Détection d'attaques sur les équipements d'accès à Internet

Gilles Roudiere

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *7 septembre 2018* **par :**

GILLES ROUDIÈRE

Détection d'attaques sur les équipements d'accès à Internet

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

LAAS-CNRS : Laboratoire d'Analyse et d'Architecture des Systèmes

Directeur de Thèse :

Philippe OWEZARSKI

Jury :

KAVÉ SALAMATIAN	Professeur des Universités	Rapporteur
SANDRINE VATON	Professeur	Rapporteur
VINCENT NICOMETTE	Professeur des Universités	Examinateur
PATRICE ABRY	Directeur de Recherche	Examinateur
PEDRO CASAS	Chargé de Recherche	Examinateur
PHILIPPE OWEZARSKI	Directeur de Recherche	Directeur de thèse
FRANÇOIS DEVIENNE	Ingénieur	Invité

Remerciements

Les travaux de thèse présentés dans ce manuscrit ont été réalisés au sein du Laboratoire d'Analyse et Architecture des systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS). J'adresse un remerciement à Messieurs Jean ARLAT et Liviu NICU pour avoir dirigé tour à tour le laboratoire durant ces trois années de travaux.

Je souhaite remercier mon directeur de Thèse, Philippe OWEZARSKI pour son aide précieuse et son support toujours sans faille. Ses conseils ont toujours été avisés. Ce manuscrit est le résultat d'une véritable collaboration. Je remercie aussi François DEVIENNE, mon co-encadrant, qui a su nous faire confiance pour mener à bien ce projet, tout en nous apportant toute l'aide nécessaire à son bon déroulement. Aussi, un grand merci à Véronique BAUDIN, cette thèse doit beaucoup à son travail.

J'adresse des remerciements particuliers aux professeurs Kavé SALAMATIAN et Sandrine VATON pour avoir accepté d'être les rapporteurs de mes travaux de thèse. Je remercie Messieurs Patrice ABRY, Pedro CASAS et Vincent NICOMETTE, pour avoir accepté d'examiner mon travail en prenant partie au jury de thèse. Un grand merci à tous les membres du jury pour leur travail et le temps qu'ils ont su m'accorder.

Je remercie tous les membres du LAAS, et particulièrement l'ensemble de l'équipe SARA, qui ont su m'accueillir avec bienveillance. Un grand merci à Khalil DRIRA et Olivier BRUN pour avoir dirigé l'équipe. J'admire d'ailleurs la patience de mes collègues qui ont su supporter mon humour douteux et mes goûts musicaux polémiques. Une qualité à reconnaître tout particulièrement à mes collègues de bureau : Juliette, Rémy, Imane et Quentin. J'adresse aussi un merci à Nicolas, Inès, Guillaume et François (respectivement Tic et Tac... ou l'inverse, je ne sais pas), Santiago, Ioanna, Nicola, Tom, Hang, Nga, Pascal, Yann et Chloé pour leur amitié et leur soutien.

Un grand merci à "la commu" pour me supporter depuis tant d'années, merci à tous mes amis. Un merci à Dj I.G.N (Benoît) et Arlord (Arnaud) pour tous les moments partagés dans notre collocation, la bien nommée "La Bolosserie". Merci au collectif 30.000% Makina.

Enfin, je remercie mes parents, mes grand-mères et Coraline pour leur soutien sans faille.

Résumé

Les anomalies réseaux, et en particulier les attaques par déni de service distribuées, restent une menace considérable pour les acteurs de l'Internet. La détection de ces anomalies requiert des outils adaptés, capables non seulement d'opérer une détection correcte, mais aussi de répondre aux nombreuses contraintes liées à un fonctionnement dans un contexte industriel. Entre autres, la capacité d'un détecteur à opérer de manière autonome, ainsi qu'à fonctionner sur du trafic échantillonné sont des critères importants.

Au contraire des approches supervisées ou par signatures, la détection non-supervisée des attaques ne requiert aucune forme de connaissance préalable sur les propriétés du trafic ou des anomalies. Cette approche repose sur une caractérisation autonome du trafic en production, et ne nécessite l'intervention de l'administrateur qu'à postériori, lorsque une déviation du trafic habituel est détectée. Le problème avec de telle approches reste que construire une telle caractérisation est algorithmiquement complexe, et peut donc nécessiter des ressources de calculs conséquentes. Cette exigence, notamment lorsque la détection doit fonctionner sur des équipements réseaux aux charges fonctionnelles déjà lourdes, est dissuasive quant à l'adoption de telles approches.

Ce constat nous amène à proposer un nouvel algorithme de détection non-supervisé plus économe en ressources de calcul, visant en priorité les attaques par déni de service distribuées. Sa détection repose sur la création à intervalles réguliers d'instantanés du trafic, et produit des résultats simples à interpréter, aidant le diagnostic de l'administrateur. Nous évaluons les performances de notre algorithme sur deux jeux de données pour vérifier à la fois sa capacité à détecter correctement les anomalies sans lever de faux-positifs et sa capacité à fonctionner en temps réel avec des ressources de calcul limitées, ainsi que sur du trafic échantillonné. Les résultats obtenus sont comparés à ceux de deux autres détecteurs, FastNetMon et UNADA.

Table des matières

Chapitre 1

Introduction

Sommaire

La détection des anomalies reste une préoccupation majeure de nombreux acteurs de l'Internet. C'est un problème complexe, ayant attiré l'intérêt à la fois des acteurs industriels, comme les Fournisseur d'Accès à Internet (FAI), et les membres de la communauté scientifique. Ces anomalies, définies comme des évènements déviant du comportement habituel du trafic, peuvent avoir des conséquences sévères sur les services rendus.

Si certaines anomalies peuvent être la conséquence d'erreurs de configuration, ce sont les scans et attaques qui intéressent le plus les acteurs du numérique. Ces anomalies sont généralement beaucoup plus difficiles à éliminer puisqu'elles sont le fruit d'une véritable volonté de nuire. Elles dépendent alors directement de la motivation et des ressources de l'attaquant. Parmi ces attaques, on trouve la fameuse attaque par Déni de Service Distribuée (DDoS). Ce type d'attaque consiste en de nombreuses sources de trafic, distribuées

en plusieurs endroits sur Internet, envoyant un très grand nombre de fausses requêtes à un système cible. Épuisé par ces requêtes, le système visé ne peut alors plus rendre un service efficace. Ces attaques peuvent avoir des conséquences graves. On prendra pour exemple l'attaque de mai 2017 [1], visant plusieurs sites d'information (dont *Le Monde* ou *Le Figaro*) ayant provoqué leur complète indisponibilité pendant une heure. Ou encore, très récemment, l'attaque DDoS ayant visé le site *Github* (pour l'hébergement et la gestion de projets de développement logiciel), ayant culminé à $1,3$ *Tbps*. Un record battu moins d'un mois plus tard par une attaque à 1.7 *Tbps* [2]. L'élimination de ces attaques, puisque particulièrement nombreuses et dangereuses, sont une priorité pour nombre d'acteurs industriels. D'autant que leur conséquences peuvent dépasser le simple aspect financier, comme lorsque des questions politiques sont en jeu [3]. Ainsi, il est alors nécessaire de se prémunir contre de telles anomalies. Cela nécessite leur détection et leur caractérisation à l'aide d'outils efficaces.

Certains détecteurs opèrent au niveau d'un hôte, une approche idéale pour détecter des anomalies ayant un impact se limitant à une seule machine. Néanmoins, certaines attaques, incluant celles par Déni de Service Distribuées, peuvent saturer complètement le système ciblé. Dans certains cas, la saturation peut arriver en aval, à l'entrée du réseau par exemple, ce qui rend une atténuation au niveau de l'hôte inutile. C'est pourquoi une détection au niveau du réseau devient nécessaire. D'autant que de telles approches permettent de mutualiser les ressources et de regrouper en un seul point les outils de détection.

Néanmoins, la détection d'anomalies réseau est un problème ayant de nombreuses composantes. Elle nécessite des outils capables d'opérer en temps réel, de façon autonome, sur un trafic à haut débit, disparate et en constante évolution. À cela s'ajoute la nature changeante des anomalies à détecter, pouvant mettre en échec les outils de détection. À toutes ses considérations s'ajoutent des contraintes industrielles, dues aux capacités technologiques des équipements actuels. En particulier, lorsque la détection doit être effectuée au niveau réseau, les équipements utilisés (routeurs ou commutateurs) utilisent en priorité leurs ressources pour la gestion du trafic. Les capacités restantes sont donc limitées, obligeant à concevoir des méthodes de détection peu gourmandes en ressources de calcul. Aussi, les architectures de surveillance actuelles opèrent sur du trafic échantillonné (reposant sur sFlow [1] ou IPFIX [2]), ce qui, certes, limite l'utilisation des ressources nécessaires aux opérations du détecteur, mais peut amener à une détection dégradée. Ainsi, cette thèse cherche à proposer de nouvelle solutions, prenant en compte l'ensemble des composantes de cette problématique.

1. https ://www.lemondeinformatique.fr/actualites/lire-une-attaque-ddos-perturbe-des-sites-de-presse-francais-68162.html
2. https ://www.numerama.com/tech/333738-ddos-record-encore-battu-avec-une-attaque-mesuree-a-17-tbps.html
3. https ://www.numerama.com/politique/325269-une-attaque-ddos-cible-les-principales-banques-des-pays-bas.html

Dans ce chapitre, nous présentons en premier lieu les motivations du projet AATAC, pour lequel ont été menés nos travaux. La problématique à laquelle nous répondons est ensuite exposée. Enfin, nous introduisons nos différentes contributions permettant de répondre à ce problème.

1.1 Le projet AATAC

Cette thèse a été menée dans le contexte du projet AATAC (Autonomous Algorithm for Traffic Anomaly Characterization), réunissant le Laboratoire d'Analyse et Architecture des Systèmes (LAAS-CNRS) et la société Border 6 (dont le logo est représenté sur la figure 1.1).

FIGURE 1.1 – Logo de la société Border 6

Border 6, désormais partie du groupe Expereo, est une PME proposant des services de surveillance des réseaux. Ils proposent *Non-Stop Internet*, une solution tout-en-un cherchant l'optimisation des coûts, l'amélioration de la disponibilité et de la performance des réseaux IP-BGP en multihoming.

Comme exposé plus tôt, certaines anomalies ou attaques peuvent provoquer de graves dysfonctionnements du système, allant jusqu'à rendre complètement inaccessibles les services touchés. Assurer la disponibilité du réseau à chaque instant nécessite alors de détecter, de caractériser et mitiger ces anomalies. Or les techniques utilisées à ce jour par Border 6 reposent sur des techniques de seuillages, généralement difficiles à configurer et pouvant amener la levée de fausses alarmes. Des approches plus modernes, reposant sur des techniques d'apprentissage automatique permettent une détection plus autonome, tout en fournissant une détection efficace et aidant l'administrateur à effectuer son diagnostic. Le projet AATAC vise donc l'élaboration de nouvelles techniques de détection tirant parti des approches récentes tout en répondant aux contraintes d'un contexte industriel.

Dans le cas de Border 6, les contraintes d'opération sont particulières, d'abord par la nature de ses équipements. En premier lieu, pour des raisons opérationnelles, ils opèrent à l'entrée du réseau de l'entreprise à surveiller. La détection doit donc être réalisée au niveau réseau. Aussi, ces équipements opèrent un certain nombre de traitements permettant la gestion du trafic. La détection ne doit donc pas avoir d'impact sur leur bon fonctionnement. Par conséquent, le détecteur doit opérer avec des ressources de calcul particulièrement

limitées. Aussi, les débits considérés nécessitent que les équipements surveillés opèrent sur du trafic échantillonné. Cela peut avoir un impact sur la qualité de la détection.

Nous notons aussi que même si l'ensemble des anomalies peut perturber le bon fonctionnement du réseau, ce sont les attaques par Déni de Service Distribuées qui intéressent en priorité Border 6. D'après leur expérience, et comme mentionné plus tôt, ce sont ces attaques qui perturbent le plus la disponibilité des services de leurs clients. Notre projet cherche donc, en priorité, à produire et évaluer de nouvelles approches répondant à cette problématique.

1.2 Problématique

Nous présentons ici notre problématique, divisée en trois parties. Tout d'abord, nous abordons l'ensemble des éléments à considérer lors de la construction d'un détecteur. Nous abordons ensuite plusieurs difficultés liées au trafic actuel. Enfin, nous discutons de la spécificité des attaques par déni de service distribuées.

1.2.1 Construire un bon détecteur

De nombreux paramètres sont à prendre en compte lors du choix d'un détecteur d'anomalies. Si la qualité de la détection est bien sûr à prendre en compte, l'utilisation d'un détecteur nécessite aussi la mobilisation de ressources, qu'elles soient liées au travail d'un administrateur ou aux infrastructures nécessaires à l'exécution du détecteur. Chaque solution nécessite alors une véritable analyse coût-bénéfice.

Exhausitivité et robustesse. L'utilité d'un détecteur est tout d'abord conditionnée par sa capacité à détecter les anomalies, de manière efficace et inconditionnelle. La qualité de la détection est donc l'un des paramètres à prendre en compte lors du choix d'une solution. La sensibilité (ou taux de vrais positifs) d'un détecteur correspond à sa capacité à lever une alerte lorsqu'une anomalie est présente dans le trafic. Un détecteur idéal est alors très sensible, détectant toutes les anomalies présentes dans le trafic.

Aussi, un détecteur utilisable doit pouvoir opérer en temps réel sur le trafic tout en détectant les anomalies au plus tôt, avant qu'elle ne cause des dommages irrémédiables au système visé.

Il doit enfin être robuste, pouvant opérer en toutes situations. Un pic d'utilisation ne doit notamment pas perturber le bon fonctionnement du détecteur. En cas d'attaque, le détecteur doit rester opérationnel autant que possible, au moins le temps qu'une contremesure puisse être mise en place.

Réduction du travail de l'administrateur. Le travail de l'administrateur est l'un des coûts importants à considérer pour l'adoption d'un détecteur. Son temps est précieux, un bon détecteur doit donc tout faire pour limiter l'intervention de l'administrateur. Plusieurs points sont alors à prendre en compte.

Si un détecteur détectant toutes les anomalies serait évidemment intéressant, le fait est que, dans un cas réaliste, un détecteur très sensible peut avoir tendance à générer des alertes alors qu'aucune anomalie n'est présente dans le trafic. On parle alors d'un faux positif. La sensibilité d'un détecteur, évoquée plus tôt, est par conséquent toujours à considérer conjointement avec le taux de faux positifs, soit la proportion d'alarmes levées ne correspondant pas à une vraie anomalie. La capacité d'un détecteur à générer peu de faux positifs est alors primordiale, l'analyse de chacune de ces fausses alertes gaspillant le temps de l'administrateur. Lorsque le détecteur génère un trop grand nombre de ces faux positifs, l'administrateur peut, dans le pire des cas, perdre confiance et finir par ignorer les alertes générées.

Ceci dit, au-delà de la qualité de la détection en elle-même, ce sont les résultats produits par le détecteur qui permettent une gestion efficace des anomalies. Lorsqu'il lève une alerte, un bon détecteur doit alors fournir à l'administrateur assez d'informations pour une prise de décision avisée, notamment lorsque le détecteur ne peut produire une information assurément fiable. Le détecteur doit notamment produire une caractérisation de l'anomalie, c'est-à-dire une liste d'attributs permettant de tracer une frontière entre le trafic normal et celui de l'anomalie. Aussi, lorsque plusieurs évènements sont détectés, mais correspondant à une même anomalie, le détecteur doit être capable de corréler ces évènements et doit fournir l'ensemble de ces informations de manière compacte. Les outils d'analyse fournis avec un détecteur, graphiques ou non, font donc partie intégrante d'une solution.

Si un détecteur fournissant des informations pertinentes est un avantage, un détecteur capable d'opérer de manière autonome la plupart du temps est bien plus appréciable encore. Cette autonomie repose sur plusieurs critères.

Tous d'abord, à condition qu'il dispose de suffisamment d'informations, un détecteur doit pourvoir appliquer automatiquement des contre-mesures si l'anomalie détectée est assurément un danger au bon fonctionnement du système. L'échelle de temps nécessaire à l'intervention d'un administrateur étant généralement bien plus grande que celle nécessaire à un traitement automatique, c'est un avantage certain.

L'autonomie d'un détecteur est aussi conditionnée par le travail nécessaire à l'installation et la mise à jour du détecteur. Le détecteur doit pouvoir opérer sur des trafics fondamentalement différents et s'adapter de manière fluide à la fois aux évolutions du trafic normal mais aussi à l'apparition de nouvelles attaques et anomalies. Un détecteur autonome propose généralement une approche proactive, il ne nécessite l'intervention de l'ad-

ministrateur que lorsque un évènement suspect a été effectivement détecté. Aussi, lorsque l'administrateur a pris une décision vis-à-vis d'une anomalie, un détecteur autonome sera capable de prendre en compte cette décision, et appliquera, si une anomalie similaire se présente, la même décision automatiquement.

Ceci étant dit, une approche totalement autonome n'est par réaliste. L'intervention de l'administrateur sera toujours nécessaire, puisque la définition même d'une anomalie dépend de la politique de sécurité à appliquer, et qu'elle est forcément en constante évolution.

Coût matériel. Le travail de l'administrateur n'est pas le seul coût à prendre en compte lors de l'utilisation d'un détecteur. Chaque solution nécessite l'utilisation de plus ou moins de ressources de calcul. Suivant l'architecture sur laquelle repose le détecteur, ces ressources peuvent être limitées, notamment lorsque la détection n'opère pas sur un matériel dédié. L'utilisation d'un détecteur nécessitant des ressources de calcul modérées, et si possible, constantes, peut donc être nécessaire.

1.2.2 Difficultés

Si la construction d'un détecteur d'anomalies nécessite de prendre en compte de nombreux paramètres, certaines difficultés ne sont pas inhérentes à la construction de tels détecteurs mais plutôt aux conditions dans lesquelles ils doivent opérer.

Évolution et disparités du trafic

Tous d'abord, force est de constater l'augmentation constante du trafic depuis plusieurs années. Le rapport d'Akamai du troisième trimestre 2017 [3] sur l'état d'Internet fait état, de 2010 à 2016, d'un quadruplement (de 20151 pétaoctets par mois à 88719) du trafic Internet et d'une augmentation de 70% du nombre d'utilisateurs (de $2,0$ à $3,4$ milliards). Cette évolution, expliquée et par le développement économique de plusieurs pays, et par le développement récent de l'Internet des objets, a conduit à la multiplication des équipements connectés à Internet, et par conséquent du trafic produit. Cette évolution a aussi un impact sur la nature du trafic. Le *streaming* vidéo, toujours plus populaire, représente désormais une grande partie du trafic traversant les réseaux (50% en France d'après le même rapport). Les applications supportées par le réseau sont toujours plus diverses.

Aussi, Internet est un réseau disparate. Chaque réseau connecté à Internet peut supporter des applications très diverses, résultant en des propriétés statistiques significativement différentes d'un point à l'autre du réseau. Les seuils de détection doivent être adaptés en conséquence, suivant le trafic, mais aussi suivant la politique de sécurité à appliquer.

Cette disparité spatiale s'ajoute à une disparité temporelle, liée par exemple aux cycles d'utilisation journaliers de la plupart des services.

La disparité et l'évolution du trafic ont deux conséquences directes : un détecteur parfait construit à un moment donné peut devenir inefficace en peu de temps, et un détecteur efficace en un endroit du réseau peut être complètement inefficace ailleurs. Construire des détecteurs capables de s'adapter au trafic de manière autonome devient plus qu'un avantage, une nécessité. Aussi, les débits actuels requièrent l'utilisation de solutions pouvant, ou passer facilement à l'échelle, ou pouvant opérer à des débits très importants (à l'aide de matériel dédié par exemple).

Le cas particulier des attaques par déni de service distribuées

S'ajoutent aux difficultés liées à l'évolution du trafic légitime, celles dues à la multiplication des anomalies, à la fois en nombre et en type. Les attaques sont des anomalies particulières puisqu'elles sont la conséquence du désir de nuire d'un attaquant. L'intérêt de l'attaquant est par conséquent que son attaque reste difficile à détecter et à contrer. Plusieurs techniques d'évasion lui sont alors disponibles afin de faire passer le trafic nuisible pour du trafic légitime. Aussi, l'attaquant peut potentiellement exploiter des faiblesses de conception inhérentes à certains détecteurs pour les rendre inopérants. Certaines attaques peuvent alors profiter de la complexité algorithmique de certains détecteurs pour épuiser leurs ressources, ce qui peut permettre de cacher une attaque sous une surcharge à priori inoffensive.

Un type d'attaque régulièrement utilisé dans ce but est l'attaque par déni de service distribuée. Ce type d'attaque vise à épuiser les ressources de la victime grâce à un très grand nombre de fausses requêtes. Comme illustré par la figure 1.2, ces attaques sont généralement lancées à l'aide de botnets, un ensemble de machines contrôlées par un attaquant et réparties sur le réseau. De telles attaques ont des conséquences graves, pouvant entraîner la paralysie des systèmes visés. En effet, d'après un rapport de Neustar [4] interrogeant un panel de 1010 entreprises, plus de 60% des entreprises interrogées estiment le coût de l'indisponibilité de leurs services à plus de $100.000 par heure. Le détail de ces réponses est détaillé sur la figure 1.3. D'après ce même rapport, plus de 50% de ces attaques nécessitent plus de 3 h pour être atténuées. Comme illustré par le graphique 1.4, les débits considérés sont conséquents.

Si le risque dû à une seule de ces attaques peut alors se chiffrer en millions de dollars, il doit encore être multiplié par leur fréquence. Toujours d'après la même étude, 84% des entreprises interrogées ont subi une attaque DDoS en 2017, contre 73% en 2016. La figure 1.5 illustre les résultats obtenus lors de la consultation sur la fréquence des attaques. Plus de 70% des entreprises ont subi au moins deux attaques en un an et 20% environ au

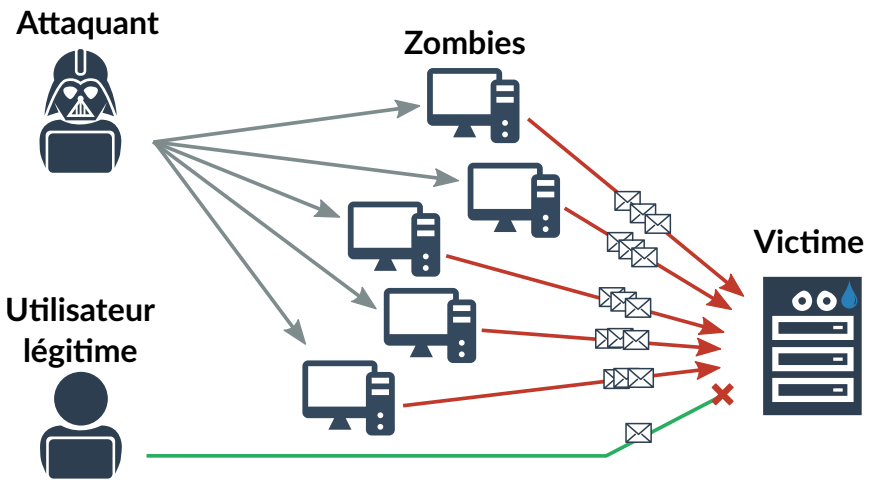


FIGURE 1.2 – Fonctionnement d'un botnet, utilisé pour une attaque DDoS

moins une par mois. Ce sont donc des attaques courantes.

La recrudescence de ces attaques est probablement due à deux facteurs. Tout d'abord, ces attaques sont faciles à mettre en place. Des outils comme LOIC [4] ont rendu accessible au grand public la mise en place de telles attaques, comme l'attaque de 2010 nommée *Operation : Payback*[5] et menée par un collectif appelé *Anonymous*. Cet outil, très simple d'installation et doté d'une interface graphique, a pu permettre à des amateurs de provoquer l'arrêt des sites de la MPAA (Motion Picture Association of America) et de l'IFPI (International Federation of the Phonographic Industry), en totalisant pour les deux sites un arrêt total de 30 heures. Aussi, la multiplication des équipements connectés à Internet (entre autre dus au développement de l'Internet des objets) ont contribué à l'utilisation de telles attaques. Ces équipements, parfois peu protégés, sont autant d'agents pouvant être détournés pour constituer un botnet, et utilisés pour exécuter une attaque distribuée. Un exemple parlant date de 2016, où un ensemble de 150000 caméras connectées a été utilisé pour opérer une attaque de presque 800 Gbit/s visant le fournisseur de cloud OVH [6].

Aussi, comme indiqué plus tôt, ces attaques sont couramment utilisées afin de masquer d'autres attaques, souvent plus pernicieuses. D'après Neustar, il existe une chance sur deux qu'une attaque DDoS soit utilisée pour masquer un autre type d'attaque. Une liste non exhaustive inclut l'installation de malwares ou de ransomwares, le vol de données clients, le vol d'une propriété intellectuelle, ou encore la destruction d'un matériel. Dans des cas moins graves, ces attaques peuvent ne faire qu'augmenter légèrement le coût d'exploitation

4. https ://sourceforge.net/projects/loic/
5. https ://www.pandasecurity.com/mediacenter/news/4chan-users-organize-ddos-against-mpaa/
6. https ://www.lesnumeriques.com/vie-du-net/attaque-ddos-botnet-cameras-securite-lache-1-tb-s-contre-ovh-n56187.html

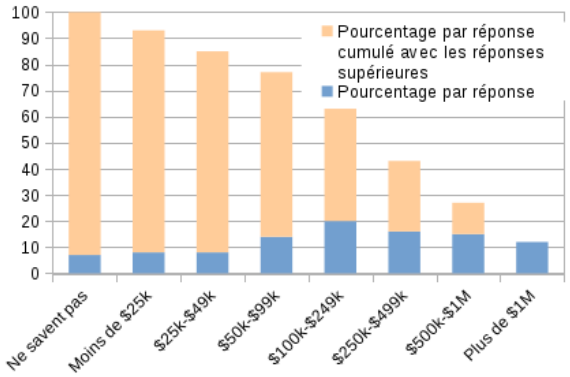


FIGURE 1.3 – Estimation du coût d'une heure d'indisponibilité des services (Source : Neustar)

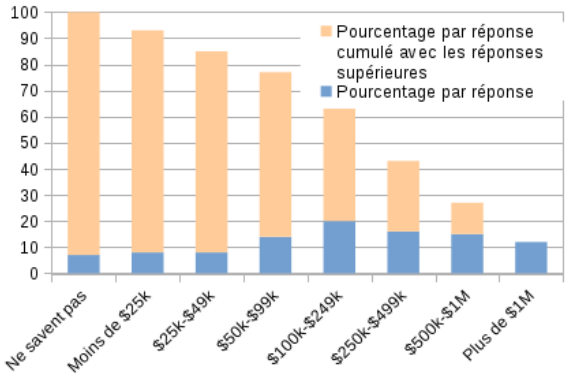


FIGURE 1.4 – Débit, en paquets par seconde, des attaques par Déni de Service Distribuées entre 2016 et 2017. (Source : Neustar)

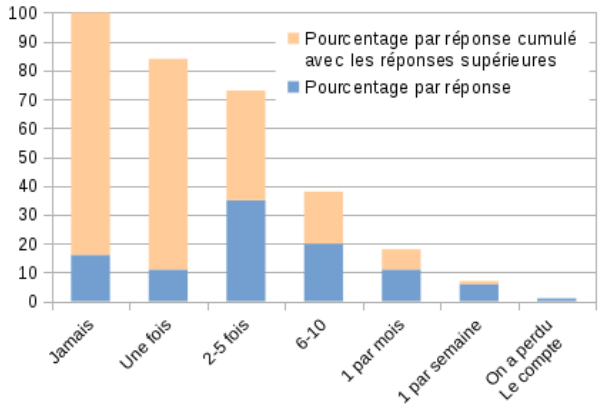


FIGURE 1.5 – Nombre d'attaques par Déni de Service Distribuées subies sur 12 mois, entre 2016 et 2017. (Source : Neustar)

par la consommation inutile de ressources nécessaires au traitement des fausses requêtes, ou entraînent un service dégradé.

Si ces attaques restent toujours aussi fréquentes aujourd'hui, c'est qu'elles restent particulièrement efficaces. Ceci s'explique notamment parce que leur détection et leur atténuation est un problème particulièrement difficile, et ce, pour plusieurs raisons : tout d'abord, la nature distribuée de ces attaques rendent l'identification de l'origine du trafic difficile. Par conséquent, caractériser le trafic d'une même attaque nécessite d'identifier d'autres critères de regroupement que l'adresse IP source. Aussi, par nature, ces attaques peuvent rapidement incapaciter le détecteur. Non seulement l'attaque doit être détectée et caractérisée très rapidement, mais l'ensemble des équipements sur lesquels reposent la détection

et l'atténuation de telles attaques doivent être robustes à de telles attaques. Enfin, les attaques par déni de service laissent à l'attaquant quantité de techniques d'évasions. Ces attaques, puisqu'elles consistent simplement en de fausse requêtes pour un système cible, peuvent alors être constituées d'un trafic très similaire à du trafic normal. L'usurpation d'adresse IP (ou *IP spoofing*) est une autre technique d'évasion couramment utilisée. Cette technique, consistant à utiliser de fausses adresses IP source pour générer le trafic illégitime, rend d'autant plus difficile l'identification et l'atténuation de ces attaques. Enfin, on notera l'attaque par *slow growth*, qui consiste alors à augmenter de façon lente le débit de l'attaque. Cela permet éventuellement de tromper certains algorithmes de détection, le trafic anormal apparaissant comme habituel au fil du temps.

1.2.3 Résumé

Comme nous avons pu le constater, il est très difficile de construire un détecteur capable d'exceller en tous points. Des compromis sont alors nécessaires. De par leur dangerosité, nous avons décidé de nous focaliser sur les attaques DDoS. Finalement, l'ensemble des questions résumant notre problématique est illustré par la carte mentale de la figure 1.6.

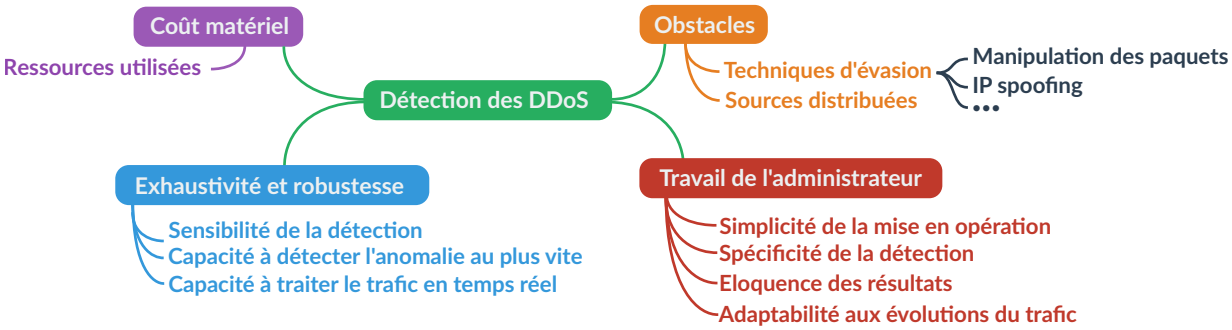


FIGURE 1.6 – Carte mentale de la détection d'anomalie

1.3 Contributions

Afin de répondre à notre problématique, cette thèse présente 3 contributions.

1.3.1 AATAC, un nouvel algorithme de détection d'anomalies

En premier lieu nous présentons un nouvel algorithme de détection d'anomalies éponymement nommé AATAC. Cet algorithme de détection non-supervisé est conçu pour opérer

en temps réel avec des ressources limitées. Il est capable de produire des informations détaillées sur les propriétés du trafic au moment de l'anomalie. Pour combiner à la fois une description des anomalies détaillées et une opération en temps réel, le traitement de notre algorithme se divise en deux parties : la première réalise une analyse statistique du trafic en un temps linéaire, alors que la deuxième opère une analyse plus complexe, à instants réguliers, permettant effectivement la détection d'anomalies.

1.3.2 Évaluation des performances d'AATAC en conditions réelles

La deuxième contribution consiste en l'évaluation de cet algorithme sur deux jeux de données construits par nos soins. Le premier jeu de données, nommé SynthONTS, est un jeu de données labellisé. Il est constitué d'un ensemble de traces capturées dans lesquelles ont été insérées plusieurs attaques générées dans un réseau émulé (incluant notamment plusieurs DDoS). Il a permis l'évaluation de la capacité de l'algorithme à détecter correctement les anomalies, tout en ne générant pas ou peu de faux positifs. Le deuxième jeu de données a été capturé à l'entrée du réseau de l'un des clients de Border 6 et a permis d'évaluer la capacité de l'algorithme à opérer en temps réel sur du trafic en production.

Les résultats obtenus avec AATAC sont comparés à ceux obtenus avec deux autres algorithmes de détection, à savoir ORUNADA et FastNetMon. Ces deux algorithmes, parmi les rares à proposer une implémentation nous étant disponible, sont représentatifs respectivement des approches non-supervisées et reposant sur une base de connaissance (voir chapitre 2).

