Détection autonome de trafic malveillant dans les réseaux véhiculaires

Quentin Ricard

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *24/09/2020* **par :**

Quentin Ricard

Détection autonome de trafic malveillant dans les réseaux véhiculaires

---

JURY

Isabelle Chrisment	Professeur des Universités	Rapporteure
Jean-François Lalande	Professeur des Universités	Rapporteur
Sandrine Vaton	Professeur	Membre du Jury
Marcello Dias De Amorim	Directeur de Recherche	Membre du Jury
Thierry Gayraud	Professeur des Universités	Membre du Jury
Philippe Owezarski	Directeur de Recherche	Membre du Jury
Laurent Charnot	Ingénieur	Invité

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Philippe Owezarski
Rapporteurs :
Isabelle Chrisment et *Jean-François Lalande*

Remerciements

« Nous ne nous élevons pas au niveau de nos attentes, nous tombons au niveau de notre entraînement. »

Voilà ce qu'écrivait Archiloque il y a bien longtemps et lorsque j'ai découvert ce proverbe pendant ma rédaction, il fit résonner en moi deux échos. Le premier, serait que la qualité de notre travail représente directement le fruit de l'enseignement que nous avons reçu. Le second, pourrait être que nous ne nous élevons que grâce à ceux qui nous donnent de l'entrain. Ainsi, je considère que même si j'ai bénéficié d'un enseignement de qualité tout au long de mon parcours, jamais je ne serai parvenu à bout de ces travaux sans mon entourage, à commencer bien sûr par Philippe mon directeur.

Je te remercie pour tout ce que tu m'as appris, pour ta disponibilité et toutes ces discussions durant lesquelles nous partagions nos intuitions, mes « problèmes » et tes pistes pour y répondre. Tout cela m'a permis d'aller de l'avant malgré les revers que j'ai pu rencontrer pendant ces trois années et je t'en suis aujourd'hui extrêmement redevable.

Je tiens également à remercier chaleureusement Isabelle Chrisment et Jean-François Lalande pour leurs rapports et commentaires bienveillants concernant mes travaux. Merci également à Sandrine Vaton, Marcello Dias de Amorim et Thierry Gayraud d'avoir accepté d'examiner la thèse et d'assister à la soutenance.

J'exprime également toute ma gratitude envers Jérôme Boyer et Alain Filipowicz pour m'avoir permis de réaliser cette thèse au sein de Continental. Je remercie également toute l'équipe sécu/safety (présente et passée) de Continental Digital Services France et tout particulièrement Silvia et Laurent pour m'avoir assisté tout au long de mes recherches.

J'adresse tous mes remerciements au LAAS-CNRS et à toute l'équipe SARA pour m'avoir hébergé pendant plus de trois ans, et surtout pour m'avoir procuré un environnement d'accompagnement et d'entraide qui m'a été d'une aide considérable dans la réalisation de cette thèse. Un grand merci donc à mes colocataires et voisins de bureau : Imane et sa chambre, Clément et son Youki « c'est honteux ! » , Pascal le punk, Marine, Tanissia, Nicolas[2], Tic et Tac, Gilles et Rémy.

Cet environnement de recherche exceptionnel a également été complété d'un ensemble de personnes qui, de près comme de loin, ont contribué à la réussite de mes travaux. Je souhaite en conséquence remercier mon professeur Cédric Lauradoux pour m'avoir initié à la recherche au sein de Privatics. Cette expérience m'a motivé, quelques années plus tard à entreprendre cette thèse. Je remercie également Pascal Lafourcade pour m'avoir fait découvrir les travaux de mon directeur Philippe qui m'ont tapé dans l'oeil et motivé à le contacter.

Je tiens également à remercier tous mes amis pour m'avoir soutenu pendant toutes ces années : Sylvan, Simon et Daniel pour nos innombrables soirées face aux lumières de nos écrans. Flavien pour ces séances salvatrices à Rive Droite. Loïc pour nos descentes enneigées. Merci également au reste de la bande Valentinoise et Toulousaine ils se reconnaîtront !

Il m'est impossible d'oublier dans ces remerciements ma mère Eliane et Franck. Je vous suis à jamais reconnaissant pour tout ce que vous avez fait pour moi. Vous n'avez cessé de m'apporter de la motivation et les valeurs que vous m'avez transmises me portent dans tout ce que j'entreprends. Un grand merci également à mon père Michel, pour m'avoir insufflé le goût de la science au grès de ses histoires de sorgho. Enfin, merci à l'ensemble de ma famille et en particulier à Nicole pour ses relectures et ses conseils éclairés.

Enfin, je souhaite remercier Julie, pour avoir été présente aux moments les plus difficiles de la rédaction. Tu as su me remonter le moral lorsqu'il était au plus bas, me faire relativiser lorsque j'étais insatisfait de mes résultats. Ta tendresse, ton amour et nos promenades bordelaises m'ont porté dans la rédaction de ce manuscrit. Je suis impatient d'entamer ce nouveau chapitre de notre vie à tes côtés.

Table des matières

Introduction

Sommaire

L'avènement des Systèmes de Transport Intelligents (ITS) entraîne l'apparition de véhicules de plus en plus connectés à leur environnement sur les réseaux routiers mondiaux. Ils découlent d'une volonté de la part des acteurs industriels de faire évoluer les moyens de transports vers plus de sûreté, moins de pollution et un voyage plus agréable pour les passagers, sans oublier la course au véhicule complètement autonome. Il ne faut pour autant pas oublier que ces véhicules, dits connectés, proposent ces nouveaux services en se reposant sur des canaux de communication et systèmes embarqués susceptibles de dysfonctionnement. Or, ces derniers pourraient venir perturber la prise de décision de ces véhicules ou de leur conducteur, au point de menacer la sécurité des passagers ou des autres usagers de la route lorsque ces décisions sont prises sur la base d'une mauvaise ou de l'absence d'informations critiques. De plus, en permettant aux véhicules d'accéder au réseau et de communiquer avec le reste du monde, ils se voient eux aussi devenir vulnérables aux divers acteurs malveillants présents sur Internet.

Conséquemment, le monde automobile se retrouve désormais confronté aux mêmes problématiques qui occupent les acteurs industriels et académiques de la communauté des systèmes d'information traditionnels depuis de nombreuses années. En effet, les anomalies dont sont victimes les réseaux informatiques peuvent être la conséquence de défauts de configuration tout comme d'attaques informatiques. Elles se définissent par la déviation du comportement des communications habituelles observées entre les différents composants d'un réseau. En pratique, elles peuvent par exemple mener à l'incapacité d'un serveur de répondre à des requêtes légitimes dans le cadre d'attaques par déni-de-service ou bien à la fuite de données personnelles lors d'une intrusion dans un système d'information. Ainsi, chercheurs et industriels ont mobilisé des ressources considérables destinées à proposer des solutions permettant de discriminer ces anomalies du trafic normal à l'intérieur des

flux de communications. Des mécanismes allants du simple pares-feux basé sur des règles de filtrage à l'utilisation d'algorithmes d'apprentissage automatique ont été imaginés afin de permettre aux opérateurs de ces réseaux de détecter et de prévenir ces anomalies et intrusions menaçant leurs systèmes.

En devenant connecté, le véhicule est donc lui aussi soumis à ces menaces pesant sur les systèmes d'information classiques, à la différence qu'en tant que système cyber-physique les anomalies et intrusions pourraient avoir des répercussions dans le monde physique. C'est pourquoi il est primordial de considérer l'application des mécanismes de détection d'anomalies et d'intrusion au domaine véhiculaire. De surcroît, le monde académique ainsi que les différentes communautés d'experts en cybersécurité ont été témoins à plusieurs reprises ces dernières années des prouesses techniques de nombreuses équipes de recherche prouvant qu'il était possible de prendre le contrôle de véhicules à distance [1]. De plus, il a aussi été démontré que les attaquants étaient en capacité d'accéder à des informations personnelles des utilisateurs par le biais de leurs véhicules ou encore d'enregistrer des conversations à leur insu à l'intérieur même de l'habitacle.

Ainsi, les méthodes de sécurisation des réseaux doivent être appliquées au monde automobile afin que les systèmes de transport intelligents puissent être déployés et utilisés en toute confiance par les acteurs industriels et les utilisateurs. Aussi, le caractère singulier du milieu automobile suggère donc une adaptation des solutions existantes afin de prendre en compte ses spécificités. Dans le cadre de la détection d'anomalies, l'approche classique de la sécurisation d'un réseau consiste à positionner un détecteur à l'entrée de ce dernier. Or, dans les systèmes de transport intelligents, les véhicules sont connectés à des réseaux divers tels que les réseaux véhiculaires Adhoc ou les réseaux cellulaires et partagent donc les canaux de communication avec d'autres équipements. Il est donc préférable de favoriser une approche au niveau de l'hôte et donc un positionnement à l'intérieur même de l'habitacle. Le détecteur se limitant de ce fait aux seules anomalies affectant le véhicule sur lequel il est installé.

De plus, les communications des véhicules diffèrent des réseaux traditionnels et doivent donc être analysées en conséquence. Celles-ci peuvent être scindées en deux groupes. Le premier concerne le trafic généré par l'utilisation du véhicule dans le cadre des services des Systèmes de Transport Intelligents. On y trouvera par exemple des informations télémétriques sur le statut du véhicule notamment sa vitesse et sa position jusqu'à une description de son environnement comme la température ou les obstacles détectés par ses capteurs et caméras. Le second quant à lui, résulte de l'utilisation du système de divertissement du véhicule à savoir les services d'écoute de musique en ligne, la consultation de mails et messageries instantanées ou encore la mise à jour de ses applications.

En conséquence, nous nous proposons dans cette thèse d'étudier une solution de détection des anomalies dans les réseaux véhiculaires prenant en compte le caractère singulier des communications des véhicules et de leurs composants. Dans ce chapitre

1. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

d'introduction, nous présentons le cadre du projet E-Horizon dans lequel s'inscrit cette thèse dont le financement est assumé par Continental Digital Services France. Ensuite, nous exposons la problématique à laquelle nous répondons. Enfin, nous énonçons nos contributions qui visent à répondre à ce problème.

0.1 Le Projet E-Horizon

FIGURE 1 – Logo de l'entreprise Continental

Cette thèse s'inscrit dans le cadre du projet E-Horizon de Continental porté par sa filiale française : Continental Digital Services France. Créée en 2016, cette nouvelle entité de Continental a été chargée de développer des services numériques basés sur le cloud pour les véhicules connectés. Le projet E-Horizon a pour objectif principal la réduction des accidents et accidents mortels sur les routes en proposant des services permettant aux conducteurs de mieux anticiper leurs actions. De plus, la réduction de l'empreinte écologique des véhicules ainsi que l'amélioration de l'expérience de circulation sont aussi des paramètres importants dans le transport intelligent de demain.

En effet, dans le bilan de l'accidentalité routière de 2018 [2], l'observatoire national interministériel de la sécurité routière (ONISR) fait état des causes principales des accidents mortels selon l'étude de ses auteurs présumés (Tableau 1).

FIGURE 2 – Les onze services du projet E-Horizon.

Il apparaît que les causes majeures d'accidents en dehors de l'alcool et de la vitesse, sont dues à des situations dans lesquelles les conducteurs n'ont pas été en

2. https://www.onisr.securite-routiere.interieur.gouv.fr/etat-de-l-insecurite-routiere/bilans-annuels-de-la-securite-routiere/bilan-2018-de-la-securite-routiere

Cause identifiée	Proportion
Vitesse	27%
Alcool	18%
Autre cause	11%
Cause indéterminée	15%
Priorité	10%
Inattention	10%
Stupéfiant	9%
Malaise	7%
Dépassement dangereux	5%
Somnolence / Fatigue	3%
Changement de file	4%
Contre-sens	2%
Obstacle	1%
Facteurs liés au véhicule	0%
Téléphone	1%
Non-respect des distances de sécurité	1%
Total	124%

TABLE 1 – Causes principales (multi-causes) des accidents mortels selon l'étude des auteurs présumés d'accidents mortels

mesure de prendre les bonnes décisions au bon moment par manque d'attention ou d'information. Ainsi, Continental souhaite agir pour la sécurité routière en mettant au point 11 services directement dédiés à prévenir ces situations. Ces derniers sont donc orientés vers l'apport d'informations complémentaires aux conducteurs de véhicules ou à faciliter leur conduite. Ces services sont illustrés par la Figure 2 et sont définis comme suit :

- Systèmes d'aide à la conduite :
  - Lecture et partage des panneaux de signalisation.
  - Assistance au péage.
  - Détection de fatigue.
  - Informations de la vitesse maximale conseillée dans certains tronçons de route.
  - Aide au parking en fournissant au conducteur la position de la place la plus proche.
  - Détection des zones de travaux et dangers proches.
- Analyse du véhicule et notamment de l'usure de ses composants.
- Évaluation de la densité de trafic et détection de bouchons.

Ces services, font donc appel à l'information engrangée grâce aux flottes de véhicules déployés sur les routes et centralisée dans des serveurs. Ces connaissances sont ensuite redistribuées dans une optique d'intelligence collaborative. Un exemple d'application se traduirait donc de la manière suivante (Figure 3) :

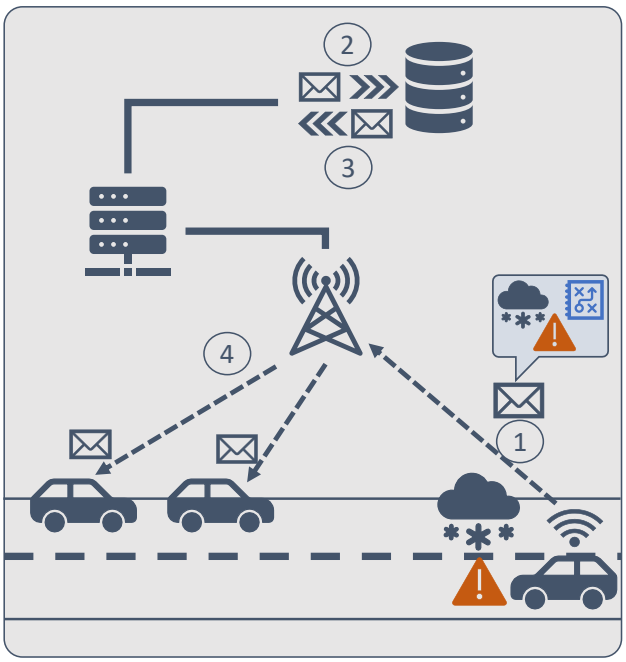


FIGURE 3 – Exemple de service dédié au partage de l'information concernant des conditions météorologiques dangereuses.

Dans ce scénario, un premier véhicule détecte grâce à ses capteurs, une situation dangereuse comme la présence de glace sur la route. Il partage l'information avec un serveur distant par le biais de communications cellulaires. Après traitement de cette information, le serveur propage cette connaissance aux véhicules en approche de la zone dangereuse afin que les conducteurs puissent adapter leur vitesse et éviter un éventuel accident.

Le bon fonctionnement des services du projet E-Horizon repose donc sur une confiance totale dans les données qui transitent sur le réseau cellulaire. Or, les véhicules désormais connectés peuvent être sujets à des dysfonctionnements des équipements embarqués dans le véhicule ou pire encore, des tentatives de piratage de la part d'acteurs malveillants. Ainsi, il est donc essentiel de pouvoir s'assurer que les communications des véhicules sont traitées de façon à détecter d'éventuelles anomalies.

0.2 Problématique

Nous présentons notre problématique sous deux aspects. Tout d'abord, nous abordons le sujet de la détection d'anomalies dans le contexte du véhicule connecté afin d'en extraire les éléments à prendre en compte dans la conception d'un bon détecteur. Ensuite, nous présentons les difficultés rencontrées dans le traitement de ce trafic particulier notamment de par la nature des anomalies ainsi que la création d'un corpus de données suffisamment représentatif pour la validation de

notre méthode de détection d'anomalies dédiée aux réseaux cellulaires de véhicules.

La détection de trafic malveillant dans les réseaux véhiculaires L'application de la détection d'anomalies aux réseaux cellulaires de véhicules implique la prise en compte des paramètres supplémentaires dans l'élaboration du détecteur. Bien que la qualité du moteur de détection soit un critère important, il faut cependant également considérer le coût du système dans son ensemble, tant sur le plan matériel qu'humain, mais aussi les limites de son application.

0.2.1 Caractéristiques du trafic véhiculaire

Nous distinguons dans les communications des réseaux cellulaires de véhicules, ou C-V2X pour *cellular vehicles-to-everything*, deux types de trafics. Le premier est lié aux services des systèmes de transports intelligents ITS, le second, quant à lui, résulte de l'interaction des utilisateurs avec le système de divertissement du véhicule. Ainsi le trafic lié aux ITS est plutôt de nature régulier où chaque véhicule envoie des messages de télémétrie destinés au bon fonctionnement des services ITS, donnant ainsi lieu à des sessions de communication de longue durée. Le trafic lié aux systèmes de divertissements quant à lui se rapproche de celui des applications mobiles présentes sur les smartphones. En effet, on retrouve sur les systèmes de divertissement des véhicules des applications d'écoute de musique, de navigation, ainsi que les mises à jour de ces applications ou des services de cartographie ou d'autres composants du véhicule. Ces applications résultent en des échanges brefs et répétés entre le véhicule et les différents serveurs impliqués dans ces services.

Ainsi, le système de détection doit être capable de s'adapter à ce trafic changeant, mais aussi aux différentes habitudes d'utilisation de ces systèmes qui peuvent avoir une forte influence sur ce trafic.

0.2.2 Nature des anomalies

Les anomalies auxquelles les véhicules sont vulnérables sont de natures très diverses. Allant de la simple attaque par déni-de-service, empêchant le véhicule d'accéder correctement aux services connectés, à l'utilisation de logiciels malveillants pouvant extraire des informations personnelles du véhicule. Le spectre large de l'impact sur les propriétés du réseau de ces attaques implique que les attributs utiles à leur détection varient grandement d'un cas à l'autre. Le détecteur doit donc faire feu de tout bois afin d'être en mesure de détecter un maximum de types d'anomalies possibles.

0.2.3 Contexte d'exécution et traçabilité

De plus, contrairement aux systèmes de détection classiques disposés à l'entrée des réseaux à protéger, nous considérons dans cette thèse un réseau cellulaire où

les véhicules sont mélangés avec les autres équipements du réseau comme les mobiles ou les objets connectés de l'internet-des-objets. Ainsi, le système de détection d'anomalie doit être exécuté à l'intérieur même du véhicule.

Conséquemment, il est donc primordial d'assurer une traçabilité des alertes détectées puisque l'interaction directe d'un administrateur avec chaque véhicule connecté n'est pas envisageable. De plus, de nombreuses applications destinées au bon fonctionnement du véhicule devront s'exécuter en concurrence avec le système de détection, celui-ci se verra donc affecté de ressources limitées.

0.2.4 Cadre légal

Enfin, le traitement des communications générées par les véhicules soulève des problématiques juridiques dues au caractère personnel de ces données qu'il faut aussi prendre en considération. En effet, l'analyse de communications induites par l'utilisation d'un véhicule s'apparente à un traitement de données à caractère personnel défini par la Commission Nationale de l'Informatique et Liberté (CNIL) comme suit [3] :

> *Toute opération portant sur des données personnelles, quel que soit le procédé utilisé. Par exemple, enregistrer, organiser, conserver, modifier, rapprocher avec d'autres données, transmettre, etc. des données personnelles.*

Ainsi, pour qu'un tel système puisse être un jour embarqué dans un véhicule, il se devra d'être conforme aux réglementations en vigueur. Dans le cas de l'Europe et particulièrement de la France, il est donc important de prendre en compte la réglementation générale pour la protection des données personnelles (RGPD), que ce soit dans l'analyse des données ou le traitement des anomalies détectées.

0.3 Contributions

Nous avons cherché à répondre à cette problématique en proposant 3 contributions que nous présentons dans cette thèse.

0.3.1 Svalinn : Approche ontologique à la détection non supervisée d'anomalies

Tout d'abord, nous avons mis au point un système de détection d'anomalies nommé **Svalinn** [4]. Ce système est basé sur une représentation ontologique des communications véhiculaires. Nous extrayons des propriétés du trafic et les introduisons dans une ontologie, puis, l'algorithme à mémoire temporelle hiérarchique (ou *hierarchical temporal memory algorithm* HTM) est en charge de déceler d'éventuelles anomalies. Afin d'archiver les alertes générées par ces dernières, le système utilise

3. https ://www.cnil.fr/fr/cnil-direct/question/un-traitement-de-donnees-caractere-personnel-cest-quoi

4. Grímnismál 38$^{\text{ème}}$ strophe

pour chaque anomalie détectée des règles d'inférence destinées à réunir des informations de contexte autour de l'anomalie. Le système procède donc en trois étapes à savoir la construction des attributs du trafic, la détection des anomalies et enfin leur représentation. Ces travaux ont été présentés au congrès Embedded RealTime Systems (ERTS) [100].

0.3.2 Autobot : Environnement d'émulation d'un réseau véhiculaire

Afin d'évaluer notre système, nous avons conçu un environnement d'émulation de communications cellulaires de véhicules nommé **Autobot**. Il n'existe pas à notre connaissance de corpus réel de trafic véhiculaire contenant des anomalies. Aussi, nous avons décidé de générer un trafic réaliste contenant plusieurs anomalies dans un environnement isolé dont les propriétés réseau respectent celles observées sur les réseaux cellulaires actuels. Plusieurs applications véhiculaires et de divertissement ont donc été créées afin de générer ce corpus. De plus, les anomalies générées dans l'environnement ont été tirées de véritables scénarios observés sur le terrain. Nous avons réalisé une étude de cet environnement afin de démontrer ses capacités à émuler un comportement réseau défini et assurer le bon fonctionnement des applications communicantes. Les résultats de ces travaux ont été présentés au symposium Distributed Simulation and Real Time Applications (DSRT) [99].

