealing if attackers prefer to use long stealth scans or short and so easily discoverable ones. Moreover it helps to fine-tune our proactive scan blocking technique in regards to the period of time an IP address should be filtered.

In Figure 9, we compute the mean value of quartiles over the FR and JP dataset on a weekly basis. The median is around 10ms, so the darknet receives an average of 100 scans probes by seconds but the frequency of probing packet tends to increase over the weeks. Therefore, scans are generating a lot of noisy traffic over Internet. They participate to congest network and are also actually part of DDoS [?]. Proactive blocking is helpful to reduce the footprint of such ever-growing attack technique.

Although the previous observation is global and confirms the general trend regarding the continuous phenomena of scanning, *i.e.* every machine in the Internet is constantly solicited by TCP connections, the inter-scan is computed considering both source and destination IP addresses (pair-wise) before being averaged in Figure 10. In this case, median time is around 100 seconds between scans. This number seems quite high because darknet input packets are not composed only of scans but can also contain packets lost because of misconfigured network elements. As this kind of misconfiguration should not be so frequent, it can explain these high median and third quartile numbers. This explanation has also the advantage to explain the gap of around 90 seconds between median and first quartile. With these considerations in mind, we can consider that this results expose the same conclusion as previous ones and so that scanning activities are constant but raises over the time of our experiment.

5. EVALUATION OF ATTACKER-BASED PORT DISTANCES

In order to assess the veracity of our inter-port distances and because no grountruth actually exists, extensive experimentations cannot be done and we illustrate

the results on some example of port numbers. However, for an extensive evaluation, next section integrates the distance into a port blocking application and shows the benefits of our distance when applying to a realistic scenario.

We select the 60 most smallest distances and represents them as annotated edges between ports in Figure 11 and 12 (respectvely for *shortest-path-based* distance and *communities-structure-based* based distance). For the distance metric *shortest-path-based*, these 60 smallest edges create a highly connected graph between most widely used ports in web applications. In particular, we clearly distinguish the smallest distances (in bold) between HTTP-related ports: 80 (HTTP), 443 (HTTPS) and 8080 (Alternative HTTP). Moreover, these ports are also connected to email services ports. It is relevant as in usual deployment, an email server is frequent, to be also used by web services for instance (to send notifications). FTP port is also very close to web ports. Indeed, FTP was largely used in the past for updating web pages (especially personal home pages). Other connections like between ports 22 and 3389 are logic because they are related to standard services (SSH and remote desktop) to open session on remote computers. This shows an ability of this semantic distance to extract and represent several type of semantics between network ports. However, for the distance metric *communities-structure-based*, we do not find the same low distance semantic and the same high connectivity than in the previous one. Ports seem to be linked in a random order and without any semantic. This would be explained by the design of the algorithm that sustains this distance metric. Indeed, in the last iterations of the edges removing phase of this algorithm, almost all the edges connect communities of two nodes that are then not especially semantically connected. Thus, real semantic distances are, for this distance metric, not located in the smallest distances of this metric.

6. PREVENTIVE PORT BLOCKING WITH SEMANTIC PORTS DISTANCE

In this section, we present an application of the attacker behavior-based distance to demonstrate the validity of the latter within a practical application.

Assuming a probed port is detected with a regular method such as those deployed in intrusion detection system (*e.g.* based on the number or ratio of TCP connection request to closed ports), our preventive port blocking scheme aims to predict future port numbers which will be probed to discard the traffic accordingly. It may be thus part of an Intrusion Prevention System (IPS). A very restrictive technique would fully blacklist an IP address performing a scan but our approach is more fine-grained as blacklisting selected ports. This avoids collateral effects when an IP address is shared by multiple users, for instance if an attacker relies on virtual machines hosted in a cloud service provider. Because probing or scanning is an initial step to discover reachable hosts and services, defeating it reduce the attacker visibility and so limits her ability to craft a very tailored attack.

Assuming an IP address detected as performing a scan towards a given port number, our method preventively block the ingoing traffic from this IP address towards the K nearest ports for a user-defined period of time. For instance, when a scan happens on port 80 (HTTP) our semantic distance will filter traffic towards 443 and 8080 assuming $K = 2$ (as well as the initial port). The method is voluntary simple (compared to sophisticated methods with advanced modeling or techniques like machine learning) in order to focus our evaluation on the veracity of our new inter-port distance. The advantage is also to limit he overhead because the distances are computed beforehand.

As with *communities-structure-based* distance metric, the semantically more relevant distances are not located in the smallest distances of the metric, we limit our usage of semantic distances to the second one, namely *shortest-path-based*.