1.3.3 Impact de différentes techniques d'échantillonnage sur les performances de l'algorithme

La dernière de nos contributions est une évaluation de l'algorithme sur des traces échantillonnées. L'échantillonnage est en effet l'une des nombreuses contraintes de l'industrie lorsqu'une opération avec des ressources limitées est nécessaire. Il peut par contre avoir un impact négatif sur la qualité de la détection. Pour notre évaluation, cinq techniques d'échantillonnage ont été utilisées : trois techniques reposant sur le nombre de paquets échantillonnés, et deux techniques d'échantillonnage temporel. Différents taux d'échantillonnage ont d'ailleurs été utilisés.

Nous évaluons à la fois l'impact sur la qualité de la détection d'AATAC et le gain en temps de calcul apportés par l'échantillonnage. Ces résultats sont comparés avec ceux obtenus par FastNetMon sur le même trafic.

1.4 Résumé

Cette thèse a été menée dans le cadre du projet AATAC, visant à proposer de nouvelles méthodes de détection d'anomalies réseaux. Ces approches, profitant des techniques récentes d'apprentissage automatique, cherchent à répondre aux problèmes principaux des industriels : à savoir une détection autonome, robuste, et visant en priorité les attaques par déni de service distribuées.

1.5 Plan

La suite de cette thèse contient 5 chapitres :

- Le chapitre 2 fait un état de l'art des techniques de détection actuelles. Illustrant l'évolution de ce domaine de recherche et la nécessité de nouvelles approches, compatibles avec les contraintes de l'industrie.
- Le chapitre 3 présente l'architecture de AATAC, notre algorithme de détection non-supervisé capable d'opérer avec des ressources de calcul limitées.
- Le chapitre 4 aborde l'évaluation de notre algorithme sur deux jeux de traces. La qualité de la détection de l'algorithme et sa capacité à opérer en temps réel avec des ressources contraintes sont alors évaluées et comparées à celles de deux autres algorithmes.
- Le chapitre 5 évalue l'impact de cinq techniques d'échantillonnage, à différents taux, sur les performances d'AATAC. Cet impact est mesuré à la fois sur la qualité de la détection et sur le gain en temps de calcul. Les résultats sont comparés à ceux obtenus avec FastNetMon.
- Le chapitre 6 conclut cette thèse en résumant l'ensemble de nos contributions et présentes des pistes d'amélioration pour nos travaux futurs.

Chapitre 2

État de l'art

Sommaire

Dans cette section, nous étudions ce qui a été réalisé dans le domaine de la détection des attaques et anomalies sur les équipements d'accès à Internet. Comme nous avons pu le constater dans le chapitre précédent, le problème de la détection d'anomalies est complexe. Comme de nombreux paramètres sont à prendre en compte pour juger de la qualité d'un détecteur, de nombreuses méthodes de classification sont possibles. Si des critères comme

la qualité de la détection ou la complexité algorithmique sont des critères particulièrement importants, ils sont difficiles à comparer objectivement d'une approche à l'autre sans mettre en place un protocole d'évaluation dédié.

Dans notre cas, nous avons choisi comme critère de classification la capacité des détecteurs à opérer de façon autonome. Le travail de l'administrateur, ou par extension d'experts en sécurité informatique, ont un coût conséquent. Ses interventions doivent donc être limitées autant que possible. Qui plus est, cette méthode de classification permet de distinguer des techniques reposant sur des principes fondamentalement différents, dont les hypothèses de départ, avantages et inconvénients, sont clairement identifiables.

Ainsi, dans ce chapitre, nous présentons en premier lieu des approches dites "systèmes experts". Ces détecteurs reposent sur une base de connaissances exhaustive, consistant en un ensemble de règles ou signatures, cherchant à reconnaitre un ensemble de motifs connus et précisément caractérisés dans le trafic. Nous présentons ensuite plusieurs détecteurs reposant sur de l'apprentissage supervisé. Ces approches entraînent un modèle à reconnaitre différentes classes de trafic grâce à un jeu de données labellisé. Enfin, nous présentons des approches non-supervisées, opérant sur le trafic en production sans connaissances préalables, et détectant les anomalies en isolant des comportements inhabituels repérés dans le trafic.

2.1 Systèmes experts

Les systèmes experts reposent fortement sur les connaissances que nous avons du trafic. Elles soumettent le trafic à un ensemble prédéfini de règles ou motifs, construits par un expert, permettant de reconnaitre des évènements bien particuliers.

Les anomalies connues sont nombreuses et variées, chacune d'entre elles se traduit par des conséquences différentes sur le trafic surveillé. Lorsque les caractéristiques d'une anomalie particulière sont clairement identifiées, il est alors possible de construire un ensemble de règles permettant de détecter cette anomalie précisément. Cet ensemble est appelé *la signature* ou *le profil* d'une attaque. On parle aussi de détection *par règles (rule-based)* ou *par signature (signature-based).*

Snort [5] and Bro [6] sont deux détecteurs open-source reposant sur des signatures. Lorsqu'un paquet traverse le détecteur, son contenu est évalué par un très grand nombre de règles permettant de vérifier si son contenu peut correspondre à une attaque connue. Ces deux détecteurs sont capables d'analyser le trafic à plusieurs niveaux d'agrégation : elles peuvent analyser chaque paquet ou suivre les connections pour détecter des motifs caractéristiques dans la charge utile.

La figure 2.1 présente un exemple de règle Snort, ici construite pour détecter une

```
# alert tcp $EXTERNAL_NET any -> $HOME_NET 143 (

    msg:"PROTOCOL-IMAP login brute force attempt";
    flow:to_server,established,no_stream;
    content:"LOGIN";
    fast_pattern:only;
    detection_filter:track by_dst, count 30, seconds 30;
    metadata:ruleset community, service imap;
    classtype:suspicious-login;
    sid:2273; rev:11;
)
```

FIGURE 2.1 – Un exemple de règle Snort

attaque par force brute sur le protocole IMAP. Comme nous pouvons le constater, la règle cherche les flux TCP à destination du port 143 (IMAP) et contenant la charge utile "LOGIN". Une alerte est levée si plus de 30 connexions sont détectées en 30 secondes. Cette règle est donc très spécifique. Elle nécessite la sélection méticuleuse de seuils de détection, un travail qui doit être réalisé par un expert pour chaque anomalie à détecter. Snort a une communauté active, mettant à la disponibilité de chacun un grand nombre de règles préfabriquées permettant de détecter les attaques connues.

Bro propose une approche plus flexible que Snort. Il utilise un ensemble de modules construits à l'aide d'un langage de programmation dédié (des scripts Bro). Chaque module peut alors détecter une à plusieurs attaques différentes avec des mécanismes de détection différents. Bro contient par défaut un module reposant sur des signatures, compatible avec les règles Snort.

À l'image de ce que propose Bro, certains travaux ont proposé des mécanismes de détection se voulant plus flexibles. Par exemple, Scheirer et Chua [7] proposent une détection utilisant l'empreinte de Rabin. Cet outil permet la création de signatures insensibles à des décalages dans la charge utile des paquets.

Les approches que nous avons présentées jusqu'à présent cherchent à caractériser les anomalies connues, et non le trafic normal. Certaines taxonomies séparent les détecteurs en deux catégories : les *détecteurs d'abus (misuse detectors)* et les *détecteurs d'anomalies (anomaly detectors)*. Les détecteurs d'abus cherchent à repérer dans le trafic des motifs correspondant à une anomalie particulière. Ils caractérisent les propriétés des attaques ou anomalies connues. À l'inverse, les détecteurs d'anomalies caractérisent le comportement normal, ou du moins attendu, du trafic. Ils détectent alors des déviations de ce comportement attendu.

Il existe très peu de systèmes experts caractérisant le trafic normal plutôt que les anomalies, et la plupart d'entre eux sont assez anciens. Certaines approches ont par exemple proposé de modéliser le fonctionnement attendu de protocoles. On parle alors de détection comportementale, ou "reposant sur les spécifications" (*specification-based*). Une telle approche a été proposée par Yoo [8], l'auteur présente une spécification du protocole TCP. Cette spécification est ensuite compilée afin d'opérer une inspection du trafic, détectant d'éventuelles anomalies dans les entêtes des paquets. Sekar et al. [9] proposent une approche combinant spécification et détection statistique. Plusieurs protocoles sont modélisés à l'aide d'une machine à états finis étendues par un mécanisme d'analyse statistique. La fréquence de déclenchement de certaines transitions ou les valeurs les plus communes de certains champs sont ainsi analysées, permettant de détecter une anomalie en cas de déviation. Ces approches, bien qu'intéressantes, sont assez difficile à mettre en place. Conserver le statut de l'ensemble des connexions traversant le réseau nécessite des ressources de calcul importantes, et n'est plus véritablement envisageable sur les débits actuels. Qui plus est, contrairement aux approches de détection d'abus, ces approches sont incapables de fournir des informations précises sur l'anomalie détectée. C'est pourquoi presque tous les systèmes experts sont des détecteurs d'abus.

Les approches par signatures ont certains avantages. Tout d'abord, le grand nombre de règles utilisées permet de distinguer les attaques les unes des autres. Chaque règle peut aussi être annotée d'un certain nombre d'informations quant à l'anomalie visée, ce qui simplifie le diagnostic. Les conditions nécessaires à la levée d'une alerte sont alors explicites, ce qui facilite significativement le travail de l'administrateur. De plus, lorsque les anomalies à détecter sont peu nombreuses, les outils de détection peuvent être optimisés en conséquence. Cela peut limiter les ressources de calcul utilisées et améliorer la détection. Notons qu'il est néanmoins assez rare d'utiliser une solution reposant sur une base de signatures pour ne détecter qu'un nombre très restreint d'anomalies, la multiplication des signatures actives peut rapidement nécessiter des ressources de calcul importantes.

Mais là n'est pas leur seul désavantage. Tout d'abord, construire les règles de détection est un travail complexe, nécessitant le travail d'un expert en sécurité. Les signatures, et particulièrement les seuils de détection associés, doivent tout d'abord être adaptées au trafic surveillé. Lorsque le trafic évolue (à l'ajout d'un nouveau service par exemple), une mise à jour des signatures est alors nécessaire. Cela nécessite un travail conséquent d'analyse et de configuration qui doit être répété à intervalles réguliers. Si, comme c'est le cas pour Snort, certaines signatures des attaques peuvent être partagées sans ajustement, ce n'est néanmoins possible que pour des règles précises, dont le critère de détection peut-être estimé comme indépendant du trafic surveillé.

Aussi, pour détecter des anomalies à la couche applicative, de tels algorithmes doivent

parfois analyser la charge utile des paquets. Cela nécessite la reconstruction de certains flux traversant le réseau. On parle alors de *Deep Packet Inspection (DPI)*, ou en français "inspection profonde des paquets". Suivant ce que la signature cherche à vérifier, l'analyse en profondeur d'un paquet peut être complexe, ce qui rend les systèmes experts gourmands en ressources de calcul si, comme évoqué plus tôt, un grand nombre de signatures est nécessaire. Enfin, ces détecteurs sont incapables de détecter des attaques *zero day*, c'est-à-dire n'ayant jamais été repérées auparavant.

Bien que les approches par signatures soient assez peu autonomes, elles restent toujours largement utilisées. En effet, le modèle commercial des entreprises du domaine de la sécurité informatique repose en grande partie sur l'élaboration de règles et contre-mesures pour répondre aux nouvelles menaces se développant continuellement. Le travail de construction de ces règles, réalisé en grande partie manuellement, est coûteux et chronophage, mais il peut être externalisé assez simplement. Au-delà du fait que ce modèle commercial soit un frein à l'adoption de solutions plus autonomes, le fait est que la capacité des approches par signatures à produire des rapports d'anomalie précis favorise leur utilisation encore aujourd'hui.

Afin d'éviter le fastidieux travail de création et de mise à jour des signatures, de nombreux chercheurs ont souhaité profiter des opportunités rendues possible par l'apprentissage automatique. De telles approches simplifient le travail de l'administrateur en étant capables de caractériser, de manière autonome, différentes classes de trafic existantes à partir d'un échantillon de trafic labellisé. Ces approches sont décrites dans la section suivante.

2.2 Approches supervisées et semi-supervisées

Les détecteurs supervisés ont été les premiers détecteurs proposés reposant sur de l'apprentissage automatique. Le fonctionnement de ces détecteurs nécessite deux phases. La première phase, dite d'entraînement, consiste à paramétrer un modèle du trafic à l'aide d'un jeu de données labellisé. En opération, ce modèle est ensuite utilisé dans la deuxième phase, dite de test, pour identifier chaque classe de trafic.

Les algorithmes semi-supervisés se distinguent des algorithmes supervisés par leur jeu de données d'entraînement. Alors que les algorithmes supervisés nécessitent un jeu de données complètement labellisé, les algorithmes semi-supervisés utilisent un jeu d'entraînement au moins partiellement labellisé. Ce jeu de données peut comprendre uniquement des classes de trafic anormales, uniquement des classes de trafic normales, ou plus rarement un mélange des deux.

Les algorithmes supervisés et semi-supervisés peuvent opérer différents niveaux de classification : les détecteurs à classe unique se contentent de classer les instances comme

anormales ou non, tandis que les détecteurs à classes multiples sont capables de repérer différents types d'anomalies. Alors que les détecteurs à classes multiples fournissent plus d'informations sur les anomalies, ils ont également besoin de labels plus spécifiques dans le jeu de données d'entraînement. Bien entendu, les différents labels présents dans le jeu d'entraînement conditionnent les informations qui pourront être données à l'administrateur lors de la détection d'une anomalie.

Différents outils d'apprentissage automatique ont été utilisés dans le domaine de la détection d'anomalies réseaux. Dans la suite de cette section, nous classons les différents détecteurs proposés dans la littérature en fonction des outils qu'ils utilisent.

2.2.1 Réseaux de neurones artificiels

Plusieurs algorithmes supervisés s'appuient sur des réseaux de neurones artificiels. Un neurone artificiel consiste en une fonction de transfert avec plusieurs entrées pour une seule sortie. Dans un réseau, ces neurones sont connectés entre eux, la sortie d'un neurone pouvant être connectée à un ou plusieurs autres neurones. Cet agencement crée alors un graphe qui peut prendre de nombreuses formes différentes. Dans certains cas, ce graphe peut contenir des cycles, on parle alors de réseaux de neurones récurrents.

Un réseau de neurones très commun est le perceptron multicouche. Le perceptron utilise un type de neurone artificiel dont la fonction de transfert est la suivante :

$$f(X,W) = \begin{cases} 1 & \text{si } \sum_{i=1}^{n} w_i x_i > \theta \\ 0 & \text{sinon} \end{cases}$$

Où $X = \{x_1, x_2, ..., x_n\}$ sont les valeurs en entrée du neurone, $W = \{w_1, w_2, ..., w_n\}$ un ensemble de poids associé à ces entrées et θ un paramètre appelé le biais. La valeur de $f(X,W)$ est celle en sortie du neurone. Dans un perceptron les neurones sont organisés en couches (une ou plusieurs) où chaque couche prend en entrée l'ensemble des sorties de la couche précédente. La figure 2.2 illustre cette organisation.

Pour entraîner un réseau de neurones, on ajuste les poids associés à chaque connexion entre deux neurones. Lors de l'entraînement sur un jeu de données labellisé, on vérifie si la sortie du réseau de neurones (soit les valeurs en sortie de la dernière couche), correspondent aux données en entrée (les valeurs en entrée de la première couche). On calcule alors une erreur qui sera compensée en modifiant les poids des différentes connexions dans le réseau. Dans la majorité des cas, cet ajustement des poids est réalisé par la méthode de rétropropagation du gradient. Elle consiste à corriger l'erreur observée selon l'importance des poids qui ont justement participé à la réalisation de cette erreurs.

Dans certains cas, les réseaux de neurones sont utilisés comme outils de classification.

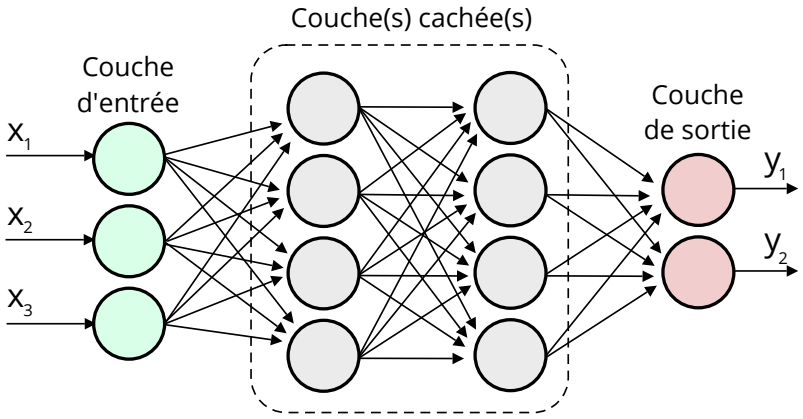


FIGURE 2.2 – Un exemple de réseau de neurones

La première couche prend alors en entrée des caractéristiques extraites du trafic (x_1, x_2 et x_3 sur la figure). La classification est ensuite réalisée grâce aux valeurs en sortie de la dernière couche : chaque neurone de la dernière couche correspond à une classe, le score produit par le neurone s'assimile alors à la probabilité que le trafic testé appartienne à la classe représentée par le neurone (y_1 et y_2 sur la figure).

Poojitha et al. [10] présentent un détecteur utilisant un réseau de neurones, entraîné à l'aide de l'algorithme de la rétropropagation du gradient sur le jeu de données KDD'99. Cette approche est simple, mais les résultats produits sont acceptables, avec un taux de détection de 94.93% et un taux de faux positifs de 0.2%. Néanmoins, l'utilisation d'un jeu de données datant de 1999 rend ces résultats obsolètes. D'autres considérations sur ce jeu de données sont évoquées dans le chapitre 4.

Hawkins et al. [11] proposent une approche semi-supervisée pour la détection d'anomalies. Ils utilisent un *Replicator Neural Network* pour modéliser le trafic normal. Ce type de réseau comporte autant d'entrées que de sorties, leur entraînement vise à produire un réseau produisant en couche de sortie un résultat identique aux données d'entrées. En opération, une anomalie est alors détectée lorsque la sortie du réseau se distingue significativement des données d'entrées. Ce réseau doit dont être entraîné sur du trafic normal, ne contenant pas d'anomalies.

Une autre approche semi-supervisée est proposée par Sun et al. [12] utilisant un réseau de neurones à ondelettes *(wavelet neural network)*, dont la fonction d'activation des neurones est construite à partir d'une ondelette. Utilisée en traitement du signal, les ondelettes permettent, entre autre, une analyse des signaux à la fois en fréquence et en temps. Les auteurs comparent les résultats de leur approche avec un réseau de neurones standard et obtiennent une meilleure détection en un temps d'entraînement plus long.

Lee et al. [13] proposent un détecteur utilisant une hiérarchie de réseaux de neurones

opérant sur des mesures statistiques. La couche la plus basse de la hiérarchie utilise des données très spécifiques à un service alors que la couche la plus haute déduit un statut général du réseau. Dans ces travaux, les auteurs se focalisent sur la détection de nouvelles attaques, les réseaux de neurones sont donc entraînés de façon semi-supervisées sur du trafic normal uniquement. Sur des traces générées par simulation, l'approche a montré sa capacité à détecter des attaques par *SYN flood* et différents types de scans. Les différents types d'anomalies sont finalement caractérisés à l'aide d'un réseau de neurones spécifique appelé "carte auto-organisatrice".

Tong et al. [14] utilisent un réseau de neurones hybride combinant un réseau RBF *(Radial Basis Function)* et un réseau Elman. Le réseau RBF, utilisant une fonction d'activation radiale, permet de capturer des répartitions complexes (non linéaires) dans l'espace d'entrée. Un réseau Elman est un type de réseau récurrents dont certains neurones sont utilisés pour stocker des valeurs. Ce type de réseau peut alors être utilisé pour de la prédiction de séquences. En combinant ces deux approches. Les auteurs obtiennent une bonne détection générant peu de faux positifs sur le jeu de données DARPA/Lincoln laboratory de 1999, notamment pour des attaques par déni de service.

Plus récemment, Aygun et Yavuz [15] ont utilisé un type de réseau de neurones appelé autoencodeur. Ce réseau multicouche a autant de neurones en première couche qu'en dernière, son objectif étant de reproduire en sortie les données d'entrées. Les couches intermédiaires contiennent moins de neurones, ce qui force une simplification des caractéristiques observées. Ici, il est entraîné de façon non-supervisée seulement sur des traces normales extraite de NSL-KDD, une version améliorée du jeu de donnée KDD'99. Le détecteur opère une bonne détection, s'approchant des performances de détecteurs utilisant des réseaux de neurones hybrides.

Les réseaux de neurones sont un outil visiblement efficace. Les détecteurs présentés ici montrent de bons résultats, détectant de nombreuses anomalies et levant peu de faux positifs. Différents types de réseaux existent répondant à différents objectifs, ce qui démontre la flexibilité de cette approche. Néanmoins, les réseaux de neurones nécessitent généralement une phase d'apprentissage plutôt longue, ce qui rend leur utilisation fastidieuse. Aussi, peu d'approches abordent le problème de la visualisation des données produites. Si les résultats que les réseaux de neurones produisent sont bons, ils restent difficiles à interpréter. Cela peut rendre difficile l'adoption de détecteurs reposant exclusivement sur des réseaux de neurones.

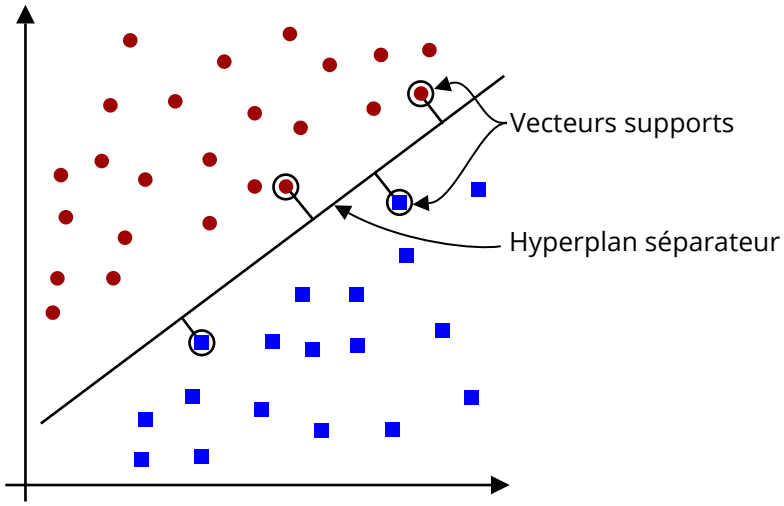


FIGURE 2.3 – Un machine à vecteur de support (SVM)

2.2.2 SVM

D'autres classifieurs supervisés utilisent une machine à vecteurs de support ou SVM (Support Vector Machine *ou* Séparateur à Vaste Marge). Cette approche cherche à créer un ou plusieurs séparateurs linéaires permettant d'isoler chaque classe de trafic. Ces hyperplans séparateurs sont entraînés sur des données labellisées de manière à ce que la marge entre les différentes classes soit la plus grande possible. Maximiser la marge à l'entraînement, revient à minimiser le risque d'erreur de classification lorsque le séparateur est utilisé sur des données non labellisées. Une illustration de ce type de séparateur est proposée sur la figure 2.3. Nous constatons que la définition du plan entre la classes des cercles et celle des carrés ne repose que sur les points les plus proches de la ligne de séparation. Ces points sont appelés vecteurs supports.

Comme tous les problèmes ne sont pas linéairement séparables, plusieurs techniques ont alors été proposées pour améliorer les SVM. La technique de la marge souple (*Soft margin* [16]) permet de séparer des ensembles pas tout à fait linéairement séparables. La méthode accepte ainsi quelques erreurs de classification pendant l'entraînement. L'astuce du noyau (*Kernel trick* [17]) permet quant à elle de transformer des données d'entrée afin de les amener dans un espace où elles sont linéairement séparables.

Dans [18] par exemple, Perdisci et al. proposent un détecteur opérant sur la charge utile des paquets. Ils combinent plusieurs classifieurs SVM mono-classe (n'isolant qu'une seule classe de trafic). Les caractéristiques extraites du trafic sont produites à l'aide d'une fenêtre glissante sur le contenu de la charge utile. En variant l'espace entre les octets analysés, l'algorithme produit des données dans plusieurs espaces différents, qui sont ensuite analysés

par les différents séparateurs à vaste marge. L'utilisation de différents classifieurs permet d'améliorer la pertinence des résultats, tout en rendant difficile l'utilisation de techniques d'évasion. Ces travaux sont étendus dans [19] où ce détecteur, finalement appelé McPAD, montre sa capacité à détecter des attaques shell-code. Cette approche est intéressante pour repérer des motifs très précis dans le contenu de la charge utile. Ceci étant dit, au niveau réseau, un traitement de la charge utile est difficilement envisageable. Néanmoins, l'apprentissage automatique proposé ici permet la création de règles plus souples et robustes à des décalages des octets anormaux dans la charge utile.

Dans [20], Khan et al. proposent une approche souhaitant résoudre le temps d'entraînement relativement long nécessaire aux SVM, notamment sur des jeux de données à haute dimensionalité. Les auteurs utilisent alors du clustering hiérarchique (évoqué dans la section 2.3) afin de fournir des points plus optimaux à un SVM. L'approche montre sa capacité à réduire le temps d'entraînement nécessaire à l'algorithme. L'outil de détection est évalué sur le jeu de données DARPA de 1998. Il s'avère donner des résultats de détection acceptables, avec $69,8\%$ de décisions exactes. Les résultats sont comparés avec une approche utilisant seulement un SVM, cette dernière obtenant $57,6\%$ de décisions exactes. Aussi l'approche proposée réduit de 17 h à 13 h le temps d'entraînement de l'algorithme.

Entraîner une machine à vecteur de support est long et fastidieux, mais cet inconvénient est compensé par une phase de détection très rapide, linéaire en fonction du nombre d'éléments et du nombre de dimensions. Généralement, les SVM génèrent des résultats de classification aussi bons que ceux obtenus avec des réseaux neuronaux, voire meilleurs.

2.2.3 Règles d'association

Plusieurs classifieurs cherchent à produire à partir des caractéristiques du trafic un ensemble de règles simples permettant de distinguer chaque classe existante. Ils recherchent alors dans les propriétés du trafic des motifs récurrents, des liens d'associations entre différentes propriétés du trafic. Ces motifs sont ensuite utilisés pour inférer un ensemble de règles régissant les propriétés du trafic. On parle en anglais de *frequent itemsets mining*.

Une règle consiste généralement à une assertion telle que $X \implies Y$, où X et Y sont deux ensembles d'attributs du trafic. Par exemple, en supposant qu'un serveur *host* héberge seulement un site Internet, nous pourrions imaginer une règle détectée sur les flux, telle que *(Destination=host)* $\implies$ *(Protocole=http)*. L'ensemble des règles créées peut éventuellement être retravaillé par la suite, si certaines règles ne sont pas pertinentes. On évalue notamment le *support* et *l'indice de confiance* de chaque règle. Le support correspond ici à la proportion d'évènements contenant X, soit ici la proportion de connexions vers *host* dans notre exemple. L'indice de confiance est alors la proportion des éléments

contenant Y parmi ceux contenant X, soit la proportion des connexions vers *host* utilisant effectivement le protocole *http*. Plus ces indices sont hauts, plus la règle est évaluée comme étant pertinente.

La plupart des outils utilisant de la recherche d'association sont anciens. Par exemple, Audit Data Analysis and Mining (ADAM) [21] et LEarning Rules for Anomaly Detection (LERAD) [22] sont deux détecteurs, proposés en 2001 et 2003, utilisant des règles d'association sur les propriétés des flux TCP. ADAM propose une approche semi-supervisée, caractérisant le trafic normal et capable de découvrir des anomalies inconnues jusqu'à présent. Il utilise une fenêtre glissante pour analyser le trafic, extrayant les règles des flux repérés dans la fenêtre. LERAD opère en deux phases. L'une extrait un sous-ensemble aléatoire du jeu d'entraînement et génère un ensemble de règles à partir de celui-ci. Les règles sont ensuite ajustées en repérant toutes les valeurs possibles pour un même antécédent (X dans le formalisme évoqué plus précédemment). Les règles n'ayant pas un indice de support ou un indice de confiance suffisant sont enfin retirées lors d'une évaluation sur une autre sous-partie du jeu de données. Notons qu'ADAM, contrairement à LERAD utilise des caractéristiques de niveau réseau (adresses et ports), plutôt que des données du niveau applicatif. Les deux détecteurs sont évalués sur le jeu de données DARPA/Lincoln Laboratory, et produisent des résultats moyens à acceptables. ADAM et LERAD détectent respectivement 41% et 64% des attaques en levant peu de faux positifs.

Des *règles d'association floues* sont utilisées dans une approche relativement plus récente [23]. Les règles d'association floues combinent la construction de règles d'association classiques avec la théorie des sous-ensembles flous. Elle introduit une notion de probabilité dans l'élaboration des règles. Les auteurs proposent d'introduire de l'apprentissage incrémental pour construire les règles, permettant une amélioration de l'apprentissage, en temps réel, lorsque le détecteur est en opération. L'outil est utilisé avec succès pour détecter des attaques DoS, tout en étant capable de détecter les anomalies en moins de deux secondes.

L'un des avantages les plus importants des règles d'association reste leur caractère explicite. En effet, la création de règles simples permet à l'administrateur de comprendre rapidement les raisons de la levée d'une alerte. Cela permet un diagnostic simplifié, permettant d'identifier certains faux positifs. Néanmoins, ces approches fonctionnent bien sur des données en entrées où les relations d'attributs à attributs restent simples. Elles fonctionnent à priori à l'échelle du flux, mais peuvent s'avérer peu robustes lorsque des changements globaux ont lieu.

2.2.4 Méthodes ensemblistes

Les détecteurs ensemblistes combinent les résultats de plusieurs méthodes de détection différentes afin d'améliorer la détection. Prendre en compte les résultats de plusieurs détecteurs peut avoir l'un des deux objectifs suivants :

- La détection d'un plus grand nombre d'attaques, si les approches sont conçues pour détecter des anomalies différentes.
- Une détection plus robuste, combinant les résultats de plusieurs détecteurs pour prendre la décision la plus avisée.

La principale différence entre deux approches ensemblistes tient dans l'algorithme utilisé pour combiner les résultats des différents détecteurs. Si l'on veut détecter le plus d'attaques possible, une approche peut consister à lever une alerte si au moins l'un des détecteurs lève une alerte. Sinon, certaines propositions utilisent un vote majoritaire entre les différents détecteurs. D'autres approches assignent enfin un indice de confiance à chaque détecteur, selon les résultats qu'il a produits (que ce soit en opération ou sur un jeu de données d'entraînement).

Dans [24], les auteurs présentent une approche ensembliste pour la détection des attaques par déni de service distribuées. Le détecteur, nommé RBPBoost, utilise les résultats produits par plusieurs classifieurs utilisant des réseaux de neurones de type RBP (Resilient Back Propagation). L'approche les combine en utilisant la stratégie de minimisation des coûts de Neyman-Pearson. Cette stratégie consiste à adapter les seuils de détection de chaque modèle en fonction d'un taux de faux positifs acceptable donné. Le taux de classification correcte varie entre 97% et 99% suivant le jeu de données utilisé dans l'évaluation (parmi CONFICKER, KDD'99, DARPA 1999 et 2000).

Un classifieur ensembliste, appelé *Super Learner*, est présenté par [25]. Il utilise une technique de validation croisée pour choisir quelles méthodes opèrent une meilleure détection. Un large panel de différentes approches est proposé, utilisant différents classifieurs reposant sur un réseau de neurones, des SVM ou encore des arbres de décisions. Le détecteur ensembliste s'avère meilleur que chacun des détecteurs pris un à un. Les auteurs relativisent néanmoins la pertinence d'une telle approche lorsque les classifieurs utilisés au départ sont déjà performants, le coût de calcul supplémentaire n'étant dans ce cas que peu justifié.

L'apprentissage ensembliste réduit le risque de construire un modèle du trafic incorrect. Suivant le trafic surveillé, cela peut permettre une meilleure détection. Néanmoins, comme évoqué dans [25], la pertinence de telles approches est limitée lorsque l'un des détecteurs

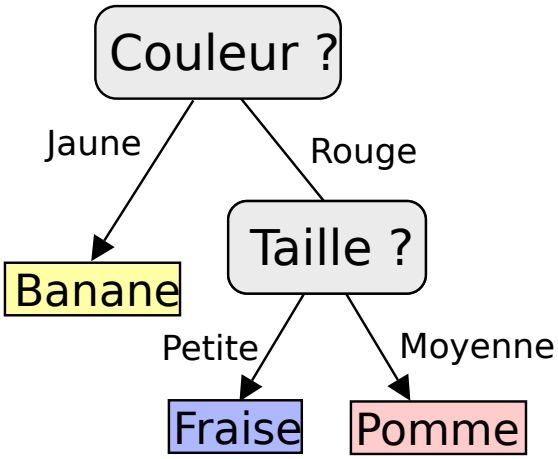


FIGURE 2.4 – Un arbre de décision

utilisé produit déjà de bons résultats de détection. Prendre en compte les résultats d'autres détecteurs moins performants se traduit par une détection plus complexe, pouvant conduire à la levée de faux positifs. Utiliser plusieurs algorithmes différents nécessite aussi plus de ressources de calcul et complexifie à la fois la mise en place et la maintenance de la solution, tout en complexifiant l'analyse des résultats de détection.