0.3.3 Évaluation des performances de détection

Notre dernière contribution porte sur une étude des attributs d'intérêt pour l'algorithme employé durant la phase de détection à savoir l'algorithme à mémoire temporelle hiérarchique. Notre objectif a donc été d'isoler les attributs du trafic les plus importants pour la détection des anomalies de notre corpus. En effet, nous démontrons qu'utiliser moins d'attributs a un effet positif sur la détection. De plus, la réduction du nombre d'attributs réduit aussi de fait le coût de traitement de l'algorithme réduisant ainsi à la fois la latence entre l'apparition d'une anomalie et sa détection, mais aussi les ressources nécessaires au traitement du trafic.

0.4 Plan

Cette thèse est donc divisée en 4 chapitres :

— Le premier chapitre introduit l'état de l'art des attaques auxquelles les véhicules sont vulnérables ainsi que les techniques de détection actuelles. Nous nous efforçons d'y présenter l'évolution des approches en lien avec le domaine industriel de l'automobile et en adéquation avec leurs contraintes de manière à illustrer le besoin d'un mécanisme de détection d'anomalies dans les communications du véhicule.

— Le chapitre 2 présente notre système de détection d'anomalies **Svalinn** basé sur la représentation ontologique du trafic véhiculaire.

— Le chapitre 3 expose l'évaluation de notre système grâce à l'utilisation d'un outil d'émulation des communications et anomalies nommé **Autobot**. La qualité de la détection de l'algorithme en fonction des attributs utilisés ainsi que la consommation de ressources y sont évaluées.
— Le dernier chapitre conclut cette thèse en résumant nos contributions et présente des pistes d'améliorations possible dans la poursuite de ces travaux.

Chapitre 1

Contexte Scientifique et État de l'art

Sommaire

Nous présentons dans ce chapitre le contexte scientifique de cette thèse et l'état de l'art autour de la détection d'intrusions et d'attaques par déni-de-service dans

les réseaux. Nous nous intéressons en particulier aux réseaux cellulaires de véhicules, nous présentons les vulnérabilités liées aux véhicules connectés et les pistes exploitées par la communauté scientifique dans la détection de l'exploitation de ces vulnérabilités par des attaquants. Nous démontrerons cependant que l'application de méthodes de détection d'anomalies basées sur des algorithmes d'apprentissage est un chemin encore peu exploré pour détecter les attaques au niveau des communications cellulaires du véhicule connecté.

1.1 Le véhicule connecté

Le développement des systèmes de transports intelligents est entièrement basé sur les capacités des véhicules à échanger de l'information avec leur environnement tel que les autres véhicules, les piétons ou encore l'infrastructure routière comme les feux de signalisation ou les barrières de péage. La Figure 1.1 illustre les moyens de communication dont le véhicule dispose. Les véhicules modernes extraient, grâce à leur réseau interne de capteurs, des connaissances sur leur environnement proche. La position d'obstacles ou de panneaux de signalisation, les conditions météorologiques dans une zone précise, ou encore les lignes de démarcation sont des informations que le véhicule peut être en mesure de partager avec l'ensemble du système de transport intelligent. Il peut utiliser pour cela deux moyens de communication sans-fil, les réseaux cellulaires ou les réseaux ad-Hoc.

Nous présentons dans cette section les composants clés des véhicules contemporains leur permettant de communiquer et de réaliser ces tâches complexes. Nous nous intéressons tout particulièrement aux composants rendant le véhicule connecté à ces différents réseaux.

1.1.1 Les ECUs et le Bus-CAN

Le fonctionnement du véhicule moderne repose donc d'abord sur un réseau interne de plus en plus complexe. Les éléments connectés à ce réseau interne sont des unités de contrôle électroniques (ou *electronic-control-unit* ECU) qui remplissent des tâches très diverses au sein du véhicule. Au fil des années, ces systèmes sont devenus de plus en plus nombreux afin de proposer aux conducteurs des fonctionna-

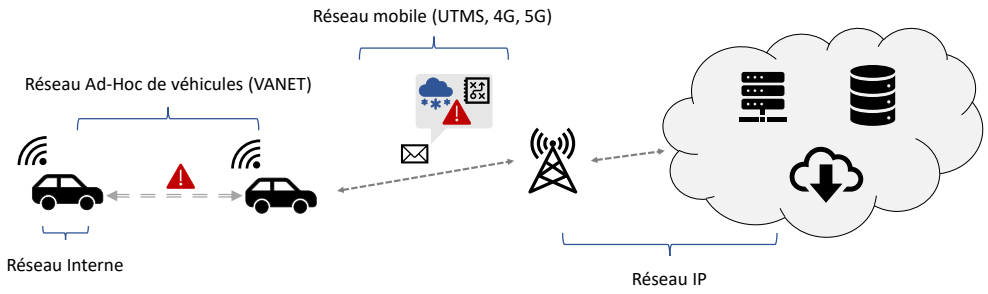


FIGURE 1.1 – Illustration des moyens de communication des véhicules

lités toujours plus innovantes. Du simple contrôle de la fréquence des essuies glace au maintien des distances de sécurité ou à la détection d'angles morts, ces systèmes deviennent incontournables dans la réduction des accidents. Ces ECUs recueillent des données contextuelles transmises par les nombreux capteurs disposés dans le véhicule comme les caméras, les radars ou encore les lidars. Historiquement, cette transmission est réalisée grâce à un bus interne au véhicule ; le bus-CAN (pour *Control-Area-Network*).

Le bus-CAN est un réseau principalement utilisé dans le domaine automobile qui a été normalisé par la norme ISO 11898. Auparavant, les ECUs des véhicules étaient connectés entre eux en point-à-point. Au fur et à mesure de la démocratisation de l'électronique embarquée, le nombre d'ECU augmenta considérablement rendant le câblage à l'intérieur du véhicule trop volumineux et coûteux. Le bus-CAN [18] a donc été développé de manière à remplacer l'approche point à point par un seul lien dans lequel tous les ECU raccordés au même bus peuvent communiquer avec tous les autres, réduisant ainsi la complexité et longueur de câblage total des véhicules. Suivant les constructeurs, il peut arriver d'observer plusieurs réseaux CAN différents à l'intérieur d'un même véhicule. Le bus-CAN est donc un réseau de broadcast où les messages CAN disposent tous d'un identifiant permettant aux ECU de déterminer les messages qu'ils doivent interpréter parmi l'ensemble des messages transmis sur le réseau.

Le Port OBD-II Parmi les autres composants connectés au bus-CAN, il en est un particulier, le port OBD-II (pour *on-board-diagnostic*). En effet, il s'agit d'un port de diagnostic accessible généralement depuis l'intérieur de l'habitacle du véhicule. Ce port est utilisé par les professionnels de l'automobile pour effectuer des diagnostics sur l'état du véhicule en lisant directement les trames apparaissant sur le bus-CAN ou en envoyant des requêtes particulières directement aux ECUs au moyen d'une «valise».

Le bus-CAN reste un composant coûteux du véhicule, ainsi le LIN (pour *Local Interconnect Network)* est souvent utilisé en complément du bus-CAN pour connecter en série plusieurs ECUs. Ce type de réseau est peu performant et peu fiable, il est donc souvent utilisé pour les ECUs responsables des fenêtres ou des équipements de climatisation, mais jamais pour des fonctions critiques comme l'accélération ou le freinage.

D'autres systèmes basés sur des communications sans-fil interagissent avec ce réseau comme les systèmes de surveillance de pression des pneus (ou *tire pression monitoring system* TPMS). Tous ces équipements offrent donc aux véhicules une vision à 360 degrés de leur environnement immédiat ou jusqu'à quelques centaines de mètres.

1.1.2 Les communications externes au véhicule

Pour autant, les constructeurs ont fait le constat que cette vision périphérique proche n'était pas suffisante pour assurer une complète sécurité des usagers des vé-

hicules. En effet, dans de nombreux cas, le véhicule peut voir son champ de vision obstrué par d'autres véhicules ou obstacles notamment à cause de la topologie de la route. Ainsi, la capacité des véhicules à communiquer et échanger de l'information entre eux constitue un palier à franchir pour la sécurisation des systèmes de transports.

Ces dispositifs de communication externes au véhicule sont regroupés au sein de TCUs (pour *telematic control units*). Ce sont des dispositifs embarqués et connectés au bus-CAN que le véhicule utilise pour ses communications externes. Ils sont équipés par exemple d'un système de géo-positionnement (GPS) et de plus en plus couramment d'un dispositif de communication cellulaire pour permettre aux véhicules d'accéder aux nouveaux services des systèmes de transports intelligents. Un autre mécanisme de communication externe au véhicule a longtemps été étudié par la communauté scientifique. Ce mécanisme basé sur les protocoles de communication de proche en proche sans fil permettent aux véhicules de former des VANETs (pour *vehicular-adhoc-networks*).

Les réseaux VANETS Les VANETS ont été conçus pour permettre aux véhicules d'échanger entre eux et avec des équipements d'infrastructure installés sur les routes dans l'objectif d'améliorer la sûreté des véhicules ainsi que la fluidité du trafic. Ces réseaux sont basés sur l'amendement 802.11p du standard IEEE 802.11 destiné à standardiser l'accès sans fil dans les environnements véhiculaires (WAVE). De cet amendement sont nés deux protocoles distincts, le premier DSRC (pour *dedicated short range communication*) porté par l'Amérique du Nord et le second porté par l'institut européen des normes de télécommunications (ETSI) concernant la standardisation des messages C-ITS (pour Cooperative intelligent transportation systems). Ces standards ont été établis pour permettre aux industriels de développer ces moyens de communication.

Dans ces réseaux, les véhicules en mouvement établissent spontanément des communications avec leurs voisins proches ainsi qu'avec les équipements installés au bord des routes. Les véhicules sont ainsi capables de transmettre des messages de proche en proche en s'appuyant sur ces équipements et sans avoir à se reposer sur le réseau cellulaire. Ces communications directes sont très efficaces et permettent d'obtenir de très faibles latences, mais leur portée est bien souvent inférieure à 1km. Les applications utilisant ces réseaux fournissent aux véhicules des services liés à la sûreté du véhicule et de ses passagers comme les systèmes d'évitement de collision, d'aide au dépassement ou à l'insertion dans le trafic voir une assistance au franchissement de croisements dangereux dans lesquels les conducteurs ont peu de visibilité.

Pour autant, bien que ces réseaux présentent des propriétés intéressantes, de nombreux verrous techniques ralentissent encore leur adoption par les acteurs du marché [73, 35]. Par exemple, la gestion du coût des équipements installés au bord des routes pour permettre aux véhicules d'accéder au réseau de l'infrastructure. Puisque ces réseaux sont uniquement créés entre les véhicules à proximité les uns

des autres, ils sont extrêmement diffus et sans infrastructure les informations captées par les véhicules peuvent mettre du temps à se propager si les véhicules ne peuvent pas établir de communication avec le réseau d'infrastructure ou d'autre véhicules. De plus, en cas de forte densité de circulation des problèmes de collision et d'ordonnancement des messages peuvent perturber le bon acheminement de ces derniers.

1.1.3 Conclusion

Tous ces équipements connectés sont donc des systèmes essentiels au bon fonctionnement des véhicules modernes et notamment des fonctionnalités de sûreté. Ainsi, la menace de voir ces systèmes affectés par des attaques informatiques est un problème qui a été soulevé très tôt par la communauté scientifique [67, 93]. Dans ces travaux, les chercheurs avaient anticipé que l'établissement de communications entre des véhicules et le reste du monde pourrait avoir un impact néfaste sur la sécurité de ces derniers.

En particulier, Lang *et al.* [67] présentent de nombreux scénarios réalistes notamment le scénario S7 où un attaquant pourrait envoyer des paquets mal-formés au véhicule, engendrant ainsi un dysfonctionnement du véhicule. Depuis, ces scénarios se sont avérés prémonitoires : de nombreux travaux ont démontré que les systèmes embarqués dans les véhicules étaient et sont toujours vulnérables à de nombreuses attaques. De plus, les véhicules autonomes sont d'avantage impactés par ces attaques à distance ou dirigées vers les capteurs du véhicule [96, 32] puisque le conducteur est remplacé par un système autonome qui s'appuie en continu sur les données accumulées par ses capteurs. Ainsi, il est nécessaire d'étudier des mécanismes de protection de ces véhicules notamment par la détection de ces attaques au niveau du réseau.

1.2 La détection d'intrusions et de déni-de-service

La détection d'intrusions ou d'attaques par déni-de-service est un domaine de recherche depuis longtemps exploré par la communauté scientifique. De nombreuses méthodes ont ainsi été conçues afin de détecter les intrusions ou attaques par déni-de-service dans les réseaux. Historiquement, celles-ci étaient d'abord basées sur les connaissances d'experts qui étaient chargés de produire des signatures précises d'attaques connues afin que ces dernières soient détectées par le système de détection si elles venaient à apparaître sur le réseau surveillé.

Par exemple, le système de détection d'intrusions open-source Suricata [1] utilise des règles comme celle présentée dans la Figure 1.2. On y distingue trois groupes, en rouge l'action qui sera réalisée si cette règle est déclenchée, en l'occurrence une alerte. L'entête de la règle est représenté en vert, dans ce cas on s'intéresse à n'importe quel flux TCP en partance du réseau surveillé (`$HOME_NET`) vers l'IP

1. https://suricata-ids.org/

```
alert tcp $HOME_NET any -> 154.35.64.82 80
(msg:"ET CNC Shadowserver Reported CnC Server Port 80";
flow:to_server,established; flags:S;
reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org;
threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil;
flowbits:set,ET.BotccIP; sid:2405000; rev:5731;
metadata:affected_product Any, attack_target Any, deployment Perimeter,
tag Shadowserver,
signature_severity Major, created_at 2012_05_04, updated_at 2020_05_12;)
```

FIGURE 1.2 – Exemple de règle utilisé dans le système de détection d'intrusions Suricata.

`154.35.64.82` sur le port `80`. Enfin, les options relatives à cette règle sont en bleu, on y trouve le message d'alerte qui sera affiché par Suricata (`msg`) et d'autres informations comme la référence vers une description de l'attaque (`ref:url`) ou encore le type d'attaque (`classtype`). Dans le cas présent, il s'agit d'une règle destinée à détecter du trafic généré par un Botnet désirant communiquer avec son centre de commande.

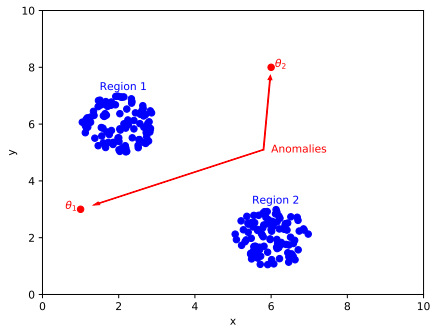
Ces méthodes sont extrêmement efficaces pour détecter les attaques connues avec grande précision. Cependant, elles sont incapables de détecter des attaques inconnues et la base des signatures doit constamment être mise à jour afin de se protéger contre les nouvelles attaques ce qui s'avère extrêmement coûteux. De plus, il peut arriver qu'il se passe plusieurs jours entre l'apparition d'une nouvelle attaque et sa découverte par les experts, durée pendant laquelle le système d'information reste à la merci de l'attaque.

Ainsi, d'autres méthodes comme celles basées sur les spécifications ont été conçues afin d'être en mesure de détecter des attaques inconnues ou des variations d'attaques existantes. Cependant, les méthodes basées sur la spécification des protocoles de communication ou des applications sont difficiles à mettre en place, car il est nécessaire d'établir le comportement de ces derniers de manière exhaustive ce qui est coûteux et complexe [16]. Ainsi, l'application de ces méthodes est restreinte à des scénarios où le nombre d'applications et protocoles est limité.

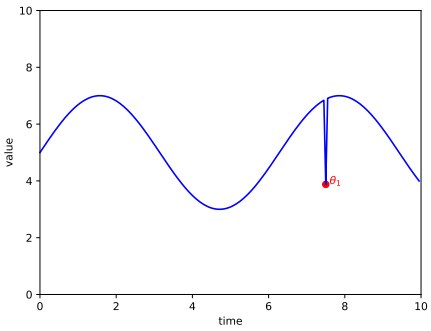
Une autre solution consiste en l'utilisation d'algorithmes d'apprentissage automatique appliqués à la détection d'anomalies. Un des premiers exemples de l'application de ces algorithmes dans les réseaux mobiles remonte à 1998 où Buschkes *et al.* [20] proposaient déjà une approche basée sur la création de profils d'utilisateurs et reposant sur des réseaux Bayésien pour la détection d'anomalies. Dans cette section, nous présentons les grandes familles d'approches par apprentissage automatique appliqués à la détection d'anomalies.

1.2.1 Terminologie

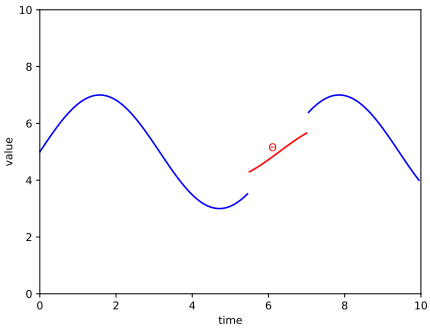
La détection d'anomalies dans les réseaux consiste en l'observation des communications à la recherche de motifs ou d'événements différents d'une situation



(a) Exemple d'anomalies ponctuelles.



(b) Exemple d'une anomalie contextuelle.



(c) Exemple d'une anomalie collective.

FIGURE 1.3 – Illustrations des différents types d'anomalies définis par [24].

normale. En effet, l'hypothèse est que les attaques informatiques telles que les tentatives d'intrusion ou les déni-de-service sont des événements rares par rapport à l'ensemble des communications sur le réseau surveillé. Chandola *et al.* [24] définissent plusieurs types d'anomalies : ponctuelle, contextuelle ou collective illustrées par la Figure 1.3.

1.2.1.1 Anomalie ponctuelle

Les anomalies ponctuelles représentent des instances de données considérées comme anormales en rapport avec le reste du corpus de données observé. La Figure 1.3a représente ces anomalies. On y observe deux régions de données dites habituelles, c'est-à-dire que l'ensemble des données du corpus ont tendance à être inscrites à l'intérieur de cette région. Les points θ_1, θ_2 quant à eux, représentent des anomalies à l'intérieur de ces données. Par exemple, durant une attaque par déni-de-service, le nombre de demandes de connexions auprès d'un serveur croît énormément de façon à priver les demandes légitimes de l'accès au serveur. Ainsi, cet afflux massif constitue une anomalie du fait du nombre aberrant de connexions au serveur à un instant donné.

1.2.1.2 Anomalie contextuelle

Les anomalies contextuelles sont des événements considérées comme anormaux par rapport à un contexte particulier, mais qui pourraient être considérés comme normaux dans un autre contexte. Une illustration de ce genre d'anomalies est représentée dans la Figure 1.3b. L'anomalie est illustrée par le point θ_1, on constate que sa valeur est présente dans le corpus de données, mais n'est pas prévue à l'instant auquel elle apparaît. Dans cet exemple, imaginons que le débit de données observé sur un poste de travail a pour habitude d'être élevé la journée et faible la nuit. Ainsi, constater un débit élevé la nuit pourrait être le signe d'une intrusion sur ce poste.

1.2.1.3 Anomalie collective

Les anomalies collectives sont liées à une collection de données liées entre elles et considérées comme anormales par rapport à l'ensemble des autres données. Ce type d'anomalie est illustré par la Figure 1.3c où la séquence Θ représente des points collectivement anormaux par rapport au reste des données. Pour autant, il n'est pas nécessaire que chaque instance à l'intérieur de la collection soit considérée comme anormale : seul l'ensemble est considéré comme tel. Un exemple d'attaque représentant ce genre d'anomalie serait une attaque par déni-de-service appliquant la méthode de *«slow growth»*. Dans cette méthode, au lieu de soumettre la victime à un trafic intense dès le début de l'attaque, celui-ci croît continuellement jusqu'à ce que la machine victime ne puisse plus répondre à de nouvelles requêtes. Cette méthode habitue le système au cours du temps à ne pas considérer l'attaque comme telle, mais simplement comme une évolution du trafic habituel.

1.2.2 Les corpus de données

Afin d'entraîner les différents algorithmes à reconnaître ces anomalies, les chercheurs doivent utiliser des corpus de données. Dans le cadre de la détection d'intrusions, ces derniers contiennent des communications jugées normales ainsi que des exemples d'attaques. Afin de vérifier les capacités d'un algorithme à détecter les attaques, ces corpus sont labellisés, c'est à dire que pour chaque instance du corpus sa classe est connue. Ainsi, un algorithme est évalué sur ses capacités à différencier une instance normale d'une instance anormale. De nombreux corpus sont cités tout au long de cet état de l'art. Leur utilisation permet aux chercheurs de comparer les résultats de leurs méthodes avec l'état de l'art. C'est pour cela que nous avons cherché à favoriser la mention dans cet état de l'art de travaux ayant recours tant que possible à ces corpus. Cependant, il est important de noter que tous les corpus ne se valent pas. En effet, pendant longtemps, le corpus KDD99 [61] a été la référence en matière de détection d'anomalies puisqu'il était un des seuls corpus proposant une grande variété d'attaques. Il a d'ailleurs subi de nombreuses modifications au cours du temps pour palier nombre de ces défauts [115]. Pour autant, ce corpus du siècle dernier n'est plus représentatif des attaques actuelles et d'autres jeux de

données créés récemment comme CICIDS2017 [104], UNSW [83] ou encore WSN [9] sont désormais privilégiés dans les travaux récents.