6.1 Evaluation methodology

The distances between ports are derived using the overall dataset (combining both the JP and FR darknet) over the period 6 month between January and June 2015. We are thus able to test our proactive blocking in the remaining data of the darknet from 1st July 2015 to 7th July 2015. The same scan extraction method as described in section 4 is used.

We define two performance metrics:

1. Blocking ratio: The percentage of probed ports pro-actively blocked (%blocked).

2. Usefulness: The percentage of blocked ports which are effectively probed afterwards (%usefulness).

In fact, quantifying the number of false positives or legitimate traffic which is blocked by our technique is impossible because darknet data does not contain mix traffic with attack and legitimate traffic. Our usefulness metric is thus more drastic by only considering as a valid, only blocked ports targeted by a scan. However, in section 6.4, a real dataset with mix traffic is used in order to evaluate the number of false positives.

6.2 Baseline scenarios

In order to assess the benefit of our metric, two baseline scenarios are considered:

- *MySelf*: this strategy employs a simple algorithm which consists in simply blocking the currently probed port number since our observations in sec-

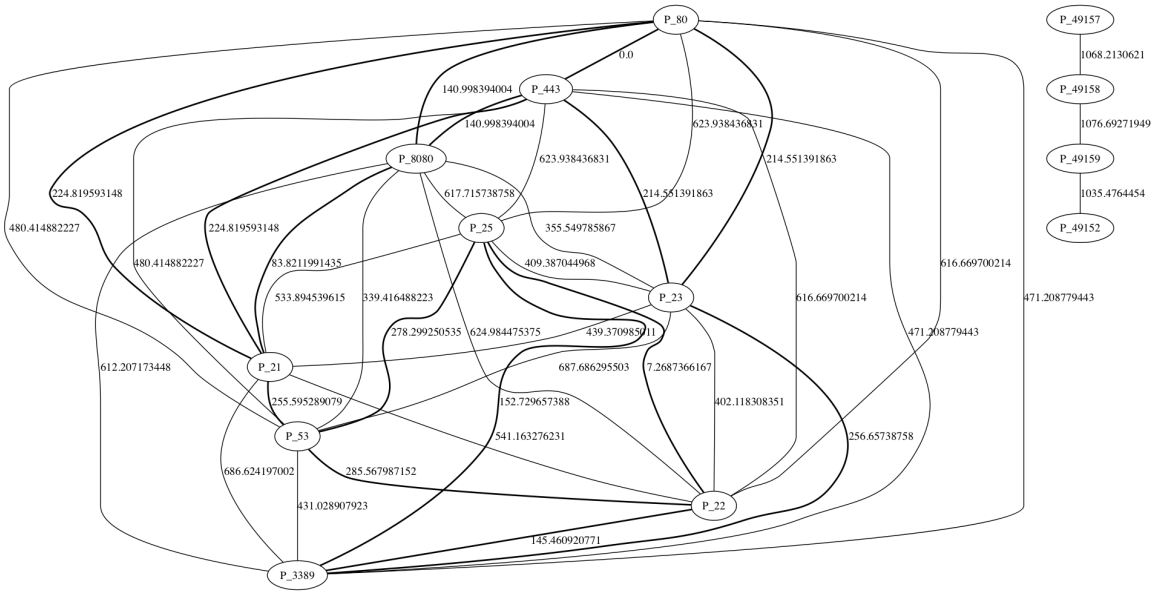


Figure 11: Network ports graph linked with the 60 smallest *shortest-path-based* distances

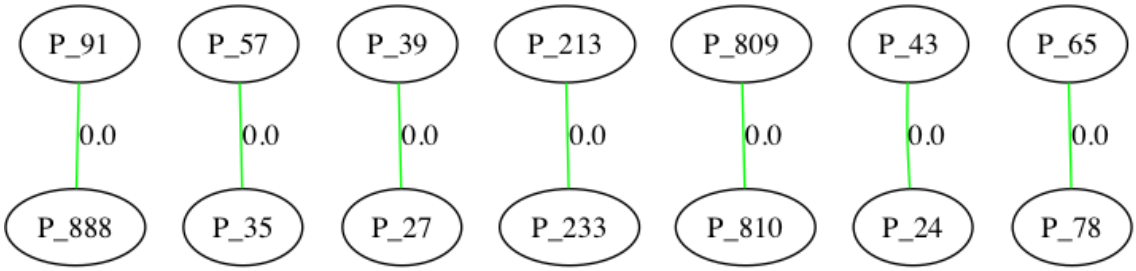


Figure 12: Network ports graph linked with the smallest *communities-structure-based* distances

tion 4 shows that an attacker usually targets the same ports multiple times.