2.2.5 Autres méthodes de classification

Nous avons présenté jusqu'à présent les principaux algorithmes d'apprentissage supervisés ou semi-supervisés utilisés pour la détection d'anomalies. Bien sûr, d'autres méthodes d'apprentissage supervisé ou semi-supervisé ont été utilisées, mais dans une moindre mesure.

Les arbres de décision en sont un exemple. Un arbre de décision consiste en une suite de prédicats organisés sous la forme d'un arbre. Les feuilles de l'arbre sont alors des prédictions. La figure 2.4 illustre ce mécanisme pour de la classification de fruits. Les arbres de décision peuvent être créés à partir d'un jeu d'entraînement. L'algorithme va alors chercher, à chaque étape, l'attribut le plus pertinent pour séparer les différentes classes contenues dans le jeu de données. Cet attribut sera alors utilisé pour l'un des prédicats. Plus on entraîne l'arbre, plus on ajoute de prédicats, ce qui rend l'arbre plus profond.

Par exemple, dans [26], l'approche présentée utilise un algorithme de clustering permettant d'améliorer la performance de classification d'arbres de décision. Plusieurs arbres sont alors entraînés, et leur performances de détection sont évaluées pour chaque région de l'espace d'entrée (chacune définie par un cluster). La détection finale est opérée de manière ensembliste, en combinant les résultats de chaque classifieur, mais pondérés par

leur performance dans la région de l'instance à classer. Évaluée sur KDD'99, l'approche produit une classification correcte à 90%, meilleure que d'autres approches ensemblistes aussi évaluées. Dans [27], une méthode de sélection des attributs est évaluée. Différents algorithmes utilisant des arbres de décision sont alors utilisés sur KDD'99, obtenant une classification correcte à 93%. Si les arbres de décision peuvent obtenir de bon résultats de classification, leur entraînement est parfois fastidieux. Il est souvent nécessaire d'entraîner un grand nombre d'arbres avant d'en trouver un efficace. Néanmoins, la phase de test d'un tel algorithme reste souvent rapide, pourvu que l'arbre soit peu profond. Aussi, une telle approche simplifie le diagnostic d'une alerte puisque la liste des décisions qui ont mené à la décision finale est contenue dans la structure de l'arbre.

La théorie des ensembles bruts (*Rough Set Theory*) est une sous-partie de la théorie des ensembles flous évoquée plus haut. Cette théorie raisonne sur des ensembles imprécis, définis par des bornes supérieure et inférieure. En effet, un ensemble brut X est défini par les ensembles d'éléments R^+ et R^-, constitués respectivement des éléments appartenant *certainement* et *possiblement* à X.

La théorie des ensembles bruts a été, par exemple, utilisée dans [28] afin de réduire le nombre d'attributs du trafic à considérer dans le jeu de données KDD'99 pour y détecter les attaques. La classification finale est réalisée par un SVM. L'approche permet une amélioration du taux d'instances bien classifiées de 86% à 90%, par rapport à un SVM appliqué sur les données brutes.

Les réseaux bayésiens sont un autre outil pouvant servir pour de l'apprentissage supervisé. Un réseau bayésien est un graphe acyclique illustrant des relations de causalité entre différentes variables d'intérêt. À chaque relation est assigné un poids, illustrant à quel point la relation de causalité est forte. Dans [29], une version étendue de ce modèle, appelée réseau bayésien en temps continu (*Continuous Time Bayesian networks*), est utilisée. Cette extension permet de caractériser des attributs évoluant au fil du temps, en utilisant des réseaux bayésiens pour décrire les états d'un processus de Markov. L'approche est évaluée sur les jeux de données MAWI [30] et LBNL[31]. Elle montre de bon résultats de détection comparés à d'autres approches plus anciennes, surtout sur le jeu de données MAWI.

2.2.6 Considérations relatives aux détecteurs supervisés

Les approches supervisées et semi-supervisées proposent généralement une détection nécessitant moins de ressources de calcul que les systèmes experts. En effet, les modèles utilisés par ce type d'approches sont plus génériques, ils ne requièrent pas un modèle de chaque attaque existante, mais vont plutôt utiliser des classes de trafic plus génériques. Habituellement, ces approches divisent le trafic en deux classes : malveillant et normal.

Par conséquent, ces approches passent généralement mieux à l'échelle. Néanmoins, selon la diversité des labels du jeu de données d'entraînement, les détecteurs supervisés peuvent fournir des labels pertinents associés aux anomalies détectées permettant de simplifier l'étape de diagnostic. Notons que les détecteurs semi-supervisés utilisant seulement du trafic sans attaques dans le jeu de données d'entraînement sont capables de détecter des anomalies inconnues.

Le principal problème de la détection supervisée reste justement la création de ce jeu de données correctement labellisé. En effet, construire un jeu de données bien labellisé et représentatif du trafic est ardu. Pour ce faire, plusieurs approches sont possibles :

- Capturer des traces réelles puis assigner manuellement un label à chaque instance du trafic. Cette méthode est en général difficile et chronophage, à moins de supposer que le trafic capturé ne contienne pas d'anomalie, ce qui peut être une hypothèse audacieuse. Aussi, cette méthode de construction ne permet pas de contrôler le contenu du jeu de données, ce qui peut rendre difficile la caractérisation d'évènements rares mais intéressants (typiquement une attaque).

- Forger des traces en générant plusieurs classes de trafic connues (normal ou anormal). Cette méthode requiert des générateurs réalistes, qui sont assez difficiles à construire.

- Utiliser une combinaison des deux, généralement en capturant du trafic supposé sans anomalie, puis en insérant des attaques générées dans le jeu de données.

Quoiqu'il en soit, construire un tel jeu de données nécessite un travail conséquent de la part d'un expert en sécurité. Aussi, ce travail doit être effectué à nouveau lorsque le trafic à surveiller évolue ou que de nouvelles anomalies sont découvertes. Ceci rend leur usage dans un contexte industriel difficile. Même si construire un tel jeu de données peut être plus simple que de construire et maintenir à jour un ensemble de signatures, un tel investissement est trop coûteux pour les acteurs industriels.

Les difficultés de construction d'un jeu d'entraînement et sa maintenance rendent les approches supervisées difficilement exploitables dans un contexte réaliste. Néanmoins, de telles solutions ont permis des avancées significatives vers des méthodes de détection plus autonomes, notamment parce qu'un grand nombre de techniques d'apprentissage supervisées ont été adaptées pour opérer de manière non supervisée.

2.3 Approches non-supervisées

Comme nous l'avons vu, à la fois les systèmes experts et les approches supervisées ou semi-surpervisées reposent sur un ensemble de données complexes à construire et à maintenir. Elles sont donc coûteuses pour une utilisation dans un contexte industriel. Pour cette raison, certains chercheurs ont proposé la création de détecteurs non-supervisés.

Les approches non-supervisées ne nécessitent ni signatures, ni jeu de données labellisé. Elles reposent à la place sur la supposition que l'essentiel du trafic reste du trafic normal, alors que les anomalies sont des évènements statistiquement rares et isolés. Bien qu'ils puissent utiliser un jeu de données séparé, la plupart des détecteurs non-supervisés opèrent directement sur le trafic en production. Ils construisent alors une structure de données représentative du trafic, regroupant les instances du trafic similaires et isolant celles aux caractéristiques inhabituelles. Lorsqu'un évènement dévie significativement du comportement habituel du trafic, une alerte est alors levée. Le regroupement d'instances similaires permet l'identification de frontières entre les différents groupes d'instances. Ceci simplifie, à postériori, la création de règles permettant de délimiter chaque classe de trafic. Ces approches sont parfois dites proactives, puisqu'elles sont capables de détecter certaines menaces avant même qu'elles soient connues.

Les approches non-supervisées ont plusieurs avantages, tels que leur capacité à détecter de nouvelles anomalies ainsi qu'à s'adapter à différents types de trafic. Comme elles ne nécessitent pas la construction et la mise à jour d'une base de signatures ou d'un jeu de données labellisé, ces approches sont plus autonomes. Complétées avec des mécanismes pertinents, elles peuvent aussi prendre une décision automatique quant aux anomalies détectées. Une approche typique consiste alors à évaluer un coefficient de dangerosité relatif à l'impact qu'a l'anomalie sur le trafic, et à bloquer le trafic si ce risque est trop élevé.

Ceci étant dit, le fait que les approches purement non-supervisées n'utilisent aucune base de données caractérisant les anomalies connues est aussi le principal problème de ces approches. En effet, de tels détecteurs doivent combler ce manque en fournissant suffisamment d'informations à l'administrateur réseau pour qu'il puisse prendre une décision avisée. Aussi, comme cette information se doit aussi d'être facilement compréhensible, les outils de diagnostic proposés sont à considérer comme faisant partie intégrante de la solution proposée par un détecteur non-supervisé. Dans certains cas, un outil de visualisation performant peut s'avérer être très intéressant.

Dans cette section, nous présentons en premier lieu une ensemble d'approches purement non-supervisées. La section 2.4 présente des approches plus hybrides, introduisant de la connaissance dans la détection.

2.3.1 Détection de changement

Les approches reposant sur de la détection de changement considèrent qu'une anomalie induit un changement significatif dans les probabilités de distribution des caractéristiques du trafic. Elles prédisent donc un ou plusieurs paramètres du trafic, et détectent d'éventuelles déviations entre la prédiction et les observations.

Yu [32] utilise l'algorithme CUmulative SUM pour détecter des anomalies dans du trafic symétrique (le nombre de paquets par seconde entrant est similaire au nombre de paquets par seconde sortant). Cet algorithme est une technique d'analyse séquentielle, impliquant le calcul d'une somme cumulée. Supposons un échantillon x_n auquel est assigné un poids w_n, la somme cumulée est calculée de la façon suivante :

$$\begin{aligned} S_0 &= 0 \\ S_{n+1} &= \max(0, S_n + x_n - w_n) \end{aligned} \tag{2.1}$$

Cette somme, lorsqu'elle atteint une valeur de seuil donnée, peut traduire un changement brusque dans les échantillons observés. Pour s'adapter à n'importe quelle bande passante, l'approche présentée utilise un seuil adaptatif, calculé de façon dynamique à partir du trafic considéré comme normal. Un trafic déséquilibré est alors considéré comme anormal. L'auteur parvient à une détection opérant en temps réel et détectant les anomalies rapidement, tout en ne nécessitant que peu d'efforts de configuration. Néanmoins cette approche se focalise seulement sur la détection d'attaques par *SYN flooding*.

Parmi quatre autres approches proposées dans l'article, [33] présente une approche utilisant CUSUM. Cette approche utilise une matrice de trafic, construite à partir du volume de trafic pour chaque couple source/destination sur un intervalle de temps donné (ici 5 minutes). Pour chaque intervalle, la matrice est comparée avec une estimation de ce que devrait être la matrice de trafic. L'erreur entre l'estimation et les valeurs observée est alors quantifiée. Les auteurs présentent alors une approche appelée *Generalized Likelihood Ratio Test*, consistant à appliquer CUSUM sur un calcul du maximum de vraisemblance (cette valeur est une estimation de l'erreur moyenne observée sur une fenêtre glissante). Cette technique peut néanmoins introduire un délai dans la détection de changement. L'approche est notamment évaluée sur le jeu de donnée Abilene, contenant du trafic échantillonné capturé sur 11 points de présence. Malheureusement l'approche en question s'avère être moins performante qu'une approche triviale, utilisant un calcul de la variance.

Comme nous pouvons le constater CUSUM est un outil simple, mais efficace seulement dans certains cas bien précis. Néanmoins il n'est pas le seul outil utilisé pour la détection de changement. Par exemple, le calcul de l'entropie a été largement utilisée comme outil de

détection de comportements inhabituels. L'entropie est en effet une mesure de la prédictibilité d'un valeur. C'est donc un outil approprié pour détecter des changements inattendus dans les propriétés du trafic.

Lakhina et al. [34] construisent une matrice contenant l'entropie calculée pour un ensemble de caractéristiques globales du trafic. Ils appliquent ensuite la *multiway subspace method*. Cette méthode utilise PCA pour séparer l'espace de données en un sous-espace normal et un autre dit résiduel. Une anomalie est détectée lorsque l'entropie calculée pour le sous-espace résiduel augmente. Dans un second temps, les auteurs utilisent l'algorithme de clustering de *k-means* combiné à du clustering hiérarchique pour grouper les anomalies similaires. Cette étape de clustering regroupe naturellement les anomalies ayant des causes similaires, ce qui simplifie le travail de l'administrateur. Bien que l'approche soit conçue pour opérer en ligne, notons que les auteurs n'abordent pas le problème formellement. Il est donc difficile d'évaluer sa capacité à opérer en temps réel. *Defeat*, présenté par [35], étend cet algorithme en agrégeant les flux IP de façons aléatoire. Cela permet une détection plus exacte et aide à l'identification des flux responsables de l'anomalie.

Dans [36], Celenk et al. utilisent la moyenne de l'entropie et la variance de l'entropie calculée sur un fenêtre temporelle. La détection est réalisée à partir d'une version modifiée du *Discriminant Linéaire de Fisher*, un outil normalement utilisé pour définir un hyperplan séparateur entre plusieurs classes de trafic. Cette version modifiée permet ici d'extraire une mesure de la dispersion des données d'entrée, appelée indice de performance du FLD (Fisher Linear Discriminant) modifié. Cet indice, lorsque dépassant un seuil donné, cause la levée d'un alerte. L'outil de visualisation proposé met en relation cet indice avec l'entropie calculée, sa moyenne et sa variance. Néanmoins, il n'aide pas vraiment à caractériser visuellement le trafic anormal. Le détecteur atteint un taux de détection allant jusqu'à 70% des attaques sur du trafic capturé sur le réseau de l'université d'Ohio.

Tellenbash et al. [37] proposent le *Téléscope entropie (Entropy telescope)*. À partir d'une mesure de l'entropie de Tsallis, leur approche divise l'activité du réseau en différentes régions. Les auteurs appliquent ensuite différents algorithmes sur les mesures extraites (Filtre de Kalman, PCA et KLE) pour détecter les anomalies :

Le filtre de Kalman est un outil d'estimation. Il utilise un modèle du trafic, corrigé de manière autonome lorsque les différentes mesures passent par le filtre, en supposant que le bruit présent dans les mesures effectuées est gaussien. Aussi, il produit, à chaque itération, une estimation de l'erreur possible conjointement à l'estimation elle-même.

PCA (Principal Component Analysis) est un outil permettant de transformer des données corrélées entre elles en de nouvelles données décorrélées. Cette technique utilise une matrice de covariance comme outil d'analyse, qui est alors utilisée pour

construire une nouvelle base pour exprimer les données. Les vecteurs de cette nouvelles base sont classés en fonction de leur apport d'information à la mesure. En ne conservant que les premiers vecteurs, on peut alors décrire les données avec moins de dimensions, sans une perte d'information significative

KLE (Karhunen–Loeve Expansion) est une extension de PCA permettant de prendre en compte des corrélations temporelles.

Pour le filtre de Kalman, c'est l'erreur estimée par le filtre qui est utilisée pour détecter les anomalies. PCA et KLE sont ici utilisés pour créer un modèle simplifié du trafic, et c'est la différence observée entre la prédiction du modèle et la mesure qui sert d'outil de détection. Finalement, les anomalies découvertes sont classées par un SVM. On notera que si cet algorithme nécessite un jeu de données d'entraînement séparé, il a l'avantage de réduire la complexité de la phase de détection. Il est aussi capable de catégoriser les anomalies, ce qui est intéressant. L'algorithme s'est avéré efficace en conditions réelles, pouvant produire une classification correcte à un taux d'environ 80%.

Lors de travaux plus récents, Callegari et al. [38] présentent un nouveau détecteur utilisant différents types d'entropies. Se voulant robuste aux techniques d'évasion, l'algorithme utilise des *reversible sketches*. Les *sketches* sont le résultat d'une projection aléatoire des données sur un espace plus réduit. Les données sont alors compressées, mais une partie de l'information initiale peut alors être perdue. Les *reversible sketches* résolvent ce problème en permettant de retrouver, au moins en partie, les données initiales. Dans le cas présent, on s'intéresse surtout à retrouver les adresses IP et ports incriminés dans une anomalie détectée. Une mesure de l'entropie est finalement utilisée pour détecter les anomalies. L'exactitude de la détection est évaluée sur le jeu de données MAWILab [30], et s'avère limitée. Néanmoins, la scalabilité de cette approche est prometteuse.

Les approches par détection de changement sont efficaces pour détecter des anomalies ayant un impact sur l'ensemble du trafic, telles que les DoS ou DDoS. Par contre, elles sont incapables de produire plus d'informations sur les causes d'une attaque, comme les adresses IP sources incriminées, sans être combinées avec des algorithmes plus élaborés. Néanmoins, elles proposent généralement une détection peu exigeante en ressources de calcul ce qui en fait de bonnes candidates pour être utilisées comme déclencheurs d'analyses plus poussées (comme proposé par [39]).

2.3.2 Approches statistiques

Certaines approches modélisent le trafic en utilisant d'autres outils statistiques. Les estimateurs paramétriques n'étant plus suffisants pour modéliser le trafic actuel, la plupart des algorithmes utilisent des modèles semi-paramétriques ou non-paramétriques. Ces

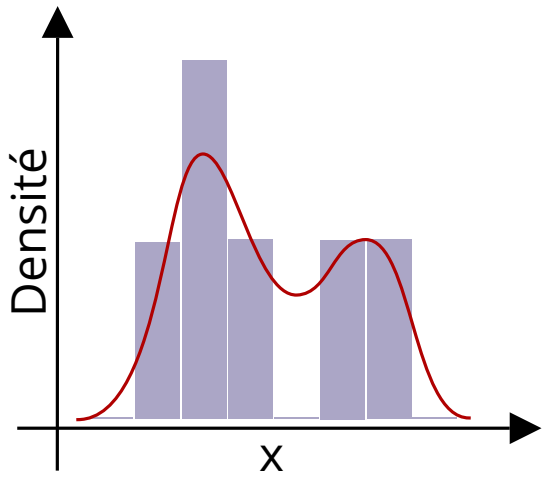


FIGURE 2.5 – Illustration de la relation entre un histogramme et un estimateur par fenêtres de Parzen

modèles ont pour objectif d'estimer la probabilité d'occurrence de chaque évènement, et de lever une alerte lorsqu'un évènement est inhabituel.

Certaines approches non-paramétriques utilisent des histogrammes, comptant et groupant les instances du trafic si elles ont des attributs similaires. PHAD [40] et le détecteur proposé par Krugel et al. [41], utilisent cette approche pour identifier des combinaisons improbables d'attributs du trafic. À partir d'histogrammes calculés sur une fenêtre temporelle, PHAD estime les distributions dans la fenêtre suivante. L'algorithme est évalué sur le jeu de donnée DARPA 99, et détecte environ 40% des attaques en général, mais 60% des DoS et des scans. Ces résultats sont relativement bas, comparés à des approches plus récentes. Dans [41], le type de requête, sa taille et la distribution de la charge utile sont utilisés comme métriques pour la détection d'anomalies. Les histogrammes sont notamment utilisés pour évaluer la fréquence de chaque valeur d'octet possible dans la charge utile. L'approche montre de bon résultats pour des attaques très spécifiques sur des traces DNS, mais sont, à priori, assez peu généralisables.

Yeung et Chow [42] utilisent des *fenêtres de Parzen (Parzen-windows)*. Cet outil d'estimation peut être construit de la même manière qu'un histogramme, en comptant les instances se trouvant dans un même intervalle. Les fenêtres de Parzen ont néanmoins la propriété de lisser les estimations entre chaque intervalle grâce à une technique d'interpolation. La figure 2.5 illustre la construction d'un estimateur par fenêtres de Parzen à partir d'un histogramme. Évaluée sur KDD'99, l'approche obtient de bon résultats avec une classification correcte à environ 90%.

NIDES [43, 44] est un algorithme assez connu, mais ancien, utilisant des histogrammes. Il a été utilisé au sein du framework EMERALD [45]. S'il n'est pas, à l'origine, destiné à

la surveillance des réseaux, NIDES a été adapté pour fonctionner sur des paquets réseau. NIDES est capable de gérer à la fois des attributs continus ou discrets. Il construit des profils à long terme du trafic par port et par adresse. Lorsqu'un changement significatif est détecté, il lève une alerte. Si l'algorithme est évalué dans [44] sur des listes d'appels systèmes, ce n'est malheureusement pas le cas sur des traces réseaux.

Un nouveau détecteur présenté par Chen et al. [46], utilise des *sketches*, pour compresser les données en entrée de l'algorithme. Il utilise alors PCA Multi-Echelle, une extension de PCA utilisant de la décomposition en ondelettes permettant d'éviter le problème de la contamination de l'espace normal. Ce problème apparait lorsque les anomalies contenues dans le trafic ont un impact sur la caractérisation construite du trafic normal. Cette approche est réaliste puisqu'elle accélère le traitement grâce à la construction de *sketches* tout en pouvant retrouver les adresses IP responsables de l'anomalie. Néanmoins, détecter les adresse IP responsables de l'anomalie n'est pas assez pour que l'administrateur puisse prendre une décision avisée. Évalué sur le jeu de données MAWI, l'approche s'avère meilleure qu'une approche utilisant PCA sans les *sketches*. Globalement, l'approche produit de bons résultats, levant peu de faux positifs pour une détection allant de 50% à 85% suivant le taux maximum de faux positifs considéré comme acceptable.

Un autre détecteur utilisant PCA est présenté par [47]. Dans cet article, Liu et al. expliquent l'impossibilité de construire une matrice de corrélation sur du trafic en temps réel. L'approche utilise un réseau de neurones pour construire les vecteurs propres utilisés par PCA à l'aide d'un jeu de données labellisé. Elle n'est donc pas à proprement parler non supervisée, mais les auteurs proposent une extension utilisant une approche par renforcement, où l'algorithme est auto-entraîné sur les anomalies qu'il détecte. Les résultats, évalués sur KDD'99, sont très bons, avec une détection de presque toutes les anomalies (autour de 98%).

D'autres approches statistiques peuvent utiliser les modèles de Markov cachés [48] ou l'analyse d'ondelette [49].

Les approches statistiques offrent généralement plus d'informations sur les anomalies détectées que les approches de détection de changement. En effet, le profil statistique qu'elles produisent permettent de comprendre en quoi l'instance détectée comme anormale est différente du trafic habituel. Néanmoins, ces approches se limitent souvent à considérer les caractéristiques du trafic indépendamment les unes des autres sans corréler les causes d'une anomalie, ce qui peut limiter l'exactitude de la détection et compliquer le diagnostic. Néanmoins, ces approches sont intéressantes puisqu'elles nécessitent généralement peu de ressources de calcul. Elles sont alors un choix avisé lorsque la détection doit opérer sur des équipements aux capacités limitées.

2.3.3 Analyse des plus proches voisins

Les algorithmes reposant sur l'analyse des plus proches voisins comparent chaque instance ajoutée au modèle à un ensemble d'autres instances. Les voisins les plus proches sont alors considérés dans la classification de l'instance. Si l'instance est proche de ses voisins, alors on peut déterminer l'instance testée comme appartenant à la même classe que ses voisins. Si au contraire l'instance est loin de ses voisins, elle est alors considérée comme anormale. L'approche habituelle vise à assigner à l'instance un score dépendant de sa distance au $k^{\text{ième}}$ voisin. Ce score est un estimateur de la densité locale autour de l'instance, qui peut alors être utilisé pour estimer à quel point l'instance est inhabituelle, et donc dangereuse. Cet outil est décrit plus en détail dans la section 3.

Un tel algorithme a été présenté dans [50] : ODIN utilise un algorithme utilisant les plus proches voisins pour détecter des anomalies dans le jeu de données KDD'99 [51]. L'algorithme produit un simple score d'anormalité, mais pas d'autre information sur la source de l'anomalie. L'algorithme est évalué notamment sur KDD'99, mais produit des résultats de détection insuffisants, avec un taux d'erreur de presque 50%.

L'algorithme LOF a été utilisé dans [52]. LOF attribue un score à chaque instance en fonction de sa densité locale, mais comparée à la densité locale autour de ses plus proches voisins. Cette variante permet de compenser le fait que certaines zones de l'espace sont naturellement moins denses que d'autres. Les instances auxquelles est assignée une densité locale sensiblement inférieure à celles de ses voisins est alors détectée comme anormale. Dans [53], les auteurs proposent une version incrémentale de LOF. Les approches incrémentales évitent d'avoir à traiter à nouveau l'ensemble des données lorsqu'une instance est mise à jour, ajoutée ou supprimée du modèle. Lorsque l'analyse est effectuée sur une fenêtre glissante, ces approches sont particulièrement appropriées. Comme elles permettent une détection continue et rapide, ces approches sont intéressantes pour de la détection en temps réel. L'algorithme est succinctement évalué sur le jeu de données DARPA 1998, prouvant sa capacité à détecter une attaque DDoS.

MINDS [54, 55] est un ensemble d'outils d'apprentissage automatique. Il est partie intégrante de Ripper, un détecteur utilisant LOF. Il compare donc les instances du trafic à un échantillon de l'ensemble des données. Ce détecteur a l'avantage de fournir un module capable de résumer les anomalies à l'aide de règles d'association. Lorsqu'une anomalie est détectée, MINDS est capable d'indiquer l'impact de chaque caractéristique du trafic dans le calcul du score d'anormalité, ce qui aide le diagnostic. Les performances de détection sont très bonnes, d'après une évaluation sur des traces capturées sur le réseau de l'Université du Minnesota. Il détecte plus de 90% des anomalies et près de 95% des alertes générées sont de vrais positifs.

RBRP [56] est un détecteur utilisant l'algorithme NL [57]. L'algorithme NL utilise l'algorithme des plus proche voisins pour identifier des évènements anormaux, mais combiné à une approche par bloc minimisant les entrées/sorties. RBRP [56] regroupe en premier lieu les instances similaires à la manière d'un histogramme. Il utilise ensuite l'algorithme NL pour comparer chaque cluster produit pour détecter d'éventuelles déviations. Cet algorithme a été créé pour faire face à la complexité élevée de l'algorithme ORCA [58]. Comme le prouve une évaluation sur KDD'99, RBRP [56] a l'avantage de passer relativement bien à l'échelle quand le nombre de dimensions considérées augmente. Cela permet d'analyser un grand nombre d'attributs du trafic. Néanmoins la qualité de la détection n'est pas étudiée.

Les approches utilisant les plus proches voisins sont efficaces. Bien qu'elles ne fournissent qu'assez peu d'informations sur les anomalies détectées, elles détectent la plupart des anomalies tout en ne nécessitant aucune hypothèse sur le trafic. Néanmoins, de telles approches sont difficilement applicables telles quelles pour de la détection en temps réel, leur complexité étant quadratique relativement au nombre d'instances. Cette complexité peut être réduite à $O(n \log n)$ lorsqu'un outil d'indexage spatial est utilisé comme KD-tree [59], R-trees [60] ou X-trees [61]. Néanmoins, de telles approches sont de complexité exponentielle relativement au nombre de dimensions considérées, ce qui rend leur utilisation difficile.

2.3.4 Techniques de clustering

Certaines méthodes utilisent du clustering pour opérer leur détection, qui consiste à grouper les instances du trafic lorsqu'elles sont proches les unes des autres. À partir de cette structure de données, il existe plusieurs manières de détecter des évènements anormaux.

Plusieurs techniques utilisent les *outliers*, un sous-produit de certains algorithmes de clustering. Lorsqu'une instance ne peut pas être classée dans un cluster pertinent (elle se trouve dans une partie de l'espace trop peu dense), elle est classifiée comme étant un outlier. Ces outliers sont alors considérés comme des anomalies.

D'autres techniques utilisent la structure des clusters créés pour évaluer un score d'anormalité. Chaque instance est comparée au centre, à la taille où encore à la densité du cluster auquel elle appartient. Comme c'est un algorithme peu complexe et ne produisant pas d'outlier, l'algorithme k-means a souvent été utilisé [62, 63]. Enfin, certaines approches peuvent classifier des clusters entiers comme anormaux.

Dans [39], Casas propose UNADA, un algorithme de détection utilisant l'algorithme de clustering DBSCAN [64]. Contrairement à k-means, cet algorithme ne nécessite pas de définir au préalable un nombre fixe de clusters. Pour chaque élément, l'algorithme compte le nombre d'éléments dans un rayon r autour de cet élément (on parle de voisins). Si ce

nombre dépasse N, alors l'élément est considéré dans une partie dense de l'espace, et est ajouté à un cluster. Deux éléments sont alors considérés dans un même clusters si il existe une suite de voisin à voisin entre eux, où tous les éléments traversés sont dans une zone dense de l'espace. UNADA applique DBSCAN à chaque sous-espace N-dimensionnel de l'espace de données. Ici, N vaut deux, donc l'algorithme calcule alors toutes les combinaisons possibles de deux attributs parmi tous les attributs du trafic considérés. Pour chaque couple d'attributs, les valeurs sont projetées sur le sous-espace à deux dimensions correspondant. Pour chaque sous-espace, les outliers détectées sont alors comparés au centre du plus gros cluster trouvé dans ce sous-espace. Un score d'anormalité leur est alors assigné. Les résultats de chaque sous-espace sont alors combinés pour détecter si l'instance ajoutée au modèle est anormale ou non. Notons qu'UNADA peut opérer en temps réel, ce traitement complexe n'étant exécuté que si un changement significatif est détecté dans les propriétés du trafic. L'algorithme produit aussi automatiquement des signatures pour les anomalies détectées, simplifiant le travail de l'administrateur. Une amélioration est proposée par [65]. Cette nouvelle version opère sur une fenêtre glissante et opère avec du *grid clustering*. Le *grid clustering* regroupe en effet les instances dans des hyperrectangles (à la manière des intervalles d'un histogramme), et opère le regroupement à partir du nombre d'éléments dans chaque hyperrectangle. En résulte un clustering moins exigeant en ressources de calcul. L'algorithme détecte alors les anomalies plus rapidement et produit de meilleurs résultats.

Dans [66], Zhao et al. présentent AWSDS. Ce détecteur utilise aussi du clustering de flux par sous-espace, et à chaque flux détecté comme un outlier dans l'un des sous-espaces est assigné un score d'anormalité. Ce détecteur produit une détection correcte sur le jeu de données KDD'99, mais sa complexité quadratique le rend difficile d'usage en temps réel.

Portnoy et al. [67] proposent un autre détecteur utilisant du clustering. Après une étape de standardisation, ce détecteur utilise une variante du *single-linkage clustering*. Cette technique de clustering hiérarchique utilise la distance entre chaque paire d'éléments. Les éléments sont alors regroupés deux à deux dans un même cluster, en considérant les distances inter-éléments de la plus courte à la plus longue. Bien que les clusters créés soient assez grossiers, cet algorithme est capable de traiter le trafic en un temps linéaire. La labellisation des clusters est réalisée en comptant le nombre d'instances dans chaque cluster. Les résultats de détection atteignent environ 50% des attaques avec un taux de faux positifs inférieur à 1%. CBUID [68] utilise une technique très similaire. La technique de clustering est différente, puisqu'elle regroupe les éléments dans des clusters en hypersphères. Le cluster le plus gros est alors considéré comme normal. Cette approche en question produit des résultats légèrement meilleurs que [67].

fpMAFIA [69] est une version optimisée de pMAFIA [70] qui lui-même est une version

étendue de l'algorithme de clustering CLIQUE [71]. La première étape de la détection utilise un ensemble de *frequent baskets* grâce à une version modifiée de l'algorithme *Adaptive Grid Algorithm*. Cet outil de clustering est très similaire à du *grid clustering*, la différence principale tenant à l'utilisation d'intervalles (dans la construction des hyperrectangles) pouvant changer de taille. Les zones les plus denses de l'espace sont alors caractérisées sur des intervalles plus fins que les zones moins denses. Cela permet de focaliser l'analyse sur les zones de l'espace les plus intéressantes. Les clusters sont ensuite extraits d'une structure d'arbre de type FP-Tree [72], qui a été inspirée par la recherche de *frequent itemsets* dans des bases de données transactionnelles (évoquée dans la partie 2.2.3). Finalement, les instances qui n'appartiennent pas à un cluster sont considérées comme anormales. Néanmoins, bien que théoriquement plus complexe que pMAFIA et CLIQUE vis-à-vis du nombre de dimensions, les auteurs soutiennent que fpMAFIA opère une meilleure détection. Ceci est probablement dû au fait que cet algorithme nécessite moins de passes lorsque la dimensionnalité des clusters augmente. Néanmoins, comparé à trois autres techniques utilisant du clustering, kNN ou des SVM, l'approche génère plus de faux positifs. C'est alors la capacité de l'approche à traiter le trafic plus rapidement qui la rend intéressante.