Les corpus de données sont essentiels au domaine de la détection d'anomalies dans les réseaux. Grâce à eux, de nombreuses méthodes ont été conçues au cours des dernières décennies. Nous présentons ci-après, les principales approches existantes à ce jour ainsi que des exemples d'utilisation des corpus pour l'évaluation de celles-ci.

1.2.3 Les modèles de Markov cachés

Les modèles de Markov cachés (ou HMM pour *hidden markov model*) sont une extension des chaînes de Markov. Celles-ci modélisent des séquences d'événements dans lesquels le passage d'un événement à l'autre dépend de l'état courant. Dans le cas d'un modèle de Markov caché, on suppose qu'il existe des états non observables dans cette chaîne.

On considère ainsi deux processus aléatoires X et Y où X n'est pas directement observable (il est caché), mais peut être observé à travers le processus Y qui produit des séquences de symboles observables. Ainsi, un modèle de Markov caché est défini par :

— N : le nombre d'état du modèle.
— M : le nombre de symboles.
— A : la distribution des probabilités de transitions d'état à état du processus X.
— B : la distribution des probabilités d'émission des symboles du processus Y
— Et λ représente l'état de départ.

L'objectif est donc d'apprendre à partir d'une séquence de données ou d'un jeu de séquences produit par Y, les meilleurs ensembles A et B des probabilités de transition et d'émission des états de X et Y. De nombreux algorithmes permettent d'estimer la vraisemblance maximale de ces ensembles comme l'algorithme Baum-Welch [13] et d'espérance-maximisation [81], l'algorithme de Viterbi [43] ou encore la méthode bayésienne de Monte-Carlo par chaînes de Markov [118].

Ariu *et al.* [11], appliquent donc les modèles de Markov cachés à la détection d'intrusions dans des flux HTTP. Ils extraient la charge utile des requêtes HTTP qu'ils modélisent sous la forme de séquences d'octets à l'aide d'une fenêtre glissante sur l'intégralité de chaque requête HTTP de leur jeu de données. Afin de réduire la complexité de leur modèle, ils échantillonnent de façon aléatoire l'ensemble des séquences créées lors de l'extraction. L'apprentissage consiste donc à estimer les probabilités de transition d'une séquence d'octets à une autre dans le cas d'un trafic normal, ce qui suppose *de facto* d'avoir accès à un corpus d'entraînement sain.

Ainsi, lors de la phase de détection, ils calculent les probabilités du modèle de Markov caché d'émettre la suite de séquences extraites d'une requête HTTP à analyser. Si cette estimation est faible alors la requête est considérée comme anormale. Afin d'augmenter leurs chances de détections les auteurs combinent plusieurs modèles de Markov cachés et discriminent chaque requête en fonction de la moyenne

des estimations de chacun des modèles.

Ils obtiennent ainsi de bons résultats dans la détection d'attaques XSS (99%) et shell code (98%) pour un faible taux de faux positifs (0.1%) en se basant sur le jeu de données de l'agence américaine pour les projets de recherche avancée de défense (DARPA) [70].

Dans d'autre travaux, Stefanidis et Voyiatzis [112] appliquent un HMM à la détection d'anomalies dans les réseaux industriels (ou SCADA pour *supervisory control and data acquisition*) et particulièrement dans les communications du protocole Modbus. Ils réalisent leurs expérimentations sur le jeu de données extrait de simulations d'un gazoduc, d'un château d'eau ainsi que d'un système de transmission d'électricité [82]. Ils comparent leurs résultats avec de nombreuses autres méthodes comme les réseaux bayésiens, les réseaux de neurones ou les forêts aléatoires et démontrent leurs capacités à détecter des injections de commandes avec 96% pour le cas simple, mais aucune détection pour les cas plus complexes. Ils sont capables de détecter à plus de 99% les attaques par déni-de-service ainsi que les processus de reconnaissance. Leur méthode montre aussi un faible taux de faux positifs (moins de 1%).

1.2.4 Les réseaux Bayésiens

Les réseaux bayésiens sont des modèles probabilistes représentés sous formes de graphes acycliques orientés. Ils permettent de modéliser les relations de dépendance entre un ensemble de variables d'intérêt. La classification naïve bayésienne est l'un des plus simples modèles de réseaux bayésien. Elle est basée sur l'hypothèse que les attributs d'une classe sont indépendants les uns des autres.

Le théorème de Bayes indique que pour chaque instance à classifier X, représentée sous la forme d'un vecteur à n attributs tel que $X = (x_1, ..., x_n)$, la probabilité qu'il appartienne à la classe k de l'ensemble des classes possibles C est défini telle que :

$$P(C_k|X) = \frac{P(C_k)P(X|C_k)}{P(X)} \tag{1.1}$$

Et puisque chaque attribut est considéré comme indépendant des autres, la distribution de C est définie telle que :

$$P(C_k|x_1, ..., x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{\sum_k P(C_k)P(X|C_k)} \tag{1.2}$$

Ainsi, les travaux présentés par Koc *et al.* [64] utilisent les réseaux cachés bayésiens naïfs pour la détection d'intrusions. Cette extension des réseaux bayésiens permet de ne plus considérer les attributs comme indépendants les uns des autres en créant un attribut caché représentant le parent d'un attribut dont la valeur peut être influencée par les valeurs des autres attributs en utilisant la formule d'infor-

mation mutuelle conditionnelle notée I_p tel que :

$$I_p(A_i, A_j|C) = \sum_{a_i,a_j,c} P(a_i, a_j, c) log \frac{P(a_i, a_j|c)}{P(a_i|c)P(a_j|c)} \quad (1.3)$$

Où A_i et A_j représentent deux attributs et a_i, a_j leur valeurs respectives, C la variable de classe de l'élément contenant les attributs de l'ensemble A et c sa valeur. Cette méthode permet de considérer les attributs qui jouent un rôle commun dans la classification d'un élément et d'influencer la classification en pondérant ces attributs.

Ils réalisent leurs expérimentations en se basant sur le corpus de données KDD99 [61] et évaluent les capacités de détection de leur méthode et la comparent notamment avec l'approche naïve bayésienne. Ils obtiennent un score de 93.7% de détection pour 6% de faux positifs avec toutefois de meilleurs résultats dans la classification des attaques par déni-de-service (99.6% pour 0.4% de faux positifs). En comparaison, l'approche par réseau bayésien n'obtient que 90% de détection pour 9.4% de faux positifs et l'approche naïve bayésienne 70% pour 21% de faux positifs.

Dans des travaux plus récents sur le même jeu de données, Zhang *et al.* [132] comparent leurs résultats avec trois autres méthodes, celle des plus proches voisins (K-NN), la machine à vecteurs de support et les arbres de décision. Leur méthode semble donner de moins bons résultats de détection, avec un ratio de 86% contre 93% pour KNN et les arbres de décision et 89% pour la machine à vecteurs de support. Cependant, la phase d'apprentissage et de validation de leur algorithme est réalisée en moins de 2 secondes contre 48 minutes pour K-NN, 10 heures pour la machine à vecteur de support et 2 minutes pour les arbres de décision.

1.2.5 Les règles d'associations

D'autre systèmes de détection utilisent les règles d'associations [4]. Là où les réseaux bayésiens admettent une indépendance entre chaque attribut d'un élément, les règles d'associations ont pour objectif de découvrir des liens entre les attributs à l'intérieur de registres de transactions. En d'autres termes, prenons $I = i_1, ..., i_n$ l'ensemble des attributs appelés objets, et $D = t_1, ..., t_m$ l'ensemble des transactions. Chaque transaction t_i contient un sous-ensemble de I à partir duquel les règles $I_k \Rightarrow I_l$ associant deux éléments entre eux sont créées. Plusieurs mesures existent de façon à choisir les règles les plus intéressantes parmi l'ensemble des règles possibles. Les principales étant l'indice de support qui représente la probabilité d'avoir les éléments I_k et I_l dans une même transaction ($P(I_l \cap I_k)$) et l'indice de confiance définis comme la probabilité conditionnelle $P(I_l|I_k)$.

L'apprentissage des règles se fait donc en deux étapes, la première consistant à appliquer un seuil minimal de support pour obtenir l'ensemble F des sous-ensembles les plus fréquents trouvés dans D. Puis, dans un second temps, déterminer un seuil de confiance minimal afin d'extraire de F les règles d'associations.

Par exemple, Tajbakhsh *et al.* [114] présentent une solution de détection d'intrusions basée sur les règles d'associations floues. Ils comparent les résultats de la

détection de leur méthode avec d'autres méthodes comme du clustering ou l'utilisation de l'algorithme des plus proches voisins (K-NN). Le jeu de données utilisé est KDD99 [61]. La notion de règle d'association floue vient du fait que certains attributs manipulés par les règles d'associations comme par exemple la durée d'une session TCP, ou le débit moyen observé au niveau d'un serveur sont des variables continues et il est donc impossible de créer une règle par valeur différente possible. Ainsi, la notion de flou sert à définir des ensembles de valeurs en prenant en compte la distribution d'une variable à l'intérieur du jeu de données. Ainsi, ils définissent donc trois ensembles représentant des valeurs basses, moyennes et hautes à partir desquelles il est possible de créer les règles d'associations. Les objets sont donc représentés par un couple attribut, valeur comme par exemple (durée, petite) ou encore (protocole, TCP) pour chaque session représentée dans le corpus de données. Leur processus de détection génère ainsi un certain nombre de règles leur permettant de classifier les objets et ainsi détecter les différentes classes d'attaques. Ils évaluent leur système en fonction du nombre de faux positifs et de ratio de détection et obtiennent un score de 80% de détection pour 2.9% de faux positifs. En comparaison, les autres approches présentent un meilleur taux de détection (plus de 90%), mais pour un taux de faux positifs supérieur à 8%.

Dans des travaux plus récents, Chan *et al.* [23] appliquent la méthode des règles d'associations floues à la détection d'attaques au niveau des services web comme les injections SQL et XML ou les déni-de-service distribués avec un taux de détection proche des 100% pour moins de 1% de faux positifs.

1.2.6 Le clustering

L'analyse par clustering est une autre méthode très utilisée dans la détection d'intrusions et en particulier des attaques par déni-de-service. L'objectif du clustering est de regrouper un ensemble d'objets de façon à ce que les objets considérés comme similaires appartiennent au même groupe (cluster). Détecter une anomalie dans ce contexte consiste à déceler parmi l'ensemble des objets ceux n'appartenant à aucun groupe ou à un groupe de petite taille. De nombreuses méthodes de clustering existent, nous citons ici les exemples les plus employés dans la détection d'anomalies dans les réseaux.

L'algorithme K-means est une méthode couramment employée dans le clustering. Par exemple, Münz *et al.* [84] présentent son utilisation pour la détection d'anomalies dans les réseaux. L'algorithme K-means regroupe les objets en fonction de leurs attributs dans k clusters différents. Il peut être résumé de la façon suivante :

1. Choisir le nombre k de clusters.
2. Initialiser le centroïde de chacun des k clusters.
3. Assigner chaque objet à un cluster en fonction de la distance euclidienne avec les centroïdes.
4. Recalculer les centroïdes en fonction des objets à l'intérieur des clusters formés.

5. Répéter l'opération jusqu'à stabilisation des centroïdes.

Dans leurs travaux, les auteurs utilisent l'algorithme pour à la fois classifier les données présentées à l'algorithme et ainsi déterminer pour un objet si celui-ci appartient à la classe normale ou anormale, mais aussi pour détecter les valeurs aberrantes (plus couramment appelés *outliers*). Une valeur aberrante est un objet pour lequel la distance avec le centroïde le plus proche est supérieure à un seuil prédéfini. Ainsi, cette valeur n'étant pas associée à l'un des clusters est donc considérée comme un outlier et donc comme une valeur anormale appartenant à une classe jusqu'alors inconnue.

D'une manière similaire, les travaux de Gaddam *et al.* [45] présentent une comparaison entre l'utilisation de K-means de façon combinée avec des arbres de décision et obtiennent plus de $99,12\%$ de précision pour 3% de faux positifs sur le corpus de données NAD-1999 [71].

Plus récemment, les travaux présentés par Dromard *et al.* [39] rapportent de l'utilisation de l'algorithme ORUNADA basé sur un algorithme de grid-clustering sur un trafic plus récent issu du corpus ONTIC [92]. Ils démontrent les capacités de leur algorithme à opérer en ligne et réaliser la détection d'anomalies en temps réel. De plus, ils obtiennent des meilleurs résultats de détection à savoir plus de 93% de taux de détection pour moins de 0.05% de faux positifs par rapport à leur solution précédente, UNADA [22] basée sur l'algorithme DBSCAN.

1.2.7 Les arbres de décision

Les arbres de décision sont des approches prédictives à la classification d'éléments. A partir d'observations d'un élément correspondant aux branches, l'arbre est utilisé de façon à déduire la classe d'un objet. Celle-ci est représentée par une feuille de l'arbre.

Par exemple, une tentative d'intrusion sur un serveur FTP par force brute pourrait ressembler de façon simpliste à l'arbre illustré en Figure 1.4. Le port 21 est celui utilisé pour le protocole de transfert de fichier FTP, la connexion au serveur requiert d'entrer un mot de passe dont l'arbre de décision estime le 100ème essai infructueux comme une tentative d'intrusion et le classifie comme étant du trafic malveillant.

Les travaux de Moon *et al.* [80] présentent un système de détection d'intrusions basé sur les arbres de décisions pour la prévention des menaces persistantes avancées ou APT (pour *advanced persistent threat*). Ainsi, ils combinent à la fois des informations tirées du réseau, mais aussi des événements occurrents sur le système où est effectuée la détection dans le but de détecter des logiciels malveillants (ou *malwares*). Ces attributs sont construits à partir de l'analyse du comportement de plusieurs malwares en comparaison avec des logiciels normaux. Les attributs liés au comportement d'un malware sont donc répertoriés dans un arbre de décision, on y trouve notamment l'action de terminaison d'un processus, le lancement d'un téléchargement ou encore l'accès à des fichiers temporaires. Les auteurs obtiennent grâce à cet arbre un taux de détection de 84.7%.

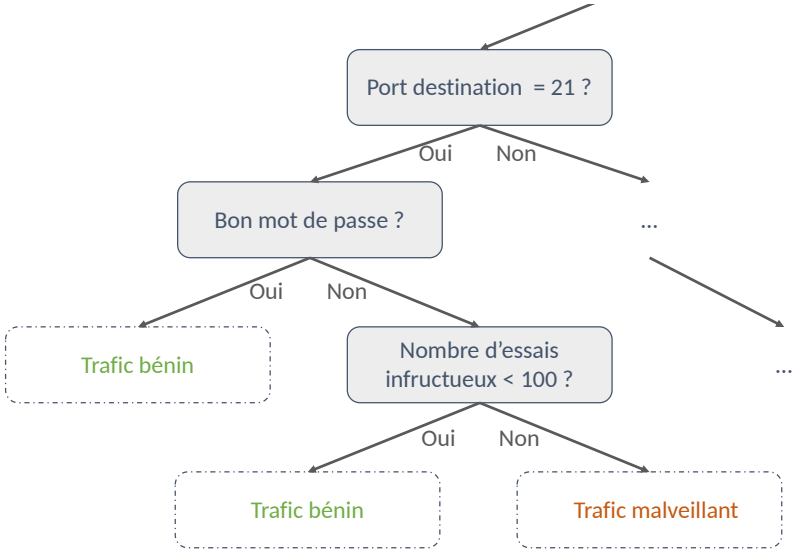


FIGURE 1.4 – Illustration d'une tentative d'intrusion sur un serveur FTP par force brute.

Dans des travaux orientés uniquement autour des communications réseaux, Peng *et al.* [95] utilisent la technique des arbres de décision pour la détection d'intrusions en utilisant le corpus de données KDD99 [61]. Ils comparent leur approche avec celle des réseaux bayésiens ainsi que l'algorithme des K plus proches voisins. Quant à la vitesse d'exécution leur algorithme est capable de traiter l'intégralité du corpus en 3 secondes. Comparativement KNN met un peu moins de 2h pour traiter 10% du corpus. Les auteurs se sont intéressés aux capacités de détection de leur méthode en fonction des attaques considérées. Par exemple, là où l'algorithme KNN est capable de détecter les attaques par déni-de-service ainsi que les scans avec 100% de fiabilité, il n'obtient que 85% de détection pour les intrusions et 0% pour les escalades de privilèges tandis que leur méthode est capable de détecter toutes les attaques. On notera cependant que le score de détection pour l'escalade de privilège bien que non nul, reste faible (35%).

Ainsi, Ahmim *et al.* [7] ont quant à eux utilisé un corpus beaucoup plus récent pour valider leur approche basée sur les arbres de décision et les règles : CICIDS2017 [104]. Dans leur approche hiérarchique, le détecteur d'anomalies base ses décisions sur 3 modèles (RepTree, Jrip et Forest PA [3]) qui améliorent la classification de l'arbre de décision construit. Ils obtiennent ainsi un taux de détection de 94.4% pour 1.1% de faux positifs.

1.2.8 L'apprentissage ensembliste

De la même façon que le clustering, il existe de nombreuses techniques permettant de construire les arbres de décision. Une approche très employée dans la détection d'intrusions est la méthode ensembliste des forêts à décisions aléatoires (Random Forests). Les méthodes ensemblistes combinent plusieurs algorithmes d'ap-

prentissage pour obtenir de meilleures prédictions. Les forêts à décisions aléatoires sont donc constituées d'un ensemble d'arbres de décision générés de façon à ce qu'ils ne soient pas corrélés entre eux. Ces forêts classifient les instances qui leurs sont présentées en prenant part à un vote. La classe majoritaire sera celle affectée à l'objet.

Par exemple, les travaux présentés par Ustebay *et al.* [117] utilisent les forêts aléatoires afin de réduire la complexité du corpus de données CICIDS2017 [104] en sélectionnant les meilleurs attributs pour la classification. Cette dernière est quant à elle, réalisée grâce à un réseau de neurones artificiels. Les auteurs parviennent à réduire la taille des données manipulées de 95% et obtiennent un score de détection de 91% pour un taux de faux positifs de 18%.

Un autre exemple d'apprentissage ensembliste appliqué à la détection d'intrusions est présenté par Zhou et Cheng [133]. Les auteurs utilisent trois classifieurs pour la détection d'intrusions basés sur les arbres de décision (C4.5 [103]), les forêts aléatoires ainsi que Forest PA [3]. Ils appliquent leur méthode sur les différents corpus disponibles à savoir KDD99 [61], CICIDS2017 [104] et AWID [12] sur lesquels ils obtiennent des résultats de détections supérieurs à 99.5% pour un taux de faux positifs compris entre 8% et 15%.

1.2.9 Les machines à vecteurs de support

Les machines à vecteurs de support ou séparateurs à vaste marge (SVM) sont souvent utilisés en tant que classificateur binaire pour la détection d'anomalies. Le principe de fonctionnement est de trouver un hyperplan séparateur à l'intérieur d'un ensemble de données à plusieurs dimensions. L'objectif est donc de trouver l'hyperplan qui maximise la marche séparant les différentes classes de trafic. Bien qu'efficace, cette méthode requiert l'accès à des données labellisées et un temps d'entraînement souvent très grand.

Par exemple, dans l'une des premières utilisations de SVM pour la détection d'anomalies [63] la durée d'apprentissage est réduite de 17h à 13h pour une précision de détection de 69%.

Ainsi, Vijayanand *et al.* [120] utilisent sept modèles de SVM. Chacun d'entre eux est dédié à la détection d'une classe d'attaque présente dans les corpus de données CICIDS2017 [104] et ADFA-LD [31]. Ainsi, pour le cas du CICIDS2017, les attaques sont toutes détectées avec un taux de plus de 99% pour un taux de faux positifs maximal de 2%.

1.2.10 Les réseaux de neurones artificiels

Enfin, les approches par réseaux de neurones artificiels ont aussi été employées dans la détection d'intrusions et de déni-de-service. Ces réseaux connectent des neurones artificiels sous la forme de graphes aux structures diverses. Les neurones artificiels qui peuplent ces réseaux réalisent des fonctions simples à plusieurs entrées et une seule sortie. Les sorties peuvent être à leur tour connectées à un ou plusieurs

autres neurones. Le fonctionnement de base de ces neurones artificiels consiste donc à calculer la somme pondérée des valeurs en entrée du neurone suivant la formule suivante [113] :

$$y = f(\sum_{i=1}^{n} W_i \times x_i) \tag{1.4}$$

Où,

— y représente la valeur de sortie du neurone et W_i, x_i représentent respectivement le poids et la valeur d'entrée du neurone.

— La fonction $f(.)$ quant à elle représente la fonction d'activation du neurone.

De nombreuses fonctions non linéaires sont utilisées pour l'activation des neurones comme les sigmoïdes, les hyperboles ou les différentes formes de fonctions de rampes : *Rectifier Linear Units, Leaky Rectifier Linear Units* ou encore *Exponential Linear Units*.

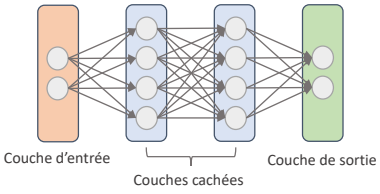
L'apprentissage profond (ou *Deep Learning*) consiste à empiler de nombreux neurones sous la forme de couches dans l'objectif de construire à l'intérieur du réseau une abstraction des attributs des données présentées à l'algorithme. Différents types de réseaux de neurones artificiels existent, les plus courants sont les réseaux profonds, récurrents ou antagonistes génératifs comme illustrés dans la Figure 1.5. Les réseaux profonds sont des réseaux de neurones disposant d'au moins une couche cachée. En d'autres termes, il existe au moins une couche de neurones entre celles disposées à l'entrée et à la sortie du réseau.