- *Euclidian*: this algorithm blocks the K nearest ports using an euclidean (non semantic) distance between the port numbers in addition to the initially probed port. If the scanned port is 80, and K = 2, the set of ports to block will be 80, 81 and 82 (when equality, the highest port number is selected in priority, *e.g.* 81 with $K = 1$)

In Figure 13a and Figure 13b, the *MySelf* strategy leads to block between 5 and 25% of scans with a maximum of 18% usefulness.

In Figure 13a, more scans are pro-actively blocked with the *Euclidean* distance algorithm though the usefulness, in Figure 13b, is lowered and so potential false positives could increase in a real case of mixed traffic.

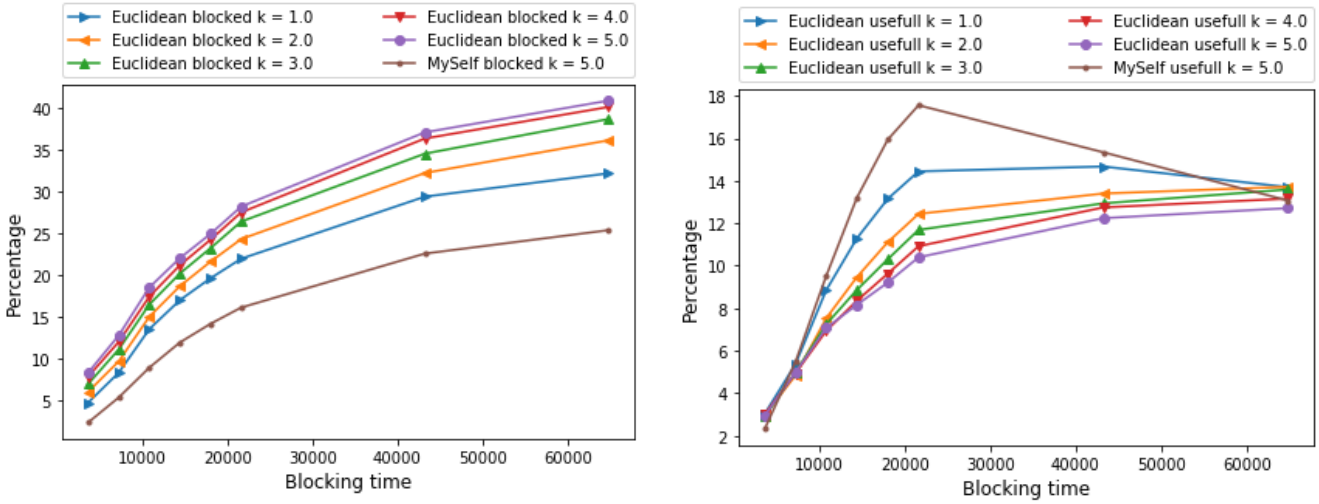
Regarding the parameters, increasing k is equivalent to increase the number of ports blocked by definition including those which are effectively probed afterwards and those which are not aimed by the attacker. As a result, this blocking ratio logically increases. However, as highlighted by the decreasing the usefulness, new ports blocked when increasing K have a higher probability to not be targeted in the future. Hence, increasing k is not a good strategy to improve the efficiency of the baseline Euclidian-based technique.

A higher blocking time also contribute to block more ports. A similar effect is thus observed on the blocking ratio. However it tends to enhance the usefulness until an upper-bound around 20000 seconds (5h30). Therefore, the port blocking is efficient with this time horizon.

6.3 Results

In this section, results of the proactive port blocking based on the attacker behavior-based distance are provided. The goal is to compare its benefit compared to the baseline scenarios while evaluating the impact of two main parameters: (1) the number of ports to be preventively blocked (K) and (2) the time that a source IP address has to be banned for. We compare results assuming JP or FR datasets, respectively in Figure 14c and Figure 14b, or both together in Figure 14a.