Nastaiinullah et al. [73] proposent une autre approche utilisant l'algorithme de clustering CLIQUE. Après avoir appliqué une standardisation *z-score*, les auteurs utilisent PCA pour réduire la dimensionnalité du problème. Enfin, l'algorithme de clustering CLIQUE est appliqué sur les données pour identifier des sous-espaces contenant des *unités denses* (des hypercubes contenant un nombre suffisant d'instances). Comme le nombre de sous-espaces à explorer peut être grand, CLIQUE utilise la technique du *Minimal Description Lenght (MDL) pruning* pour mettre de côté les sous-espaces les moins intéressants. Dans chaque sous-espace, les unités denses sont regroupées pour constituer des clusters, classifiés suivant leur taille et nombre d'instances. Enfin, CLIQUE résume chaque cluster créé à l'aide d'un hyperrectangle, ce qui permet de simplifier la phase de détection. Évalué sur KDD'99, atteint 94.5% des attaques détectées pour un taux de faux positifs à 1.5%. Néanmoins, dans cette configuration, l'analyse du jeu de donnée prend 2752.1 secondes avec 10 attributs du trafic surveillés contre 154.6 secondes pour 6 attributs. Cela démontre la difficulté de l'approche à passer à l'échelle, lorsque de nombreux attributs du trafic sont à surveiller.

L'approche proposée par Bhuyan et al. [74] repose sur deux algorithmes. Une première étape de clustering, utilisant des arbres et appliqué sur des sous-espaces, regroupe les instances similaires même dans un espace à hautes dimensions. La technique MMIFS [75] de sélection d'attributs est alors utilisée pour simplifier la phase de détection. Cette technique sélectionne les attributs un par un suivant la quantité d'information (au sens de la théorie de l'information de Shannon) qu'ils apportent. Cette technique est ici utilisée pour construire récursivement des sous-espaces plus petits, à partir de chaque sous-espace

construit précédemment, qui sont alors organisés sous la forme d'un arbre de clusters. Lorsque les clusters sont stables, l'algorithme CLUSLab est alors utilisé pour assigner un label à chaque cluster en fonction de trois mesures : sa taille, sa compacité et ses attributs dominants. Une amélioration du détecteur est présentée dans [76], qui propose d'utiliser la technique de sélection MIGE-FS avec l'algorithme de clustering TCLUS. Ces algorithmes ont une complexité quadratique relativement au nombre d'instances, mais proposent une bonne détection des anomalies. Les techniques de sélection d'attributs aident ces algorithmes à être plus autonomes, évitant la levée de fausses alarmes. Comme peu de détecteurs abordent le problème de la sélection des attributs du trafic à surveiller, de telles approches sont pertinentes puisqu'elles ne demandent pas à l'administrateur de choisir de manière arbitraire des attributs du trafic à surveiller.

Par nature, les approches utilisant du clustering ont un avantage significatif, notamment parce qu'elles construisent une structure de données représentative du trafic. Différentes classes de trafic sont ainsi caractérisées par les clusters créés. La structure de donnée créée est alors souvent utilisée pour fournir à l'administrateur plus d'informations sur la nature des anomalies détectées, ce qui simplifie la phase d'analyse. Par conséquent, ces approches peuvent être à privilégier par rapport aux approches utilisant les plus proches voisins.

2.3.5 Techniques supervisées adaptées

Plusieurs techniques, traditionnellement utilisées par des approches supervisées, ont été adaptées pour travailler de façon non-supervisée.

NSOM [77] par exemple, utilise des SOM pour détecter des anomalies en temps réel. Après un apprentissage non-supervisé, la méthode proposée détecte les anomalies en évaluant à quel point l'entrée sur réseau de neurones est différente de celle proposée par le neurone gagnant. Comme pour les réseaux de neurones supervisés, de tels algorithmes nécessitent une période d'entraînement assez longue avant d'être efficaces. Néanmoins, la phase de détection est généralement assez simple, ce qui en fait de bons candidats pour une détection en temps réel. Un autre détecteur utilisant un réseau de neurones non supervisé est RT-UNNID [78]. Pour détecter les anomalies connues, un jeu de données labellisé est utilisé. Néanmoins, cet algorithme est aussi capable de détecter des anomalies inconnues à l'aide d'un apprentissage non-supervisé. RT-UNNID est mise en œuvre avec trois réseaux de neurones différents : SOM, ART-1 et ART-2. D'après les résultats obtenus par les auteurs, ART produit de meilleurs résultats que SOM en général, et ART-2 est l'algorithme le plus adapté à de la détection en temps réel.

Dans [79], Wagner et al. cherchent à détecter des flux IP anormaux. Ils utilisent OCSVM (One-class SVM) à la fois de façon supervisée et non-supervisée. La détection

non-supervisée produit de bons résultats tout en conservant une complexité limitée.

2.3.6 Autre techniques

Otey et al. [80] présentent un détecteur capable de détecter des anomalies dans des données contenant à la fois des attributs catégoriques et continus. Pour ce faire, l'algorithme construit une fonction de distance pour comparer les instances entre elles. Deux instances sont considérées comme proches si elles partagent des attributs catégoriques ou si leurs attributs sont cohérents vis-à-vis d'une matrice de covariance maintenue par l'algorithme. Les instances anormalement distantes des autres sont alors considérées comme anormales. Notons que ce détecteur est capable d'opérer de façon distribuée.

Dans [81], Song et al. proposent un détecteur utilisant du filtrage par grille, du clustering et des SVM. Pendant une phase d'entraînement, l'algorithme utilise du filtrage par grille pour supprimer les potentielles anomalies présentes dans le jeu de données. Les données supposées normales extraites sont ensuite divisées en un nombre k (choisi dynamiquement) de clusters. Les frontières de chaque cluster sont alors déterminées à l'aide d'un SVM. Pendant la phase de détection, chaque instance est associée au cluster le plus proche, puis testée par le SVM correspondant. Ce détecteur est très autonome, la plupart des paramètres de l'algorithme étant définis automatiquement. La phase de détection a aussi une complexité relativement basse, permettant une opération nécessitant peu de ressources.

Fernandes et al. [82] présentent un détecteur divisé en deux modules, le premier opère la caractérisation du trafic à partir de données collectées au fil du temps. Deux méthodes sont proposées pour ce premier module : une utilisant PCA extrayant l'évolution temporelle du trafic et l'autre reposant sur de l'*Ant Colony Optimization*, une technique de clustering de flux. Chaque méthode crée une *Signature Digitale* grâce à de l'analyse de flux, c'est-à-dire un ensemble de séries temporelles caractérisant une journée de trafic. Le deuxième module opère la détection en utilisant l'algorithme de Dynamic Time Warping (DTW), un algorithme de reconnaissance de motifs utilisé pour comparer les séries temporelles. L'algorithme semble produire de nombreux faux positifs, mais détecte la plupart des anomalies. L'algorithme est prometteur puisqu'il donne à l'administrateur un ensemble pertinent de données statistiques sur les anomalies détectées, tout en proposant une détection très rapide.

2.3.7 Considérations relatives aux détecteurs non-supervisés

Les techniques non-supervisées ont l'avantage d'être complètement autonomes, tout en ayant la capacité de détecter de nouvelles anomalies. Limitant au maximum le travail de l'administrateur, elles sont le type de techniques vers lesquelles se tourner à l'avenir. Le

principal problème avec de telles approches c'est qu'elles nécessitent un compromis entre les ressources de calcul consommées et les informations fournies à l'administrateur.

Comme nous avons pu le constater, les approches les plus légères, reposant sur de la détection de changement, fournissent assez peu d'informations sur les anomalies détectées. Seul un indice d'anormalité est fourni, ce qui rend difficile le diagnostic de l'alerte. Si en général le problème n'est pas étudié en profondeur, les approches plus avancées permettent en général une classification plus fine des données d'entrée. C'est notamment le cas des approches utilisant du clustering. Mais cette analyse a un coût, puisque regrouper les instances similaires nécessite souvent la comparaison de chaque instance (généralement un flux) avec un grand ensemble d'autres instances. C'est une analyse qui peut être algorithmiquement complexe, rendant une détection en temps réel difficile.

Il est donc nécessaire, lors de l'utilisation de techniques non-supervisées, de choisir ou d'adapter la méthode de détection aux ressources de calculs disponibles. Cela peut alors nécessiter des compromis, par exemple en focalisant la méthode de détection sur certains types d'attaques.

2.4 Détecteurs hybrides

Bien que visant une très haute autonomie, les détecteurs purement non-supervisés doivent faire face à plusieurs problèmes. Tout d'abord, même si certains détecteurs non-supervisés sont capables de diviser le trafic en plusieurs groupes distincts, une labellisation multi-classes n'est possible que si une forme de connaissance est introduite dans le système. Comme l'analyse des anomalies est un travail fastidieux, un détecteur autonome devrait être capable de reconnaitre certains motifs déjà vus auparavant. Par conséquent, plusieurs chercheurs proposent de combiner certaines techniques basées sur des connaissances ou supervisées avec un détecteur non-supervisé.

Dans [83], Lu et Tong proposent un détecteur hybride qui combine le détecteur Snort avec deux détecteurs non-supervisés utilisant CUSUM et du clustering *Expectation-Maximisation (EM)*. CUSUM est utilisé pour détecter des changements dans le profil du trafic, alors que le clustering EM est utilisé pour paramétrer un modèle combinant plusieurs estimations Gaussiennes. Ceci permet la détection à la fois d'évènements improbables dans le temps, grâce à CUSUM, mais aussi des combinaisons d'attributs du trafic anormales grâce au clustering EM. Ce détecteur hybride devrait être capable d'opérer en temps réel suivant le nombre de règles Snort utilisées. Néanmoins, la configuration de ces règles reste fastidieuse.

Zhang et al. [84, 85] présentent un détecteur hybride utilisant des forêts aléatoires. Ce détecteur est divisé en deux parties, un traitement hors-ligne et un en ligne. Il propose une détection à la fois supervisée et non-supervisée. Les forêts aléatoires sont utilisées pour

construire un arbre de décision à partir d'un ensemble d'autres arbres de décision construits à partir d'un sous-ensemble d'attributs choisis aléatoirement. Lorsque le processus non-supervisé détecte une anomalie, une signature de l'anomalie est produite. Cette signature est alors réintroduite dans la partie supervisée pour détecter des anomalies similaires dans le futur. C'est un avantage non négligeable sur les détecteurs purement supervisés, puisque de telles techniques évitent à l'administrateur une analyse déjà effectuée sur des anomalies plus anciennes. De plus, l'approche séparant le traitement en deux parties distinctes de complexités différentes permet son opération sur du trafic en temps réel. Comme chercher dans le trafic des motifs connus est généralement plus simple que d'exécuter une analyse en profondeur non-supervisée, cette approche rend ce détecteur beaucoup plus réaliste pour un contexte industriel.

Dans [86], Yamanishi et Takeuchi utilisent SmartSifter, un détecteur utilisant un modèle reposant sur une combinaison de gaussiennes. Ce détecteur, calculant un score d'anormalité par instance, est combiné dans cet article à un détecteur supervisé. Pour ce faire, les anomalies avec un score très haut sont labellisées, en production, par un expert. Le détecteur utilise ensuite un mécanisme de renforcement, de manière à ce que cette décision soit réintroduite dans le système, formant un ensemble de règles permettant de filtrer le trafic en entrée.

Les détecteurs hybrides font un très bon compromis entre des informations exhaustives fournies à l'administrateur (grâce à la base de connaissance) avec l'autonomie de la détection. L'approche par renforcement apparait comme la plus pertinente, puisque qu'elle permet un lien naturel et cohérent entre la partie non-supervisée et celle supervisée (puisque les informations produites par la détection non-supervisée sont utilisées pour entraîner le modèle supervisé). Néanmoins et par conséquent, ces approches nécessitent une détection non-supervisée efficace avant tout.

2.5 Résumé

Dans ce chapitre, nous avons présenté de nombreuses approches existantes cherchant à résoudre le problème de la détection d'anomalies. Les approches non-supervisées (parfois étendues pour une détection hybride) sont les plus autonomes et résolvent la plupart des problèmes liés à l'utilisation de base de connaissances ou de modèles entraînés du trafic. Néanmoins, elles peuvent s'avérer algorithmiquement complexes, ce qui peut nécessiter des ressources de calcul conséquentes. Afin de palier ce problème, nous proposons une nouvelle approche de détection non-supervisée, qui, pour limiter la complexité de son opération, se focalise sur la détection d'anomalies ayant un impact sur l'ensemble du trafic. Cela convient particulièrement à la détection d'attaques par déni de service distribuées.

Chapitre 3

AATAC

Sommaire

Afin de répondre à la problématique énoncée dans le premier chapitre, nous avons conçu un nouvel algorithme appelé AATAC, pour Autonomous Algorithm for Traffic Anomaly Characterization, soit *Algorithme Autonome pour la Caractérisation des Anomalies du Trafic*. AATAC est tout d'abord un algorithme se voulant autonome. Il est donc un algorithme non supervisé, qui, par conséquent, repose sur l'hypothèse suivante :

Hypothèse 1. *Une anomalie est un évènement statistiquement rare, dont les propriétés diffèrent significativement du trafic en condition normale.*

La notion d'évènement est néanmoins sujette à interprétation. Un détecteur peut en effet opérer une détection à différents niveaux de granularité. Dans la littérature, les approches les plus récentes basées sur de l'apprentissage automatique, utilisent généralement

le flux comme unité d'analyse. À l'inverse, des approches basées sur de la détection de changement considèrent un évènement comme issu d'un changement dans l'état global du trafic à un instant donné. Cette interprétation a des conséquences, puisqu'elle impacte la qualité de la détection, les données pouvant être fournies à l'administrateur lors de la levée d'une alerte, ainsi que les ressources nécessaires à l'exécution de l'algorithme.

Dans notre cas, nous étendons donc l'hypothèse 1 de la manière suivante :

Hypothèse 2. *Une anomalie est un évènement caractérisé par sa date d'occurrence et impactant significativement les valeurs et probabilités de distribution des caractéristiques spatiales du trafic.*

Concrètement, cette hypothèse revient à considérer le trafic dans son ensemble et considérer qu'il est, à un instant donné, soit dans un état normal, soit dans un état anormal. Cette hypothèse est particulièrement appropriée vis-à-vis de notre problématique, la priorité de notre algorithme restant la détection des attaques par déni de service distribuées. Comme indiqué dans le chapitre 1, ces attaques ont généralement un impact significatif sur l'ensemble du trafic.

Dans la suite de cette section, nous décrivons tout d'abord l'algorithme dans son ensemble dans la partie 3.1. La section 3.2 décrit le traitement *continu* de l'algorithme tandis que la section 3.3 décrit son traitement *discret*. Ensuite, la section 3.4 explique comment diagnostiquer une anomalie grâce aux résultats produits par l'algorithme. La section 3.5 conclut ce chapitre en résumant son contenu.

3.1 Vue d'ensemble

AATAC repose sur la construction de ce que nous appelons un *instantané* du trafic. Chaque instantané est une représentation à court terme de l'état du trafic à un instant t.

Comme illustré par la figure 3.1, AATAC opère un traitement en deux parties distinctes. Un premier traitement, dit *continu*, utilise tout d'abord en entrée des données agrégées par flux. Elles sont ensuite utilisées pour alimenter une structure de données constituée d'un ensemble de compteurs. Le traitement continu se veut être un processus rapide, capable de traiter chaque instance de trafic dans un temps linéaire.

La deuxième partie de l'algorithme, dite *discrète* produit à intervalles réguliers un instantané du trafic. Chaque instantané est alors comparé à ceux précédemment créés. Une méthode basée sur l'analyse des k plus proches voisins nous permet alors d'identifier si l'instantané est statistiquement anormal, et si tel est le cas de lever une alerte.

Les instantanés produits ont l'avantage de pouvoir être simplement affichés comme un ensemble de courbes en deux dimensions. Cet outil simplifie grandement le travail de

diagnostic de l'anomalie, permettant à l'administrateur de prendre une décision éclairée vis-à-vis de l'anomalie détectée.

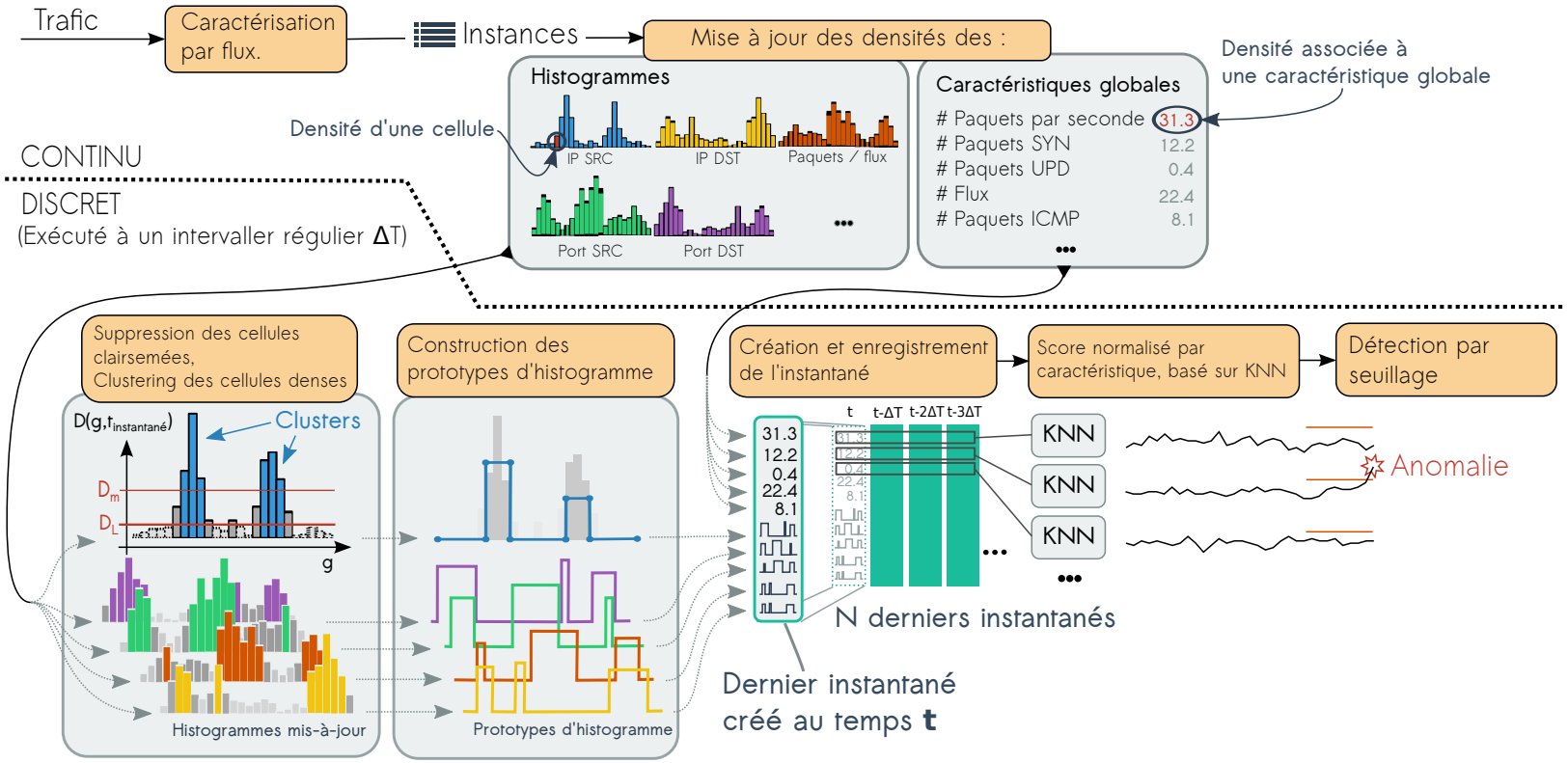


FIGURE 3.1 – Architecture de l'algorithme AATAC

Type	Protocole	Description
Estampille	∅	Date de début ou de fin du flux
Agrégation	IP	Protocole
	IP	Adresse IP source
	IP	Adresse IP destination
	TCP ou UDP	Port source
	TCP ou UDP	Port destination
Métrique	∅	Nombre de paquets
	∅	Nombre d'octets
	UDP	Nombre de paquets UDP
	TCP	Nombre de paquets TCP
		Nombre de paquets SYN
		Nombre de paquets URG
		Nombre de paquets PSH
		Nombre de paquets RST
		Nombre de paquets FIN
		Nombre de paquets CWR

TABLE 3.1 – Exemple des clés et métriques associées à un flux

3.2 Traitement continu

3.2.1 Construction des attributs du trafic

En entrée de l'algorithme, sont utilisées des données agrégées par flux caractérisés par le 5-tuple (IP_{source}, $IP_{destination}$, $Port_{source}$, $Port_{destination}$, $Protocole$), auxquels sont associées plusieurs métriques, telles que le nombre de paquets, d'octets ou la proportion de paquets SYN. Ces données constituent un ensemble d'instances. Ensemble, elles forment les **métriques de flux**. Le tableau 3.1 liste l'ensemble des données utilisées permettant de caractériser un flux ainsi qu'un certain nombre, non exhaustif, de métriques de flux. Parmi ces données, l'estampille est la seule caractéristique véritablement nécessaire à l'exécution de l'algorithme.

L'extraction des données du trafic par flux a été choisie de manière à pouvoir s'adapter à plusieurs situations. Dans un contexte industriel, la surveillance du réseau passe par la mise en place d'une architecture dédiée. De telles solutions permettent d'obtenir une vue cohérente du réseau, tout en limitant la bande passante nécessaire à récolter les données. Pour ce faire, ces approches reposent sur un protocole d'exportation de flux tel que Netflow [87] ou IPFix [2]. La plupart des équipements réseaux profitent de technologies permettant cet export de manière optimisée, limitant l'impact sur la mémoire et les ressources de calcul nécessaires au bon fonctionnement des équipements réseau. Des technologies reposant sur

de l'export échantillonné de paquets, telles que sFlow [1] restent néanmoins compatibles avec cette approche.

Pour être réalistiquement utilisable dans un contexte industriel, notre solution se devait d'être compatible avec de telles solutions, c'est pourquoi notre algorithme repose sur des données agrégées par flux en entrée.

3.2.2 Mise à jour incrémentale de la densité des cellules

Une fois les caractéristiques du trafic agrégées par flux, la partie continue de l'algorithme (partie supérieure de la figure 3.1) construit une structure de données permettant de caractériser les attributs du trafic à une échelle plus globale. Puisque ces métriques sont ensuite utilisées pour constituer un instantané du trafic, elles constituent l'ensemble des **métriques d'instantanés**. L'ensemble de ces métriques est noté F et peut être divisé en deux sous-ensembles :

$F_{globales}$ dont chaque métrique caractérise une propriété globale à l'ensemble du trafic.
Exemples : bande passante, nombre de paquets par seconde, etc.

$F_{distributions}$ dont chaque métrique caractérise la distribution d'un attribut du trafic.
Exemples : Nombre de paquets par adresse IP, nombre de flux pour un port donné, etc.

Pour évaluer ces caractéristiques du trafic, nous utilisons une méthode inspirée de l'algorithme D-Stream [88], un algorithme de *grid clustering* opérant sur des flux de données. D-Stream groupe les instances similaires lorsqu'elles appartiennent à une même partie assez dense de l'espace. Pour estimer la densité de l'espace autour d'une instance, il divise l'espace de valeurs en p partitions (ou cellules) de taille égale. Il attribue ensuite une densité à chaque cellule notée g en fonction du nombre d'instances qu'elle inclut à l'instant t.

Pour le cas d'une métrique f caractérisant une distribution du trafic ($f \in F_{distribution}$), notre algorithme dérivé de D-Stream est appliqué, par métrique, sur un espace à une dimension. Cette méthode simplifie le traitement et évite la *malédiction de la dimension* [89]. La structure finale, assignant une densité à chaque partition d'un espace mono-dimensionnel, s'assimile donc à un histogramme. Cela simplifie aussi la visualisation des données lors de la phase de diagnostic de l'anomalie décrite en section 3.4. Pour les métriques globales ($f \in F_{globales}$), le partitionnement n'est pas nécessaire. On évalue ainsi une caractéristique globale du trafic en supposant une cellule recevant toutes les instances du trafic, ou répondant à un critère particulier. Par exemple, pour évaluer le nombre de flux TCP, cette cellule ne recevrait que les instances (ou flux) contenant du trafic TCP.

Pour déterminer le poids de chaque instance dans le modèle, D-Stream attribue à chaque instance un *facteur de densité* qui dépend du temps écoulé entre sa date d'ajout au modèle et l'instant présent. Les instances plus âgées se voient ainsi assigner un poids plus faible dans le modèle que les instances récentes. Considérons une instance x ajoutée au modèle à l'instant t_x. Son *facteur de densité* à l'instant t est noté $D(x,t)$ et est calculé de la façon suivante :

$$D(x,t) = w_x \lambda^{t-t_x} \tag{3.1}$$

avec λ le *facteur d'évanouissement*, un paramètre de l'algorithme impactant la vitesse à laquelle le poids de chaque instance se réduit au fil du temps. Le poids w_x associé à l'instance x permet une caractérisation du trafic dépendante des propriétés de chaque instance. Ainsi, dans notre cas, ce poids w_n est fixé à 1 lorsque une caractérisation par flux est désirée, ou multiplié par le nombre de paquets dans le flux pour une caractérisation par paquet.

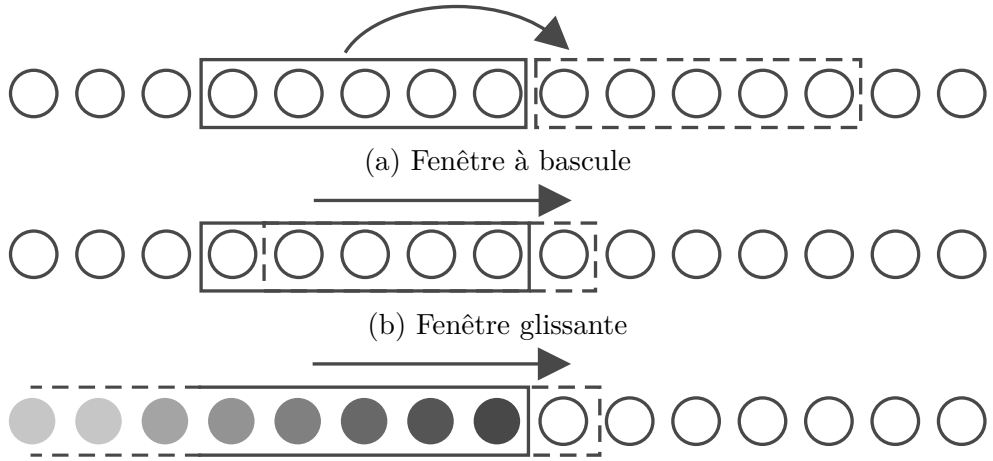
Considérons l'ensemble $E(g,t)$ des instances incluses dans la partition g à l'instant t. On peut alors assigner à g une densité $D(g,t)$ calculée de la façon suivante :

$$D(g,t) = \sum\nolimits_{x \in E(g,t)} D(x,t) \tag{3.2}$$

Ainsi une cellule recevant fréquemment des instances aura une densité élevée, alors qu'une cellule en recevant rarement aura une densité faible.

Le choix des instances à considérer et leur poids dans le modèle en fonction de leur date d'arrivée est un problème commun dans le domaine de l'analyse de signaux. Il existe plusieurs techniques, dites de fenêtrage, répondant à cette problématique. Puisque le poids d'une instance réduit de manière exponentielle au fil du temps, la technique que D-Stream propose correspond à une fenêtre à amortissement. Elle est illustrée sur la figure 3.2 conjointement avec les deux autres techniques de la fenêtre glissante et de celle à bascule.

Bien que nous aurions pu utiliser l'une de ces deux autres techniques, elles restent moins intéressantes pour le calcul d'une somme de densités que la fenêtre à amortissement. La fenêtre à bascule (figure 3.2a) consiste à considérer les N dernières instances, et d'attendre N nouvelles instances pour en calculer la somme. Suivant la quantité de données nécessaire à produire une caractérisation pertinente, cet intervalle peut être long, limitant la capacité de l'algorithme à traiter en temps réel les données. La fenêtre glissante (figure 3.2b) répond à ce problème en considérant les N dernières instances toutes les n instances ($n < N$). Néanmoins, afin d'évaluer la somme de la densité des N dernières instances, cette technique nécessite l'enregistrement des N dernières densités, ce qui peut représenter une quantité de donnée importante.



(a) Fenêtre à bascule

(b) Fenêtre glissante

(c) Fenêtre à amortissement

FIGURE 3.2 – Différentes techniques de fenêtrage

La fenêtre à amortissement (figure 3.2c) que nous utilisons considère l'ensemble de toutes les instances passées mais avec un poids réduisant exponentiellement au fil du temps. Elle produit une caractérisation cohérente à chaque instant, tout en ne nécessitant la sauvegarde que d'une quantité de données limitée pour produire la somme de densités des instances enregistrées dans le modèle. En effet, dans notre cas, $D(g,t)$ peut être calculé de façon incrémentale à l'ajout d'un instance en ne stockant que deux valeurs : la date d'ajout de la dernière instance t_l et sa densité associée $D(g,t_l)$. À l'ajout d'un nouvelle instance de poids w_n arrivée à l'instant $t_n > t_l$, la densité est mise à jour de la façon suivante :

$$D(g,t_n) = \lambda^{t_n - t_l} D(g,t_l) + w_n \tag{3.3}$$

Démonstration. Supposons l'instance x_n ajoutée à la cellule g à l'instant t_n, et qu'elle est la seule instance ajoutée depuis t_l. On a donc :

$$E(g,t_n) = E(g,t_l) \cup \{x_n\} \tag{3.4}$$

Ainsi :

$$\begin{aligned} D(g,t_n) &= \sum\nolimits_{x\in E(g,t_n)} D(x,t_n) \\ &= \sum\nolimits_{x\in E(g,t_l)} (D(x,t_n)) + D(x_n,t_n) \\ &= \sum\nolimits_{x\in E(g,t_l)} (w_x \lambda^{t_n - t_x}) + D(x_n,t_n) \\ &= \sum\nolimits_{x\in E(g,t_l)} (w_x \lambda^{t_n - \cancel{t_l} + \cancel{t_l} - t_x}) + D(x_n,t_n) \\ &= \lambda^{t_n - t_l} \sum\nolimits_{x\in E(g,t_l)} (w_x \lambda^{t_l - t_x}) + D(x_n,t_n) \\ &= \lambda^{t_n - t_l} D(g,t_l) + D(x_n,t_n) \end{aligned} \tag{3.5}$$

or $D(x_n,t_n) = \lambda^{t_n - t_n} w_n = w_n$ d'où :

$$D(g,t_n) = \lambda^{t_n - t_l} D(g,t_l) + w_n \tag{3.6}$$

□

Notons qu'à l'initialisation, la densité à $t = 0$ est définie à zéro, puisqu'à cet instant la cellule n'a jamais reçu d'instance :

$$D(g,0) = 0 \tag{3.7}$$

Cette méthode a l'avantage d'être de complexité linéaire relativement au nombre d'instances et ne nécessite pas de stocker les instances pour leur analyse.

3.3 Traitement discret

Le traitement hors-ligne, exécuté à intervalles réguliers, peut être divisé en plusieurs parties. Comme illustré dans la partie inférieure de la figure 3.1, un premier traitement est effectué sur l'ensemble des cellules (3.3.1). Le résultat de ce traitement est ensuite utilisé pour construire un instantané du trafic (3.3.3), qui sera enfin soumis à une détection d'anomalie (3.3.4).

3.3.1 Mise à jour de la structure de données

Comme indiqué dans la section précédente, la densité associée à chaque cellule est mise à jour de façon incrémentale à l'ajout d'une instance. Par conséquent, lorsque le traitement hors-ligne est exécuté, les densités des cellules ne sont pas cohérentes entre elles, puisqu'elles correspondent à des dates de mise à jour différentes. La première étape du traitement hors

ligne consiste alors à mettre à jour la densité de toutes les cellules de manière à ce qu'elles soient cohérentes avec la date de création de l'instantané notée t_h. La densité de chaque cellule g est alors mise à jour de la façon suivante :

$$D(g, t_h) = \lambda^{(t_h - t_l)} D(g, t_l) \tag{3.8}$$

Suivant l'espace des valeurs possibles, le nombre de cellules à considérer dans le modèle peut être très grand. Par conséquent, le stockage des données nécessaire à l'exécution de l'algorithme doit être fait de manière dynamique. Ainsi l'algorithme n'enregistre la densité et la date de dernière mise à jour associées à une cellule uniquement si elle a reçu au moins une instance.

Néanmoins, au fil du temps, le nombre de cellules enregistrées ne peut qu'augmenter, et éventuellement devenir très grand, ce qui impacterait les performances. Pour éviter une explosion de la quantité de mémoire utilisée, les données associées à une cellule ayant une densité très faible sont supprimées. En effet, les cellules associées à une densité très faible ont reçu peu d'instances récemment, elles ne sont donc plus utiles à la représentation du trafic actuel. Le seuil utilisé pour sélectionner ces cellules est noté D_l, elles sont dites *clairsemées*.