Dans les réseaux de neurones récurrents, la valeur de sortie dépend de l'entrée qui est présentée au réseau, mais aussi de la sortie du réseau à l'itération précédente, ils sont notamment utilisés dans l'apprentissage de séquences.

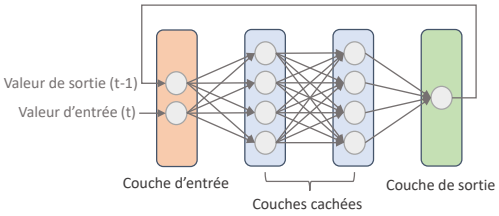
Enfin, les réseaux de neurones antagonistes génératifs mettent en compétition deux réseaux afin de générer à partir d'un jeu de données d'entraînement, de nouvelles données présentant les mêmes propriétés. Le processus implique de présenter à un réseau discriminant des données générées par le réseau générateur et de faire en sorte d'entraîner le réseau générateur à créer des données que le discriminant ne sera pas en mesure de différencier des véritables données d'entraînement.

Ainsi, les travaux présentés par Vinayakumar *et al.* [121] illustrent l'utilisation d'un réseau de neurones profond pour la détection d'intrusions. Ils procèdent à une comparaison approfondie des capacités de détection des différents réseaux de neurones qu'ils ont créé pour chacun des corpus de données qu'ils utilisent (KDD99 [61], NSL-KDD [115], UNSW [83], WSN [9], CICIDS2017 [104], Kyoto [110]). Ils comparent également la méthode des réseaux de neurones avec d'autres méthodes de détection d'anomalies plus classiques comme Adaboost, les forêts aléatoires, SVM, KNN et les arbres de décision. Leurs comparaisons portent sur la classification binaire et multi-classes à savoir la capacité de l'algorithme à déterminer si la donnée traitée est anormale ou non, mais aussi la classification directe à une attaque particulière. Ils constatent que la méthode des réseaux de neurones profonds est meilleure que les méthodes plus classiques.

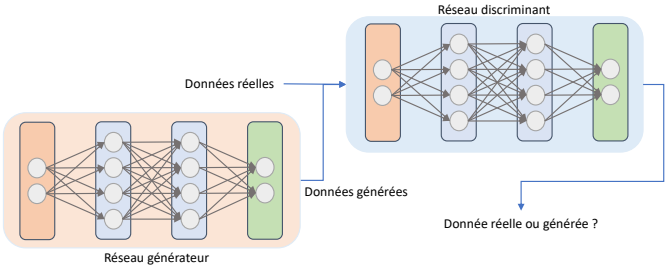
Dans une autre approche Yin *et al.* [127] s'inspirent des progrès des résultats



(a) Réseau de neurones profond.



(b) Réseau de neurones récurrent.



(c) Réseau de neurones antagonistes génératifs.

FIGURE 1.5 – Différents types de réseaux de neurones artificiels.

des réseaux de neurones pour la détection d'anomalies et notamment ceux présentés par Javaid *et al.* [57]. Ils utilisent un modèle d'apprentissage supervisé basé sur un réseau de neurones récurrent (RNN ou *recurrent neural network*) et comparent leurs résultats avec d'autres algorithmes grâce au corpus NSL-KDD [115]. Ce corpus contient un dataset d'apprentissage ($KDDTrain^{+}$) sur lequel leur modèle obtient une précision de 99.81%. Dans la phase de tests leur modèle n'obtient plus que 83.28% de précision sur $KDDTest^{+}$ et 68.55% sur $KDDTest^{-21}$. Le corpus utilisé dans ces travaux contient 4 types d'attaques à savoir :

— Probe : Reconnaissance.
— Dos : Déni-de-service.
— U2R : Escalade de privilèges de l'utilisateur de la machine.
— R2L : Accès distant non autorisé par un attaquant à la machine de la victime.

Ainsi, lorsqu'on observe les capacités de détection par attaque de la méthode présentée, on constate qu'elle est très efficace à détecter des attaques par déni-de-service ou de reconnaissance, mais échoue pour les autres types d'attaques avec 11.5% et 24.7% de précision pour les attaques U2R et R2L.

Enfin, les travaux de Zenati *et al.* [130] proposent une solution à base de réseaux de neurones antagonistes génératifs qu'ils testent notamment sur le corpus de données KDD99 [61] et comparent avec SVM et d'autres algorithmes basés sur les réseaux de neurones. Ils obtiennent une précision de détection de 94% et un rappel de 95%.

1.3 Les vulnérabilités du véhicule connecté

Il existe donc quantité de méthodes pour la détection d'anomalies dans les réseaux. Nous avons présenté dans la section précédente les principales familles d'algorithmes et leurs capacités à détecter certains types d'attaques dans les réseaux traditionnels ou sans-fil.

Dans cette section, nous présentons les différentes vulnérabilités ayant été exploitées par des attaquants afin de compromettre à distance la sécurité des véhicules connectés. En particulier nous nous intéressons tout particulièrement à la compromission des composants de communication cellulaires donnant lieu à un accès au bus-CAN.

D'autres études traitent plus en détail de la surface d'attaque des véhicules et notamment des équipements accessibles uniquement par des attaques à proximité physique comme les lecteurs CDs ou les équipements Bluetooth. Par exemple, les travaux de Checkoway *et al.* [27] présentent l'une des toutes premières études destinées à établir la surface d'attaques des véhicules contemporains (2011) et propose des recommandations pour la sécurisation de futurs véhicules. De plus, Miller et Valasek [77] présentent une étude de la surface d'attaque à distance des véhicules connectés reposant sur l'étude de nombreux modèles du marché produits entre 2006 et 2014. Les chercheurs y rapportent notamment des vulnérabilités sur les composants Bluetooth, TPMS, ou encore sur le système de Radio ainsi que sur le système de communication cellulaire et les applications tierces connectées à Internet.

1.3.1 Le bus-CAN, objectif privilégié des attaquants

A l'instar des réseaux traditionnels où l'objectif ultime des attaquants est de devenir *«root»*, à savoir l'utilisateur disposant d'un maximum de privilèges sur une machine d'un réseau attaqué. Le Graal des pirates en ce qui concerne les véhicules est son réseau interne ou bus-CAN. Ils doivent, pour y accéder, exploiter de multiples vulnérabilités présentes sur le véhicule en formant ainsi une chaîne d'exploits (de l'anglais *exploits chain*).

Cependant, la sécurité des communications n'a pas été pensée dans le développement du standard. En effet, chaque ECU communique en broadcast et sans chiffrement, rendant chaque message lisible par les autres membres du réseau. Ainsi, dès lors qu'un attaquant obtient l'accès à ce réseau, il est en capacité de stocker et analyser chacun des messages transmis sur ce bus. De plus, si un attaquant est capable de contrôler un composant connecté au bus-CAN, il sera alors en capacité d'envoyer des messages sur celui-ci et perturber les autres ECUs comme ceux responsables du freinage ou de l'accélération du véhicule. Dans les exemples d'attaques que nous présentons dans cet état de l'art nous nous attarderons particulièrement sur celles permettant aux attaquants l'accès à ce bus.

1.3.2 Le port OBD-II

Le port OBD-II a directement accès au bus-CAN du véhicule, il représente donc lui aussi une cible potentielle pour un attaquant. Ainsi, même si ce port était auparavant uniquement accessible physiquement par l'attaquant, il existe désormais un marché proposant des services connectés basés sur l'installation d'un équipement connecté sur ce port. Les propriétaires du véhicule peuvent ainsi accéder à des services comme le suivi de la consommation, ou procéder à des diagnostics eux-mêmes sans avoir recourt à un professionnel. Pour autant, ces équipements représentent un vecteur d'attaque supplémentaire contre le véhicule puisqu'ils peuvent permettre à un attaquant d'accéder au bus-CAN en les compromettant.

Ce genre d'appareil est particulièrement populaire aux États-Unis pour les services d'assurance au kilomètre. Or, il a été démontré par Foster *et al.* [44] que ces dispositifs pouvaient être exploités de façon à obtenir le contrôle de certains composants du véhicule.

1.3.3 Les réseaux véhiculaires ad-hoc (VANET)

Enfin, les réseaux VANETS sont prompts à de nombreux scénarios d'attaques [98, 52] dus à la nature décentralisée du réseau et du manque d'authentification des nœuds participant aux communications. Ces attaques sont principalement orientées autour de l'information partagée par les véhicules plutôt que la prise de contrôle du véhicule par l'exploitation d'une vulnérabilité. Ainsi, les attaques Sybil [38] par exemple, consistent à faire croire aux autres usagers qu'un grand nombre de véhicules circulent sur une portion de route afin d'influencer leurs comportements. Pour ce faire, l'attaquant forge des fausses identités de véhicules que les autres vont considérer comme authentiques ce qui constitue un réel problème pour la sécurité et la fiabilité de ces réseaux. D'autres attaques comme les trous noirs ou trous de vers, sont liées au routage des paquets entre les véhicules. Elles consistent à forcer les autres véhicules à rediriger leurs messages vers un nœud contrôlé par l'attaquant qui pourra ainsi décider ou non de les transmettre.

Bien que de nombreux scénarios d'attaques existent au niveau des réseaux VANETS, nous constatons ces dernières années que la plupart des constructeurs de véhicules favorisent déjà l'utilisation des réseaux cellulaires [2] à défaut des VANETs dont la pénétration du marché sera plus longue compte tenu des coûts d'infrastructure engendrés par la mise en place de ces réseaux [3]. De plus, les nouveaux standards de communication notamment celui de la 5G [4] prévoient des cas d'utilisation de communications entre véhicules en passant par les antennes relais déjà installées pour les réseaux cellulaires [25]. En effet, des études comme [126, 14, 15, 123] démontrent les capacités des réseaux cellulaires actuels à fournir une qualité de service suffisante

2. Voir le rapport Ericsson sur la mobilité https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf

3. Voir l'étude de 5GAA https://5gaa.org/wp-content/uploads/2019/01/5GAA-BMAC-White-Paper_final2.pdf

4. https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf

pour les applications dans lesquelles le temps de propagation de l'information vers un véhicule est critique.

1.3.4 Le système d'infotainment et applications smartphone

Dans les exemples présentés précédemment, les attaquants interagissent directement avec le véhicule depuis internet. Cependant, il a aussi été démontré que certains véhicules étaient attaquables en passant par un intermédiaire. En effet, de nombreux constructeurs proposent désormais des applications pour smartphone permettant aux propriétaires d'accéder à des fonctionnalités supplémentaires comme le préchauffage du véhicule ou l'accès au taux de chargement des batteries pour les véhicules électriques. D'autres applications plus controversées comme le Smart Summon disponible sur les véhicules de la marque Tesla permet aux propriétaires de faire se déplacer leur véhicule de façon à ce qu'il se gare de façon autonome. Dans ces cas d'utilisation, le véhicule est connecté avec un serveur qui reçoit les requêtes des applications installées sur les smartphones des utilisateurs.

Ainsi, si un attaquant est capable de communiquer avec ce serveur et que celui-ci interprète les requêtes, alors il peut être possible de communiquer avec n'importe quel véhicule proposant cette fonctionnalité, comme ce fut le cas pour le modèle Leaf (2016) de Nissan via l'application NissanConnectEV [1] ou le modèle 320d Touring (2015) de BMW et son application ConnectedDrive [2].

1.3.5 Le smartphone comme vecteur d'attaque

Les travaux présentés par Woo *et al.* [124] s'intéressent à un scénario d'attaque via l'utilisation d'une application smartphone vérolée. Dans leurs expérimentations, les chercheurs ont démontré qu'ils étaient en mesure de diffuser largement une application de diagnostic automobile malveillante sur le marché d'applications smartphone Android. Sous couvert de permettre aux utilisateurs de connecter leur téléphone au véhicule, leur permettant ainsi de consulter la consommation ou les erreurs éventuelles détectées sur les ECUs de ce dernier. L'application agit en réalité comme un intermédiaire entre l'attaquant et le véhicule. Elle est ainsi capable de propager un message depuis le serveur contrôlé par l'attaquant vers un véhicule connecté à l'application qui, par exemple, pourrait stopper le moteur ou déclencher une accélération.

De façon similaire, Mazloom *et al.* [75] présentent un autre moyen d'attaque dirigée vers un véhicule par le biais de l'appariement du véhicule avec le smartphone de l'utilisateur dans lequel serait installée une application malveillante permettant à un attaquant de compromettre le véhicule de la victime. Cette attaque profite d'une vulnérabilité dans l'implémentation du protocole Mirror-Link sur un véhicule produit en 2015. Cette vulnérabilité permet l'exécution de code arbitraire qu'il est possible d'exploiter afin d'afficher des messages sur l'écran du véhicule. Le protocole Mirror-Link étant très répandu dans le monde automobile pour l'appariement des smartphones aux systèmes de divertissement à l'intérieur des véhicules (IVI pour In-

Vehicle-Infotainment systems), cette attaque représente elle aussi un danger sérieux pour de nombreux véhicules.

1.3.6 L'unité de Contrôle Télématique (TCU)

L'unité de contrôle télématique constitue le point d'entrée des communications à longues distances du véhicule. Ainsi, il constitue un vecteur d'attaque privilégié puisqu'il permet aux attaquants d'accéder au véhicule à distance.

Peut-être la plus célèbre occurrence d'exploitation de vulnérabilité à distance de véhicules connectés est celle rapportée par les chercheurs Miller et Valasek [78] en 2015. Forts de leur expérience précédente dans la manipulation du bus-CAN de véhicules [76], ils rapportent de façon extrêmement détaillée de nombreuses vulnérabilités permettant de prendre le contrôle du véhicule à distance. Leurs découvertes ont engendré un rappel historique de plus de 1.2 million de Jeep Cherokee (2014).

En particulier, ils ont découvert que le système d'infotainment installé sur le véhicule (Uconnect) exécutait un service de communication inter-processus (D-Bus) et que ce service était connecté à un port du véhicule traditionnellement utilisé pour irc [5] (6667). Or, ce véhicule était aussi équipé d'un modem de communication cellulaire et il s'est avéré qu'un fournisseur d'accès américain permettait la communication IP entre équipements. Ainsi, les chercheurs étaient capables de déclencher des attaques à distance permettant de contrôler des composants du véhicule comme les freins, la musique, ou encore le système de ventilation. De plus, ils ont réalisé que le fournisseur d'accès affectait des adresses IPs aux véhicules sur une plage prédéfinie. Ils ont scanné cette plage afin de détecter tous les véhicules vulnérables à cette attaque, entraînant ainsi d'autres marques comme Dodge, Chrysler ou Ram dans ce rappel massif de véhicules.

Ces découvertes ont motivé d'autres travaux dans de nombreuses équipes de recherche. Ceux des chercheurs de KEEN Security Lab ont été considérés comme les dignes successeurs du duo Miller, Valasek. En effet, ils ont eux aussi démontré à plusieurs reprises que des véhicules d'autres marques comme Tesla ou BMW étaient aussi affectés par de nombreuses vulnérabilités. Leurs premières découvertes se sont concentrées sur les véhicules de la marque Tesla [88]. La première vulnérabilité repose sur une version (v7.1) du navigateur internet installé sur les modèles S (2016). Son exploitation permet l'exécution de code arbitraire pouvant conduire à la manipulation du véhicule grâce à l'envoi de messages sur le bus-CAN. Les chercheurs ont démontré la faisabilité de l'attaque en accédant au service WiFi du véhicule. Cependant, ils estiment qu'il serait possible de piéger les utilisateurs du véhicule à accéder à des noms de domaines contrôlés par un attaquant afin d'exploiter cette même vulnérabilité.

Dans d'autres travaux [89], la même équipe s'est concentrée sur la fonctionnalité de mise à jour à distance (ou OTA pour *over-the-air* update). Cette fonctionnalité, était l'une des forces de la marque Tesla qui depuis a été reprise par de nombreux

5. Inter-relay-chat, est une application populaire de discussion dans la communauté informatique

autres constructeurs. En effet, ces mises à jour à distance permettent de diffuser largement des améliorations ou correctifs directement sur les véhicules sans passer par les garagistes comme c'était traditionnellement le cas. Ainsi, les chercheurs ont découvert qu'il était possible de compromettre le module de conduite autonome en suivant une chaîne d'exploits découverte sur le système d'infotainment (Center Information Display) puis en se déplaçant latéralement dans le réseau interne au véhicule pour atteindre l'ECU responsable de l'«AutoPilot». Ils démontrent ainsi qu'un attaquant pourrait décider d'attenter à la sécurité de l'utilisateur si ce dernier décide d'utiliser le mode «AutoPilot» après que l'attaquant ait modifié le code de ce dernier.

Enfin, dans les travaux de Cai *et al.* [21] présentés eux aussi à la célèbre conférence de sécurité BlackHat (USA), les chercheurs ont découvert des vulnérabilités sur les véhicules BMW et notamment une autre chaîne d'exploits à distance de ces derniers. Dans ce scénario, une attaque de l'homme-du-milieu est possible afin d'intercepter les requêtes du service *ConnectedDrive* installé sur le véhicule et de modifier son contenu pour compromettre le navigateur internet du véhicule. Il est ensuite possible là aussi de se déplacer latéralement à l'intérieur du véhicule afin d'injecter des messages sur le bus-CAN de ce dernier.

Plus récemment, une équipe de chercheurs des laboratoires de Computest ont découvert des vulnérabilités majeures sur les systèmes d'infotainment des véhicules de marque Volkswagen et Audi [29]. Certains véhicules de ces marques sont équipés d'un boîtier comportant une carte SIM permettant à la voiture de communiquer par Internet. Or, une des vulnérabilités rapportées dans ces travaux est notamment accessible directement en effectuant un scan de l'adresse IP du véhicule. Ainsi, et puisqu'aucun correctif ne peut être propagé à distance sur ces véhicules, les chercheurs ont décidé de ne pas détailler publiquement la vulnérabilité dans son ensemble. Il est intéressant de noter la différence dans la gestion de l'incident avec l'exemple Miller et Valasek où les chercheurs avaient décidé de publier les résultats en détails malgré l'impossibilité de corriger les véhicules à distance. Ainsi, ils ont simplement informé les constructeurs de cette vulnérabilité. Il est cependant important de noter que pour que cette attaque soit réalisable, le fournisseur d'accès doit autoriser les véhicules à avoir une adresse IP publique ainsi qu'autoriser les communications entre membres du réseau ce qui n'est pas toujours le cas.

1.4 Détection d'anomalies appliquée au véhicule connecté

Comme nous l'avons présenté dans le début de ce chapitre, le véhicule moderne est un système complexe dans lequel de nombreux composants sont amenés à communiquer. Il serait tentant d'espérer que le système responsable des éléments critiques du véhicule puisse tout simplement être isolé des autres composants du réseau de façon à éviter que des attaques puissent impacter le bon fonctionnement du véhicule. Cependant, l'architecture interne du véhicule ne permet pas une telle

isolation. Il est à ce jour impossible d'à la fois isoler les composants critiques du véhicule tout en fournissant aux utilisateurs des applications leur permettant d'interagir avec leur véhicule grâce à leur smartphone. Par exemple, démarrer le véhicule afin de le mettre en préchauffage ou encore des fonctionnalités plus avancées comme le Smart Summon de la marque Tesla que nous avons déjà mentionné dans la section précédente.

C'est pourquoi, l'agence fédérale américaine pour la sécurité routière (*National Highway Traffic Safety Administration* (NHTSA)) définit dans son rapport de 2016 [87] 4 grands axes permettant de limiter la probabilité de succès ainsi que l'impact des attaques sur les véhicules connectés. Elle recommande ainsi de structurer les mécanismes de protection du véhicule autour des points suivants :

1. Identifier et favoriser la protection des éléments critiques du système de contrôle du véhicule ainsi que des informations à caractère personnel contenues dans le véhicule.
2. Assurer l'intégrité du système interne du véhicule en temps réel en surveillant les indices de compromission.
3. Assurer la reprise en temps réel du contrôle du véhicule par le conducteur en cas d'incident de sécurité.
4. Favoriser le partage d'information en source ouverte par les différents partenaires de l'industrie afin d'apprendre à se prémunir contre les attaques.

Dans cette thèse, nous nous intéressons tout particulièrement au second point à savoir l'utilisation de mécanismes de détection d'anomalies dans les communications du véhicule connecté. Nous avons présenté dans la section précédente (section 1.3) de nombreuses attaques et anomalies auxquelles les véhicules peuvent être sujets. Dans des études récentes, Dibaei *et al.* [37] et Cui *et al.* [32] présentent de nombreux travaux destinés à la sécurisation des véhicules connectés suivant l'état des connaissances actuelles sur les vulnérabilités existantes. Parmi ces solutions, de nombreuses sont basées sur une analyse des communications du véhicule. Cependant, la majorité des travaux répertoriés ciblent les réseaux VANETs ou le bus-CAN et se focalisent sur l'utilisation des réseaux cellulaires par les véhicules.

Dans cette section, nous présentons quelques approches de détection d'anomalies appliquées aux véhicules connectés. Ces approches sont classées en fonction de la nature du réseau surveillé : interne (bus-CAN), VANETs et réseaux cellulaire.

1.4.1 Le réseau interne (bus-CAN)

Nous commençons par étudier les travaux qui se focalisent sur le bus-CAN que nous avons présenté comme étant l'objectif principal des attaquants dans la section 1.3.

Par exemple, Song *et al.* [109] étudient les intervalles entre les messages CAN pour la détection d'intrusions. Ils utilisent une approche par spécification pour détecter les injections de messages dans le bus-CAN. Pour ce faire, ils ont étudié les intervalles entre les messages qui apparaissent sur le CAN par type de message

particulier pour déterminer les conditions normales de trafic. Lorsque l'intervalle entre des messages n'est pas respecté alors ils considèrent qu'une anomalie a lieu. Ils testent leur mécanisme en générant plusieurs attaques par déni-de-service sur le bus-CAN d'un véhicule et obtiennent 100% de détection et aucun faux positif.