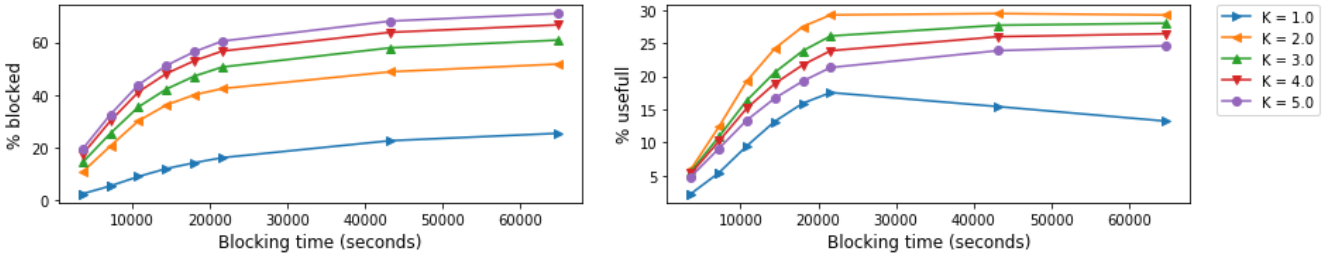
There is a significant improvement in both blocking



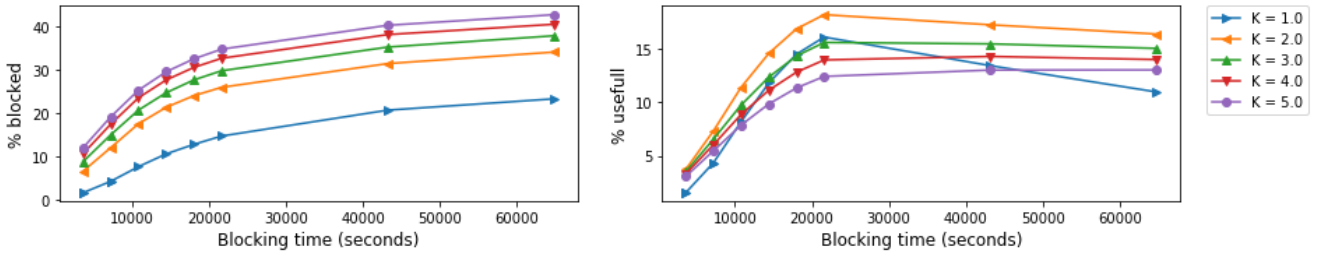
(a) Blocking performance

(b) Usefulness performance

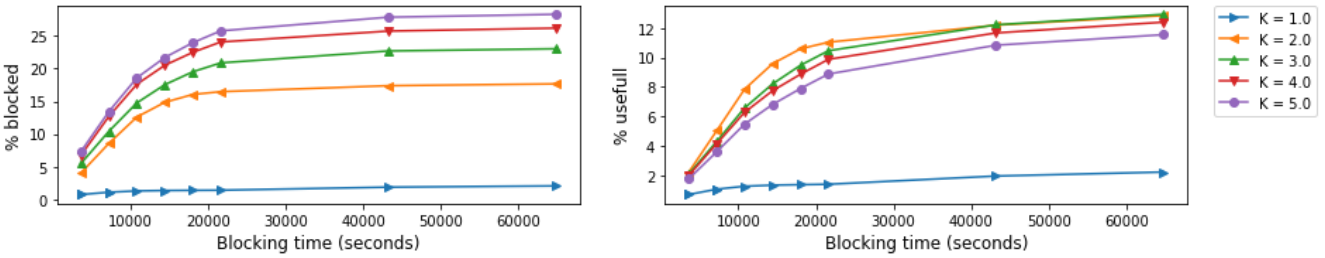
Figure 13: Port blocking - baseline scenarios



(a) Results for both darknets (FR, JP)



(b) Results for french darknet



(c) Results for japanese darknet

Figure 14: Blocking statistics for K Nearest ports algorithm with the attacker behavior-based

percentage and usefulness compared to baseline scenarios. Up to 40% and 30% of scans are blocked for the FR and JP dataset respectivelyin Figure 13. Globally, this percentage can reach around 70% in Figure 14a. Besides, the percentage of usefulness is about 15% for FR and near 12.5% for JP giving a maximum global

Table 2: Application of preventive port blocking in traditional network with K=3

	TPR	FPR	Usefulness
Minimum	99.94%	0.0000012%	47.79%
Mean	99.98%	0.0015%	66.97%
Maximum	99.99%	0.0091%	83.33%

usefulness of 30% whereas in the baseline scenarios is 18%.

Regarding the impact of parameter values, the usefulness increases when the blocking period of time increases until around 5h30 with the overall dataset. Unlike baseline scenarios, the relation between the value of K and accuracy is not similar to baseline cases. A good tradeoff between blocking ratio and usefulness is $K = 3$ in order to block around 50% of all scans with a usefulness of 25% assuming an optimal blocking time of 20000 seconds.