3.3.2 Création des prototypes d'histogrammes

Afin de simplifier la phase de détection d'anomalie, chaque distribution du trafic est simplifiée. On limite ainsi la quantité de données à stocker dans un instantané en transformant un ensemble de cellules caractérisant une distribution en un *prototype d'histogramme*. Fondamentalement, un prototype d'histogramme est une courbe linéaire par parties représentant de manière simplifiée la distribution caractérisée par un histogramme.

Pour construire ce prototype, les cellules ayant une densité supérieure à un seuil D_m, dites denses, sont utilisées. Les cellules adjacentes sont ainsi groupées en clusters dont plusieurs caractéristiques sont enregistrées, à savoir : les extrémités du cluster (min_c et max_c) sur la dimension considérée, ainsi que la densité moyenne des cellules incluses dans le cluster (avg_c). Puisque l'algorithme travaille sur des espaces à une dimension, l'algorithme de clustering est ici trivial : il parcourt l'ensemble des cellules denses et groupe deux cellules denses si elles sont adjacentes.

Ces données sont utilisées pour produire une représentation simplifiée de la distribution appelée prototype d'histogramme. La courbe est produite en ajoutant, pour chaque cluster, les 4 points $(min_c, 0)$, (min_c, avg_c), (max_c, avg_c) et $(max_c, 0)$, à l'ensemble des points constituant le prototype. Cette construction est illustrée par la figure 3.3.

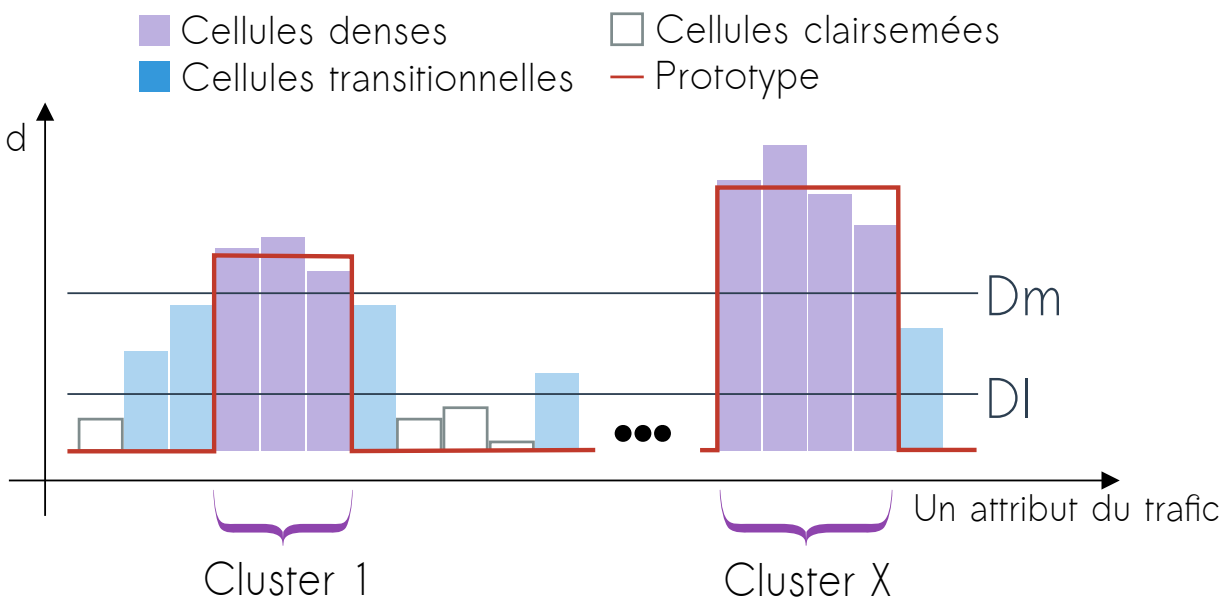


FIGURE 3.3 – Suppression des cellules clairsemées et construction du prototype d'histogramme

3.3.3 Construction et stockage de l'instantané du trafic

L'ensemble des prototypes produits est ensuite associé aux valeurs de densité des cellules globales au trafic, formant l'ensemble F des métriques d'instantanés. Elles sont alors regroupées pour former un instantané du trafic, lequel est alors enregistré avec l'ensemble des N derniers instantanés produits.

3.3.4 Détection des instantanés anormaux

Pour évaluer à quel point l'état du trafic actuel est différent du trafic habituel, l'algorithme compare le dernier instantané créé aux N derniers instantanés enregistrés. La méthode des k *plus proches voisins*, utilisée notamment comme un estimateur de densité locale, est souvent utilisée comme un score d'anormalité. La figure 3.4 illustre le fonctionnement de cet algorithme dans un espace à deux dimensions. Pour l'élément considéré en rouge sur la figure, l'algorithme calcule la distance avec tous les autres éléments. Il trie ensuite les éléments par ordre de proximité et considère seulement les k plus proches éléments. La densité locale peut alors être estimée de plusieurs façons différentes. La méthode la plus commune est d'utiliser la distance au k^{ieme} voisin, mais d'autres méthodes peuvent dépendre de la distance aux autres plus proches voisins. Par exemple, ce score peut être calculé en utilisant la moyenne des distances aux k plus proches voisins, ou encore en

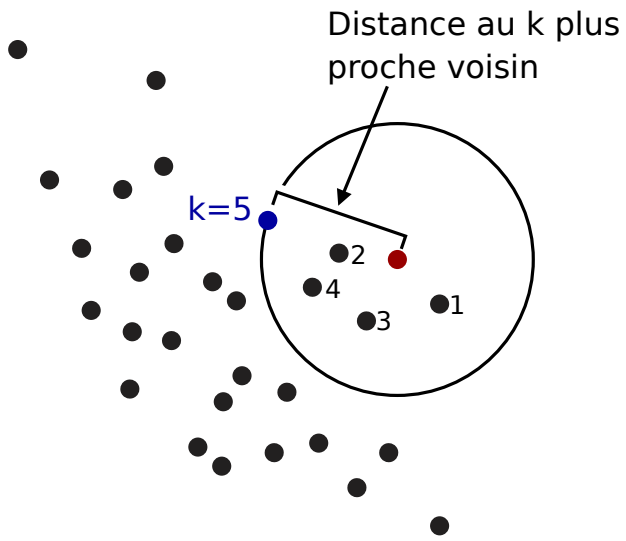


FIGURE 3.4 – Calcul de la distance au k plus proche voisin

utilisant une pondération gaussienne en fonction du rang du voisin.

Dans le cas d'AATAC, l'algorithme des k plus proches voisins est alors appliqué par métrique d'instantané pour estimer à quel point le dernier instantané est différent des autres. Ici, le score est calculé en utilisant la distance au $k^{\text{ième}}$ voisin, les autres méthodes de pondération ayant produit des résultats sensiblement similaires. Un instantané est noté S, et la distance entre deux instantanés S_i et S_j suivant la métrique f est notée $d(S_i, S_j, f)$.

La façon dont est calculée la distance entre deux métriques est différente suivant si elle caractérise une distribution ou une caractéristique globale au trafic. Pour une métrique caractérisant une distribution, la fonction de distance doit pouvoir comparer deux prototypes d'histogramme. Puisqu'un histogramme est une courbe constante par partie, AATAC utilise l'aire entre les deux courbes comme fonction de distance entre deux prototypes. Pour les métriques globales, c'est la distance euclidienne qui est utilisée pour comparer deux densités.

Notons qu'avant de calculer le score d'anormalité, l'ensemble des distances calculées est standardisé sur l'ensemble des N derniers instantanés. Pour ce faire, chaque distance est divisée par la variance σ, soit la moyenne du carré des distances :

$$\begin{aligned} d_{\text{normalisée}}(S_i, S_j, f) &= \frac{d(S_i, S_j, f)}{\sigma} \\ avec : & \\ \sigma &= \sum_{i=0}^{N} \sum_{j=i+1}^{N} d(S_i, S_j, f)^2 \end{aligned} \tag{3.9}$$

Suivant la métrique d'instantané considérée, la distance entre deux valeurs peut avoir différents ordres de magnitude. Cette standardisation, si elle n'est pas nécessaire à l'exécution de l'algorithme des k plus proches voisins, permet à postériori de simplifier la sélection d'un seuil utilisé sur le score d'anormalité produit. En effet, pour chaque métrique, le score produit est comparé à un seuil $\tau_{anomalie}$: si le score d'anormalité passe au-dessus de cette limite, une alerte est alors levée.

3.4 Diagnostic de l'anomalie

Lorsqu'une anomalie est détectée, les N derniers instantanés sont toujours disponibles. Comme un instantané consiste en un ensemble de prototypes d'histogrammes, il peut alors être affiché comme un ensemble de courbes. Ainsi, les N derniers instantanés sont alors proposés à l'administrateur, fournissant une vue dynamique du trafic à l'instant de l'anomalie. Cet outil permet de comprendre simplement quelles caractéristiques du trafic ont causé la levée de l'alerte, mais aussi de vérifier si l'anomalie n'a pas aussi eu un impact sur d'autres propriétés du trafic, dans une moindre mesure.

La figure 3.5 est un exemple de représentation d'un instantané. Sur la gauche de la figure sont affichés un certain nombre de prototypes d'histogrammes alors que les densités globales sont visibles sur la droite. Sous chaque métrique, un graphique montre l'évolution du score calculé au fil du temps permettant d'estimer le début de l'anomalie et les métriques impactées. Les graphiques en rouge indiquent quelles métriques sont détectées comme anormales à l'instant de l'anomalie.

3.5 Paramètres de l'algorithme

Comme vu précédemment dans cette section, AATAC nécessite la sélection de plusieurs paramètres pour opérer, à savoir D_l, D_m, λ, N, k, ΔT et enfin $\tau_{anomalie}$.

D_l et D_m sont deux paramètres dépendant directement de la densité des cellules lorsque le traitement discret est effectué, ils dépendent par conséquent des paramètres λ et ΔT ainsi que des poids w_x assignés aux instances. Aussi, il est possible et préférable de fixer D_l et D_m par expérimentation avec le trafic sur lequel va opérer l'algorithme. Les seuils D_l et D_m ne sont applicables que dans le cas de cellules caractérisant une distribution du trafic, ils peuvent être fixés par distribution. Dans notre cas, nous avons évalué, par distribution, la moyenne des densités des cellules lors du traitement discret. D_l et D_m sont alors fixés par rapport à cette moyenne.

Comme indiqué plus tôt, D_l est un paramètre d'optimisation évitant la multiplication

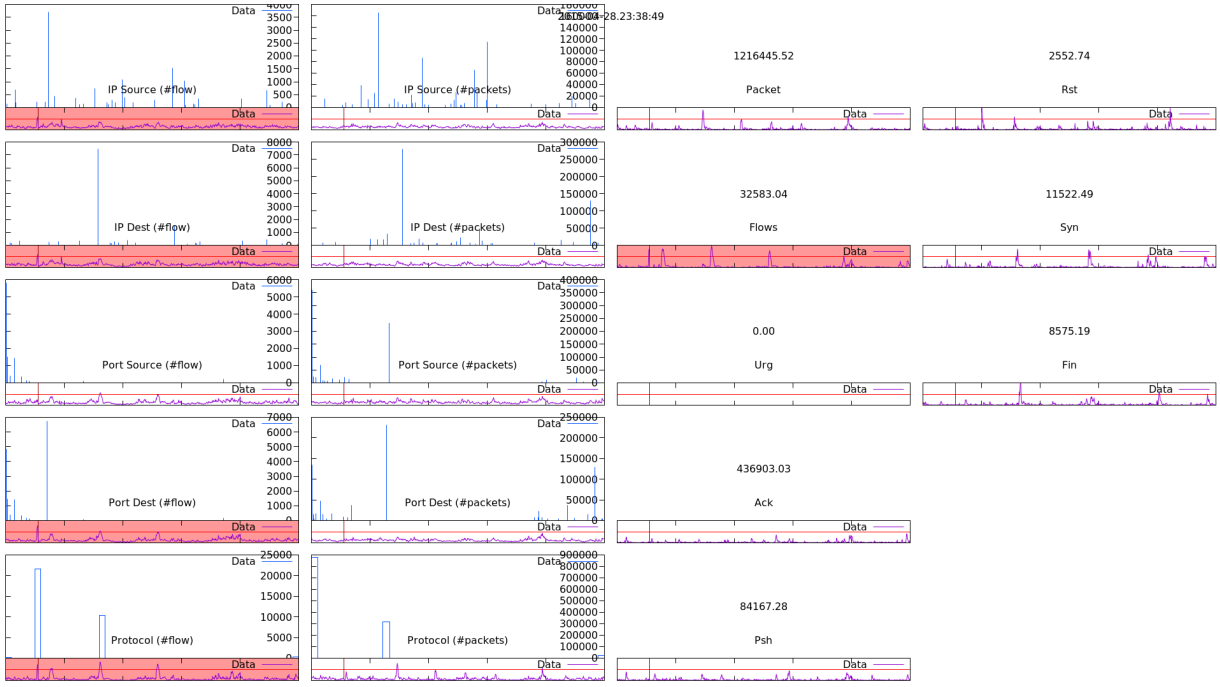


FIGURE 3.5 – Exemple d'un instantané tracé

des cellules à considérer dans le modèle. Comme la densité d'une cellule décroit de manière exponentielle au fil du temps, D_l peut être choisi très proche de zéro, de manière à ce que l'impact sur la caractérisation du trafic soit minimal.

Le paramètre D_m détermine à partir de quel point une cellule est considérée comme assez dense pour caractériser véritablement le trafic. Plus ce seuil est bas, plus le prototype d'histogramme créé par le traitement discret décrit précisément l'histogramme original. Un seuil bas crée des instantanés alors plus complexes, ce qui rend plus difficile la phase de comparaison des instantanés. Inversement, une valeur plus grande simplifie les prototypes créés et accélère le traitement discret. Le but est alors de déterminer à partir de quelle densité une cellule est pertinente dans la caractérisation du trafic. En raisonnant par rapport à la moyenne, nous pouvons considérer qu'une cellule est pertinente si elle représente au moins 1% de la moyenne des densités d'un histogramme. Cette valeur a produit des résultats acceptables lors de nos expérimentations.

λ caractérise la vitesse à laquelle diminue l'influence de chaque instance dans le modèle. C'est un paramètre néanmoins difficile à fixer tel quel, puisque son impact sur la caractérisation du trafic lors du traitement discret est aussi dépendante du paramètre ΔT. Néanmoins, nous pouvons simplifier son choix en introduisant à la place le paramètre $R = \lambda^{\Delta T}$. Pour comprendre ce paramètre, il faut noter qu'à chaque création d'un instantané, les instances considérées lors la création de l'instantané précédent ont toujours un

impact sur les densités des cellules actuelles. Considérons un instantané créé à l'instant t, et supposons qu'à l'instant $t + \Delta T$, une cellule g n'ait reçu aucune instance. Sa densité à l'instant $t + \Delta T$ est alors égale à :

$$D(g, t + \Delta T) = \lambda^{\Delta T} D(g, t) = R \times D(g, t) \tag{3.10}$$

Ainsi, R illustre à quel point les densités d'un instantané pèsent dans l'instantané suivant, ce qui conditionne à quel point deux instantanés successifs sont corrélés. Lorsque R est grand, la caractérisation du trafic produite est alors moins sensible aux changements soudains des propriétés du trafic.

Le paramètre ΔT correspond à l'intervalle entre deux créations d'instantanés, il est préférable de le choisir assez court lorsque c'est possible. En effet, comme la détection est effectuée lors du traitement discret, un intervalle faible permet de détecter l'anomalie au plus tôt. Néanmoins, augmenter la fréquence du traitement discret nécessite aussi plus de ressources de calcul.

Lorsque N est choisi grand, des instantanés plus anciens vont être considérés dans la détection. Ceci complexifie le traitement discret mais a aussi tendance à rendre la détection moins sensible aux changements à court terme du trafic. Ce paramètre N reste à fixer conjointement avec R et ΔT, ces trois paramètres affectant la temporalité de la caractérisation du trafic. En effet, augmenter n'importe lequel de ces paramètres produit une représentation à plus long terme du trafic, et inversement. Ces trois paramètres sont donc à équilibrer suivant la variabilité du trafic à surveiller, une caractérisation trop sensible pouvant générer des faux positifs.

Lorsque l'algorithme des k plus proches voisins est utilisé comme outil de classification, la valeur du paramètre k peut être fixée par validation croisée. Lorsqu'il est utilisé pour produire un score d'anormalité, ce ne n'est malheureusement pas applicable. Fixer le paramètre k par expérimentation est alors possible, mais de façons générale sa valeur a peu d'influence sur les scores calculés. Une règle empirique consiste alors à fixer $k = \frac{1}{2}\sqrt{N}$, ce qui a produit un résultat plus que convenable lors de nos expérimentations, quelle que soit la valeur de N choisie.

Enfin, le paramètre $\tau_{anomalie}$ permet d'ajuster directement la sensibilité du détecteur. Sa valeur dépend alors de la criticité de la détection, et doit alors être fixée par l'administrateur.

3.6 Résumé

Dans cette section nous avons présenté AATAC, un nouvel algorithme de détection d'anomalie visant en priorité les attaques par déni de service distribuées. AATAC est un algorithme non supervisé. Il produit et utilise dans son processus de détection des instantanés du trafic, reposant entre autres sur l'estimation de la distribution de plusieurs attributs du trafic. Les instantanés produits peuvent alors être affichés, fournissant à l'administrateur une vue dynamique des caractéristiques du trafic et lui donnant des informations pertinentes quant à l'anomalie détectée. Nous avons aussi listé l'ensemble des paramètres de l'algorithme et discuté de leur impact sur les performances du détecteur.

Chapitre 4

Évaluation

Sommaire

Dans ce chapitre, nous évaluons deux caractéristiques d'AATAC : la qualité de sa détection ainsi que sa capacité à traiter le trafic en temps réel avec des capacités de calcul limitées.

4.1 Évaluation de la qualité de la détection

4.1.1 Jeu de données

Pour évaluer la capacité de notre algorithme à détecter correctement une anomalie, un jeu de donnée labellisé est nécessaire. Pour assurer une évaluation significative, le jeu de données doit répondre à plusieurs critères.

Tout d'abord ce jeu de données se doit d'être réaliste. Les protocoles, les usages et le volume de données échangées évoluent constamment, par conséquent un bon jeu de données doit être constitué d'un trafic représentatif des conditions actuelles d'utilisation. De plus, il doit contenir un ensemble d'anomalies suffisamment réalistes et variées pour que les résultats de l'évaluation soient représentatifs d'une opération en conditions réelles. En particulier, puisque notre algorithme vise en priorité la détection d'attaques par déni de service distribuées, plusieurs attaques de ce type doivent être incluses dans le jeu de données. Enfin, un jeu de données de qualité implique des labels appropriés. Le processus de labellisation utilisé doit donc assurer la qualité des labels assignés aux instances de trafic.

Ceci étant dit, très souvent pour des raisons légales, les jeux de données utilisés pour l'évaluation de détecteurs d'anomalies ne sont pas publics. Peu d'entre eux sont alors à la fois disponibles, labellisés et représentatifs. Néanmoins, nous avons envisagé d'utiliser certains d'entre eux pour nos évaluations :

KDD 1999 [51]. La compétition KDD de 1999 a mis plusieurs équipes en concurrence pour créer un détecteur d'intrusion le plus efficace possible. Un jeu de données a alors été créé pour l'occasion, contenant neuf semaines de trafic généré manuellement dans un réseau local simulant un réseau militaire de l'*U.S. Air Force*. Pendant la génération, plusieurs attaques ont alors été effectuées et labellisées en conséquence. Ce jeu de données est fourni au format *CSV* associant à chaque flux contenu dans le trafic plusieurs métriques. Ce jeu de données est très souvent utilisé comme référence, mais bien que labellisé avec un protocole exemplaire, KDD'99 contient des attaques dans des proportions irréalistes. Aussi le jeu de données reste très ancien et ne contient pas d'estampilles temporelles qui seraient nécessaires à l'exécution de notre algorithme.

MAWILab [30]. Le jeu de données MAWILab est un ensemble de labels associés au trafic de l'archive MAWI. Cette archive est construite à partir de trafic capturé à raison de 15 minutes par jour entre le backbone WIDE et un fournisseur d'accès à Internet en amont. Bien que le trafic soit récent, le trafic capturé ne correspond pas entièrement à notre cas d'utilisation. Aussi, les labels associés au trafic sont générés automatiquement en combinant les résultats de plusieurs détecteurs. La qualité de ces labels n'est donc pas assurée, puisque qu'elle dépend de détecteurs par nature imparfaits.

UNB ISCX Intrusion Detection Evaluation DataSet [90]. Ce jeu de données de 2012 est constitué de trafic généré artificiellement. Pour qu'il soit réaliste, les générateurs utilisés ont été construits à l'aide de traces capturées et de l'analyse du comportement des utilisateurs. Pendant la génération du trafic, plusieurs scénarios d'attaques sont déroulés. Le jeu de donnée final ne contient malheureusement que

deux types d'attaques DDoS, un scan et une attaque par force brute. À nos yeux, cela c'est pas suffisant pour évaluer notre détecteur dans des conditions assez variées. Une version mise à jour du jeu de données, plus récente et publiée trop tard pour nos évaluations, est présentée par [91].

Puisque ces jeux de données ne remplissent pas tous les critères que nous nous sommes fixés, nous avons choisi de construire notre propre jeu de données. Pour ce faire, plusieurs solutions sont alors possibles :

- Capturer des traces réelles, puis les labelliser à la main ou à l'aide d'un ou plusieurs détecteurs jugés fiables : faire entièrement confiance à un détecteur n'est pas envisageable pour s'assurer de la qualité des labels, une analyse manuelle du trafic est donc nécessaire, ce qui est difficile et chronophage. Aussi, les anomalies étant généralement rares, il se peut que la capture ne contienne pas d'anomalie, cette méthode est donc un pari risqué.
- Créer des traces à l'aide de générateurs de trafic ou d'un simulateur : Cette approche nécessite un simulateur ou des générateurs capables de créer du trafic réaliste, ce qui est en réalité très difficile à construire et nécessite de nombreuses simplifications.
- Utiliser une combinaison des deux, en capturant des traces réelles puis en générant (avec un générateur ou manuellement) plusieurs attaques. Cette technique assure à priori la diversité des attaques générées artificiellement, ainsi que la qualité des labels qui leur sont associés. Aussi, cette approche garantit le réalisme au moins du trafic capturé (puisqu'il s'agit de trafic réel). Néanmoins, cette technique n'évite pas une analyse à priori, afin de vérifier si le trafic capturé ne contient pas d'anomalies. On peut néanmoins supposer les traces réelles comme ne contenant pas d'anomalie si elles sont créées dans un environnement sûr.

La dernière méthode, plus simple à mettre en place, est donc celle que nous avons choisie pour construire SynthONTS. Ce jeu de données est constitué d'une heure de traces en conditions moyennes, rejoué plusieurs fois avec une attaque différente générée. Il contient 12 types d'attaques dont plusieurs DDoS.

Pour construire ce jeu de données, nous avons capturé, dans le contexte du projet ONTIC [92] plusieurs jours de traces à l'entrée d'un fournisseur de cloud. Les traces en questions, nommées traces ONTS sont disponibles sur demande sur le site du projet. Après analyse, nous avons choisi une heure de traces contenant un minimum d'anomalies parmi les fichiers capturés. Le fichier en question contient une heure de traces, capturées en 2015, un mardi, entre 15 h 15 et 16 h 15. Le trafic a un débit moyen de 76000 paquets par

seconde, ce qui correspond à une bande passante de 475 Mbit par seconde.

Avant toutes choses, nous avons analysé le fichier afin d'y repérer d'éventuelles anomalies. Nous y avons trouvé 11 évènements apparaissant comme inhabituels, que nous avons, par prudence, décidé d'ignorer pour notre évaluation (ni leur détection, ou leur non-détection ne compte dans nos résultats). Notre analyse étant limitée aux entêtes des paquets, il nous était difficile d'identifier formellement ces évènements comme anormaux.

Une fois cette trace choisie, nous avons décidé de répliquer un réseau similaire au réseau original à l'aide d'une simulation. Pour ce faire, nous avons utilisé les outils *dig* et *traceroute* pour analyser l'architecture du réseau original. À ce réseau, nous avons connecté un ensemble de machines représentant l'Internet qui seront utilisées pour générer des attaques. La figure 4.1, illustre l'architecture en question. Comme on peut le voir, le routeur au centre de la figure, labellisé "Internet" fait frontière entre le réseau local, à droite, et Internet, à gauche.

Pour générer les attaques, nous avons utilisé l'outil CORE, un outil utilisant des machines virtuelles afin de simuler le comportement d'un réseau. Le serveur exécutant CORE que nous avons utilisé est équipé de 28 cœurs, il est capable de générer une quantité de trafic réaliste vis-à-vis du réseau simulé. Le tableau 4.1 résume l'ensemble des attaques que nous avons générées. Elles sont de plusieurs types :

DDoS Smurf. Cette attaque par déni de service distribuée utilise le protocole ICMP. L'attaquant usurpe l'adresse IP de la victime et envoie à une adresse IP de broadcast un grand nombre de paquets *echo request*. Les machines recevant la requête répondent alors à l'adresse IP source, saturant le réseau de la victime.

DDoS Fraggle. Cette attaque est une variante de l'attaque Smurf, mais exploite des protocoles reposant sur UDP. En général, cette attaque vise les protocoles *echo* (port 7) ou *chargen* (port 19), deux protocoles de mesure ou de surveillance des réseaux.

DDoS / DoS SYN flooding. Cette attaque par déni de service, distribuée ou non, repose sur une exploitation malicieuse du protocole TCP. Elle consiste à envoyer à la victime un grand nombre de demandes de connexions (de paquets SYN) sans répondre aux demandes de connexion reçues en retour.

Pour suivre l'état d'une demande de connexion, la machine cible doit allouer une zone de la mémoire. Or, pour ne pas que l'utilisation de la mémoire explose, le nombre de connexions semi-ouvertes est par conséquent limité. Par conséquent, au-delà d'un certain nombre de demandes de connexions restées sans réponse, la machine victime devra refuser de nouvelles demandes de connexion, ce qui peut empêcher des utilisateurs légitimes d'accéder aux services.

DoS UDP flooding. Cette attaque par déni de service nécessite l'envoi d'un grand nombre

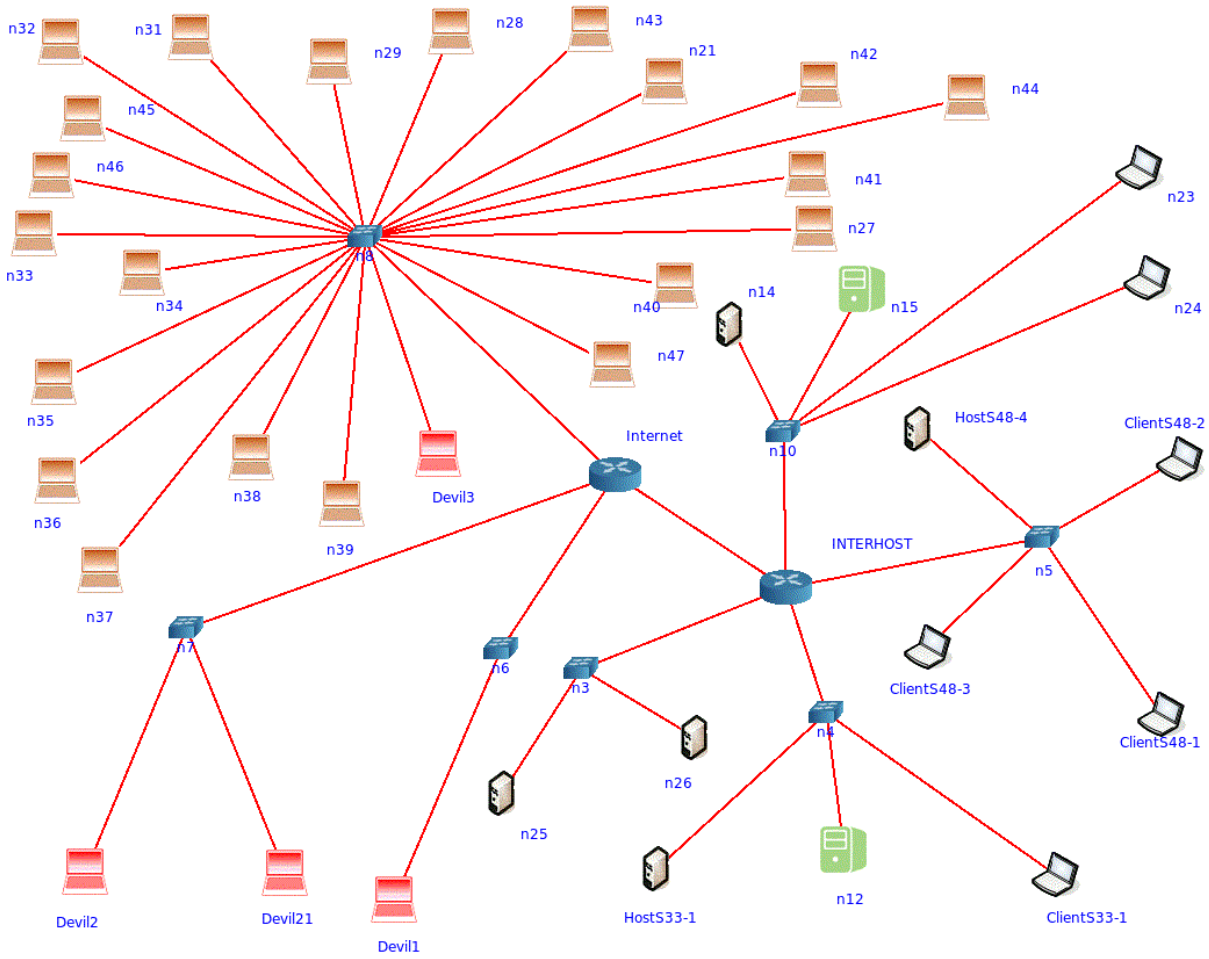


FIGURE 4.1 – L'architecture utilisée avec CORE pour générer les attaques

Type d'attaque	Outils	Durée	Nombre de paquets	Débit moyen
DDoS Fraggle	nmap	3 :55	28k	9 ko/s
DDoS Fraggle	nping	3 :37	7104k	259 Mo/s
DDoS Smurf	hping3	1 :33	12645k	143 Mo/s
DDoS Smurf	nping	3 :51	1141k	5.26 Mo/s
DDoS Smurf	nping	2 :31	6826k	47 Mo/s
DDoS Syn flooding	hping3	2 :07	4796k	21 Mo/s
DDoS Syn flooding	nping	2 :53	7247k	16 Mo/s
DoS Syn flooding	hping3	3 :38	5228k	17 Mo/s
DoS Syn flooding	nping	2 :58	7053k	10 Mo/s
DoS UDP flooding	nping	2 :30	4127k	24 Mo/s
FTP brute force cracking (rockyou.txt)	ncrack	4 :28	5373k	1,53 Mo/s
FTP brute force cracking (500-worst-password.txt)	ncrack	2 :07	2558k	1,55 Mo/s

TABLE 4.1 – Liste des attaques insérées dans le jeu de données SynthONTS

de paquets UDP à la victime. Le système doit alors traiter chaque paquet, et parfois répondre par un message *ICMP Destination Unreachable*. Cela épuise alors les ressources de la victime.

FTP brute force cracking. Cette attaque consiste à tester un grand nombre de mots de passe FTP pour accéder à ce service (et possiblement d'autres) de la machine.

L'ensemble de ces attaques forme un échantillon des anomalies visées en priorité par AATAC.

4.1.2 Courbes ROC

Un outil commun d'évaluation de la qualité de détection est la courbe d'efficacité du récepteur, ou courbe Receiver Operating Characteristic (ROC). Elle nécessite d'évaluer deux valeurs, à savoir le *Taux de Vrais Positifs (TVP)* et le *Taux de Faux Positifs (TFP)* du détecteur. Le taux de vrais positifs, aussi appelé "sensibilité", correspond à la probabilité qu'un détecteur lève une alerte lorsqu'un évènement anormal se produit. Le taux de faux positifs est la probabilité que le détecteur lève une alerte alors qu'aucune anomalie ne se produit.