Kang et Kang [59] cherchent à détecter les intrusions au niveau du réseau interne du véhicule grâce à un réseau de neurones profond. Ils se basent sur un corpus généré grâce à l'outil OCTANE (pour *Open Car Test-bed and Network Experiments*) [17]. Les attaques ne sont pas générées de la même manière que les travaux présentés plus haut. Ici, les paquets d'attaques sont injectés au fur et à mesure dans le réseau interne. Ils comparent l'utilisation d'un réseau de neurones avec SVM et obtiennent 99% de détection pour moins de 2% de faux positifs.

Wang *et al.* [122] étudient les incohérences dans les séquences de messages CAN observées sur le réseau interne du véhicule. Ils emploient une approche distribuée dans laquelle l'algorithme à mémoire temporelle hiérarchique (HTM) [55] est utilisé pour détecter les anomalies. L'algorithme est employé en parallèle pour détecter les anomalies sur chaque type de message CAN. Pour évaluer leur solution, ils ont capturé 20h de communication du bus-CAN d'une Subaru Impreza durant lesquelles ils ont pu générer des attaques de rejeu de messages ainsi que d'injection de messages aléatoires. Ils comparent leurs résultats avec un réseau de neurones récurrent et un modèle de Markov caché. La solution basée sur HTM est celle obtenant les meilleurs résultats de détection.

Enfin, une méthode ensembliste est employée par Theissler [116] où il cherche à détecter des fautes à l'intérieur des communications du bus-CAN. Dans son approche, il utilise plusieurs algorithmes comme SVM, les réseaux Bayésiens, les forêts aléatoires, et une évolution de SVM : SVDD (pour *support vector data description*) qui utilise une hypersphère au lieu d'un hyperplan pour classer les instances de données. Son corpus est constitué d'un enregistrement du bus-CAN d'une Renault Twingo de 2002 durant lequel il a injecté plusieurs fautes. Par exemple, il déclenche un dysfonctionnement des capteurs de température du moteur ainsi qu'une erreur dans l'injection de carburant à l'intérieur d'un des cylindres du moteur. Son évaluation compare les résultats de détection de l'approche ensembliste avec les résultats qu'obtiennent chacun des algorithmes utilisés dans cet ensemble.

Il définit deux scénarios d'apprentissage en fonction des algorithmes entraînés. Dans le premier, aucune anomalie n'est présente pendant la phase d'apprentissage. Dans le second, les anomalies sont à la fois présentes durant l'entraînement et la validation. Ainsi, il apparaît que SVM, les réseaux Bayésiens et les forêts aléatoire obtiennent de meilleurs résultats quand ils sont entraînés avec des instances d'anomalies. Les résultats qu'obtient l'ensemble sont constants en fonction des scénarios et SVDD obtient les meilleurs résultats tous scénarios confondus.

1.4.2 Les réseaux VANETs

D'autres mécanismes de protection se focalisent sur les attaques dirigées vers les communications des réseaux VANETs et leurs vulnérabilités notamment dans

le routage de l'information et son intégrité.

Une approche distribuée est présentée par Zaidi *et al.* [129] pour détecter les attaques Sybil et la dissémination de fausses informations de la part de véhicules malveillants. Chaque véhicule utilise le modèle de Greenshields [49] qui décrit les relations entre vitesse, densité et le flux de véhicules par heure sur une route. La relation entre la vitesse et la densité de véhicules d'un flux ininterrompu y est décrite comme étant une relation linéaire dans laquelle la densité est négativement corrélée à la vitesse. En d'autre termes, plus il y a de véhicules sur une route moins la vitesse de ces derniers est grande.

Les chercheurs utilisent ce modèle afin de prédire les flux de trafic à partir des messages échangés avec les véhicules à proximité. Ils en déduisent le nombre maximum de véhicules qui peuvent emprunter la même route à un instant donné et ainsi détecter ceux qui semblent diffuser des informations incorrectes concernant le nombre de véhicules présents sur la route. Ils réalisent leurs expérimentations grâce à la combinaison des outils de simulation de communications OMNET++ [119] et de trafic SUMO [66]. Puisque le modèle est basé sur la coopération des nœuds non malveillants du réseau, ils évaluent la capacité du détecteur à opérer suivant un nombre croissant de véhicules malveillants. Ils constatent ainsi que le taux de détection est de 100% sans faux positif en présence de 5 à 20% de véhicules malveillants. En cas d'une proportion de véhicules malveillants de 40%, le taux de détection est de 97.6% pour 2.1% de faux positifs.

Zeng *et al.* [131] cherchent à détecter des attaques par déni-de-service, trou noir et trou de vers, mais aussi les attaques Sybil opérant sur les réseaux VANETs ainsi que le trafic généré par les logiciels malveillants issu du corpus de données ISCX2012 [105]. Ils présentent une approche basée sur des réseaux de neurones profonds et comparent les capacités de détection de trois réseaux de neurones LSTM[56], CNN, et DeepVCM. Ce dernier est lui-même composé de LSTM et CNN. Ils comparent également les résultats des algorithmes SVM et des arbres de décision. Ils constatent que leur méthode (DeepVCM) obtient de meilleurs résultats de détection que les autres solutions étudiées tout en étant plus économe en stockage.

1.4.3 Le réseau cellulaire

Enfin, nous nous intéressons aux méthodes employées pour détecter des anomalies dans les communications des véhicules sur le réseau cellulaire.

Levi *et al.* [69], étudient les anomalies sur l'ensemble du véhicule à savoir le bus-CAN, les communications via le réseau cellulaire ainsi que le système d'infotainment. Leur système de détection est basé sur un modèle de Markov caché. Ils utilisent des règles afin de construire une abstraction des événements observés sur les différents composants. Ainsi, les événements produits à partir de ces règles sont par exemple :

— Au niveau du bus-CAN : Ouverture, fermeture des portes.
— Au niveau du réseau : Processus d'authentification, ports ouverts.
— Au niveau du système d'infotainment : installation d'applications, mise à

jour des applications.

Ils utilisent ensuite un modèle de Markov caché entraîné de manière supervisée pour détecter des événements anormaux sur l'un des systèmes surveillés. L'approche est accompagnée d'un mécanisme de partage de ces anomalies détectées via un SIEM (pour *Security Information and Event Management*) déployé dans un serveur distant auquel accèdent les véhicules.

Enfin, un autre exemple non directement lié au domaine du véhicule est présenté par Al Mamun et Valimaki [8]. Ces travaux se concentrent sur la détection d'anomalies dans les réseaux cellulaires au niveau du réseau opérateur et de sa gestion. Ils utilisent une machine à vecteur de support pour détecter les anomalies et appliquent un réseau de neurones récurrent (LSTM) aux anomalies détectées afin de comprendre l'évolution de l'anomalie au cours du temps. Ils étudient les anomalies à partir des attributs clé des réseaux 4G comme le nombre d'établissement de connexion entre les mobiles et le réseau-cœur 4G ou le ratio de transfert intercellulaire (ou *handover*[6]) ayant eu lieu avec succès. Ils comparent leur solution avec une méthode de classification automatique. Cette dernière définit à partir d'un corpus d'entraînement, les différentes valeurs possibles des attributs des communications réseau. Al Mamun et Valimaki constatent que cette méthode est plus efficace que celle basée sur la détection d'anomalies et qu'elle reste applicable pour de larges corpus de données.

Enfin, les travaux présentés par Aloqaily *et al.* [10] présentent l'utilisation d'un algorithme basé sur un arbre de décision pour la détection d'intrusions. Ils proposent un cadre dans lequel les véhicules accèdent aux services proposés par des services cloud en communiquant avec un véhicule intermédiaire (cluster head) élu au sein d'un groupe (cluster) de véhicules proches.

Leur corpus de données est construit à partir d'informations tirées d'une simulation de trafic générée grâce à NS3 [101] qui sert d'entraînement à l'algorithme. Ils utilisent NSL-KDD [115] comme corpus pour la détection d'anomalies. Ils obtiennent 99.92% de taux de détection pour 0.96% de faux positifs. L'utilisation de l'outil de simulation permet aux chercheurs de générer un corpus d'entraînement dans lequel les propriétés des communications des véhicules sont proches de ce que l'on pourrait rencontrer dans un cas réel. D'autre part, l'utilisation d'un corpus d'attaque connu permet aussi de vérifier que l'algorithme est capable de détecter chaque type d'attaques.

Cependant, les attaques présentes dans le corpus NSL-KDD sont des attaques spécifiques aux réseaux classiques et qui datent de plus de 20 ans. Par conséquent, un biais est ainsi introduit dans l'évaluation de leur système de détection puisque de nombreuses attaques présentes dans le corpus ne sont pas pertinentes dans le contexte des réseaux véhiculaires actuels. Par exemple, la réalisation d'une attaque par Teardrop suppose d'envoyer des paquets fragmentés malformés à un hôte de manière à faire échouer le processus de ré-assemblage. Or, cette attaque n'est réalisable

6. Lorsqu'un téléphone mobile passe d'une cellule géographique à une autre. Celle-ci étant gérée par une autre antenne un transfert intercellulaire a lieu entre les deux antennes pour assurer la continuité de connexion entre le téléphone et le réseau.

que sur de vieux systèmes d'exploitation (Windows 95, NT et 3) et sur d'anciens noyaux linux (antécédent au 2.1.63).

De plus, les communications dites normales présentes dans NSL-KDD sont *de facto* très différentes des communications véhiculaires car les réseaux sont destinés à des usages très différents. Par exemple, le corpus contient du trafic issu du protocole Finger [7], qui était utilisé au début d'internet pour obtenir des informations sur des personnes à partir de leur adresse mail. Il est donc difficile de juger si leur méthode est réellement capable de détecter les attaques avec grande précision ou si elle est simplement capable de différencier le trafic issu de la simulation NS3 de celui présent dans le corpus NSL-KDD. En effet, des événements semblables dans les applications d'apprentissage automatique ont démontré que le biais introduit pendant la phase d'apprentissage pouvait avoir un impact négatif lors de l'utilisation du même modèle en conditions réelles. Nous faisons référence par exemple à l'utilisation de la reconnaissance d'images pour la classification de tanks par l'armée américaine. L'objectif était de pouvoir distinguer des tanks camouflés dans des forêts. Il s'est avéré que l'ensemble des images utilisées pour entraîner le modèle contenaient uniquement des photographies de tanks prises sous un temps nuageux et des photographies de forêts prises par beau temps. L'algorithme a donc «appris» à reconnaître un ciel bleu plutôt que les contours de tanks [128].

1.4.4 Conclusion

Nous constatons que peu de travaux se sont concentrés sur l'application de la détection d'anomalies dans les communications véhiculaires cellulaires. Pourtant, Parkinson *et al.* [94], dans leur l'état de l'art des vulnérabilités déjà découvertes sur les véhicules connectés et véhicules autonomes, présentent de nombreux verrous scientifiques qu'il convient de traiter afin de minimiser les risques identifiés. Notamment, ils estiment que des attaques à grande échelle seront réalisables de plus en plus facilement au cours des années à venir. Cette facilité entraînera une augmentation de la fréquence des attaques puisqu'elles nécessiterons moins de connaissances de la part des attaquants. En effet, jusqu'à présent, les attaques sont souvent des tentatives de chercheurs qui agissent sur des véhicules d'expérimentation et dans des environnements contenus, au nom du bien commun afin d'améliorer la sécurité des véhicules connectés. Si les véhicules deviennent facilement identifiables sur le réseau les attaques pourraient causer de sérieux dégâts. C'est pourquoi de nombreux chercheurs [41] ainsi que la NHTSA recommandent la mise en place de politiques de réponses à incidents qui permettront :

— De se préparer aux attaques éventuelles par la mise en place d'équipes dédiées à la gestion des événements de sécurité.
— D'identifier les attaques lorsqu'elles ont lieu.
— D'être en mesure de contenir ces attaques et éviter toute perte de contrôle du véhicule par le conducteur ou le système de conduite automatisé.
— De restaurer les systèmes des véhicules touchés par ces attaques.

7. https://linux.die.net/man/1/finger

En ce sens, un système de détection d'anomalies pourrait assister les industriels dans leur lutte contre la prolifération de nouvelles attaques. L'utilisation de mécanismes capables d'identifier rapidement les attaques permet ainsi aux industriels de concevoir des correctifs afin de protéger les véhicules n'ayant pas encore été affectés.

Pour autant, nous considérons que la mise en place de ces mécanismes doit impérativement prendre en compte les moyens de communication externes aux véhicules et en particulier le réseau cellulaire. En effet, celui-ci représente un vecteur d'attaque considérable puisqu'il permet aux attaquants d'accéder aux véhicules à distance dans le but d'interagir avec leur réseau interne.

Ainsi, nous proposons dans cette thèse de fournir une ligne de défense supplémentaire au véhicule afin de rompre la chaîne d'exploits en détectant les anomalies et tentatives d'intrusions directement dans les échanges du véhicule avec le monde extérieur. Pour ce faire, nous analysons les propriétés des communications effectuées sur le réseau cellulaire grâce à un algorithme d'apprentissage automatique.

1.5 Résumé

Dans ce chapitre nous avons tout d'abord présenté les dispositifs de communication dont sont équipés les véhicules modernes. Ils permettent notamment le bon fonctionnement des systèmes internes au véhicule, mais aussi le partage d'informations critiques avec les autres véhicules et usagers de la route. Ainsi, ces dispositifs essentiels à la sûreté et à l'organisation des systèmes de transport intelligents doivent être protégés contre les anomalies d'origine accidentelle ou volontaire. C'est pourquoi nous avons présenté de nombreuses approches visant à détecter ces anomalies ou intrusions en se basant sur l'étude des caractéristiques des communications réseaux.

Nous nous sommes ensuite saisi des spécificités induites par le contexte automobile que nous devons prendre en compte dans la conception d'un détecteur d'anomalies dédié aux communications cellulaires du véhicule. Nous avons donc présenté en détails les différents vecteurs utilisés par les attaquants pour exploiter les vulnérabilités des véhicules et en particulier ceux exploitables par les communications cellulaires.

Enfin, nous avons présenté de nombreuses études proposant des mécanismes de détection dédiés aux véhicules. Cependant, nous avons pu constater que ces études se concentrent principalement sur la détection d'anomalies au niveau du réseau interne du véhicule. Or, celui-ci n'est en réalité que le dernier maillon de la chaîne d'exploits du véhicule puisque les attaquants cherchent à tout prix à interagir avec les ECUs qui y sont connectés.

Ainsi, ils doivent préalablement avoir été en mesure d'exploiter d'autres vulnérabilités et notamment celles des différents dispositifs de communication externe du véhicule. Ainsi, les capacités de communication externes du véhicule représentent donc un vecteur permettant aux attaquants d'accéder au réseau interne. C'est pourquoi nous proposons dans cette thèse une nouvelle approche autonome de détection

des anomalies et intrusions dans les communications externes du véhicule, permettant ainsi d'ajouter une ligne de défense supplémentaire afin de maintenir les attaquants hors de portée du réseau interne.

Chapitre 2

Approche ontologique à la détection d'anomalies

Sommaire

Afin de répondre à la problématique que nous avons formulée dans le premier chapitre, nous avons conçu un système de détection d'anomalies. Ce système, baptisé **Svalinn** en référence au légendaire bouclier scandinave protégeant la terre des

rayons brûlants du soleil, s'appuie sur la modélisation temporelle sous la forme d'une ontologie des communications véhiculaires.

Dans la section 2.1, nous commençons dans ce chapitre par établir les besoins et difficultés rencontrés dans la détection d'anomalies. Puis nous introduisons de façon globale les composants principaux de **Svalinn** dans la section 2.2. Ensuite, dans la section 2.3, nous détaillons le fonctionnement de chacun des composants de **Svalinn** en commençant par le processus de création des *fenêtres de description instantanée* qui sont utilisées durant le processus de détection. Subséquemment, nous présentons l'algorithme utilisé pour la détection d'anomalies dans la section 2.4. L'ontologie et le moteur d'inférence qui permettent le traitement des anomalies sont présentés dans la section 2.5. Enfin, le dispositif expérimental que nous avons mis en place pour l'implémentation de **Svalinn** est discuté dans la section 2.6.

2.1 Les besoins et difficultés de la détection

Le contexte singulier des réseaux véhiculaires nous pousse à considérer l'emploi d'une méthode de détection en accord avec la nature du trafic et l'ensemble du spectre des anomalies auquel le système doit faire face. Cette méthode doit répondre aux différentes contraintes qui pèsent sur l'exécution d'un tel système à l'intérieur d'un véhicule.

2.1.1 Nature du trafic

Tout d'abord, les réseaux véhiculaires ont ceci de particulier qu'ils opèrent des services de deux natures.

Premièrement, ceux liés à l'utilisation du véhicule dans le contexte des systèmes de transports intelligents requièrent que le véhicule partage à intervalle régulier l'interprétation de son environnement ainsi que son état à un interlocuteur distant. En effet, les véhicules modernes sont équipés d'un ensemble de capteurs et actionneurs reliés par un réseau interne. Les informations que ces équipements génèrent sont traitées par des unités de commande électronique (ECU) dans le but de permettre aux véhicules de prendre des décisions complexes comme la détection et l'évitement d'obstacles. En partageant ces connaissances avec un serveur de centralisation, les autres véhicules peuvent ainsi être avertis d'éventuels dangers et adapter leur conduite. De même, le suivi de l'usure des équipements ou de la consommation énergétique d'une flotte de véhicules est ainsi rendu possible par ces services de télémétrie.

Ensuite, les services de divertissement accessibles aux utilisateurs du véhicule partagent quant à eux les propriétés des communications mobiles. On y retrouve des sessions liées à l'utilisation d'internet et du protocole HTTP ainsi que de nombreux services streaming de musique, voire de films. Ainsi, il s'agit de flux dont les propriétés sont éloignées des services ITS en ceci que la durée des sessions est de facto plus courte que celle des services de télémétrie. De plus, les fréquences des

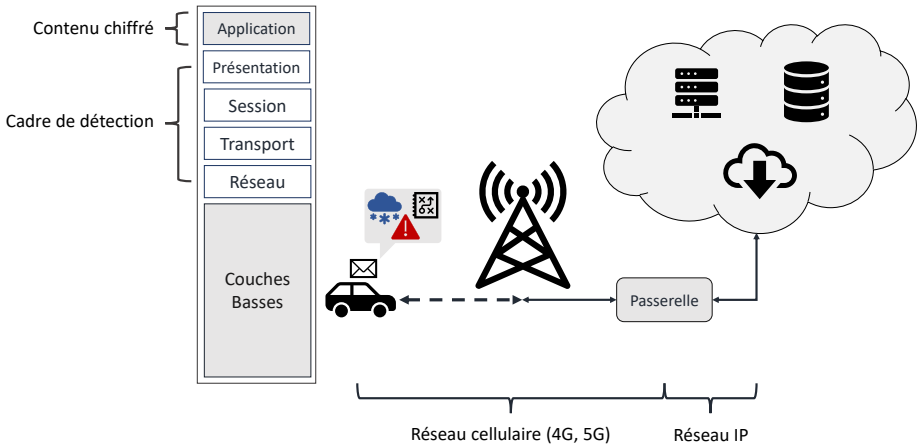


FIGURE 2.1 – Illustration des couches protocolaires employées dans le cadre des communications C-V2X.

communications liées à ces services sont, elles aussi, inférieures à celles observées pour les applications de télémétrie des véhicules.

Cette différence dans les caractéristiques des flux observés dans les communications véhiculaires constitue donc un des premiers éléments que la détection doit prendre en compte. Effectivement, cette différence pourrait induire la détection à considérer les flux de divertissement comme anormaux si aucune précaution n'était prise.

Enfin, la majorité de ces communications, notamment celles de télémétries, contiennent des informations sensibles et à caractère personnel comme les données de localisation liées aux utilisateurs du véhicule. Ces transmissions emploient donc des méthodes de chiffrement afin d'en assurer la confidentialité. À ce titre, l'analyse profonde du contenu applicatif des communications est donc exclue de la conception de notre méthode de détection.

En effet, si nous nous reportons aux piles protocolaires employées dans les communications C-V2X illustrées par la Figure 2.1, nous constatons que le champ d'action de notre détection est limité aux couches réseau, transport, session et présentation du modèle OSI puisque l'analyse de la couche applicative se limite aux attributs accessibles sans déchiffrer le contenu des messages. De plus, notre solution se veut indépendante des communications cellulaires employées par le véhicule, fussent-elle 4G ou 5G. Il est donc exclu d'analyser les couches les plus basses de ces protocoles.

2.1.2 Nature des anomalies

Les anomalies et attaques auxquelles les véhicules connectés sont soumis sont de diverses natures. Les nombreux scénarios présentés dans le second chapitre de cette thèse mettent en lumière cette diversité. Par exemple :

— Anomalies ponctuelles :
  — Les attaques par déni-de-service de forte intensité.

  - La recherche de ports ouverts (scan) qui peuvent entraîner une augmentation de l'envoi de paquets RESET de la part de l'hôte interrogé.
- Anomalies Contextuelles :
  - Les anomalies télémétriques ayant, par exemple, pour conséquence la réduction de la fréquence d'envoi de messages du véhicule vers le serveur de centralisation.
- Anomalies collectives :
  - Les communications des botnet ou logiciels malveillants dissimulées en détournant l'utilisation légitime de protocoles tel que Domain Name System (DNS).