6.4 Test with real traffic

Based on distance learn on the darknet data (both JP and FR) and the best tuning of parameters highlighted in the previous section, the proactive blocking technique is applied on real data from the MAWI Labs dataset[20]. This dataset contains packet level labeled network data gathered from an oceanic backbone between United States and Japan. It is composed of 15 minutes long by day labeled network raw network packets captures. Instead of exclusive abnormal traffic as collected data, it contains benign traffic. Therefore, the false positive rate (FPR) can be calculated.

Having a low usefulness (always lower than 30% in our previous experiments) may not be a problem if the useless predicted ports are not use by legitimated communication also. In that case, a port is blocked but without discarding any traffic, the FPR remains low. However, if the automatically blocked port affects benign traffic, the FPR increases.

We used labeled dataset from 2 to 9 September 2015 (except the 5th and 7th of September because of dataset unavailability). Each day dataset is composed of 15 minutes of scan that represent all together a total of 590,173,645 packets in 6 days with a mean of 98,362,274 packets a day.

Because only 15 minutes of data per day are labeled, using a 5h30 our blocking time is not relevant and it is so set to 15 seconds.

As shown in Table 2, 99.9% of scans are effectively blocked in a proactive manner. It is due to lower variety of the targeted ports compared to a dedicated security sensor such as the darknets. The usefulness presents also higher values but the most interesting is the low FPR largely under 0.01% decreasing to near 0%. These results prove that the new semantic port distance, defined regarding attack behaviors, allows to cleverly block network ports before an attack or a network scan occur.

7. CONCLUSION

In this paper, a new attacker behavior-based inter-port distance is introduced to catch automatically two types of natural semantics: the service- and context-semantic. The metric that is presented here extracts attackers' port semantic knowledge from scanning activities collected from darknet. Observations done with the darknet over a long time period highlights scanning behaviors and motivate the definition of the new distance between port numbers. In order to assess its veracity in an extensive manner, a proactive blocking technique has been defined using this distance. Using real data, we showed that more than 99% of scans can be blocked in advance with less than 0.1% of legitimate traffic blocked.

Possible future work would compare our semantic port distance to other ones that would use a document and text based approach to network port semantic.

8. REFERENCES

[1] S. Lagraa and J. François, "Knowledge discovery of port scans from darknet," in *Symposium on Integrated Network and Service Management (IM)*. IFIP/IEEE, 2017.

[2] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[3] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, C. Dovrolis, Ed. Springer, 2005.

[4] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[5] A. K. J. Michael, E. Valla, N. S. Neggatu, and A. W. Moore, "Network traffic classification via neural networks," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-912, Sep. 2017. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-912.pdf

[6] W. D. Donato, A. Pescape, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *Network*, vol. 28, no. 2, pp. 56–64, March 2014.

[7] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification,"

Transactions on Networking, vol. 23, no. 4, Aug. 2015.

[8] L. Grimaudo, M. Mellia, E. Baralis, and R. Keralapura, "Select: Self-learning classifier for internet traffic," *Transactions on Network and Service Management*, vol. 11, no. 2, pp. 144–157, June 2014.

[9] S. E. Coull, F. Monrose, and M. Bailey, "On measuring the similarity of network hosts: Pitfalls, new metrics, and empirical analyses," in *Network and Distributed System Security Symposium*, 01 2011.

[10] M. H. Bhuyan, K. Bhattacharyya, and J. K. Kalita, "Surveying port scans and their detection methodologies," in *The Computer Journal*, vol. 54, 10 2011, pp. 1565–1581.

[11] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks."

[12] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *Conference on Security*. USENIX Association, 2013.

[13] M. Coudriau, A. Lahmadi, and J. Francois, "Topological Analysis and Visualisation of Network Monitoring Data: Darknet case study," in *International Workshop on Information Forensics and Security (WIFS)*. Abu Dhabi, United Arab Emirates: IEEE, 2016. [Online]. Available: https://hal.inria.fr/hal-01403950

[14] C. Fachkha and M. Debbabi, "Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization," *Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1197–1227, 2016.

[15] E. Balkanli, J. Alves, and A. N. Zincir-Heywood, "Supervised learning to detect ddos attacks," in *Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on*, Dec 2014, pp. 1–8.

[16] P. Mell and R. Harang, "Limitations to threshold random walk scan detection and mitigating enhancements," in *Communications and Network Security (CNS)*. IEEE, 10 2013, pp. 332 – 340.

[17] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Security and Privacy*. iEEE, 5 2004, pp. 211 – 225.

[18] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," in *PNAS*, 2002.

[19] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *SIGCOMM Conference on Internet Measurement (IMC)*. ACM, 10 2004, pp. 27 – 40.

[20] (2018, 9). [Online]. Available: http://www.fukuda-lab.org/mawilab/v1.1/index.html