Pour évaluer ces deux probabilités, nous exécutons l'algorithme AATAC sur les traces SynthONTS. La sensibilité est évaluée en raisonnant sur le nombre d'anomalies détectées parmi les 12 attaques intégrées. Si le détecteur lève une alerte pendant la durée de l'anomalie, nous comptons un *vrai positif* ajouté au nombre de vrais positifs noté *VP*. Si l'attaque n'a pas été détectée, c'est un *faux négatif*, le nombre de faux négatifs est alors noté *FN*. Le

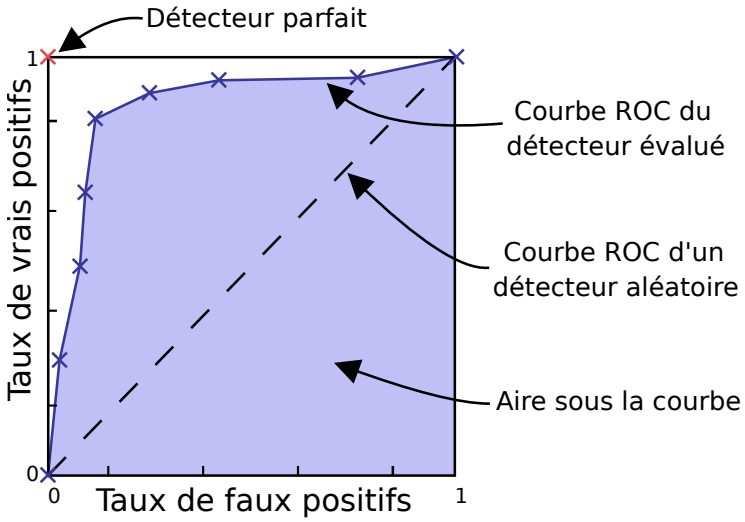


FIGURE 4.2 – Interprétation de la courbe ROC

taux de vrais positifs calculé est alors :

$$\begin{aligned} TVP = & \frac{VP}{VP + FN} \\ = & \frac{VP}{Nombre\ d'attaques} \end{aligned} \tag{4.1}$$

Si le détecteur lève une anomalie lorsqu'il n'y a pas d'attaque, nous comptons un *faux positif*. Le nombre de faux positifs est noté *FP*. Lorsqu'il n'y a pas d'attaque en cours, pour chaque test effectué par le détecteur n'ayant pas levé d'alerte nous comptons un *vrai négatif* ajouté au nombre de vrais négatifs noté *VN*. Le taux de faux positifs est enfin calculé de la façon suivante :

$$TFP = \frac{FP}{FP + VN} \tag{4.2}$$

Pour produire la courbe ROC, on calcule les valeurs de TVP et TFP en faisant varier l'un des paramètres de l'algorithme AATAC, dans notre cas le seuil $\tau_{anomalie}$. La courbe en question est illustrée par la figure 4.2. Un détecteur idéal doit détecter toutes les anomalies tout en ne générant pas de faux positifs, soit des valeurs de $TVP = 1$ et $TFP = 0$. Un bon détecteur doit ainsi s'approcher de ces valeurs. Aussi, un détecteur est plus intéressant si la modification de ses paramètres impacte peu ses performances.

Pour évaluer le détecteur indépendamment du paramètre choisi, l'*aire sous la courbe* ROC est alors souvent calculée. Plus l'aire sous la courbe est grande (et donc proche de 1), plus le détecteur est efficace.

4.1.3 Limites des courbes ROC

Bien que la courbe ROC soit un outil efficace pour évaluer la capacité d'un détecteur à classifier correctement un évènement, il souffre notamment de l'*oubli de la fréquence de base*. Ce biais existe lorsque l'on cherche à évaluer la probabilité d'occurrence d'un évènement dépendant d'un autre, tout en ne considérant pas la fréquence de base d'occurrence de l'évènement en question. Dans notre cas, la probabilité que le détecteur détecte correctement une anomalie dépend de la probabilité qu'une anomalie arrive effectivement.

Ainsi, la courbe ROC considère de manière équivalente le taux de vrais positifs, calculé dans notre cas sur un ensemble très limité d'attaques, et le taux de faux positifs, calculé sur un très grand nombre de tests effectués par le détecteur. Par conséquent, plusieurs faux positifs générés par jours ont peu d'impact sur le taux de faux positifs. Ainsi, les résultats exposés par une courbe ROC peuvent amener à penser qu'un détecteur générant de nombreux faux positifs, mais détectant plus d'anomalies, est meilleur qu'un détecteur ne générant aucun faux positif mais ne détectant pas toutes les anomalies.

Afin de palier ce problème, nous étendons notre analyse avec le calcul de la courbe d'opération du détecteur et l'efficacité de la détection d'intrusion associée.

4.1.4 Courbe d'opération et efficacité de la détection

La *courbe d'opération*, proposée par [93], trace la *Valeur Prédictive Positive (VPP)* en fonction du taux de faux positif. La valeur prédictive positive correspond à la probabilité qu'une alarme levée corresponde véritablement à une anomalie. Elle est donc calculée de la façon suivante :

$$\begin{aligned} VPP &= \frac{VP}{FP+VP} \\ &= \frac{VP}{\text{Nombre d'alertes générées}} \end{aligned} \tag{4.3}$$

La valeur prédictive positive, contrairement au taux de vrais positifs, est dépendante du nombre d'anomalies contenues dans le trafic. Pour évaluer le détecteur indépendamment de la fréquence de base, la courbe d'opération est alors comparée à une autre courbe dépendante de la fréquence de base, la *courbe de référence zéro (ZRC)*. Cette courbe de référence zéro correspond à un compromis entre la VPP et le taux de faux positifs. Elle est construite en traçant la courbe d'opération d'un détecteur qui détecterait toutes les anomalies, puis produisant un nombre croissant de faux positifs. Cette courbe est facile à construire dans le cas d'une évaluation puisque nous connaissons le nombre d'anomalies

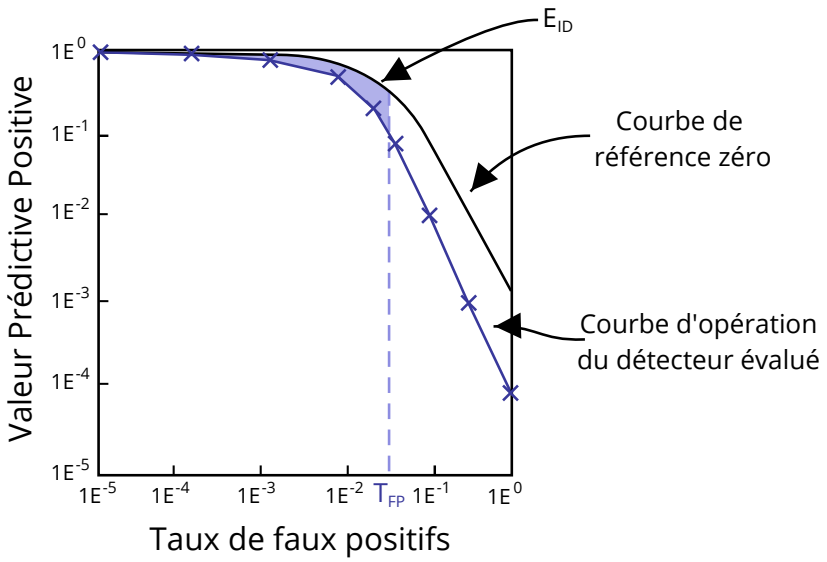


FIGURE 4.3 – Interprétation de la courbe d'opération

dans le jeu de données, et donc la fréquence de base.

Pour comparer les deux courbes, les auteurs proposent une nouvelle mesure, dite *Efficacité de la détection d'intrusion* et est notée E_{ID}. Cette mesure correspond à l'aire entre la courbe d'opération du détecteur et de la ZRC, en dessous d'un taux de faux positifs seuil noté T_{FP}. Ce seuil correspond au taux de faux positifs maximum acceptable, et permet d'ignorer dans le calcul de l'efficacité de la détection des valeurs pour lesquelles le taux de faux positifs est trop grand. En effet, au-delà d'un certain nombre de faux positifs levés, un détecteur devient vite inutilisable. Il est donc avisé de ne pas considérer les résultats au-delà d'un certain taux de faux positifs. Comme E_{ID} illustre la différence entre la courbe d'opération et la ZRC, plus cette valeur est petite, meilleur est le détecteur.

La courbe d'opération, la ZRC les valeurs de T_{FP} et E_{ID} sont illustrés par la figure 4.3.

4.1.5 Choix des paramètres

Dans cette section, nous présentons les différents résultats obtenus pour différentes configurations de l'algorithme. Nous avons évalué l'algorithme sur SynthONTS pour plusieurs valeurs de R, N, ΔT et $\tau_{anomalie}$.

Tout d'abord ΔT est fixé à 1 seconde. Cette valeur a été choisie de manière à obtenir une détection rapide, tout en permettant l'obtention de résultats rapidement. Les trois paramètres R, N et ΔT étant interdépendants, nous avons préféré fixer ce paramètre.

Pour R, nous avons choisi d'évaluer les performances d'AATAC pour les valeurs $R =$

0.01, 0.02, 0.05, 0.10, 0.50 et 0.90. Cette large plage permet d'évaluer l'impact de la corrélation inter-instantanés sur la qualité de la détection. En effet, pour $R = 0.01$ les instantanés créés sont presque indépendants, alors que pour $R = 0.90$, un instantané pèse pour 90% des densités constituant le prochain instantané qui sera créé.

L'algorithme est évalué pour les valeurs de $N = 100$, 500 et 1000, soit l'équivalent, pour $\Delta T = 1s$, d'une plage allant de 100 secondes à environ 16 minutes. Ce sont des valeurs raisonnables, permettant de caractériser le trafic sur une durée assez large compte tenu de la nature des attaques.

Pour choisir une plage de valeurs cohérente pour le seuil $\tau_{anomalie}$, nous avons exécuté AATAC sur une partie des traces ONTS. A la suite de cette étude, nous avons constaté qu'en conditions normales et pour la plupart des caractéristiques d'instantanés, le score produit par AATAC oscille entre 1 et 2. Un seuil acceptable doit à priori se situer un peu au-dessus de ces valeurs : il doit être suffisamment haut pour éviter la génération de faux positifs lorsque le trafic observe de faibles variations habituelles, mais doit aussi rester suffisamment bas pour détecter toutes les anomalies. Dans ces conditions, on peut supposer qu'un seuil $\tau_{anomalie}$ aux alentour de 3 devrait produire de bons résultats.

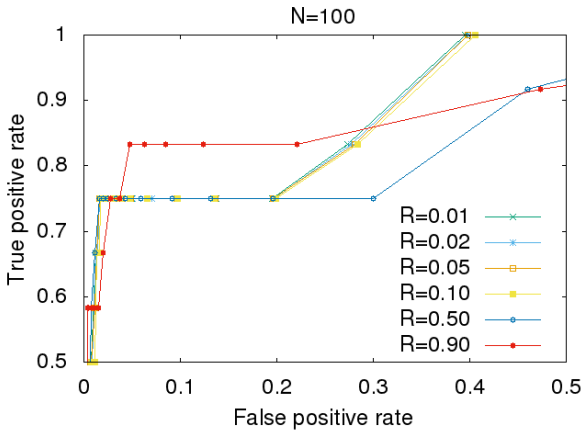
Dans le cas de notre évaluation, le paramètre $\tau_{anomalie}$ a varié entre 0 et 20. Cette large plage de valeurs a permis d'obtenir les qualités de détection allant d'un détecteur levant des alertes constamment à un détecteur ne levant aucune alerte. Néanmoins, la répartition des valeurs choisies est faite de telle façon qu'un nombre important de points soit situé un peu au-dessus des valeurs moyennes, l'objectif étant d'obtenir, sur la courbe ROC, un nombre suffisant de points proches de la condition d'opération optimale du détecteur.

Concernant les caractéristiques d'instantanés utilisées, nous en avons choisi 18. 10 d'entre elles caractérisent la distribution d'un attribut du trafic, et 8 autres caractérisent chacune un attribut global.

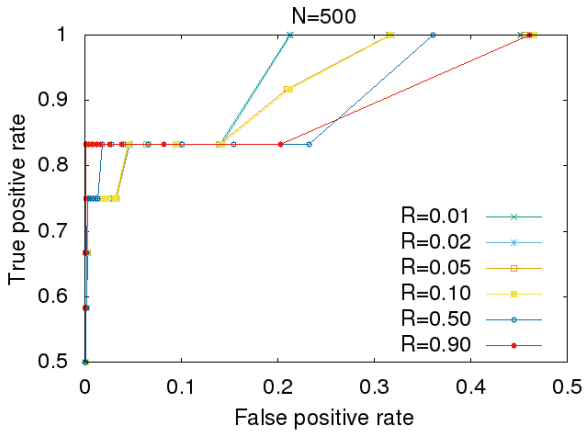
4.1.6 Résultats

Considérant ces paramètres, nous avons obtenu sur le jeu de donnée SynthONTS les courbes ROC présentées sur la figure 4.4. Comme on peut le constater sur la figure 4.4b, la meilleure détection est obtenue pour $N = 500$, $R = 0.90$ et un seuil $\tau_{anomalie}$ de 3.00. Globalement, nous constatons qu'une augmentation de N semble produire de meilleurs résultats. Notamment vis-à-vis du nombre de faux positifs qui s'avère très proches de zéro pour une détection de 83% des attaques. Aussi, de grandes valeurs pour N semblent produire des résultats plus indépendants de R. On constate que diminuer R produit une détection plus sensible, aux prix d'une augmentation du nombre de faux positifs. Le tableau 4.2 expose les résultats obtenus pour l'aire sous la courbe pour différentes valeurs de R et N. Les

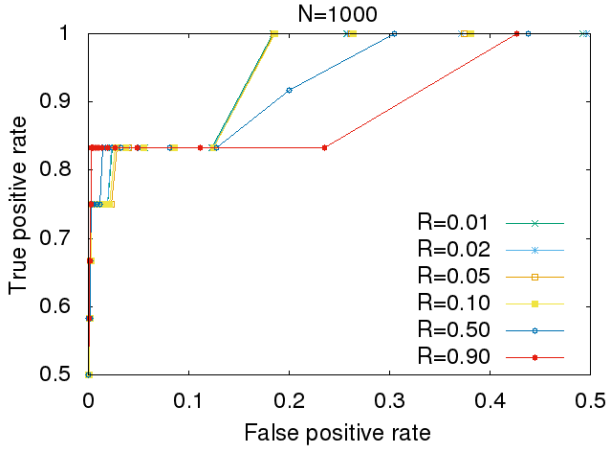
valeurs obtenues, indépendantes du seuil de détection, confirment qu'une augmentation de N produirait globalement de meilleurs résultats, tout comme la réduction de R.



(a) Courbes ROC pour $N = 100$



(b) Courbes ROC pour $N = 500$



(c) Courbe ROC pour $N = 1000$

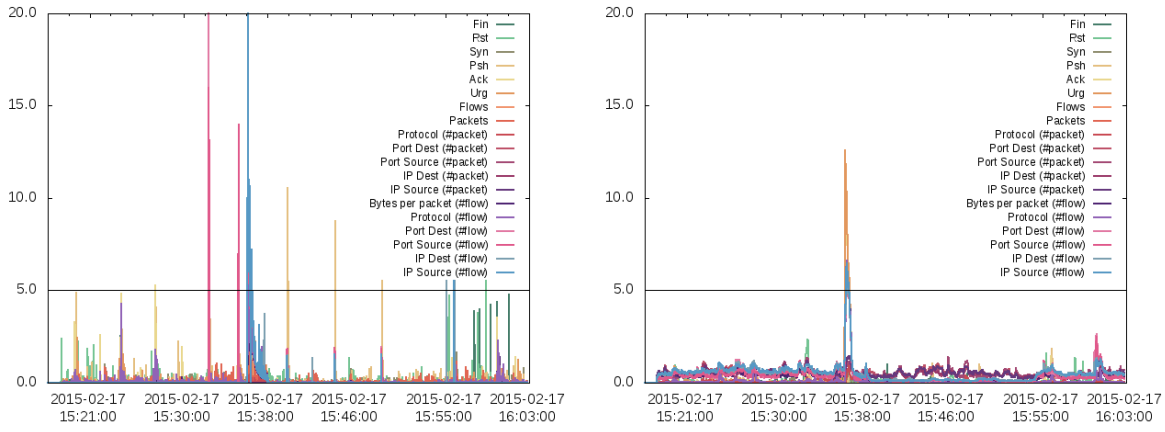
FIGURE 4.4 – Courbes ROC pour plusieurs valeurs de N et R

Néanmoins, comme indiqué précédemment, les courbes ROC souffrent de l'oubli de la fréquence de base. Ce biais est illustré par la figure 4.5 illustrant l'évolution des scores produits par l'algorithme sur l'une de nos évaluations. Comme on peut le constater, la sous-figure 4.5b pour $R = 0.90$ correspond à une détection bien plus acceptable que celle de la sous-figure 4.5a pour $R = 0.05$. En effet, pour une seule anomalie insérée dans le jeu de données, la détection pour $R = 0.05$ produit une proportion de faux positifs inacceptable.

Les courbes d'opération correspondantes, visibles sur la figure 4.6, permettent de répondre aux limites des courbes ROC. Ces courbes sont ici visibles jusqu'à un taux de faux positif maximum de 10^{-2}. On constate ainsi que pour toute valeur de N, ce sont les courbes pour $R = 0.50$ ou $R = 0.90$ qui se situent au-dessus des autres lorsque le taux

R \ N	100	500	1000
0.01	0.9211	0.9668	0.9722
0.02	0.9202	0.9666	0.9721
0.05	0.9197	0.9598	0.9716
0.10	0.9181	0.9595	0.9717
0.50	0.8860	0.9487	0.9639
0.90	0.9083	0.9445	0.9443

TABLE 4.2 – Aire sous la courbe pour les différentes valeurs de R et$N$



(a) $R = 0.05$ (b) $R = 0.90$

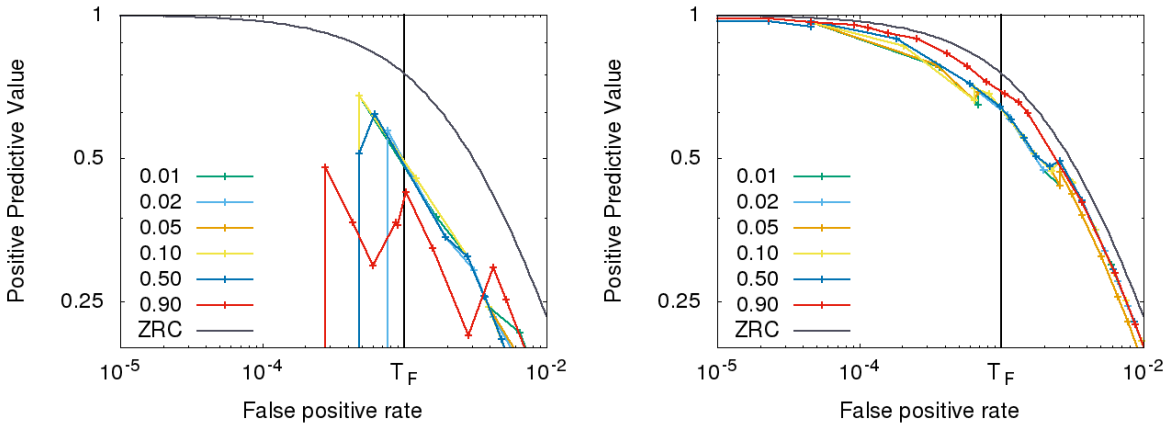
FIGURE 4.5 – Évolution des scores sur SynthONTS

T_{FP}	10^{-2}			10^{-3}			10^{-4}		
R \ N	100	500	1000	100	500	1000	100	500	1000
0.01	0.4286	0.1627	0.1579	0.5520	0.1221	0.1054	1.0	0.0200	0.0163
0.02	0.4794	0.1628	0.1570	0.7772	0.1217	0.1011	1.0	0.0201	0.0183
0.05	0.4278	0.1616	0.1587	0.5602	0.1174	0.1088	1.0	0.0201	0.0185
0.10	0.4414	0.1599	0.1593	0.5602	0.1140	0.1073	1.0	0.0209	0.0178
0.50	0.4372	0.1537	0.1465	0.5930	0.0989	0.0796	1.0	0.0218	0.0205
0.90	0.4636	0.0790	0.1114	0.6973	0.0586	0.0843	1.0	0.0158	0.0216

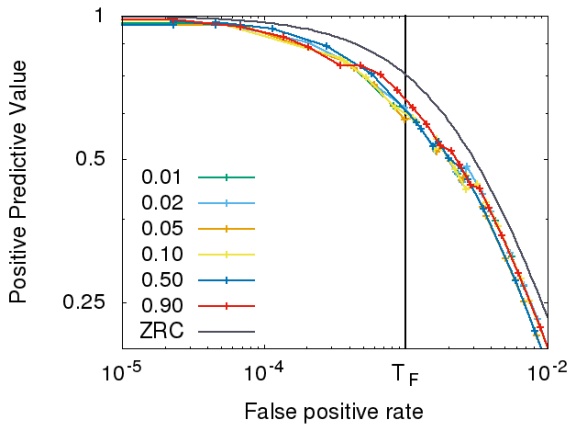
TABLE 4.3 – Efficacité de la détection d'intrusion pour différentes valeurs de N, T_{FP} et R

de faux positifs est en dessous d'une valeur acceptable de 10^{-3}. Aussi, l'augmentation de N semble produire des résultats plus stables. Comme le laissaient envisager les courbes ROC, nous obtenons les meilleurs résultats pour $R = 0.90$ et $N = 500$. Ces observations se traduisent dans les calculs de l'efficacité de la détection présentés dans le tableau 4.3. L'efficacité pour $T_{FP} = 10^{-3}$ ou $T_{FP} = 10^{-4}$ est la meilleure (valeur la plus basse) pour $R = 0.90$ et $N = 500$. On notera cependant que pour $T_{FP} = 10^{-4}$ et $N = 1000$, que c'est une diminution de R qui améliore la qualité de la détection.

L'ensemble de ces résultats s'explique tout d'abord par le fait que R a un impact sur la caractérisation du trafic à plus ou moins long terme. Un R faible a tendance à créer des instantanés du trafic moins corrélés entre eux, ce qui produit une caractérisation à plus court terme, et donc plus sensible à de faibles variations des propriétés du trafic. De manière similaire, augmenter N conduit à considérer relativement plus d'instantanés dans l'application de kNN. Les instantanés étant plus nombreux, la distance au $k^{\text{ième}}$ voisin est plus faible, ce qui par conséquent réduit les scores produits par l'algorithme. Globalement, un équilibre entre ces deux valeurs est alors à trouver afin d'opérer une détection optimale. Un avantage d'AATAC sur la sélection de ses paramètres est notamment dû à la possibilité d'afficher les instantanés du trafic, ce qui permet d'estimer visuellement si la caractérisation du trafic produite apparait comme trop sensible à des changements attendus du trafic. Le dernier paramètre $\tau_{anomalie}$ devient alors une variable d'ajustement, permettant d'adapter la détection aux risques que représente l'occurrence d'une anomalie.



(a) N=100 (b) N=500



(c) N=1000

FIGURE 4.6 – Courbes d'opération pour plusieurs valeurs de N et R

4.2 Évaluation de la capacité du détecteur à opérer en temps réel

4.2.1 Traces capturées

Afin d'évaluer la capacité de l'algorithme à traiter du trafic en temps réel, nous avons décidé d'utiliser du trafic réel. Le projet pour lequel cette thèse a été mené nous a permis de capturer des traces à l'entrée du réseau de l'un des clients de Border 6. Ce réseau supporte notamment plusieurs sites de vente en ligne. Les traces ont été capturées pendant plusieurs semaines, permettant d'obtenir une vue sur le long terme du trafic. Pour des raisons légales, les traces capturées sont limitées aux 64 premiers octets de chaque paquet.

Techniquement, la capture a été réalisée à l'aide de DPDKCap [94], un outil à l'origine développé par Wouter de Vries, alors doctorant de l'Université de Twente. Nous avons amélioré l'outil pour qu'il puisse répondre aux contraintes de notre capture, en implémentant notamment une meilleure gestion des erreurs, la suppression de la charge utile en temps réel ainsi qu'une meilleure gestion des statistiques de captures et du multi-cœurs. Cela nous a permis de répondre aux contraintes de notre campagne de capture, à savoir une capture en continu, sans perte de paquets, sur un trafic pouvant monter jusqu'à 20 Gbit/s en cas de saturation de la bande passante.

Pendant la capture, nous avons eu la chance de pouvoir capturer une attaque par déni de service distribuée. Bien qu'elle ait été atténuée par le fournisseur d'accès à Internet, le début de l'attaque est bien visible dans les traces capturées, permettant d'évaluer la capacité de notre algorithme à opérer en temps réel, même en situation de stress.

4.2.2 Mesure des performances

Pour évaluer les ressources nécessaires à l'exécution de notre algorithme, nous avons décidé d'instrumenter notre implémentation de notre algorithme. Puisque notre implémentation peut opérer sur des traces hors-ligne, nous pouvons mesurer les temps nécessaires à traiter l'équivalent d'une seconde de trafic pour évaluer les ressources consommées en situation réelle. Ainsi, à la création d'un instantané, nous évaluons trois valeurs :

- Le temps nécessaire pour agréger les paquets par flux et en extraire les métriques associées,
- Le temps nécessaire à l'exécution du traitement continu de l'algorithme, c'est-à-dire la mise à jour des densités des cellules,

- Le temps nécessaire à l'exécution du traitement discret de l'algorithme, c'est-à-dire le temps de création d'un instantané.

L'évaluation a été effectuée à l'aide d'une implémentation de l'algorithme en C. L'outil a été exécuté sur une seule machine équipée d'un processeur Intel Xeon CPU (E5-2623 v3) à 3.0 GHz. Bien qu'il soit doté de 8 cœurs, notre implémentation n'en tire parti que partiellement. En effet, seule la partie discrète de l'algorithme est parallélisée : sur ce traitement, les caractéristiques d'instantanés peuvent être traitées indépendamment les unes des autres.

Lors de cette évaluation, notons que, par cohérence, nous avons surveillé les même caractéristiques que pour l'évaluation sur SynthONTS. Cela nous permet d'affirmer que les conditions pour obtenir à la fois une détection correcte et une opération en temps réel sont les mêmes. Pour cette évaluation, nous avons fait varier les paramètres $R \in \{0.05, 0.90\}$, $\Delta T \in \{1, 5, 10, 30, 60\}$ et $N \in \{100, 1000, 5000\}$. Le paramètre $\tau_{anomalie}$ n'ayant aucun impact sur le temps de calcul, il a été ignoré.

4.2.3 Résultats

Les résultats que nous avons obtenus sont présentés dans le tableau 4.4. Nos études ayant montré que les paramètres R et ΔT ont un impact complètement négligeable sur le temps de calcul de l'algorithme, les résultats sont affichés comme moyennés suivant ces paramètres. Tout d'abord, l'extraction des flux s'avère être le traitement le plus long opéré, avec un temps moyen de calcul de $0.58s$ pour une seconde de trafic. Néanmoins, on ne peut pas vraiment considérer ce traitement comme faisant partie de l'algorithme, puisqu'il est en général exécuté par les équipements réseaux. Nous constatons que pour le traitement continu, puisqu'il atteint au maximum une valeur de $0,89s$ pour une seconde traitée, ce traitement peut être considéré comme réalisé en temps réel. En moyenne, ce traitement nécessite aux alentours de $0,24$ seconde pour traiter 1 seconde de trafic, ce qui laisse envisager qu'il serait possible de multiplier par quatre le nombre de caractéristiques surveillées. Quant au traitement discret, les valeurs obtenues apparaissent complètement négligeables. On constate une légère augmentation du temps de calcul, mais qui reste limitée malgré les 5000 instantanés considérés. Nous expliquons l'aspect limité de cette augmentation par le fait que notre implémentation de kNN est optimisée à notre cas d'utilisation : à la création d'un instantané, elle se contente en effet de calculer la distance du dernier instantané créé vers tous les autres (et non pas de tous vers tous). Cela limite la complexité de l'application de kNN, qui passe de $O(N^2)$ à $O(N)$.

Comme nous pouvons le constater, même en cas d'attaque, le détecteur nécessite moins

N	100	1000	5000
Extraction des flux (moyen pour 1s de trafic)	0.58	0.57	0.59
Continu (max pour 1s de trafic)	0.80	0.79	0.89
Continu (moyen pour 1s de trafic)	0.21	0.25	0.26
Discret	0.028	0.027	0.033

TABLE 4.4 – Temps de calcul en secondes

d'une seconde pour traiter l'équivalent d'une seconde de trafic. On peut donc assurer que l'algorithme peut opérer en temps réel.

4.3 Comparaison avec FastNetMon et ORUNADA

Afin d'obtenir une évaluation juste d'AATAC, nous avons comparé les résultats obtenus avec deux autres détecteurs, à savoir FastNetMon [95] et ORUNADA [65].

FastNetMon est l'un des rares détecteurs d'anomalies disponibles que nous avons pu installer. Ce détecteur, spécialisé dans la détection des attaques DDoS/DoS, répond en priorité à des contraintes industrielles. L'outil est capable d'opérer sur des exports SFlow, Netflow ou directement sur une interface. Il est entièrement configurable, notamment pour ce qui tient à la notification de l'administrateur. Malheureusement pour nous, il n'a pas été possible de le faire fonctionner directement sur des fichiers de capture, nous avons donc à la place rejoué les fichiers sur une interface virtuelle (à l'aide de TCPReplay [96]) sur laquelle FastNetMon opérait. Aussi, nous ne pouvons pas ici mesurer le temps nécessaire au traitement d'un seconde de trafic, nous comparons pour cette évaluation l'utilisation des ressources CPU moyennes lors de l'utilisation de chaque détecteur.

FastNetMon est représentatif des approches reposant sur nos connaissances des attaques, il utilise simplement des seuils sur le nombre de paquets par seconde ou le débit pour des protocoles souvent utilisés pour des attaques DDoS (TCP pour les *SYN flooding*, UDP pour les *UDP flooding* et ICMP pour les *Fraggle attacks*). Ce sont ces seuils que nous faisons varier pour notre évaluation.

Quant à ORUNADA, c'est la dernière version d'un autre détecteur d'anomalie construit par notre équipe. Ce détecteur non-supervisé, reposant sur du clustering de sous-espaces, opère une détection capable de corréler de nombreux attributs du trafic, Il vise une opération en parallèle sur de nombreuses machines.

À la suite de notre évaluation, ORUNADA a été capable, à son meilleur point d'opération, de détecter l'ensemble des anomalies contenues dans le jeu de données SynthONTS, tout en ne générant pas de faux positifs. Néanmoins l'algorithme a nécessité une grande quantité de ressources de calcul pour opérer : en moyenne, l'exécution de l'algorithme a

nécessité l'équivalent de 220% de la puissance d'un cœur du CPU pour opérer en temps réel. FastNetMon, quant à lui, a opéré une détection en temps réel avec une charge à 60%. Il n'a néanmoins pu détecter dans le jeu de données que 6 des 12 anomalies présentes, et a levé un faux positif. Avec les meilleurs paramètres obtenus dans notre précédente évaluation, AATAC a opéré, sur du trafic en temps réel, avec un seul cœur à 21% de charge, et pour rappel avec une détection de 83% des anomalies, et un *TFP* de 0.13%.

En conclusion de cette évaluation, nous pouvons assurer qu'AATAC est un détecteur efficace permettant un compromis entre la qualité de la détection et les ressources de calcul utilisées. La principale surprise de cette évaluation vient des performances de FastNetMon, qui s'avère plutôt lent pour la technique de détection utilisée. Il est probable que l'ensemble de ses fonctionnalités répondant aux besoins des industriels ait un impact sur les performances de l'implémentation.

4.4 Conclusion

L'évaluation d'AATAC que nous avons menée nous a permis de prouver la capacité de notre algorithme à fonctionner en temps réel, sur du trafic capturé, tout en produisant une détection d'une qualité acceptable. Sur le jeu de données SynthONTS, AATAC a détecté jusqu'à 80% des attaques pour un taux de faux positifs très acceptable de 0.13%. Sur les traces capturées, l'algorithme a opéré une détection complètement temps réel, même en cas d'attaque. Comparé à FastNetMon et ORUNADA, AATAC apparait être un compromis vis-à-vis de la qualité de la détection, mais nécessite moins de ressources de calcul que chacun des deux autres détecteurs.

Chapitre 5

Impact de l'échantillonnage sur les performances d'AATAC

Sommaire

L'échantillonnage est une technique relativement commune permettant de limiter la quantité de données à traiter pour opérer la détection. Il consiste à sélectionner, de façon dynamique, un sous-ensemble des données d'entrée. Dans notre domaine, l'échantillonnage consiste à sélectionner certains paquets dans le trafic pour opérer la détection. Plusieurs techniques d'échantillonnage existent, mais toutes ont un impact différent sur les performances de la détection. En effet, si l'objectif principal de l'échantillonnage est de limiter les ressources de calcul nécessaires à l'opération de l'algorithme, son impact sur la qualité de la détection et la pertinence des données exposées à l'administrateur ne sont pas négligeables.

Si l'échantillonnage a plusieurs avantages, le fait est qu'il est surtout nécessaire pour des raisons purement opérationnelles. Dans un conteste industriel, surveiller le trafic est une nécessité. Cette surveillance nécessite l'installation de plusieurs sondes réparties sur

le réseau, collectant des informations en différents points. S'il est possible d'utiliser des équipements dédiés à ce travail, ce n'est une solution réaliste que sur des réseaux de petite taille. Lorsque la complexité du réseau ou les débits augmentent, cette solution devient difficile à opérer et coûteuse à mettre en place. C'est pourquoi sur les architectures modernes ce sont les commutateurs et routeurs eux-mêmes qui collectent et exportent ces données. Le fait est que la collecte d'information de surveillance n'est pas la tâche principale de ces équipements, les capacités de calcul allouées à ce traitement sont par conséquent limitées. Lorsque l'analyse du contenu des paquets est utile, utiliser de l'échantillonnage est alors une nécessité. Cette nécessité est d'autant plus flagrante lorsque l'analyse doit être centralisée, regrouper l'ensemble du trafic vers un seul point n'étant par une solution réaliste. Le protocole de rapatriement utilisant de l'échantillonnage sFlow [1] est d'ailleurs un standard le l'industrie.