Ainsi, l'impact de ces anomalies et attaques sur les propriétés des communications supervisées par le détecteur sont elles aussi très diverses par leur nature contextuelle, séquentielle ou ponctuelle. De ce fait, la procédure de détection doit être en mesure de discriminer les anomalies en fonction de ces propriétés afin d'assurer la plus grande couverture possible au véhicule.

2.1.3 Contexte d'exécution

En plus des propriétés du trafic et des anomalies s'ajoute le contexte d'exécution du système de détection lui aussi particulier. La détection des anomalies dans les communications se doit d'être réalisée directement dans le véhicule afin d'en assurer la sécurité.

En conséquence, le détecteur doit être en mesure de répondre aux contraintes matérielles et humaines qu'implique son déploiement à l'intérieur du véhicule. De plus, il doit aussi être en mesure d'assurer l'archivage des alertes de détection ainsi que la création de rapports automatiques permettant aux opérateurs d'analyser ces événements a posteriori.

Les communications des véhicules avec l'extérieur sont assurées par l'unité de contrôle télématique (TCU). Celle-ci est directement connectée au bus de données CAN (controller area network) ainsi qu'au système d'aide à la conduite automobile (ADAS). L'analyse des communications entrantes et sortantes doit donc être réalisée en prenant en compte les capacités de calcul de cette unité afin de ne pas perturber les services qu'elle se doit d'assurer. Si nous prenons l'exemple d'une attaque par déni-de-service, le détecteur pourrait être soumis à un trafic dont l'intensité entraîne une surconsommation de ressources de calcul et de mémoire pouvant impacter les autres services exécutés sur le TCU, amplifiant ainsi l'attaque initiale destinée à seulement empêcher la réception ou l'envoi de messages par le véhicule. Il est donc important que l'exécution du détecteur soit isolée du reste des services de l'unité de contrôle télématique et que ses ressources soient limitées.

Ensuite, compte tenu que l'utilisation du réseau par les véhicules peut être limitée pour des raisons de coût ou simplement d'accès réseau, l'emploi de méthodes de détection embarquées à l'intérieur du véhicule limite les possibilités d'une supervision directe par un opérateur. De plus, permettre la supervision directe d'un véhicule pose d'autres problèmes comme la protection de la vie privée des utilisa-

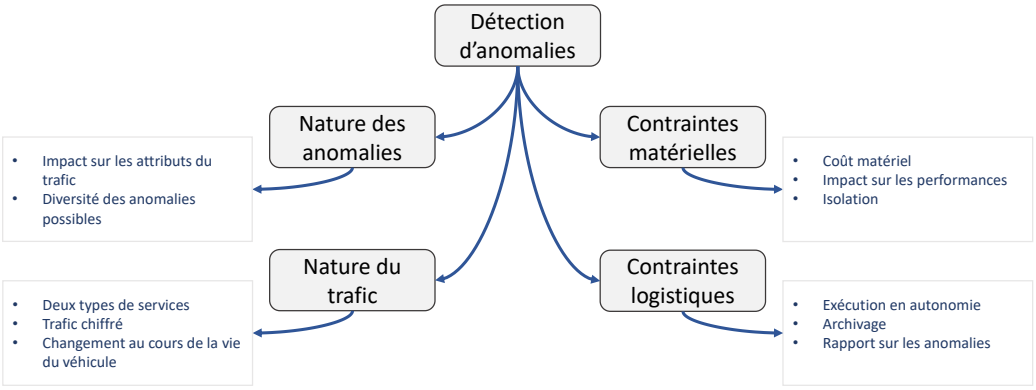


FIGURE 2.2 – Résumé des difficultés et contraintes de la détection d'anomalies dans les réseaux véhiculaires

teurs du véhicule, sans compter que le risque d'exposer le véhicule à des attaques via le système de supervision n'est pas nul. Il faut donc que le détecteur soit en mesure de fonctionner avec une certaine autonomie en procédant à l'archivage des anomalies détectées. De cette façon, il pourrait alors transmettre des rapports détaillés à un service de centralisation lorsque le réseau le permet. Ce service serait ainsi en charge de surveiller des flottes de véhicules et de gérer les anomalies pour permettre la diffusion des mises à jour de sécurité.

Enfin, le cycle de vie étendu des véhicules induit que le détecteur devra faire face au changement des propriétés des communications qu'il est tenu d'analyser. En effet, les mises à jour ou le déploiement de nouvelles fonctionnalités pourront avoir un impact sur le trafic dit «*habituel*» que le détecteur analyse. Celui-ci doit donc être en mesure de s'adapter au changement du trafic tout en maintenant la détection.

2.1.4 Résumé

Ainsi, les difficultés et contraintes auxquelles un détecteur d'anomalies dans les réseaux véhiculaires doit faire face peuvent être résumées par la carte heuristique illustrée par la Figure 2.2.

2.2 Vue d'ensemble du fonctionnement de Svalinn

Nous illustrons par la Figure 2.3 le fonctionnement de **Svalinn** qui nous permet de répondre à notre problématique. Dans un premier temps, les paquets échangés par le véhicule sont regroupés au sein de *fenêtres de description instantanée*. Ces *fenêtres* sont ensuite analysées afin d'en extraire un ensemble d'attributs. Ceux-ci sont liés au propriétés du trafic réseau comme le débit ou la latence. Ils sont soumis au processus de détection d'anomalies et d'intrusions réalisé grâce à l'algorithme à mémoire temporelle hiérarchique (HTM). Pour chaque entrée, l'algorithme produit

un score permettant de décider si une *fenêtre* est anormale, et de lever une alerte si tel est le cas.

Dans le cas où une *fenêtre de description instantanée* est considérée comme anormale par l'algorithme, un traitement est alors déclenché afin de générer un rapport concernant l'anomalie détectée.

Les sections qui suivent présentent en détail les choix qui ont motivé le design de ces différents composants et leur articulation. La section 2.3 introduit les *fenêtres de description instantanée* pour la représentation du trafic du véhicule. La section 2.4 énonce les alternatives possibles à l'algorithme HTM et la raison pour laquelle ces dernières n'ont pas été retenues dans nos travaux. Elle présente également des arguments nous permettant de choisir HTM pour la détection. La section 2.5 détaille le processus de traitement des anomalies qui repose sur une ontologie et un moteur d'inférence. Enfin, la section 2.6 détaille l'implémentation d'un prototype pour procéder à la détection d'anomalies dans les réseaux cellulaires de véhicules.

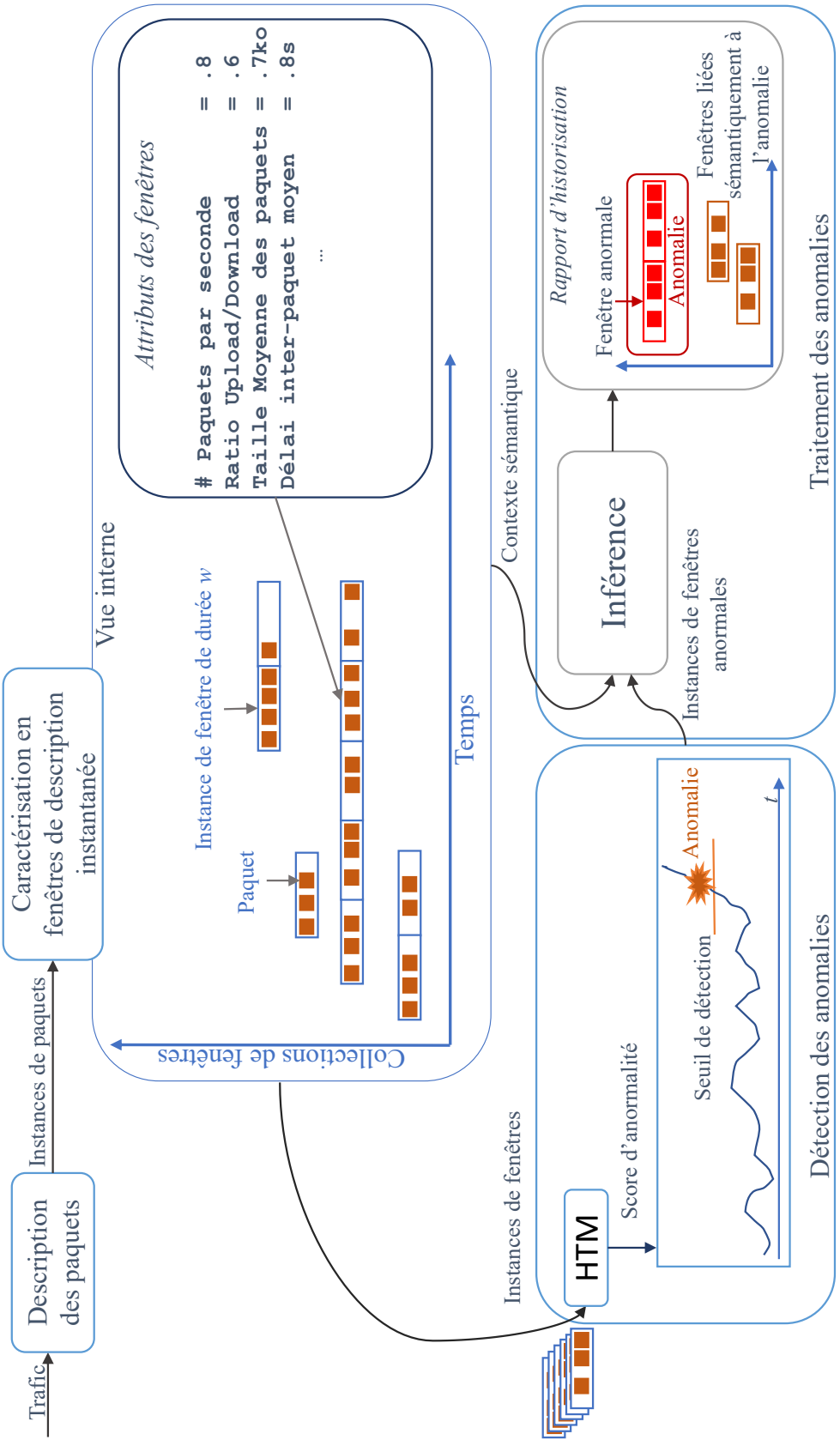


FIGURE 2.3 – Vue d'ensemble du processus de détection de **Svalinn**

2.3 La représentation des communications

Dans cette section, nous décrivons en détail le processus de création des *fenêtres de description instantanée* à partir des paquets échangés sur le réseau par le véhicule. Les termes *fenêtres* et *fenêtres de description instantanée* sont équivalents dans le reste de cette thèse.

2.3.1 Vers une représentation temporelle du trafic

Tout d'abord, l'exécution d'un système de détection d'anomalies à l'intérieur même du véhicule nous incite à limiter la consommation des ressources par le système au profit des autres services exécutés en parallèle sur celui-ci. En effet, une analyse de chaque paquet reçu ou émis par le véhicule aurait un impact néfaste sur les performances de l'ECU en particulier en cas de fort trafic.

Nous nous sommes donc intéressés aux méthodes de représentation des communications dans un réseau. Le moyen le plus répandu est celui des *flux*. Il est défini comme suit :

Définition 1 *Un flux est défini par le type de protocole de transport ainsi que les adresses IP et ports de la source et de la destination d'un paquet réseau. Un flux est traditionnellement défini par le quintuplet suivant*[1] *:*

$$\begin{aligned} F(X,Y,k,l,p) = (&IP_{source}(p_i) = X,\ IP_{destination}(p_i) = Y,\\ &Port_{source}(p_i) = k,\ Port_{destination}(p_i) = l,\ Transport = p) \end{aligned} \tag{2.1}$$

Où p_i représente le paquet i de l'ensemble de paquets P capturés sur le réseau.

Dans la plupart des approches que nous avons présentées dans l'état de l'art (section 1.2), les chercheurs se basent sur l'étude des attributs de ces *flux* afin de détecter le plus précisément possible tous types d'attaques. De plus, nous avons constaté que la plupart des corpus d'entraînement (sous-section 1.2.2) généralement utilisés dans ces travaux se basent sur cette notion de *flux* pour classer les trames capturées sur le réseau. Ils en extraient ensuite un ensemble d'attributs pouvant être utilisés pour la détection des intrusions ou anomalies.

Par exemple, les corpus CICIDS2017/2018 [104] mettent à disposition, en plus des captures réseau, une représentation du trafic générée avec l'outil CICFLOWMETER [68]. Cet outil extrait des *flux* 152 attributs[2] décrivant la taille des paquets échangés pendant la durée du *flux*, le délai-inter-paquet, le débit montant et descendant et bien d'autres.

Dans les corpus KDD99 [61] et NSL-KDD [115], les attributs sont générés sur des *flux* considérés comme terminés. Les informations des *flux* n'y sont pas contextualisées et les attributs sont liés aux sessions par protocole. Ils conservent par exemple le nombre d'octets échangés, de flags urgents, de paquets corrompus dans ces communications ainsi que leur durée.

1. https://tools.ietf.org/html/rfc3697
2. http://www.netflowmeter.ca/netflowmeter.html

De façon similaire, les corpus UNSW-2015 [83] et Kyoto [110] construisent pour chaque *flux* les informations sur le nombre d'octets échangés, retransmis ou perdus. On y trouve aussi l'information du débit montant et descendant.

Or, dans chaque situation, les attributs décrivent uniquement des *flux* considérés comme terminés. Cela pose deux problèmes. En premier lieu une attaque ne peut donc être détectée que lorsque le *flux* sur lequel elle a lieu est clos. D'autre part, le second est qu'en pratique un *flux* peut évoluer d'une situation normale à anormale et vice versa. Or, si le *flux* est uniquement considéré dans son ensemble, il devient difficile de détecter les anomalies dès qu'elles apparaissent dans le trafic par le changement des attributs des *flux*.

En particulier, si nous considérons les propriétés temporelles des communications observées sur les réseaux cellulaires de véhicules tant en matière de trafic habituel que d'anomalies ou d'attaques. En conséquence, Il paraît important de pouvoir étudier chaque *flux* au fur et à mesure que les paquets sont capturés sur le réseau.

C'est pourquoi nous extrayons des communications du véhicule un ensemble d'attributs temporels décrivant ces échanges et étudions leur évolution au cours du temps grâce à ce que nous appelons les *fenêtres de description instantanée*.

2.3.2 Fenêtres de description instantanée

Les *fenêtres de description instantanée* sont définies comme suit :

Définition 2 *Une fenêtre de description instantanée est constituée d'un ensemble de paquets capturés pendant une période donnée. Elle décrit grâce à un ensemble d'attributs temporels les propriétés des paquets qu'elle contient.*

Ainsi, pour un ensemble de paquets horodatés $P = [p_0, .., p_n]$ où $H(p_i)$ représente le temps écoulé entre la réception des paquets p_0 et p_i et w représentent la durée des *fenêtres*, chaque paquet p_i appartient à une *fenêtre de description instantanée* f_j de l'ensemble des *fenêtres* F où j est défini de 0 à m tel que $F = [f_0, ...f_j, ...f_m]$ si et seulement si :

$$j \times w < H(p_i) \leq (j+1) \times w \tag{2.2}$$

La liste des attributs que nous utilisons est disponible en annexe (Appendice A). Nous avons adapté les attributs utilisés dans les travaux sur la caractérisation des communications sur le réseau TOR [3] [68] pour définir cette liste.

2.3.3 Limitations de la définition de flux

Grâce aux *fenêtres de description instantanée* nous sommes capables d'observer les variations des attributs des *flux* afin d'y détecter des anomalies au fur et à mesure que les paquets des différents *flux* sont capturés. Cependant, la définition d'un *flux*

3. TOR, pour *The Onion Router*, est un réseau superposé permettant l'anonymisation de l'origine des connexions TCP.

est limitée par le fait que les paquets de deux *flux* différents peuvent partager un même contexte.

Par exemple, si un même hôte communique avec le véhicule par plusieurs *flux*, il pourrait être pertinent de regrouper l'ensemble des paquets au sein d'une même *fenêtre*. Par exemple, des outils comme `nmap` ou `scapy` [4] sont souvent utilisés par les attaquants pour réaliser des scans. Dans la plupart des cas, l'outil crée une succession de paquets à destination de différents ports de la machine ciblée, mais il conserve le même port source pour chacun d'entre eux. Ainsi, pour détecter ce genre de situation, il est important de considérer l'ensemble des paquets provenant de cette entité.

Pour définir ces ensembles de paquets nous avons introduit deux autres notions pour décrire les communications : les *conversations* et les *liens*.

– **Conversation** – La notion de *conversation* est définie comme suit :

Définition 3 *Une conversation est définie par les IPs sources et destination d'un paquet réseau ainsi que par le port utilisé par l'entité communiquant avec le véhicule. Elle permet donc de regrouper des paquets par le triplet suivant :*

$$C(X, Y, k) = (IP_{véhicule}(p_i) = X,\ IP_{entité}(p_i) = Y,\ Port_{entité}(p_i) = k) \qquad (2.3)$$

Supposons qu'un attaquant souhaite scanner les ports les plus courants (de 0 à 1023 souvent définis comme *well-known* [5]) d'un hôte. Dans la représentation par *flux* (Définition 1) un tel scan engendrerait la création de 1024 *flux*. Or, il pourrait être intéressant de réunir au sein d'une seule *conversation* l'ensemble des paquets de façon à permettre l'analyse de l'anomalie dans son ensemble.

Supposons maintenant que le véhicule communique avec un serveur distant dans le cadre des services de e-horizon. À chaque nouvelle requête vers ce serveur, la représentation des communications par la méthode des *flux* entraîne la création d'un nouveau *flux* puisque le port source entre chaque requête du véhicule va changer alors que l'ensemble des paquets transmis sont liés entre chacune des requêtes puisqu'ils concernent le même serveur. Par conséquent, la *fenêtre description instantanée* de *conversation* semble plus appropriée puisqu'elle se contente d'ajouter les paquets à une même *conversation* permettant ainsi d'analyser l'ensemble des paquets répartis dans différentes *fenêtres*.

– **Lien** – Cependant, la *conversation* n'est pas suffisante pour représenter les communications. En effet, il pourrait aussi être intéressant de prendre en compte tout les paquets provenant d'un même hôte au sein d'un même type de fenêtre. Par exemple, des applications comme FTP établissent deux sessions entre les mêmes hôtes sur des ports différents afin de permettre le transfert de fichiers. C'est pourquoi nous avons défini la notion de *lien* comme suit :

4. https://nmap.org et https://scapy.net/

5. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=6

Définition 4 *Un lien est défini par les IPs source et destination d'un paquet réseau. Il permet donc de regrouper des paquets par le couple suivant :*

$$L(X,Y) = (IP_{véhicule}(p_i) = X,\ IP_{entité}(p_i) = Y) \tag{2.4}$$

En regroupant ainsi tous les paquets échangés entre deux entités nous pouvons étudier les propriétés des échanges à un degré d'abstraction supplémentaire en ne focalisant plus la détection d'anomalie au niveau d'un *flux* applicatif, mais plutôt au niveau du *lien*.

2.3.4 Exemples

2.3.4.1 Création des Liens, Conversations et Flux

Nous illustrons les notions de *liens*, de *conversations* et de *flux* en présentant un exemple d'anomalie que nous souhaitons détecter. Supposons que l'utilisateur du véhicule a, par mégarde, installé une application sur son système de divertissement via une plateforme d'application tierce-partie. Or, il se trouve que l'application est vérolée par un logiciel malveillant qui extrait des informations sur le véhicule et son propriétaire pour ensuite les partager avec un serveur contrôlé par des acteurs malveillants. Afin de ne pas attirer les soupçons, les développeurs du logiciel malveillant ont décidé de dissimuler les communications de ce dernier en utilisant le protocole DNS (pour *domain name system*).

Ce dernier est chargé de traduire des noms de domaines comme laas.fr en adresse IP. Son fonctionnement est basé sur un système hiérarchique distribué, composé d'une racine ou domaine de premier niveau (TLD) ainsi que de sous-domaines. Ainsi, la résolution du nom `REDEOZIE4654TRPOIDDEZ.t.hacker.org` avec le protocole DNS peut être illustrée par la Figure 2.4.

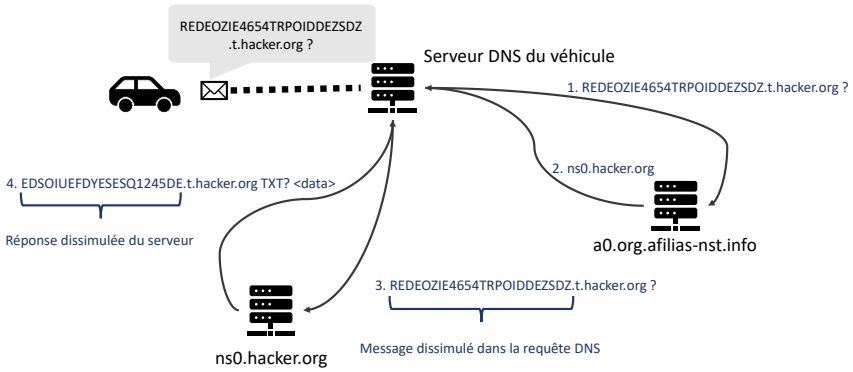


FIGURE 2.4 – Illustration de la résolution DNS d'un message caché. Le premier serveur DNS reçoit la requête du véhicule et déclenche la résolution récursive en questionnant le serveur racine (message 1). Le serveur racine redirige la requête vers le serveur contrôlé par les attaquants (message 2). Enfin le serveur DNS du véhicule transmet la requête au serveur des attaquants (message 3). Ce dernier interprète le message caché et répond avec un autre message au véhicule (message 4).