Par conséquent, si l'échantillonnage n'est pas nécessaire au bon fonctionnement d'un détecteur (et a fortiori d'AATAC, compte tenu de l'évaluation réalisée dans le chapitre précédent), il est une réalité de l'industrie à prendre en compte dans l'évaluation de notre algorithme. Nous devons donc évaluer l'impact qu'aurait un échantillonnage sur ses performances.

Pour ce faire, nous avons utilisé à nouveau le jeu de données SynthONTS. Nous avons mesuré l'impact de l'échantillonnage sur la qualité de la détection ainsi que le gain en termes de performance apporté par l'échantillonnage. Différentes techniques d'échantillonnages ont été utilisées, et à différent taux.

Dans ce chapitre, nous présentons en premier lieu dans la section 5.1 les différentes techniques d'échantillonnage existantes et expliquons le choix des techniques que nous avons utilisées pour notre évaluation. Ensuite, la section 5.2 décrit le protocole d'évaluation plus précisément. Les résultats de l'évaluation son présentés dans la section 5.3. Enfin, la section 5.4 résume le chapitre.

5.1 Techniques d'échantillonnage

La sélection d'un paquet peut être effectuée de différentes façons. Les méthodes les plus simples sélectionnent les paquets indépendamment de leur contenu et des conditions du réseau. Ces approches peuvent opérer en comptant le nombre de paquets entre chaque sélection (chaque paquet est alors choisi avec une probabilité constante au fil du temps), ou en fonction du temps. Le taux d'échantillonnage (par paquet ou par unité de temps) est fixe avec de telles approches.

Certaines approches [97, 98], dites adaptatives, proposent d'améliorer ces techniques en utilisant un taux d'échantillonnage variable. Celui-ci est alors adapté en fonction de la

charge du système, rendant la détection plus robuste à des surcharges inattendues (en cas d'attaque par exemple). Ces approches peuvent utiliser le nombre de paquets traités par seconde, mais aussi d'autres indicateurs de performance, comme le taux d'utilisation du CPU par exemple.

Enfin, certaines techniques (par exemple [99]), dites de filtrage, proposent d'assigner à chaque paquet une probabilité différente d'être échantillonné, en fonction de son contenu.

Ceci étant dit, dans les faits, les techniques reposant sur de la sélection aléatoire sont les plus répandues. Il y a plusieurs raisons à cela. Tout d'abord, si les approches adaptatives sont intéressantes, elles nécessitent un retour d'information entre le traitement complexe effectué en aval et l'étape d'échantillonnage. Dans un contexte industriel, ces deux équipements sont généralement séparés, ce qui rend la mise en place de tels mécanismes complexe pour un gain estimé comme insuffisant.

Si les techniques de filtrage de paquet peuvent nécessiter un mécanisme similaire, elles ont aussi tendance à être plus complexes. En effet, déterminer une probabilité de sélection par paquet nécessite d'évaluer plusieurs de ses caractéristiques. Certaines approches vont jusqu'à utiliser du Deep Packet Inspection (DPI) pour déterminer cette probabilité. Le principal apport de l'échantillonnage étant la réduction des ressources de calcul utilisées, il serait dommage d'utiliser une technique de sélection utilisant trop de ressources au point d'en être inutile.

Dans notre cas, le caractère non-supervisé de notre détecteur limite d'autant plus les techniques d'échantillonnage que nous pouvons utiliser. En effet, les approches par filtrage nécessitent une forme de connaissance permettant de déterminer quels paquets devraient être échantillonnés ou non. Or, pour opérer leur détection, les approches non-supervisées nécessitent la création d'une caractérisation fidèle du trafic. Influencer les paquets fournis au détecteur à l'aide d'une base de connaissance biaiserait donc la caractérisation produite, ce qui va à l'encontre du principe de la détection non-supervisée.

Les approches adaptatives, si elles ont un véritable intérêt en production, sont moins intéressantes pour une évaluation des performances de l'algorithme, puisqu'elles prennent le contrôle du paramètre du taux d'échantillonnage. Comme c'est un paramètre important, nous avons préféré le fixer manuellement, nous permettant de tester plusieurs valeurs et d'évaluer les performances de l'algorithme en fonction du taux d'échantillonnage.

Pour notre évaluation, nous nous sommes donc focalisés sur des approches à taux d'échantillonnage fixe, choisissant les paquets indépendamment de leur contenu.

5.2 Protocole d'évaluation

5.2.1 Techniques d'échantillonnage utilisant le nombre de paquets

Nous avons utilisé trois techniques d'échantillonnage reposant sur le nombre de paquets, à savoir : l'*échantillonnage systématique*, l'*échantillonnage n-parmi-N* et l'*échantillonnage probabiliste*.

L'échantillonnage systématique est le plus simple, il consiste à échantillonner systématiquement un paquet tous les N paquets. L'intervalle entre deux sélections est donc toujours de $N-1$ paquets. Il est à noter que cette technique peut souffrir d'un biais lorsque qu'une caractéristique du trafic montre un comportement cyclique.

L'échantillonnage *n-parmi-N* consiste à choisir tous les N paquets n indices compris entre 1 et N. Ces indices sont alors utilisés pour sélectionner les n parmi N prochains paquets. Dans notre cas, n est choisi à 1. Comme cet indice est choisi aléatoirement, cette technique ne souffre pas du biais de l'approche systématique. Notons que pour ces deux premières approches, le taux d'échantillonnage est de $1/N$.

L'échantillonnage probabiliste consiste à effectuer un test aléatoire à l'arrivée de chaque paquet. Le paquet est échantillonné avec une probabilité p donnée en fonction du résultat du test. Cette technique est probablement la plus évidente, néanmoins lorsque p est bas, elle fait qu'il existe une probabilité non nulle qu'aucun paquet ne soit échantillonné pendant un laps de temps assez long. Par conséquent, une limite de paquets non échantillonnés successifs est souvent utilisée. Pour cette technique, le taux d'échantillonnage est donc $\frac{1}{N} = p$.

Pour ces trois approches reposant sur le comptage de paquets, nous avons modifié les paramètres de ces approches (p ou N) pour obtenir les taux d'échantillonnage suivants : 1/500, 1/1000, 1/2000, 1/5000 et 1/10000. D'après Border 6, une valeur de 1/2000 est une valeur habituelle utilisée pour la surveillance des réseaux. Les valeurs pour notre évaluation ont donc été choisies en conséquence.

5.2.2 Techniques d'échantillonnage temporel

L'échantillonnage temporel raisonne, non pas sur le nombre de paquets présentés au système mais sur l'intervalle temporel entre deux sélections. Par conséquent, ces approches peuvent réguler le nombre de paquets par seconde sélectionnés, ce qui peut être un avantage pour stabiliser la consommation de ressources utilisées par le détecteur. Nous avons utilisé deux types de techniques d'échantillonnage temporel : *systématique* et *aléatoire*.

L'approche systématique utilise un intervalle constant entre deux sélections de paquets. De façon comparable à l'approche systématique par nombre de paquets, cette approche peut souffrir d'un biais si le trafic montre un motif cyclique.

L'approche aléatoire, comme son nom l'indique, consiste à générer, à chaque sélection, un intervalle d'attente avant la prochaine sélection. Dans notre cas, cet intervalle δt suit une distribution exponentielle, soit $P(\delta t = t) = \lambda e^{-\lambda t}$. La sélection des paquets suit ainsi un processus de Poisson. Le temps moyen d'attente entre deux paquets est alors $\mathbb{E}(\delta t) = \frac{1}{\lambda}$.

Pour ces deux approches, nous avons choisi les paramètres de manière à obtenir un intervalle moyen entre deux sélections ayant les valeurs suivantes : $1ms$, $10ms$, $100ms$ et $1s$. Cette plage se veut assez large pour évaluer le comportement de l'algorithme dans des conditions plus extrêmes.

5.3 Résultats de l'évaluation sur du trafic échantillonné

5.3.1 Impact sur la qualité de la détection

Les courbes ROC que nous avons obtenues pour les cinq techniques d'échantillonnages sont présentées sur la figure 5.1. Les courbes ROC sont limitées sur l'abscisse à un TFP de 0.01. Comme nous pouvons le constater, les résultats sont bons pour l'ensemble des techniques reposant sur le nombre de paquets. Pour les taux d'échantillonnage supérieurs à 1/2000, la qualité de la détection reste sensiblement la même pour un TFP inférieur à 0.002. Au-delà de cette valeur, le TVP est même meilleur pour le trafic échantillonné que sur le trafic brut. Néanmoins, pour un taux de faux positifs proche de zéro, il semblerait que le détecteur reste plus performant sur du trafic non-échantillonné. Cette tendance est similaire pour l'échantillonnage temporel probabiliste, notamment pour les taux d'échantillonnage assez hauts (pour δt égal à $1ms$ ou $10ms$). L'approche systématique temporelle semble produire des résultats très surprenants, puisque meilleurs que le trafic brut sur presque toute la courbe (sauf pour un TFP de zéro, même si c'est difficile à constater sur la figure).

Ces résultats sont corroborés par ceux proposés par les courbes d'opération présentées sur la figure 5.2. Les différentes valeurs calculées de l'efficacité de la détection correspondant à ces courbes sont présentées sur la figure 5.1. Pour des taux de faux positifs inférieurs à 10^{-3}, l'efficacité de la détection s'avère très peu sensible à l'échantillonnage. Seul l'échantillonnage par paquet probabiliste semble décrocher pour un taux d'échantillonnage à 1/10000, ainsi que pour les taux d'échantillonnage à 1 paquet/s ou 10 paquets/s pour l'échantillonnage temporel. Le biais de l'oubli de la fréquence de base n'étant pas présent sur ces courbes, on constate que la détection sur du trafic non-échantillonné montre en général de meilleures performances. Aussi, on peut constater qu'AATAC obtient de très

T. E.* / T_{FP}	Non échantillonné					Par paquet systématique				
						1/500	1/1000	1/2000	1/5000	1/10000
0.01	0.1543					0.0853	0.1023	0.1197	0.1051	0.1568
0.001	0.0680					0.1026	0.0819	0.0789	0.0958	0.1053
0.0001	0.0155					0.2602	0.0208	0.0163	0.0263	0.0256
	Par paquet probabiliste					1-out-of-N				
	1/500	1/1000	1/2000	1/5000	1/10000	1/500	1/1000	1/2000	1/5000	1/10000
0.01	0.0758	0.0822	0.0909	0.1285	0.2127	0.0804	0.0903	0.0968	0.1371	0.1204
0.001	0.0661	0.0649	0.1014	0.1092	0.2328	0.0910	0.0804	0.0851	0.0902	0.0899
0.0001	0.0232	0.0136	0.0276	0.0390	0.0610	0.0286	0.0197	0.0274	0.0351	0.0654

T. E.* / T_{FP}	Systematic time-based				Random time-based			
	1ms	10ms	100ms	1s	1ms	10ms	100ms	1s
0.01	0.0631	0.0394	0.2959	0.8424	0.0851	0.0939	0.2954	0.2402
0.001	0.2308	0.0242	0.2113	1.0000	0.1364	0.1127	0.1580	0.1607
0.0001	0.8993	0.0101	0.1238	1.0000	0.2495	0.0207	0.0274	0.0927

TABLE 5.1 – Efficacité de la détection d'intrusion pour les différentes *techniques d'échantillonnage** et plusieurs valeurs pour T_{FP}.

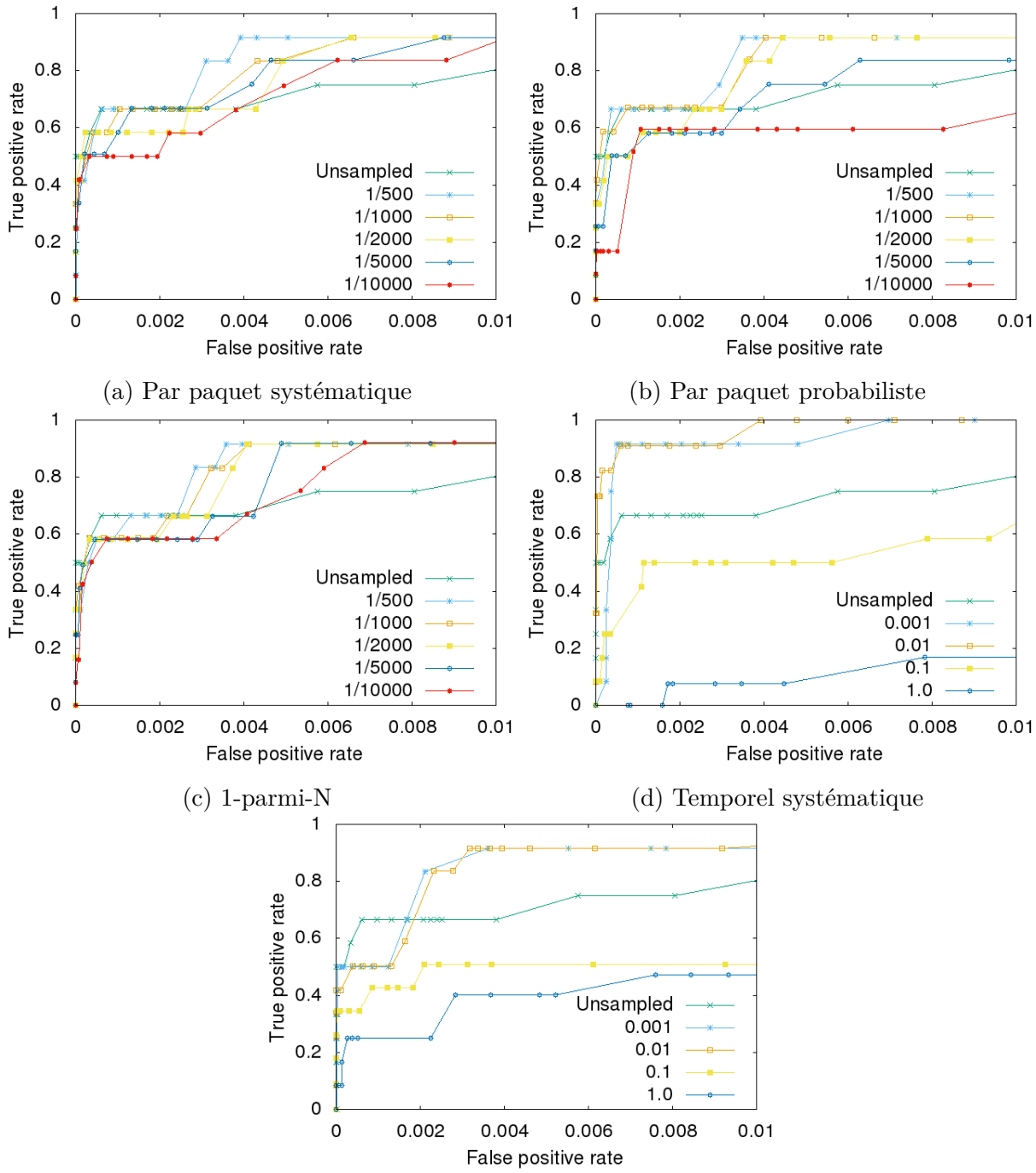
bons résultats pour l'échantillonnage temporel à haut taux.

Comme mentionné plus tôt, l'échantillonnage temporel a tendance à lisser les propriétés du trafic. En effet, ces techniques proposent d'échantillonner à intervalles réguliers un paquet, peu importe le nombre de paquets par seconde dans le trafic original. Par conséquent, cette approche peut rendre le détecteur moins sensible à des changements soudains dans les propriétés du trafic, ce qui a visiblement amélioré la détection pour les intervalles d'échantillonnage courts. Néanmoins, ce résultat est à pondérer avec le fait que, si l'on considère le nombre de paquets par seconde dans le trafic original de $67k$ paquets par seconde, un paquet toutes les $1ms$ ou $10ms$ revient à des taux d'échantillonnage de respectivement 1/67 ou 1/670. Ces taux sont relativement élevés et ne correspondent pas véritablement à un taux d'échantillonnage réaliste. On constate que pour un intervalle de $100ms$, soit un taux équivalent d'environ 1/6700, les techniques d'échantillonnage temporel montrent une efficacité moins bonne que l'échantillonnage par paquet à 1/5000.

5.3.2 Impact sur les ressources de calcul

La figure 5.3 illustre le calcul moyen du temps nécessaire à l'algorithme pour opérer sur le trafic. Comme dans le chapitre précédent, les valeurs sont, pour la partie discrète de l'algorithme, calculées pour la création d'un instantané, alors que pour la partie continue, il s'agit du temps nécessaire au traitement d'une seconde de trafic.

On peut constater que l'impact sur le traitement continu est linéaire. Cette observation était attendue puisque la partie continue d'AATAC a été conçue de manière à avoir une complexité linéaire par rapport au nombre d'instances. Le gain en temps de calcul est donc



(a) Par paquet systématique

(b) Par paquet probabiliste

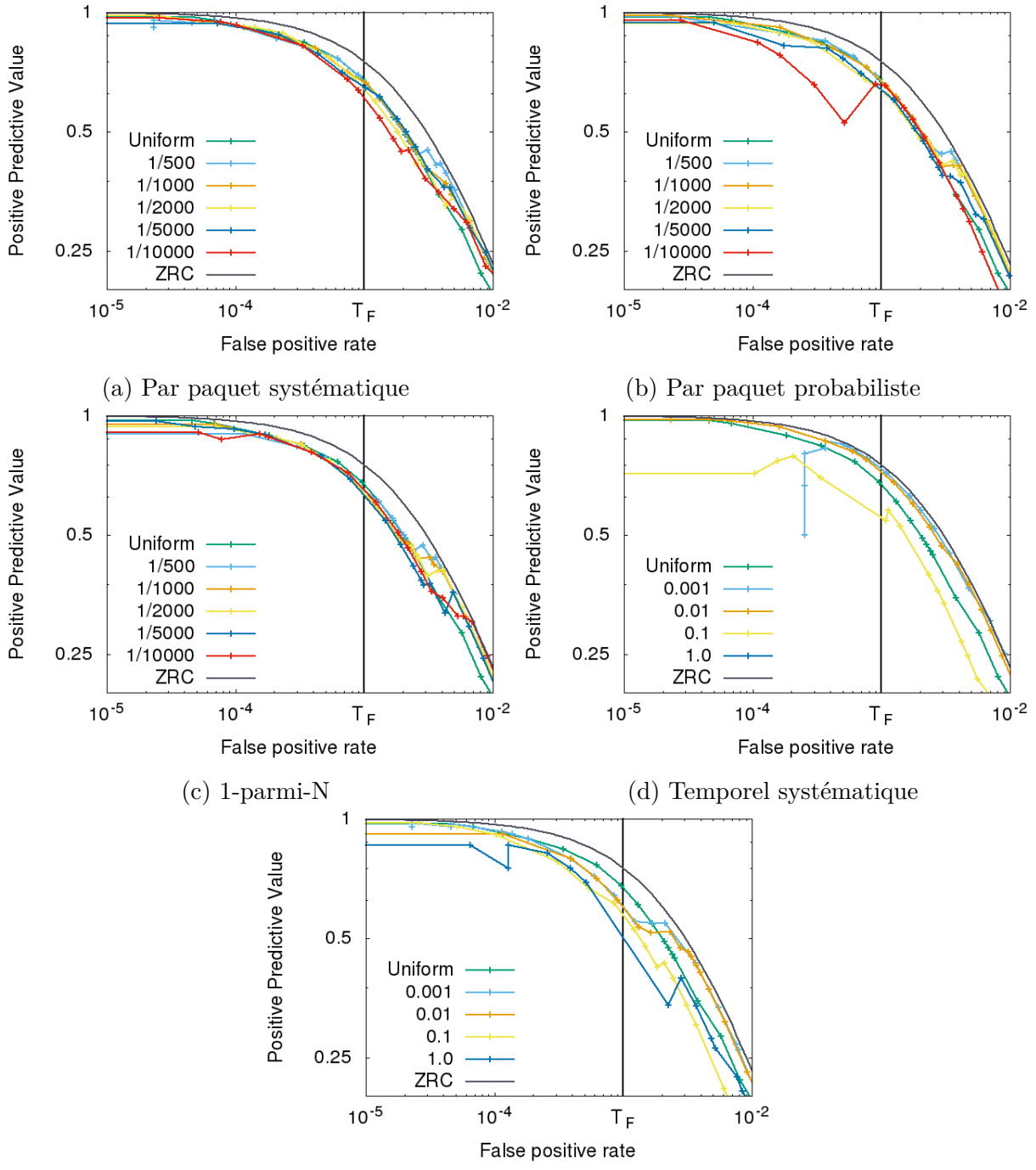
(c) 1-parmi-N

(d) Temporel systématique

(e) Temporel aléatoire

FIGURE 5.1 – Courbes ROC pour différents types et taux d'échantillonnage.

directement proportionnel au nombre de paquets par seconde en entrée de l'algorithme. Le choix de la technique d'échantillonnage n'a visiblement pas d'impact sur le temps nécessaire au traitement.



(a) Par paquet systématique

(b) Par paquet probabiliste

(c) 1-parmi-N

(d) Temporel systématique

(e) Temporel aléatoire

FIGURE 5.2 – Courbes d'opérations pour différents types et taux d'échantillonnage.

Sur le traitement discret, on constate que l'échantillonnage n'a que très peu d'impact sur le temps de calcul. Il reste relativement constant, peu importe le taux d'échantillonnage utilisé. On constate néanmoins une augmentation d'environ 18% pour l'échantillonnage à

Type d'échantillonnage	Continu (max)	Continu (variance)	Discret (max)	Discret (variance)
Temporel systématique (0.01s)	0.001597	$4.774e^{-05}$	0.098	0.002059
Temporel aléatoire (0.01s)	0.001034	$5.816e^{-05}$	0.105	0.001872
Par paquet 1-parmi-N (1/500)	0.001614	$7.184e^{-05}$	0.099	0.001880
Par paquet probabiliste (1/500)	0.001549	$8.421e^{-05}$	0.102	0.001633
Par paquet systématique (1/500)	0.001817	$7.382e^{-05}$	0.102	0.001675

TABLE 5.2 – Temps de traitement (maximum et variance) pour différentes techniques d'échantillonnage mais des taux similaires

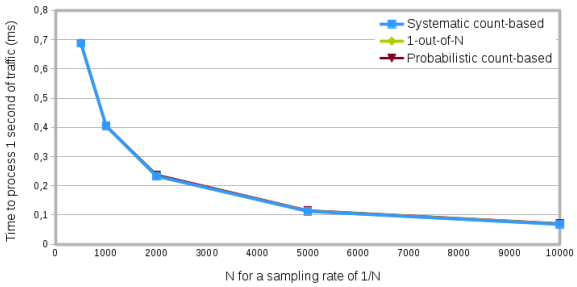
1/10000, qui est probablement due à la nature des instantanés créés. Comme indiqué précédemment, le poids de chaque paquet dans le modèle est pondéré par le taux d'échantillonnage. Par conséquent, à ce taux d'échantillonnage très faible, chaque paquet a un impact très important sur l'instantané créé. Les instantanés créés dépendent véritablement de quelques paquets seulement, ce qui les rend plus complexes, leurs propriétés n'étant pas lissées par un grand nombre de paquets. Cette complexité a, à priori, un impact sur le traitement discret, qui nécessite alors plus de temps de calcul.

Pour compléter notre étude, nous avons aussi évalué le maximum et la variance des temps mesurés. Ces valeurs, pour des taux d'échantillonnage comparables (1/500 et environ 1/670 pour un intervalle de $0.01s$), sont exposés dans le tableau 5.2. Ces mesures nous montrent si certaines techniques d'échantillonnage permettent d'obtenir une consommation de ressources plus stable. Bien que l'on aurait pu imaginer que l'échantillonnage temporel conduise à moins de variations dans les ressources utilisées, on constate que, globalement, la variance reste assez indépendante de la technique d'échantillonnage utilisée. De même, le temps de calcul maximal obtenu pour une seconde de trafic (pour la partie continue) ou pour la création d'un instantané (pour la partie discrète) ne sont pas influencés par le choix de la technique d'échantillonnage.

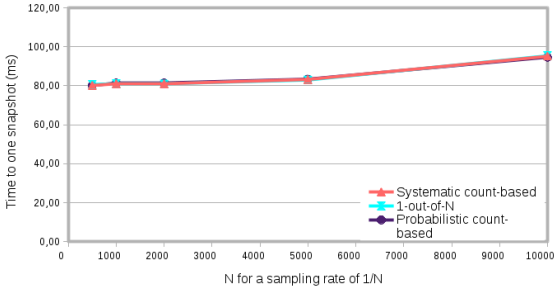
On retiendra de cette évaluation que le taux d'échantillonnage a un impact très significatif sur le temps de traitement continu, mais assez peu sur le traitement discret. Aussi, la technique d'échantillonnage utilisée n'a aucun impact significatif sur la variabilité des ressources consommées.

5.3.3 Comparaison avec FastNetMon

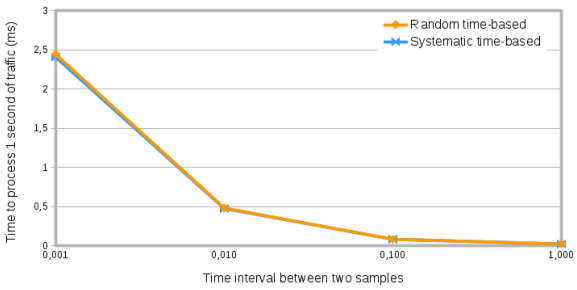
Pour comparer nos résultats avec ceux obtenus à l'aide d'un autre détecteur, nous avons à nouveau utilisé FastNetMon. L'évaluation étant plus longue que pour AATAC



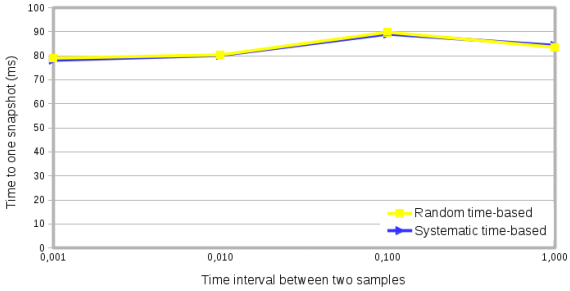
(a) Traitement continu : temps nécessaire à traiter 1s de trafic pour l'échantillonnage par paquet



(b) Traitement discret : temps nécessaire pour la production d'un instantané pour l'échantillonnage par paquet



(c) Traitement continu : temps nécessaire à traiter 1s de trafic pour l'échantillonnage temporel



(d) Traitement discret : temps nécessaire pour la production d'un instantané pour l'échantillonnage temporel

FIGURE 5.3 – Temps de traitement suivant la technique et le taux d'échantillonnage

(FastNetMon n'ayant pu être adapté pour opérer sur des fichiers de capture), nous nous sommes limités à une évaluation sur du trafic échantillonné par la technique *1-parmi-N* à un taux d'échantillonnage de 1/2000. Les paramètres de l'algorithme ont été adaptés en conséquence.

Afin d'obtenir les meilleurs résultats possibles, nous avons fait varier le seuil de détection. Malheureusement, dans sa meilleure situation, FastNetMon n'a détecté aucune des 12 attaques contenues dans le jeu de données mais a produit 7 faux positifs. La réduction du seuil n'ayant produit qu'un nombre croissant de faux positifs.

En termes de ressources nécessaires à l'exécution, les résultats obtenus ont été plutôt bons puisque la charge du processeur a été réduite à moins de 1% soit une charge complètement négligeable.

AATAC est plus robuste à l'utilisation sur des données échantillonnées que FastNetMon, il est donc probablement une solution plus réaliste dans un contexte industriel.

5.4 Résumé

Lors de cette seconde évaluation, nous avons pu constater la capacité d'AATAC à opérer sur du trafic échantillonné. Même lorsque le taux d'échantillonnage est très fort (jusqu'à 1/5000), nous constatons que la qualité de la détection n'est pas significativement réduite. L'échantillonnage temporel, lorsque appliqué à très fort taux, a même produit des résultats, sur certains points, meilleurs que sur le trafic non échantillonné. Au-delà de ça, l'échantillonnage est particulièrement efficace sur le temps de calcul de l'algorithme, puisqu'il permet une réduction significative des ressources de calcul nécessaires à l'opération. Cette réduction est d'ailleurs directement proportionnelle au taux d'échantillonnage utilisé.

Chapitre 6

Conclusion

Sommaire

Le problème de la détection d'anomalies est complexe. La diversité et l'évolution constante des réseaux, conjointement à la multiplication des anomalies ont conduit les chercheurs à expérimenter des approches plus flexibles, capables de s'adapter de manière autonome au trafic surveillé et à ses évolutions.

En effet, le problème de la détection d'anomalie dans les réseaux a été souvent abordé dans la littérature. Nous avons fait état de nombreuses méthodes de détection proposées, que nous avons choisi dans cette thèse de classifier en fonction de leur capacité à opérer de manière autonome. Nous distinguons trois catégories :

Les systèmes experts. En général, ces approches utilisent un ensemble de signatures, dont chacune est conçue pour détecter un type d'attaque précis. Si ces approches peuvent fournir des informations précises sur les anomalies détectées, elles nécessitent un lourd travail de mise à jour lorsque le trafic évolue, et d'adaptation des signatures au trafic surveillé. Le coût de ce travail est conséquent, rendant ces approches peu intéressantes.

Supervisées. Ces approches reposent sur un modèle générique du trafic, à partir duquel est instancié un modèle adapté au trafic spécifique à surveiller. Cette spécialisation nécessite un entraînement sur du trafic labellisé. Ces approches, bien que plus autonomes que celles nécessitant des signatures, nécessitent aussi un important travail de construction du jeu de données labellisées. Ce jeu de données doit être représentatif et assez varié, ce qui rend sa construction difficile.

Non-supervisées. Les approches non-supervisées reposent sur l'hypothèse que le trafic anormal est rare et statistiquement différent du trafic normal. Ces approches opèrent directement sur le trafic en production, elles construisent une caractérisation du trafic habituel et détectent d'éventuelles déviations comme anormales. Ces approches ne nécessitent qu'un travail de configuration limité et ne requièrent le retour de l'administrateur qu'à postériori. Néanmoins, construire une caractérisation pertinente et exhaustive du trafic sans connaissances préalables nécessite généralement d'analyser de nombreuses caractéristiques du trafic et de les comparer entre elles. Cette analyse est souvent algorithmiquement complexe, ce qui rend l'usage de telles approches difficile en temps réel.

Si les systèmes experts et les approches supervisées ont leurs avantages, ce sont les approches non-supervisées qui proposent la détection la plus autonome. Elles simplifient grandement le travail de l'administrateur mais ont un coût algorithmique conséquent. Pour profiter à la fois d'une détection autonome, mais reposant sur un algorithme pouvant être exécuté en temps réel sur des équipements aux capacités de calculs contraintes, nous souhaitions, dans le cadre de cette thèse, explorer de nouvelles solutions. Plus spécifiquement, notre problématique se divise en plusieurs points :

- Proposer une détection efficace, à la fois sur du trafic complet et échantillonné. L'algorithme se devait d'être notamment efficace pour la détection des attaques par déni de service distribuées, puisque ce sont des menaces particulièrement grandes pour les acteurs industriels.
- Proposer une détection autonome, nécessitant peu d'intervention de la part de l'administrateur réseau. Son travail étant un coût non négligeable de la détection, il est à limiter autant que possible.
- Proposer une détection robuste, capable d'opérer en temps réel avec des ressources de calcul limitées.
- Proposer une détection fournissant à l'administrateur des résultats simples à interpréter, facilitant le diagnostic des anomalies détectées.

Les solutions proposées jusqu'à présent ne répondaient pas parfaitement à notre problématique. Si certaines approches non-supervisées visent une détection en temps réel c'est souvent au prix d'une caractérisation plus simple et des résultats plus difficiles à interpréter (certaines approches reposant sur un calcul de l'entropie ou CUSUM en sont des exemples). En conséquence, nous souhaitions proposer un nouvel algorithme capable de réaliser une détection non-supervisée efficace, capable d'opérer en temps réel, tout en proposant des résultats simples à analyser pour simplifier le travail de l'administrateur.

Ce constat nous a conduit à la conception et l'évaluation, dans des conditions réalistes, du nouvel algorithme nommé AATAC. Sa conception repose sur différents principes :

- Une détection non-supervisée, permettant une détection en grande partie autonome.
- Sa conception en deux parties (traitement continu et traitement discret) isole le traitement le plus exigeant en ressources de calcul pour ne l'exécuter qu'à intervalles réguliers. Cela limite au maximum les ressources nécessaires à l'exécution de l'algorithme. Notre évaluation a confirmé la capacité d'AATAC à traiter le trafic en temps réel, même lorsqu'une attaque conséquente est présente dans le trafic.
- La création à intervalles réguliers d'instantanés permet d'obtenir une vue dynamique de l'état du trafic à l'instant de l'anomalie. La possibilité de présenter ces informations sous la forme de graphiques permet un diagnostic simplifié de l'anomalie.