Cette technique de tunnel-DNS (ou *dns-tunneling*) est une méthode populaire employée pour l'exfiltration de données ou le contrôle de réseaux de botnets. En effet, les communications DNS sont rarement contrôlées par les systèmes de sécurité et permettent aux pirates de communiquer avec les hôtes contaminés même si ceux-ci sont dans des réseaux isolés. Les attaquants utilisent cette technique comme un moyen de communication avec un serveur distant en ajoutant des données à l'intérieur de leurs requêtes afin de les transmettre au serveur DNS qu'ils contrôlent. Les informations extraites sont ainsi dissimulées dans le reste des communications DNS de l'hôte attaqué.

Dans ce scénario les messages envoyés en réponse par le serveur DNS de l'attaquant pourraient correspondre à des commandes destinées à prendre le contrôle du véhicule ou à en extraire des informations. Il est donc important de détecter ce genre de communications cachées.

Nous illustrons les échanges du véhicule dans le contexte de cette attaque grâce à la Figure 2.5. Elle présente les *flux* de communications du véhicule sur l'axe des ordonnées évoluant au cours du temps (abscisse) entre un véhicule et d'autres entités du réseau. Les *flux* y sont représentés dans l'ordre d'occurrence et les *fenêtres de description instantanée* qui les composent sont délimitées par des rectangles dont les frontières en marquent le début et la fin. Le début et la fin des *flux* sont définis par le début et la fin des premières et dernières *fenêtres* de chaque *flux*.

Dans le cas présent, le *flux* F_6 représente des communications DNS classiques et le *flux* F_7 représentent celles du logiciel malveillant. Les autres *flux* (F_1, F_2, F_3, F_4 et F_5) sont d'autres échanges qui nous permettent d'illustrer les différents types de *fenêtres*.

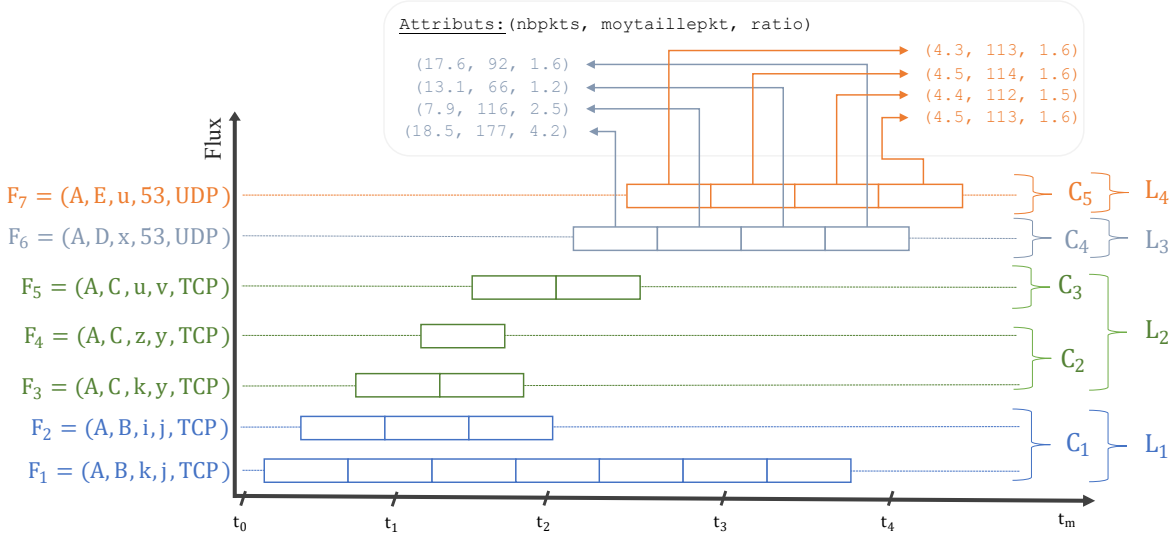


FIGURE 2.5 – Représentation des *flux* de communication du véhicule d'adresse IP A vers les entités d'adresse IP B, C, D et E.

Ces *flux* sont définis par les quintuplets suivants (Définition 1) :

— $F_1 = (A, B, k, j, TCP)$, $F_2 = (A, B, i, j, TCP)$,
— $F_3 = (A, C, k, y, TCP)$, $F_4 = (A, C, z, y, TCP)$, $F_5 = (A, C, u, v, TCP)$
— et $F_6 = (A, D, x, 53, UDP)$, $F_7 = (A, E, u, 53, UDP)$, puisque 53 est le port utilisé par le protocole DNS.

Par conséquent, les *conversations* C_1, C_2, C_3, C_4 et C_5 sont définies par les triplets suivants (Définition 3) :

— $C_1 = (A, B, j)$, $C_2 = (A, C, y)$, $C_3 = (A, C, v)$, $C_4 = (A, D, 53)$ et $C_5 = (A, E, 53)$

Enfin, les *liens* L_1, L_2, L_3 et L_4 par les couples suivants (Définition 4) :

— $L_1 = (A, B)$, $L_2 = (A, C)$, $L_3 = (A, D)$, $L_4 = (A, E)$

Ce sont les attributs des *fenêtres de description instantanée* des différents *flux* qui sont utilisés pour la détection des anomalies. Nous représentons pour cet exemple les 3 attributs suivants :

— Nombre de paquets par seconde : `nbpkts`,
— Taille moyenne des paquets : `moytaillepkt`,
— Le ratio entre le trafic montant et le trafic descendant (en taille) : `ratio`.

On remarque ainsi une différence entre les attributs des *fenêtres* du *flux* anormal F_7 par rapport au *flux* F_6, notamment le nombre de paquets par seconde et le ratio du trafic montant par rapport au trafic descendant. En effet, le *flux* F_6 présente une plus grande variance pour chacun de ces attributs avec 23.6 contre 0.009 pour le nombre de paquets par seconde et 1.7 contre 0.002 pour le ratio. Ce sont ces différences ou fluctuations dans les attributs des *flux* que nous souhaitons détecter grâce à l'algorithme.

2.3.4.2 Création des différents types de fenêtres

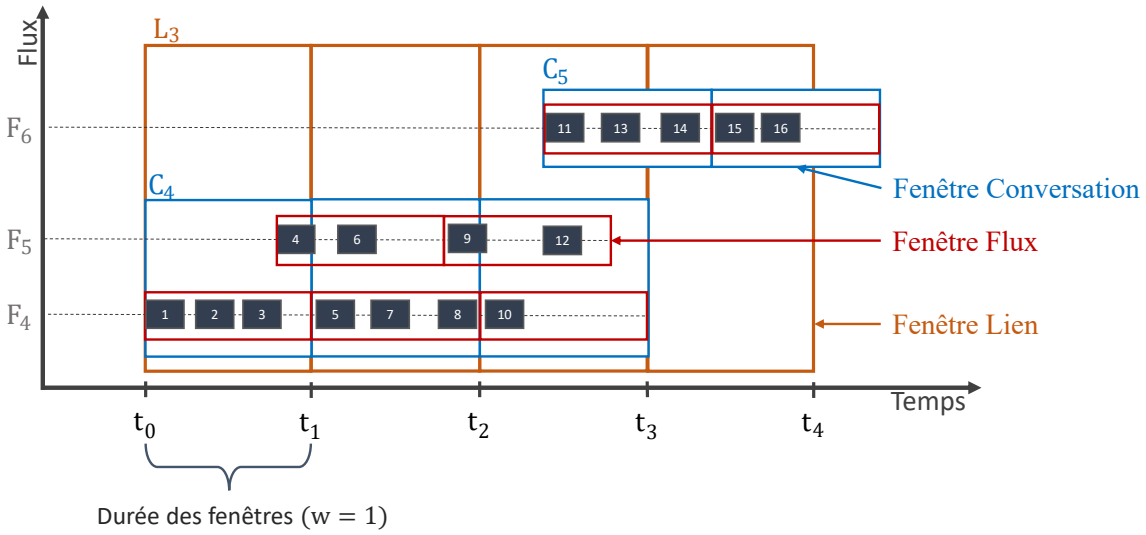


FIGURE 2.6 – Représentation de la création des fenêtres de *flux* sous la forme d'un chronographe.

Dans l'exemple précédent nous nous sommes surtout intéressés aux modes de représentation des communications (*flux*, *liens* et *conversations*) et nous avons illustré les attributs des *fenêtres de description instantanée* de deux *flux* dans le contexte d'une anomalie.

Nous présentons maintenant comment les différentes *fenêtres de description instantanée* sont extraites des communications grâce à la Figure 2.6. Les *fenêtres* de *flux* sont représentées par des rectangles rouges, les *fenêtres* de *conversation* par des rectangles bleus, et enfin, les *fenêtres* des *liens* sont délimitées par des rectangles oranges. Dans cet exemple, la durée de tous les types de *fenêtres* est égale à 1, celles-ci sont créées dans l'ordre de réception des paquets et nous obtenons ainsi :

- Les *fenêtres* de *flux* suivantes :
  - *Flux* F_4 : $f_1 = \{p_1, p_2, p_3\}$, $f_2 = \{p_5, p_7, p_8\}$, $f_3 = \{p_{10}\}$
  - *flux* F_5 : $f_1 = \{p_4, p_6\}$, $f_2 = \{p_9, p_{12}\}$
  - *flux* F_6 : $f_1 = \{p_{11}, p_{13}, p_{14}\}$, $f_2 = \{p_{15}, p_{16}\}$
- Les *fenêtres* de *conversation* suivantes :
  - C_4 : $f_1 = \{p_1, p_2, p_3, p_4\}$, $f_2 = \{p_5, p_6, p_7, p_8, p_9\}$, $f_3 = \{p_{10}, p_{12}\}$
  - C_5 : $f_1 = \{p_{11}, p_{13}, p_{14}\}$, $f_2 = \{p_{15}, p_{16}\}$
- Les *fenêtres* de *lien* suivantes :
  - $f_1 = \{p_1, p_2, p_3, p_4\}$
  - $f_2 = \{p_5, p_6, p_7, p_8, p_9\}$
  - $f_3 = \{p_{10}, p_{11}, p_{12}, p_{13}\}$
  - $f_4 = \{p_{14}, p_{15}, p_{16}\}$

2.3.5 Conclusion

Ainsi, nous avons décrit notre modèle de représentation des communications et illustré son utilisation grâce à des exemples. Dans la section suivante, nous présentons l'algorithme que nous utilisons pour détecter les anomalies à partir des *fenêtres de description instantanée*.

2.4 L'algorithme de détection

Le processus de détection d'anomalies que nous avons conçu repose sur l'utilisation d'un algorithme non-supervisé de détection d'anomalies.

2.4.1 Choix de l'algorithme de détection

Nous avons présenté dans la première section de ce chapitre (section 2.1) les contraintes particulières induites par le contexte d'exécution du processus de détection. Par conséquent, le choix de l'algorithme a été motivé par les aspects suivants :

- Le coût matériel et l'impact sur les performances.
- Le changement de la nature du trafic au cours du temps.
- La diversité dans les propriétés temporelles des anomalies que nous désirons détecter.

— L'autonomie d'exécution du système.
— L'impossibilité d'accéder à la charge utile au niveau applicatif.
— L'autonomie dans l'apprentissage de l'algorithme.

Les méthodes couramment utilisées dans la détection autonome d'anomalies sont les suivantes :

— Approche par clustering.
— Approche des plus-proches voisins.
— Approche par composante principale.
— Approche par classification (SVM).
— Approche par l'étude des séquences temporelles.

Les approches par clustering sont principalement capables de déceler des anomalies ponctuelles. Dans notre cas, nous désirons également pouvoir détecter des anomalies contextuelles ainsi que collectives telles que nous les avons définies dans notre état de l'art (sous-section 1.2.1).

L'approche des plus-proches voisins est généralement coûteuse en temps pour effectuer les prédictions ainsi que gourmande en ressources mémoires [36]. De plus, elle est particulièrement sensible au fléau de la dimensionnalité [65].

OCSVM (pour *one-class SVM*) est l'algorithme le plus souvent utilisé pour la détection d'anomalies. Cependant, il n'est pas très performant lorsque le corpus de données utilisé contient du bruit.

L'approche par composante principale [42] repose sur l'utilisation exclusive de communications normales lors de la phase d'apprentissage de l'algorithme. Cependant, dans un contexte industriel il est difficile de garantir qu'aucune anomalie n'aura lieu pendant cette phase d'apprentissage puisque celle-ci doit impérativement se dérouler lorsque les véhicules sont utilisés par leurs propriétaires. En effet, chaque utilisateur peut avoir différents comportements d'utilisation du véhicule et par conséquent engendrer des routines différentes dans les communications.

La dernière approche est celle de l'étude des séquences temporelles, c'est à dire, déceler les changements de l'état du trafic au cours du temps. Il existe de nombreux algorithmes capables d'étudier des séquences. Cependant, l'une des seules approches non-supervisée est celle des autoencodeurs [48] organisés sous la forme de LSTM [79] capables d'apprendre à reconnaître de façon autonome les anomalies à l'intérieur de séquences comme des communications.

Toutefois, ces approches ont toutes un point commun : **lorsque la phase d'apprentissage non-supervisé de l'algorithme prends fin il n'est plus capable d'apprendre de nouvelles choses**.

Par conséquent, si le trafic venait à évoluer, la détection serait de moins en moins fiable et l'algorithme devrait être ré-initialisé de façon à s'adapter au nouveau trafic.

À notre connaissance, le seul algorithme non-supervisé capable d'apprendre en permanence et de s'adapter à l'évolution des séquences auxquelles il est sujet (comme l'évolution du trafic) est l'algorithme à mémoire temporelle hiérarchique ou **HTM** (pour *hierarchical temporal memory*).

La sous-section qui suit présente les propriétés principales de l'algorithme permettant d'argumenter en faveur de son utilisation dans le cadre de la détection

d'anomalies. Ces propriétés sont extraites des détails du fonctionnement de l'algorithme (Appendice B).

2.4.2 HTM

Initialement présenté dans «On Intelligence» par Hawkins et Blakeslee [55], HTM a déjà démontré de bonnes capacités d'apprentissage de séquences [33], mais aussi de détection d'anomalies dans des données temporelles ou séquentielles [6, 51, 122].

2.4.2.1 Propriétés

L'algorithme HTM présente de nombreuses propriétés le rendant propice à son utilisation dans le contexte de la détection d'anomalies dans les réseaux véhiculaires.

– **Résistance au bruit** – Dans le contexte de la détection d'anomalies, le bruit peut être assimilé à de faibles variations dans les attributs des communications. Par exemple, de petites variations de débit ne signifient pas forcément qu'une attaque est en cours ou qu'une anomalie a eu lieu. Il en est de même pour la perte de quelques paquets. Tout ceci peut être simplement la conséquence d'événements bénins dus au temps de propagation des paquets sur le réseau, à la congestion du réseau ou à la qualité du signal cellulaire du véhicule. Il est donc nécessaire pour un bon détecteur d'anomalies de ne pas générer de faux positifs à cause de ces événements qui, du point de vue de la sécurité, n'ont pas un impact important. L'apprentissage de HTM est résistant de par la structure et l'organisation des neurones utilisés par l'algorithme.

– **Apprentissage continu** – Contrairement aux autres méthodes comme HMM ou LSTM, HTM modifie de façon continue la structure du réseau de neurones en manipulant constamment les liens entre chaque neurone. Ceux-ci sont renforcés ou réduits en fonction des prédictions réalisées par l'algorithme. Cette plasticité le rend donc également capable d'oublier des liens s'ils ne sont pas souvent sollicités. Or, dès lors que l'on considère que le trafic du véhicule est soumis à une quelconque évolution, cette propriété est indispensable au bon fonctionnement d'un détecteur d'anomalies.

– **Les prédictions sont sensibles au contexte** – La structure des neurones procure une mémoire à l'algorithme qui lui permet de baser ses prédictions sur l'événement courant ainsi que les événements qu'il a pu rencontrer par le passé. Une même entrée peut ainsi produire des résultats différents en fonction du contexte passé et l'algorithme est théoriquement capable de reconnaître un trafic peu fréquent sans pour autant le catégoriser comme anormal.

– **Chaque prédiction détecte des anomalies** – Chaque prédiction réalisée par l'algorithme indique si l'événement courant était prévu ou non, ce qui est crucial pour la détection d'anomalies. L'algorithme n'est peut-être pas capable de prédire précisément l'événement suivant, mais la modélisation de la mémoire de l'algorithme le rend capable de prédire plusieurs événements possibles à l'instant suivant. Par conséquent, s'il détecte qu'un événement n'est en aucun cas ressemblant à ce qu'il avait prévu c'est donc qu'il est anormal. Il faut cependant noter que pour que l'algorithme soit capable de détecter avec précision ces anomalies, celles-ci doivent obligatoirement rester peu fréquentes par rapport au trafic normal comme nous avons pu le constater dans l'évaluation de HTM (sous-section 3.5.2).

Par ailleurs, contrairement aux modèles de réseaux de neurones artificiels classiques, HTM n'a pas besoin de grandes quantités de données pour obtenir de bons résultats de classification [125, 34].

2.4.3 Vue d'ensemble

HTM est un ensemble de structures et d'algorithmes, reposant sur les composants suivants (Figure 2.7) :

— L'encodeur (ou *Encoder*) des données présentées en entrée à l'algorithme
— Un réseau de neurones composé d'un ensemble de mini-colonnes dans lesquelles résident plusieurs neurones pyramidaux [111].
— L'algorithme de la représentation spatiale (ou *Spatial Pooler*) et de la Mémoire Temporelle (ou *Temporal Memory*) qui manipulent les liens entre les neurones du réseau en fonction des entrées de l'algorithme.
— Un classifieur (ou *Classifier*) chargé de juger si une entrée est anormale ou non.

Le fonctionnement de l'algorithme peut être résumé ainsi :

1. À l'instant `(t)`, l'encodeur convertit les entrées présentées à l'algorithme en vecteurs binaires de taille fixe, aussi appelés **représentations distribuées éparses** (ou **SDR** pour *Sparse Distributed Representations*).
2. L'algorithme de la représentation spatiale, représenté en turquoise sur la figure, déclenche à partir de la SDR de l'entrée, l'activation d'une **petite quantité** de mini-colonnes du réseau de neurones. L'ensemble des mini-colonnes du réseau est représenté sous la forme d'une autre SDR où les bits actifs représentent les mini-colonnes actives à l'instant `(t)`.
3. L'algorithme de la mémoire temporelle, représenté en jaune sur la figure, apprend les séquences de l'activation des mini-colonnes et effectue des prédictions sur l'état futur du réseau. Il produit donc une SDR représentant les mini-colonnes dont l'activité à l'instant `(t)` a été prédite à l'instant `(t-1)`.
4. Ces prédictions sont ensuite présentées au classifieur. Dans notre cas, nous utilisons une fonction qui calcule le score d'anormalité d'une entrée en comparant la SDR produite par la représentation spatiale à l'instant `(t)` et la SDR prédite à l'instant `(t-1)` par l'algorithme de la mémoire temporelle.

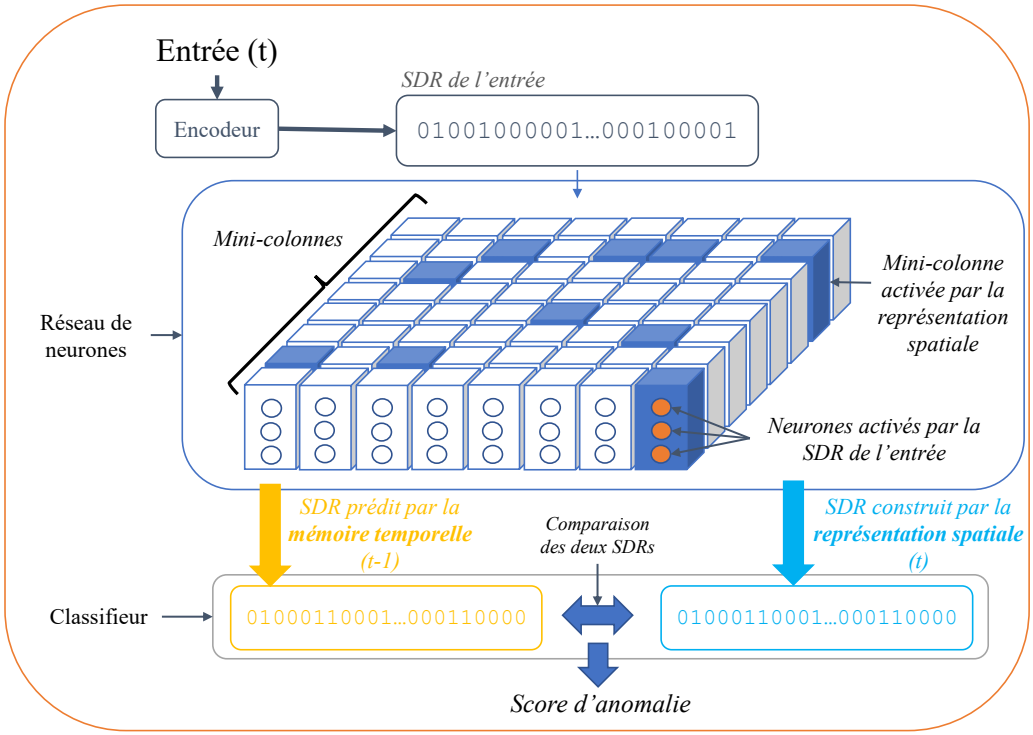


FIGURE 2.7 – Représentation du fonctionnement de HTM

La représentation spatiale et la mémoire temporelle sont donc deux concepts algorithmiques qui gouvernent ensemble le processus d'apprentissage de HTM en manipulant les liens des neurones du réseau en fonction des SDR présentées par l'encodeur à l'entrée du réseau.