Les performances de l'algorithme, incluant notamment la qualité de la détection et de sa capacité à opérer en temps réel, ont été évaluées. Cela a alors nécessité la construction de deux jeux de données. Le premier, nommé SynthONTS, a servi à l'évaluation de la qualité de la détection de l'algorithme. Le deuxième consiste en un ensemble de traces capturées à l'entrée du réseau de l'un des clients de Border 6, et a servi à l'évaluation de la capacité de l'algorithme à opérer en temps réel, en condition d'opération.

Cette évaluation a été complétée par une évaluation de l'impact de différentes techniques d'échantillonnage et à différents taux sur les performances d'AATAC. L'impact sur la qualité de la détection, ainsi que sur les gains en termes de temps de calcul ont été mesurés.

Dans la suite de ce chapitre, nous résumons tout d'abord les contributions de cette thèse. La seconde partie évoque les limites de notre approche. Nous présentons en conséquence différentes améliorations possibles de l'algorithme et envisageons de meilleures techniques.

6.1 Contributions

Cette section résume l'ensemble de nos contributions, à savoir l'outil AATAC, son évaluation, ainsi que l'évaluation de l'impact de différentes techniques d'échantillonnage du trafic sur ses performances.

6.1.1 AATAC

AATAC se veut être un algorithme de détection non-supervisé visant à opérer en temps réel tout en respectant des contraintes industrielles. Il doit pouvoir être exécuté sur des équipements aux ressources de calcul limitées car consommées par d'autres traitements nécessaires à la gestion du trafic. Il repose sur l'hypothèse que le trafic anormal est statistiquement rare et significativement différent du trafic normal.

Comme déjà vu, nous proposons un algorithme divisé en deux types de traitements. Un premier, dit continu, traite le trafic en un temps linéaire et construit une caractérisation du trafic à chaque instant. Cette caractérisation nécessite deux types de données, l'une permettant de caractériser la distribution d'un attribut du trafic, l'autre une caractéristique globale au trafic. Le traitement continu utilise comme outil de base une valeur de densité, diminuant de manière exponentielle au fil du temps. Le deuxième traitement, dit discret, produit à intervalles réguliers une photographie du trafic. Cet instantané est utilisé à la fois pour opérer la détection et pour aider l'administrateur au diagnostic, en lui fournissant une vue graphique et dynamique du trafic à l'instant de l'anomalie.

L'algorithme profite de plusieurs optimisations ayant permis de simplifier, et donc d'accélérer, le traitement. Par exemple les distributions produites et enregistrées dans les instantanés sont représentées par des prototypes d'histogrammes. Ces prototypes d'histogrammes sont simples à comparer, ce qui permet une application accélérée de l'algorithme des k plus proches voisins, utilisé pour détecter les instantanés significativement différents de ceux précédemment créés.

6.1.2 Évaluation d'AATAC sur deux jeux de données

Pour évaluer les performances d'AATAC, nous avons construit deux jeux de données :

SynthONTS. Pour évaluer la capacité d'AATAC à détecter les anomalies correctement sans lever de faux positifs, nous avions besoin d'un jeu de données à la fois assez varié et labellisé suivant un protocole rigoureux, en lequel nous pouvions avoir confiance. Malheureusement, les jeux de données publiquement accessibles ne répondaient pas parfaitement à ce besoin. En conséquence, nous avons choisi de créer notre propre jeu de données : SynthONTS. Ce jeu de données consiste en un ensemble de traces

capturées à l'entrée d'un fournisseur de cloud. Plusieurs anomalies, générées par l'émulation d'un réseau correspondant à ces traces, ont alors été intégrées dans le jeu de données. Il contient notamment plusieurs types de DDoS, le type d'attaque que l'algorithme vise en priorité.

Cette approche nous a permis d'obtenir un trafic réaliste (puisque le trafic de fond capturé est réel et récent), tout en incluant des attaques parfaitement labellisée. L'analyse préalable du trafic capturé a d'ailleurs permis d'isoler d'éventuelles attaques déjà présentes dans trafic. La construction de ce jeu de données nous a permis l'obtention de résultats pertinents sur les performances d'AATAC.

SynthONTS a aussi servi à évaluer l'impact de cinq techniques différentes d'échantillonnage, à la fois sur la qualité de la détection et sur le gain en temps de calcul.

Traces Border 6. Ces traces ont été capturées à l'entrée du réseau de l'un des clients de Border 6, hébergeant notamment plusieurs sites de vente en ligne. Ce jeu de données a été utilisé notamment pour vérifier la capacité de l'algorithme à opérer avec des ressources quantitativement contraintes sur du trafic réel.

Pour chaque évaluation, AATAC a répondu à l'ensemble de nos attentes. À son meilleur point d'opération il a détecté 83% des anomalies tout en ne générant que 0.13% de faux-positifs sur SynthONTS. Sur les traces Border 6, il a en moyenne nécessité $0,2s$ pour traiter une seconde de trafic, confirmant sa capacité à opérer en temps réel.

L'ensemble des résultats obtenus a été comparé à ceux produits par UNADA et FastNetMon. À son meilleur point d'opération, AATAC a montré une détection plus exacte que FastNetMon en opérant une détection nécessitant trois fois moins de ressources. Si UNADA a montré de meilleurs résultats de détection qu'AATAC, c'est au prix d'un temps de traitements dix fois plus long. Ces résultats nous permettent d'affirmer qu'AATAC est une solution permettant une haute qualité de la détection sans pour autant nécessiter beaucoup de ressources de calcul.

6.1.3 Impact de l'échantillonnage sur la détection

Lors d'une deuxième évaluation, nous avons évalué l'impact d'un large ensemble de techniques d'échantillonnage sur les performances d'AATAC. Nous avons utilisé cinq algorithmes d'échantillonnage différents, couvrant les principales familles existantes. Pour des taux d'échantillonnage au-dessus de 1 paquet pour 2000, AATAC a montré une capacité de détection très similaire à celle obtenue sur du trafic non-échantillonné. La détection ne s'est sensiblement dégradée qu'au-delà d'un taux de 1 paquet pour 5000. Cette évaluation nous a aussi permis de constater que la détection est restée particulièrement bonne (pour des taux d'échantillonnage élevés) avec des techniques d'échantillonnage temporel.

Cette évaluation a montré que l'échantillonnage a aussi permis une réduction significative du temps de calcul, proportionnelle au taux d'échantillonnage. Aussi, l'impact de l'échantillonnage sur les performances de FastNetMon a été plus important, rendant sa détection complètement inefficace.

6.2 Perspectives

Nous présentons dans cette section les limites et améliorations possibles de notre outil de détection et son opération sur du trafic échantillonné.

6.2.1 Une meilleure caractérisation des anomalies

AATAC a l'avantage de produire une caractérisation du trafic permettant un diagnostic simple des anomalies détectées. Néanmoins, la sélection des caractéristiques à surveiller reste un problème. Si la complexité limitée de l'algorithme permet la surveillance d'un grand nombre de paramètres, certains d'entre eux peuvent alors s'avérer plus pertinents que d'autres pour la détection d'anomalies. Dans notre cas, les caractéristiques surveillées ont été choisies en connaissance des anomalies à détecter. Une parfaite autonomie nécessiterait la sélection autonome des paramètres les plus pertinents, à la fois pour produire une meilleure détection et pour faciliter le diagnostic de l'administrateur.

Certaines techniques, comme MMIFS [75] permettent alors une sélection des attributs en fonction de l'information qu'ils apportent à la caractérisation du trafic. Elles suppriment alors du modèle du trafic des attributs redondants, trop corrélées à d'autres attributs. Néanmoins, dans un contexte de détection non-supervisée, ces méthodes de sélection n'ont pas véritablement de sens. En effet, les algorithmes de détection non-supervisés sont entraînés sur le trafic en production, la corrélation des attributs du trafic serait alors observée sur le trafic en production, supposé comme constitué majoritairement de trafic normal. Or, par définition, une anomalie est un évènement rare et inattendu pouvant avoir un impact significatif sur les propriétés du trafic. De tels évènements peuvent alors provoquer des changements significatifs dans les corrélations observées. Ne pas surveiller un attribut parce qu'il est corrélé à d'autres en temps normal peut priver le détecteur et l'administrateur d'indices significatifs sur la nature et les caractéristiques de certaines anomalies. De manière générale, les méthodes de détection non-supervisées peuvent difficilement tirer parti de l'analyse des propriétés du trafic en conditions normales pour en déduire quelles seront les caractéristiques du trafic les plus pertinentes pour caractériser une anomalie.

Ainsi, construire une méthode de sélection des attributs plus pertinente nécessite de raisonner sur les caractéristiques des anomalies, et non celles du trafic normal. Cela nécessite

la réintroduction d'une forme de connaissance dans l'outil de détection. En général, une solution pourrait alors être de combiner le détecteur avec une approche supervisée ou reposant sur une base de connaissances, pour une détection hybride. Néanmoins, si l'approche choisie nécessite un travail de configuration, de création de signatures ou d'entraînement conséquent, la solution perdrait son aspect proactif.

Pour AATAC, il serait par conséquent plus intéressant de construire cette base de connaissances à partir du retour de l'administrateur, à postériori, lorsqu'une anomalie est détectée. Ces mécanismes d'apprentissage, dits "par renforcement", pourraient alors être utilisés. Un mécanisme envisageable pourrait consister à attribuer un indice de confiance à chaque attribut du trafic surveillé (augmentant lorsque l'attribut a permis de détecter uns véritable anomalie, et diminuant lorsqu'il a conduit à la levée d'un faux positif).

6.2.2 Échantillonnage idéal

Comme nous l'avons vu dans le chapitre 5, AATAC s'est avéré particulièrement efficace sur du trafic échantillonné. Afin de contrôler les ressources de calcul utilisées par l'algorithme, la mise en place d'un mécanisme d'échantillonnage adaptatif pourrait bénéficier au fonctionnement de l'algorithme. Pour des forts taux d'échantillonnage, l'algorithme s'est avéré plus efficace avec les techniques d'échantillonnage temporel, alors qu'à faible taux, c'est l'échantillonnage reposant sur le nombre de paquets qui a montré les meilleurs résultats. Un mécanisme d'échantillonnage adaptatif pour AATAC pourrait alors utiliser un mélange de ces deux techniques. Une telle technique d'échantillonnage adaptative permettrait une utilisation de ressources de calcul plafonnée tout en assurant les meilleures performances possibles.

6.2.3 Résistance aux *slow growth*

Comme indiqué dans le chapitre 1, des techniques d'évasion sont disponibles pour les attaquants. Dans le cas des attaques DDoS, l'attaque par *slow growth* s'avère simple à mettre en place et pourrait être particulièrement efficace pour contourner la technique de détection d'AATAC. En effet, la caractérisation sur trafic construite par AATAC pour opérer la détection repose sur les N derniers instantanés produits par l'algorithme. En augmentant petit à petit le débit d'une attaque, l'attaquant peut facilement influencer les instantanés produits. Dans ces instantanés, l'attaque finira par apparaitre comme du trafic normal.

Pour faire face à cette technique d'évasion, il serait alors possible de conserver les instantanés sur une échelle temporelle plus large. Des instantanés plus anciens sont nécessaires

afin de caractériser le trafic avant l'attaque. Pour pallier ce problème, une solution pourrait être de conserver certains instantanés à part, afin de construire par exemple un profil journalier. Pour obtenir des instantanés plus représentatifs sur le long terme, nous pouvons envisager de fusionner (en moyennant leur densités par exemple) certains instantanés. Cela permettrait d'obtenir une représentation du trafic à une échelle de temps plus large.

6.3 Résumé

Pour conclure cette thèse, ce chapitre a récapitulé l'ensemble de nos contributions, incluant la conception d'un nouvel algorithme non-supervisé de détection d'anomalies réseau, son évaluation et celle de l'impact de l'échantillonnage sur son opération. Différentes possibilités d'amélioration de l'approche ont été évoquées, comme une amélioration de la caractérisation des anomalies, la conception d'une technique d'échantillonnage idéale à l'opération de l'algorithme, ainsi qu'une possible amélioration permettant une résistance aux attaques par *slow growth*.

Ainsi, nos travaux ont prouvé qu'il était non seulement possible, mais aussi efficace, d'utiliser des approches non-supervisées pour la détection des attaques DDoS. L'approche proposée par AATAC, reposant sur la création d'instantanés du trafic, a montré une bonne capacité de détection avec des ressources de calcul contraintes. Nous espérons que nos travaux pousseront la communauté à expérimenter d'autres approches non-supervisées pour la détection des anomalies réseaux. La complexité des approches actuelles rendant leur adoption difficile par les acteurs industriels, nous espérons que d'autres approches limitant leur consommation en ressources de calcul verront le jour dans les prochaines années.

Acronymes

AATAC Autonomous Algorithm for Traffic Anomaly Characterization. 11, 18–20, 51, 52, 54, 62, 63, 66, 67, 72, 73, 75, 76, 79, 83, 84, 86, 89, 90, 93–95, 99–104

ADAM Audit Data Analysis and Mining. 31

ART Adaptive Resonance Theory. 46

CBUID Clustering-Based Method for Unsupervised Intrusion Detections. 44

CLIQUE CLustering In QUEst. 45

CLUSLab CLUSter Labelling technique. 46

CORE Common Open Research Emulator. 70, 71

CUSUM CUmulative SUM. 37, 48

DBSCAN Density-Based of Application with Noise. 43, 44

DDoS Distributed Denial of Service. 9, 10, 19, 39, 42, 69, 70, 72, 101, 103, 104

DoS Denial of Service. 31, 39, 40, 70, 72

DPI Deep Packet Inspection. 25, 87

DTW Dynamic Time Warping. 47

EM Expectation-Maximisation. 48

EMERALD Event Monitoring Enabling Responses to Anomalous Live Disturbances. 40

FAI Fournisseur d'Accès à Internet. 9

FLD Fisher Linear Discriminant. 38

fpMAFIA frequent pattern in pMAFIA. 44, 45

FTP File Transfer Protocol. 72

ICMP Internet Control Message Protocol. 70, 72

IFPI International Federation of the Phonographic Industry. 16

IP Internet Protocol. 17, 41, 55, 56, 70

KDD Knowledge Discovery and Data Mining. 27, 28, 32, 34, 40–45, 68

KLE Karhunen–Loeve Expansion. 38, 39

kNN k-Nearest Neighbour. 45

LERAD LEarning Rules for Anomaly Detection. 31

LOF Local Outlier Factor. 42

MAWI Measurement and Analysis on the WIDE Internet. 68

MDL Minimal Description Lenght. 45

MIGE-FS Mutual Information and Generalized Entropy based Feature Selection. 46

MINDS Minnesota Intrusion Detection System. 42

MMIFS Modified Mutual Information-based Feature Selection. 45, 102

MPAA Motion Picture Association of America. 16

NIDES Next-Generation Intrusion Detection Expert System. 40, 41

NL Nested Loop. 43

NSOM Network-based detector using SOM. 46

OCSVM One-Class Support Vector Machine. 46

ODIN Outlier Detection using Indegree Number. 42

ONTIC Online Network Traffic Characterization. 69

ONTS ONTIC Network Traffic Summary. 69

ORUNADA Online and near Real-time UNADA. 19, 83, 84

PCA Principal Component Analysis. 38, 39, 41, 45, 47

PHAD Packet Header Anomaly Detection. 40

pMAFIA Parallel Merging of Adaptive Finite IntervAls. 44, 45

RBP Resilient Back Propagation. 32

RBRP Recursive Binning and Re-Projection. 43

ROC Receiver Operating Characteristic. 72–74

RT-UNNID Real-Time Unsupervised Neural-Net-based Intrusion Detector. 46

SOM Self-Organizing Map. 46

SVM Support Vector Machine *ou* Séparateur à Vaste Marge. 29, 30, 32, 34, 39, 45–47

TCLUS Tree-based CLUSTering algorithm. 46

TCP Transmission Control Protocol. 55, 56, 70

TFP Taux de Faux Positifs. 72, 73, 89

TVP Taux de Vrais Positifs. 72, 73, 89

UDP User Datagram Protocol. 55, 70, 72

UNADA Unsupervised Network Anomaly Detector Algorithm. 43, 44, 101

VPP Valeur Prédictive Positive. 74

WIDE Widely Integrated Distributed Environment. 68

ZRC Zero Reference Curve. 74, 75

Bibliographie

[1] S. Panchen, P. Phaal, and N. McKee, "Inmon corporation's sflow : A method for monitoring traffic in switched and routed networks," RFC 3176, RFC Editor, September 2001.

[2] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for ip flow information export (ipfix)," RFC 3917, RFC Editor, October 2004.

[3] Akamai, "L'état d'internet," tech. rep., Akamai, 2017.

[4] T. Back and U. Hand, "Worldwide DDoS Attacks & Cyber Insights Research Report," Tech. Rep. May, Neustar, 2017.

[5] M. Roesch, "Snort : Lightweight Intrusion Detection for Networks.," LISA '99 : 13th Systems Administration Conference, pp. 229–238, 1999.

[6] V. Paxson, "Bro : a system for detecting network intruders in real-time," Comput. Netw., vol. 31, no. 23-24, pp. 2435–2463, 1999.

[7] W. Scheirer and M. Chuah, "Syntax vs. semantics : competing approaches to dynamic network intrusion detection," International Journal of Security and Networks, vol. 3, no. 1, pp. 24–35, 2008.

[8] I. S. Yoo, "Protocol anomaly detection and verification," ... Assur. Work. 2004. Proc. from ..., pp. 10–11, 2004.

[9] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, "Specification-based Anomaly Detection : A New Approach for Detecting Network Intrusions," CCS '02 : Proceedings of the 9th ACM Conference on Computer and communications security, pp. 265–274, 2002.

[10] G. Poojitha, K. Kumar, and P. JayaramiReddy, "Intrusion Detection using Artificial Neural Network," Second International conference on Computing, Communication and Networking Technologies, pp. 1–7, 2010.

[11] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier Detection Using Replicator Neural Networks," in Data Warehousing and Knowledge Discovery, pp. 170–180, 2002.

[12] J. Sun, H. Yang, J. Tian, and F. Wu, "Intrusion Detection Method Based on Wavelet Neural Network," 2009 Second International Workshop on Knowledge Discovery and Data Mining, pp. 851–854, 2009.

[13] S. C. Lee and D. V. Heinbuch, "Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks," IEEE Transactions on Systems, Man, and Cybernetics, vol. 31, no. 4, pp. 294–299, 2001.

[14] X. Tong, Z. Wang, and H. Yu, "A research using hybrid RBF/Elman neural networks for intrusion detection system secure model," Computer Physics Communications, vol. 180, no. 10, pp. 1795–1801, 2009.

[15] R. C. Aygun and A. G. Yavuz, "Network Anomaly Detection with Stochastically Improved Autoencoder Based Models," in 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 193–198, 2017.

[16] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[17] A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," Automation and Remote Control, vol. 25, pp. 821–837, 1964.

[18] R. Perdisci, O. Gu, and W. Lee, "Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems," in Sixth International Conference on Data Mining (ICDM'06), pp. 488–498, 2006.

[19] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD : A Multiple Classifier System for Accurate Payload-based Anomaly Detection," Computer Networks, vol. 53, no. 6, pp. 864–881, 2009.

[20] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," The VLDB Journal, vol. 16, no. 4, pp. 507–521, 2006.

[21] D. Barbará, J. Couto, S. Jajodia, and N. Wu, "Adam," ACM SIGMOD Record, vol. 30, no. 4, p. 15, 2001.

[22] M. V. Mahoney and P. K. Chan, "Learning Rules for Anomaly Detection of Hostile Network Traffic," International Conference on Data Mining (ICDM), pp. 601–604, 2003.

[23] M.-Y. Su, G.-J. Yu, and C.-Y. Lin, "A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach," Computers & Security, vol. 28, no. 5, pp. 301–309, 2009.

[24] P. A. Raj Kumar and S. Selvakumar, "Distributed denial of service attack detection using an ensemble of neural classifier," Computer Communications, vol. 34, no. 11, pp. 1328–1341, 2011.

[25] J. Vanerio and P. Casas, "Ensemble-learning Approaches for Network Security and Anomaly Detection," in Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, pp. 1–6, 2017.

[26] H. H. Nguyen, N. Harbi, and J. Darmont, "An Efficient Local Region and Clustering-Based Ensemble System for Intrusion Detection," Proceedings of the 15th Symposium on International Database Engineering & Applications - IDEAS '11, p. 185, 2011.

[27] J. Song, Z. Zhu, P. Scully, and C. Price, "Modified Mutual Information-based Feature Selection for Intrusion Detection Systems in Decision Tree Learning," Journal of Computers, vol. 9, no. 7, pp. 1542–1546, 2014.

[28] Y. Qiao, X. W. Xin, Y. Bin, and S. Ge, "Anomaly Intrusion Detection Method Based on Rough Set Theory," Electronics Letters, vol. 38, no. 13, pp. 663–664, 2002.

[29] J. Xu and C. Shelton, "Intrusion Detection using Continuous Time Bayesian Networks.," Journal of Artificial Intelligence Research, vol. 39, pp. 745–774, 2010.

[30] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab : Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," in Proceedings of the 6th International Conference on Emerging Networking Experiments and Technologies, Co-NEXT'10, 2010.

[31] "Lbnl dataset." `https://www.icir.org/enterprise-tracing/Overview.html`.

[32] M. Yu, "A Nonparametric Adaptive Cusum Method And Its Application In Network Anomaly Detection," International Journal of Advancements in Computing Technology, vol. 4, no. 1, pp. 280–288, 2012.

[33] A. Soule, K. Salamatian, and N. Taft, "Combining Filtering and Statistical Methods for Anomaly Detection," in Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, vol. 5, pp. 331–334, 2005.

[34] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," ACM SIGCOMM Computer Communication Review, vol. 35, no. 4, p. 217, 2005.

[35] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and Identification of Network Anomalies Using Sketch Subspaces," in Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pp. 147–152, 2006.

[36] M. Celenk, T. Conley, J. Willis, and J. Graham, "Predictive Network Anomaly Detection and Visualization," IEEE Transactions on Information Forensics and Security, vol. 5, no. 2, pp. 288–299, 2010.

[37] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, and D. Sornette, "Accurate network anomaly classification with generalized entropy metrics," Computer Networks, vol. 55, no. 15, pp. 3485–3502, 2011.

[38] C. Callegari, S. Giordano, and M. Pagano, "Entropy-based Network Anomaly Detection," in 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 334–340, 2017.

[39] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems : Detecting the Unknown without Knowledge," Computer Communications, vol. 35, no. 7, pp. 772–783, 2012.

[40] M. V. Mahoney and P. K. Chan, "PHAD : Packet Header Anomaly Detection for Identifying Hostile Network Traffic," tech. rep., Florida Institute of Technology, 2001.

[41] C. Krügel, T. Toth, and E. Kirda, "Service Specific Anomaly Detection for Network Intrusion Detection," in Proceedings of the 2002 ACM symposium on . . . , pp. 201–208, 2002.

[42] D.-Y. Yeung and C. Chow, "Parzen-window Network Intrusion Detectors," Object recognition supported by user interaction for service robots, vol. 4, pp. 385–388, 2002.

[43] D. Anderson, T. Frivold, and A. Valdes, "Next-generation Intrusion Detection Expert System (NIDES) : A summary," Tech. Rep. May 1995, Computer Science Laboratory, SRI International, 1995.

[44] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, and A. Valdes, "Detecting Unusual Program Behavior Using the Stastistical Component of the Next-generation Intrusion Detection Expert System (NIDES)," Computer Science Laboratory SRI-CSL, no. 910097, pp. 6–95, 1995.

[45] P. a. Porras and P. G. Neumann, "EMERALD : Event Monitoring Enabling Responses to Anomalous Live Disturbances," Proc. 20th NIST-{NCSC} National Information Systems Security Conference, pp. 353–365, 1997.

[46] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "A Novel Anomaly Detection System using Feature-based MSPCA with Sketch," in 2017 26th Wireless and Optical Communication Conference (WOCC), pp. 1–6, 2017.

[47] G. Liu, Z. Yi, and S. Yang, "A hierarchical intrusion detection model based on the PCA neural networks," Neurocomputing, vol. 70, no. 7-9, pp. 1561–1568, 2007.

[48] D. Ariu, R. Tronci, and G. Giacinto, "HMMPayl : An intrusion detection system based on Hidden Markov Models," Computers & Security, vol. 30, no. 4, pp. 221–241, 2011.

[49] W. Lu and A. A. Ghorbani, "Network Anomaly Detection Based on Wavelet Analysis," EURASIP Journal on Advances in Signal Processing, vol. 2009, no. 1, p. 837601, 2009.

[50] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier Detection Using k-Nearest Neighbour Graph," in Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), vol. 3, pp. 430–433, 2004.

[51] U. K. Archive, "KDD Cup 1999 Data." `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. Accessed : 2018-01-24.

[52] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF : Identifying Density-Based Local Outliers," in Proceedings of the 2000 ACM Sigmod International Conference on Management of Data, pp. 1–12, 2000.

[53] D. Pokrajac, A. Lazarevic, and L. Latecki, "Incremental Local Outlier Detection for Data Streams," in 2007 IEEE Symposium on Computational Intelligence and Data Mining, pp. 504–515, 2007.

[54] L. Ertöz, E. Eilertson, A. Lazarevic, P. N. Tan, V. Kumar, J. Srivastava, and P. Dokas, "MINDS - Minnesota Intrusion Detection System," Next generation data mining, pp. 199–218, 2004.

[55] V. Chandola, E. Eilertson, and L. Ert, "Data Mining for Cyber Security," in Data Warehousing and Data Mining Techniques for Computer Security, pp. 1–20, Springer, 2006.

[56] A. Ghoting, S. Parthasarathy, and M. E. Otey, "Fast Mining of Distance-Based Outliers in High-Dimensional Datasets," Data Mining and Knowledge Discovery, vol. 16, no. 3, pp. 349–364, 2008.

[57] E. M. Knorr and R. T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," 24th International Conference on Very Large Data Bases, pp. 392–403, 1998.

[58] S. Bay and M. Schwabacher, "Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule," in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 29–38, 2003.

[59] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," Communications of the ACM, vol. 18, no. 9, pp. 509–517, 1975.

[60] Antonin Guttman, "R-trees : A Dynamic Index Structure for Spatial Searching," in Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 47–57, ACM, 1984.

[61] S. Berchtold, D. A. Keim, and H.-p. Kriegel, "The X-tree : An Index Structure for High-Dimensional Data," in Proceedings of the 22th International Conference on Very Large Data Bases, pp. 28–39, 1996.

[62] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Survey on Incremental Approaches for Network Anomaly Detection," International Journal of Communication Networks and Information Security, vol. 3, no. 3, pp. 226–239, 2011.

[63] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-Means+ID3 : A Novel Method for Supervised Anomaly Detection by Cascading K-Means Clustering and ID3 Decision Tree Learning Methods," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 3, pp. 345–354, 2007.

[64] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231, 1996.

[65] J. Dromard and P. Owezarski, "Online and Scalable Unsupervised Network Anomaly Detection Method," IEEE Transaction on network and service management, vol. 14, no. 1, pp. 34–47, 2017.

[66] X. Zhao, G. Wang, and Z. Li, "Unsupervised Network Anomaly Detection Based on Abnormality Weights and Subspace Clustering," in Sixth International Conference on Information Science and Technology, ICIST 2016, pp. 482–486, 2016.

[67] L. Portnoy, E. Eleazar, and S. Stolfo, "Intrusion Detection with Unlabeled Data Using Clustering," in In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA'2001), pp. 1–14, 2001.

[68] S. Jiang, X. Song, H. Wang, J. J. Han, and Q. H. Li, "A clustering-based method for unsupervised intrusion detections," Pattern Recognition Letters, vol. 27, no. 7, pp. 802–810, 2006.

[69] K. Leung and C. Leckie, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters," in Proceedings of the Twenty-eighth Australasian Conference on Computer Science, vol. 38, pp. 333–342, 2005.

[70] H. S. Nagesh, S. Goil, and A. Choudhary, "Scalable Parallel Subspace Clustering Algorithm for Massive Data Sets," in Proceedings 2000 International Conference on Parallel Processing, pp. 477–484, 2000.

[71] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," ACM SIGMOD Record, vol. 27, no. 2, pp. 94–105, 1998.

[72] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," ACM SIGMOD Record, vol. 29, no. 2, pp. 1–12, 2000.

[73] N. Nastaiinullah and a. P. Kurniati, "Anomaly Detection on Intrusion Detection System Using CLIQUE Partitioning," in 2014 2nd International Conference on Information and Communication Technology (ICoICT), pp. 7–12, 2014.

[74] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "An Effective Unsupervised Network Anomaly Detection Method," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI '12), p. 533, 2012.

[75] F. Amiri, M. Rezaei Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1184–1199, 2011.

[76] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "A multi-step outlier-based anomaly detection approach to network-wide traffic," Information Sciences, vol. 348, pp. 243–271, 2016.

[77] K. Labib and R. Vemuri, "NSOM : A Real-Time Network-Based Intrusion Detection System Using Self-Organizing Maps," Networks and Security, 2002.

[78] M. Amini, R. Jalili, and H. R. Shahriari, "RT-UNNID : A practical solution to real-time network-based intrusion detection using unsupervised neural networks," Computers and Security, vol. 25, no. 6, pp. 459–468, 2006.

[79] C. Wagner, R. State, T. Engel, and M. Learning, "Machine Learning Approach for IP-Flow Record Anomaly Detection," in Networking, pp. 28–39, 2011.

[80] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast Distributed Outlier Detection in Mixed-Attribute Data Sets," Data Mining and Knowledge Discovery, vol. 12, no. 2-3, pp. 203–228, 2006.

[81] J. Song, H. Takakura, Y. Okabe, and K. Nakao, "Toward a more practical unsupervised anomaly detection system," Information Sciences, vol. 231, pp. 4–14, 2013.

[82] G. Fernandes, L. F. Carvalho, J. J. Rodrigues, and M. L. Proença, "Network anomaly detection using IP flows with Principal Component Analysis and Ant Colony Optimization," Journal of Network and Computer Applications, vol. 64, pp. 1–11, 2016.

[83] W. Lu and H. Tong, "Detecting Network Anomalies Using CUSUM and EM Clustering," Proceedings of the 4th International Symposium on Advances in Computation and Intelligence, pp. 297–308, 2009.

[84] J. Zhang, M. Zulkernine, and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," IEEE Transactions on Systems, Man, and Cybernetics, vol. 38, no. 5, pp. 649–659, 2008.

[85] J. Zhang and M. Zulkernine, "A Hybrid Network Intrusion Detection Technique Using Random Forests," in First International Conference on Availability, Reliability and Security (ARES'06), pp. 262–269, 2006.

[86] K. Yamanishi and J.-i. Takeuchi, "Discovering Outlier Filtering Rules from Unlabeled Data," Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 389–394, 2001.

[87] B. Claise, "Cisco systems netflow services export version 9," RFC 3954, RFC Editor, October 2004.

[88] Y. Chen and L. Tu, "Density-Based Clustering for Real-Time Stream Data," in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 133–142, 2007.

[89] H.-P. Zimek, Arthur ; Schubert, Erich ; Kriegel, "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data," Wiley Periodicals, vol. 4, no. 5, pp. 363–387, 2012.

[90] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," Computers & Security, vol. 31, no. 3, pp. 357–374, 2011.

[91] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1 : ICISSP,, no. Cic, pp. 108–116, 2018.

[92] "Ontic." `http://ict-ontic.eu/`. Accessed : 2017-05-12.

[93] K. Nasr, A. A.-e. Kalam, and C. Fraboul, "Performance Analysis of Wireless Intrusion Detection Systems," in Internet and Distributed Computing Systems : 5th International Conference, IDCS 2012, Wuyishan,, (Wuyishan), pp. 238–252, Springer Berlin Heidelberg, 2012.

[94] "DPDKCap." `http://github.com/dpdkcap/dpdkcap.html`. Accessed : 2018-02-26.

[95] "Fastnetmon." `https://fastnetmon.com/`.

[96] "Tcpreplay." `http://tcpreplay.synfin.net/`.

[97] E. a. Hernandez, M. C. Chidester, and A. D. George, "Adaptive Sampling for Network Management," Journal of Network and Systems Management, vol. 9, no. 4, pp. 409–434, 2001.

[98] B.-Y. Choi, J. Park, and Z.-L. Zhang, "Adaptive packet sampling for accurate and scalable flow measurement," in Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE, vol. 3, pp. 1448–1452 Vol.3, Nov 2004.

[99] M. Canini, D. Fayt, D. J. Millert, A. W. Mooret, and R. Bolla, "Per flow packet sampling for high-speed network monitoring," 1st International Conference on Communication Systems and Networks and Workshops, COMSNETS 2009, 2009.