2.4.4 Les Encodeurs et SDR et leurs propriétés

Les encodeurs transforment donc les attributs des entrées de l'algorithme, que ce soient des valeurs scalaires ou des données sous forme de texte en SDR. Le fonctionnement de l'algorithme est intrinsèquement lié aux SDR et leurs propriétés fondent le socle sur lequel repose le processus d'apprentissage de HTM. Ce sont des vecteurs binaires définis comme suit [97] :

— Leur création est déterministe et leurs dimensions constantes.
— Le nombre de bits actifs dans le vecteur (bit à 1) est le même pour toutes les valeurs encodées (*sparsity*).
— Les données similaires sont encodées dans des vecteurs dont une partie des bits actifs sont les mêmes.

Les encodeurs peuvent ainsi être conçus de n'importe quelle manière pourvu que les SDR produites respectent ces contraintes.

La première propriété intéressante de ces SDR est leur capacité. Par exemple, prenons une SDR de 2048 bits pour 40 bits actifs, celle-ci dispose d'une capacité d'encodage de $\binom{2048}{40}$ [97]. Cela représente un total colossal de 2.37×10^{84} valeurs dif-

férentes [6] qui permet d'encoder très finement les différentes valeurs que les attributs peuvent prendre.

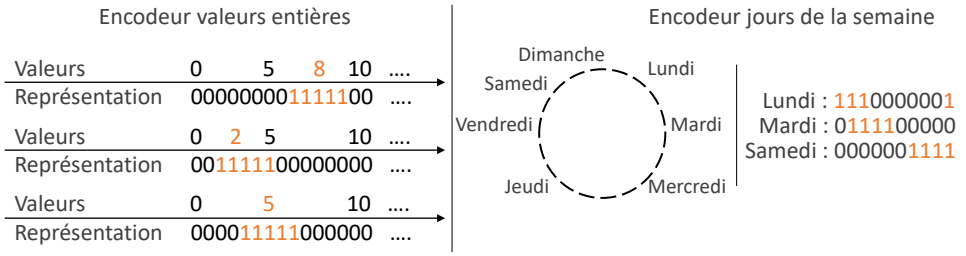


FIGURE 2.8 – Exemples d'encodeurs de valeurs entières et des jours de la semaine.

Purdy [97] détaille quelques exemples d'encodeurs que nous reprenons dans la Figure 2.8. Ils illustrent la création des SDR pour l'encodage des jours de la semaine et d'entiers. On remarque par exemple, que les représentations de Lundi et Mardi ont 2 bits actifs en commun tandis que Mardi et Samedi n'en partagent pas. De même pour la représentation des entiers où on constate que la valeur 2 est très différente de 8, mais similaire à 5.

– **Intérêt** – Cette propriété de superposition est particulièrement importante lorsque nous nous intéressons à la détection d'anomalies puisque des SDR similaires auront tendance à déclencher l'activation des mêmes mini-colonnes rendant ainsi le processus d'apprentissage particulièrement sensible aux similarités observées dans les données d'entrée. Nous revenons sur l'intérêt de cette propriété dans la sous-section suivante.

2.4.5 L'apprentissage de HTM et la structure des neurones utilisés

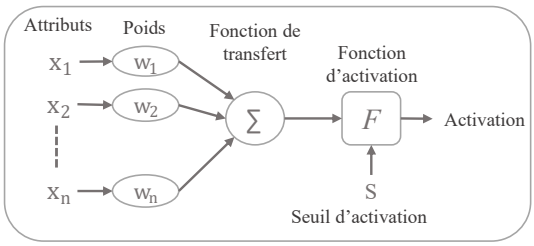
2.4.5.1 Apprentissage

Avant de détailler les points clés du fonctionnement de l'apprentissage de HTM, nous devons tout d'abord présenter le modèle de neurone utilisé par l'algorithme car ils se distinguent des réseaux de neurones plus répandus notamment par leur structure pyramidale.

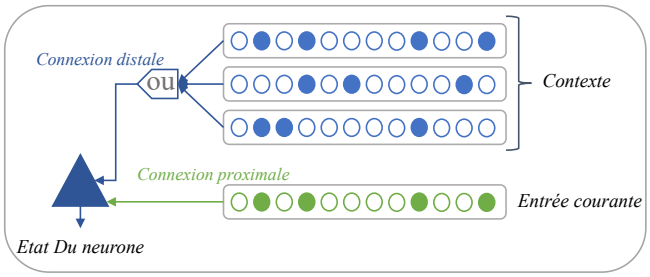
Les réseaux de neurones classiques sont composés de neurones dont l'état de sortie (actif ou inactif) dépend de : la somme pondérée des attributs présentés en entrée, d'une fonction d'activation et du seuil à partir duquel le neurone est considéré comme actif ou inactif (Figure 2.9a). Les neurones utilisés dans HTM disposent quant à eux de deux types de connexions : proximale et distale (Figure 2.9b) et 3 états de sorties possibles : **actif par connexion proximale**, **actif par connexion distale** ou **inactif**.

Ces états de sorties sont directement liés à la structure du réseau de HTM et aux algorithmes de la représentation spatiale et de la mémoire temporelle.

6. Les auteurs de l'algorithme comparent souvent cette valeur avec le nombre d'atomes dans l'univers observable qui est d'environs 10^{80}



(a) Structure du neurone couramment utilisé dans les réseaux de neurones.



(b) Structure pyramidale des neurones utilisés par HTM.

FIGURE 2.9 – Comparaison de la structure du neurone pyramidal utilisé dans HTM et celle des neurones classiques.

– **Connexion Proximale** – Les connexions proximales lient le réseau avec l'espace des entrées grâce à des segments composés d'un ensemble de connexions proximales (en vert sur la Figure 2.9b). Cet espace sert de réceptacle aux différentes SDR présentées en entrée du réseau où un bit actif est représenté par un cercle vert plein, et un bit inactif simplement par un cercle vert. Les neurones d'une même mini-colonne sont tous connectés aux mêmes bits de l'espace des entrées par le même segment. L'algorithme de la représentation spatiale est chargé de déclencher l'activation d'une **petite quantité** de mini-colonnes en fonction des segments auxquels chacun des bits actifs des SDR présentées en entrée du réseau est connecté.

Ces connexions proximales sont créées, maintenues et détruites par l'algorithme de la représentation spatiale en fonction de la fréquence d'apparition de chaque bit des SDR et d'activation des mini-colonnes.

– **Intérêt** – Le fait que chaque mini-colonne contiennent plusieurs neurones permet à l'algorithme de produire différentes prédictions pour une même entrée. Plus précisément, pour des mini-colonnes de x neurones et w mini-colonnes actives par itération, il existe x^w façons différentes de décrire la même entrée dans différents contextes [5].

– **Connexion Distale** – Les connexions distales quant à elles, lient les neurones du réseau entre eux (en bleu sur la figure) sous la forme d'autres segments composés, dans ce cas, d'un ensemble de connexions distales. Chaque neurone peut ainsi disposer de plusieurs segments distaux le connectant à différents ensembles de neu-

rones qui peuvent ou non appartenir à d'autres mini-colonnes. Les neurones actifs sont alors représentés par des cercles pleins bleus et les neurones inactifs par un simple cercle bleu. Lorsque suffisamment de neurones d'un des segments distaux du neurone sont actifs, celui-ci est passé en état prédictif par l'algorithme de la mémoire temporelle.

– **Intérêt** – Ce procédé sert de mécanisme de prédiction où chaque fois qu'un neurone est activé par un segment proximal il a une chance d'activer d'autres neurones via les segments distaux auxquels il est connecté. Si un neurone en état prédictif est alors activé par un segment proximal à l'itération suivante, c'est à dire par une entrée présentée au réseau, c'est que la prédiction était bonne et que les connexions distales doivent être renforcées.

Le renforcement de ces liens aura pour conséquence d'entraîner les neurones d'une mini-colonne à reconnaître un certain nombre de motifs et de leur permettre de prédire ceux sensés apparaître dans les itérations suivantes [7].

2.4.5.2 Exemple

Afin de clarifier la notion d'état prédictif et d'apprentissage nous décrivons un exemple simplifié dans le contexte de la détection d'anomalies. Pour ce faire, nous étudions l'activation des neurones d'un réseau simplifié lorsqu'il est soumis à une séquence de flags TCP. Ce scénario est inspiré des illustrations de Hawkins et Ahmad [54] et décrit par la Figure 2.10.

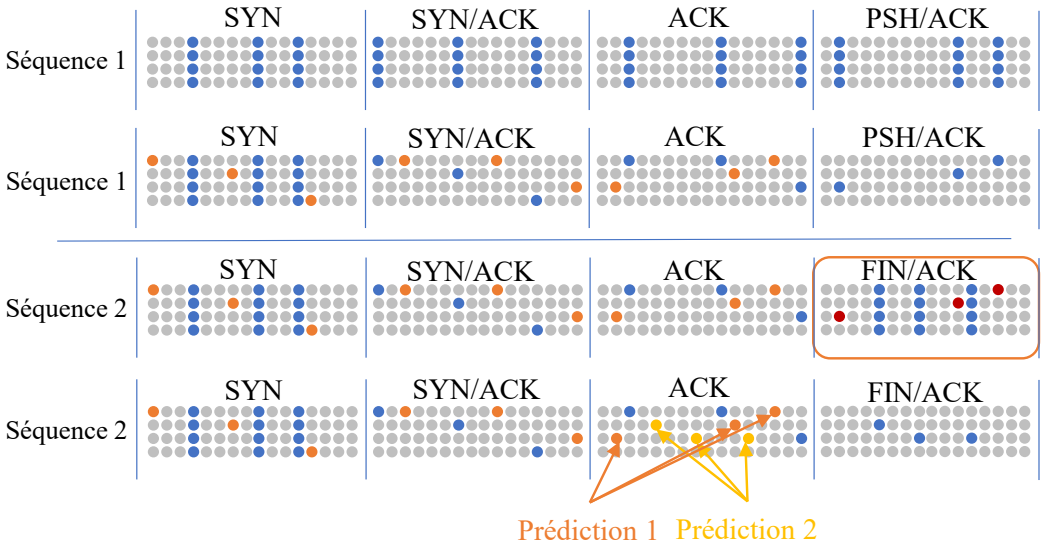


FIGURE 2.10 – Illustration du processus d'apprentissage et de prédiction des neurones de HTM

7. formant ainsi ce que les auteurs de l'algorithme appellent de «petits détecteurs de coïncidences»

Cette figure représente l'activation des neurones du réseau sous la forme de mini-colonnes en fonction de séquences de flags TCP soumises à l'algorithme.

Dans notre exemple le réseau de neurones est composé de 16 mini-colonnes de 4 neurones chacune.

À l'initialisation de l'algorithme, chaque flag présenté pour la première fois au réseau entraîne le déclenchement de chaque neurone des mini-colonnes qui ont été activées par les différents flags, ce qui est illustré par la première ligne de la figure. Supposons maintenant que nous soumettons un nombre suffisant de fois cette même séquence à l'algorithme. Les liens distaux entre les neurones d'une mini-colonne activée par un flag seront renforcés jusqu'à ce qu'à la vue d'un flag, les neurones soient capables de prédire le flag suivant. Ce comportement est illustré par la seconde ligne. Par exemple, lorsque le flag SYN est reçu, alors 3 mini-colonnes sont activées (en bleu) et 3 neurones passent en état prédictif (en orange). On remarque alors que lorsque le prochain flag est reçu (SYN/ACK), les mêmes neurones qui étaient dans l'état prédictif deviennent actifs à cette itération tandis que trois autres prédisent que le prochain flag sera ACK.

L'algorithme a donc été capable d'apprendre la séquence SYN, SYN/ACK, ACK, PSH/ACK qui correspond à l'établissement du handshake TCP puis de la transmission d'un paquet de données.

Si maintenant nous nous intéressons à la troisième ligne de la figure où nous soumettons une autre séquence à ce même réseau, nous remarquons que la prédiction du dernier flag (FIN/ACK) est erronée. En effet, le flag prévu était ACK et les neurones qui étaient en état prédictif (en rouge) ne sont pas ceux actifs lorsque FIN/ACK est traité par l'algorithme. Ainsi, la séquence SYN, SYN/ACK, ACK, FIN/ACK serait considérée comme anormale par l'algorithme.

Toutefois, si la séquence venait à être présentée un nombre suffisant de fois à l'algorithme celui-ci serait capable de s'habituer à cette nouvelle séquence. Ce comportement est illustré par la dernière ligne où lorsque le ACK est reçu, 2 prédictions sont alors possibles soit le PSH/ACK de la séquence 1, soit le FIN/ACK de la séquence 2. Et lorsque le FIN/ACK est finalement reçu on constate que les neurones actifs correspondent à la seconde prédiction (en jaune).

– **Intérêt** – Par conséquent, l'algorithme est alors capable de s'adapter en continu à l'évolution du trafic, limitant ainsi les probabilités que l'algorithme devienne inefficace. Cette propriété d'apprentissage est cruciale car elle permet la reconnaissance de différentes séquences dans les données observées par l'algorithme, ce qui le rend robuste aux changements de comportement du réseau tout en étant capable de détecter les anomalies lorsqu'elles apparaissent.

2.4.5.3 Confiance dans la prédiction

Nous avons indiqué comment le processus d'apprentissage avait lieu dans HTM et combien la structure et l'organisation des neurones en mini-colonne sont importantes. Nous nous intéressons maintenant à la confiance que nous pouvons avoir

dans les prédictions de l'algorithme compte tenu de la structure de ses neurones et la représentation des données sous la forme de SDR.

À chaque itération, l'algorithme n'active qu'une petite partie (environ 2%) des mini-colonnes du réseau. Pour ce faire, le nombre de connexions actives entre une mini-colonne et la SDR d'entrée doit être suffisamment élevé. Il convient alors de fixer un seuil à partir duquel on considère que la mini-colonne est activée par les bits actifs de la SDR d'entrée.

En effet, est-il nécessaire que chacun des bits actifs de la SDR soit connecté à la mini-colonne pour que celle-ci soit activée ? Pour répondre à cette question, Ahmad et Hawkins [5] étudient la probabilité que deux SDR partagent un nombre θ de bits en commun.

Tout d'abord, le nombre de SDR partageant au moins θ bits actifs est exprimé par la formule suivante :

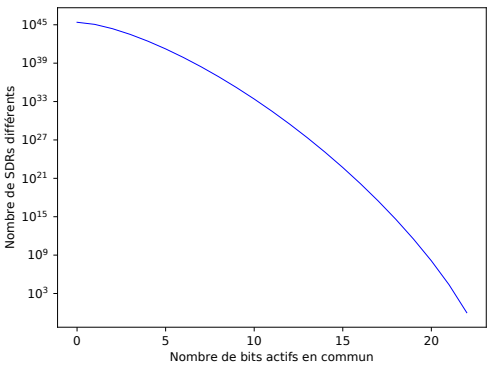
$$|\Omega_x(n, w, \theta)| = \binom{w}{\theta} \times \binom{n-w}{w-\theta} \tag{2.5}$$

Où :

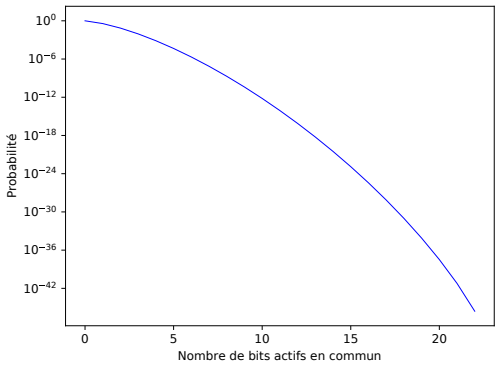
— n représente le nombre de bits de la SDR, w le nombre de bits actifs de la SDR, et θ le nombre de bits actifs en commun.

La probabilité que deux SDR partagent θ bits en commun est alors exprimée par la formule suivante [5] :

$$fp_w^n(\theta) = \frac{\sum_{b=\theta}^{w} |\Omega_x(n, w, b)|}{\binom{n}{w}} \tag{2.6}$$



(a) Nombre de SDR différentes en fonction du nombre de bits actifs en commun pour des SDR où $n = 1080$ et $w = 22$. (échelle logarithmique)



(b) Taux de faux positifs en fonction du nombre minimum de bits actifs en commun pour des SDR où $n = 1080$ et $w = 22$. (échelle logarithmique)

La Figure 2.11a représente l'évolution du nombre de SDR différentes en fonction du nombre de bits en commun (avec $n = 1080$ et $w = 22$) [8]. Par exemple, avec $\theta = 21$

8. Il s'agit de la plus petite SDR manipulable par la librairie python qui implémente l'algorithme

il existe 23276 SDR différentes partageant 21 bits actifs en comparaison le nombre de SDR partageant $\theta = 18$ bits actifs est de 3.81×10^{14}. On remarque que le nombre de SDR différentes décroit de façon exponentielle plus le nombre de bits actifs en commun est grand.

La Figure 2.11b représente la probabilité pour que deux SDR de $n = 1080$ et $w = 22$ partagent θ bits actifs en commun. Par exemple, avec $\theta = 21$ la probabilité est de 5.97×10^{-42} en comparaison avec $\theta = 10$ nous obtenons 6.48×10^{-13} ou encore 4.56×10^{-5} avec $\theta = 5$. Ainsi, dès lors que des SDR ne partagent ne serait-ce que 20% de bits actifs, il est très peu probable que l'algorithme se trompe en considérant les SDR comme identiques.

– **Intérêt** – Par conséquent, cette propriété rend l'algorithme extrêmement résistant au bruit, puisque deux SDR peuvent être considérées comme identiques avec peu de chance d'erreur bien que les SDR ne partagent qu'une petite proportion de bits en commun. De plus, cela permet aussi à l'algorithme de généraliser ses prédictions en fonction des SDR auxquelles il est soumis assurant ainsi que les neurones ne sont pas sujet au surapprentissage, mais plutôt apprennent à reconnaître des motifs généraux. Par exemple, si nous étudions le débit d'un *flux*, une mini-colonne ne sera pas entraînée à s'activer uniquement si la SDR représente la valeur 80.3Mbps, mais si la valeur appartient à la plage entre 78 et 85Mbps.

2.4.6 Conclusion

Nous avons présenté dans cette section les raisons qui nous ont poussés à sélectionner HTM comme algorithme de détection d'anomalies. HTM démontre :

— que son apprentissage est résistant au bruit,
— qu'il est capable d'apprentissage en continu,
— que ses prédictions sont sensibles au contexte des séquences d'événements traitées par le passé,
— que la détection est basée sur la vraisemblance d'apparition d'un événement.

Ainsi, HTM est à ce jour le meilleur candidat pour la détection d'anomalies du point de vue de ses capacités d'adaptation à un trafic changeant, de ses capacités d'apprentissage autonome et en continu, de l'impact sur les performances et enfin, de la diversité des anomalies que nous souhaitons être capables de détecter.

La section suivante présente le dernier problème que nous avons cherché à résoudre, à savoir le traitement des anomalies détectées par l'algorithme.

2.5 Traitement des anomalies

La détection des anomalies n'est qu'un aspect de la gestion des événements de sécurité dans le contexte du véhicule connecté. Nous souhaitons également explorer

HTM https://github.com/numenta/nupic

la définition de mécanismes permettant de gérer efficacement les anomalies détectées.

En effet, la gestion des alertes constitue un problème récurrent dans les systèmes de détection d'intrusions ou d'anomalies [108]. Un système générant trop d'alertes représente une charge supplémentaire pour les opérateurs qui peuvent avoir tendance à les ignorer lorsque leur fréquence d'apparition est trop élevée. De plus, pour qu'une alerte soit exploitable et compréhensible par les opérateurs ou simplement pour assurer la traçabilité des anomalies, il faut que suffisamment d'informations soient réunies au sein d'un rapport présentant le contexte dans lequel l'anomalie a eu lieu ainsi que les acteurs impliqués dans celle-ci.

Ce contexte pourrait être décrit par l'ensemble des autres communications qui avaient lieu lorsque l'anomalie est apparue, les hôtes impliqués, les propriétés du trafic à cet instant. En somme, suffisamment d'informations pour que l'anomalie puisse être comprise et étudiée par un opérateur ou exploitée dans le cadre d'une analyse forensique.

L'objectif du traitement des anomalies est donc double :

— S'assurer que les paquets responsables des anomalies soient identifiés.

— Assurer la traçabilité des anomalies détectées grâce à des rapports.

Par conséquent, nous avons conçu un mécanisme qui collecte les informations nécessaires à la génération d'un rapport lorsqu'une anomalie est détectée et qui s'assure que tous les paquets responsables d'une anomalie soient identifiés.

Cette section présente le processus de gestion des *fenêtres de description instantanée* anormales détectées par HTM. Ce traitement est basée sur la définition d'une ontologie décrivant les relations entre les *fenêtres* qui servent à la détection, les hôtes du réseau et les paquets échangés. Des règles d'inférences sont définies afin de construire de façon simple une description du contexte dans lequel l'anomalie est apparue ainsi que l'ensemble des paquets responsables.

2.5.1 Exemple de traitement d'une anomalie

Grâce à la définition des *fenêtres de description instantanée*, nous sommes en mesure d'extraire des attributs décrivant la nature du trafic auquel est soumis le véhicule à chaque instant. Cependant, nous souhaitons exploiter le contexte lié à ces *fenêtres de description instantanée* dans la phase de traitement puisque celui-ci pourrait permettre aux opérateurs de mieux les anomalies détectées.

Afin d'illustrer cette modélisation, nous présentons un autre exemple d'anomalie que nous cherchons à détecter : le dysfonctionnement du service de télémétrie du véhicule.

Ce scénario est représenté par la Figure 2.12, laquelle illustre une panne du système en charge du partage des informations relevant de l'état du véhicule et de son environnement. Il s'agit donc d'une situation critique puisque le bon fonctionnement des services ITS repose sur la centralisation et la dissémination de l'information de l'état des routes aux autres véhicules.

Nous nous concentrons dans cet exemple sur les attributs suivants :

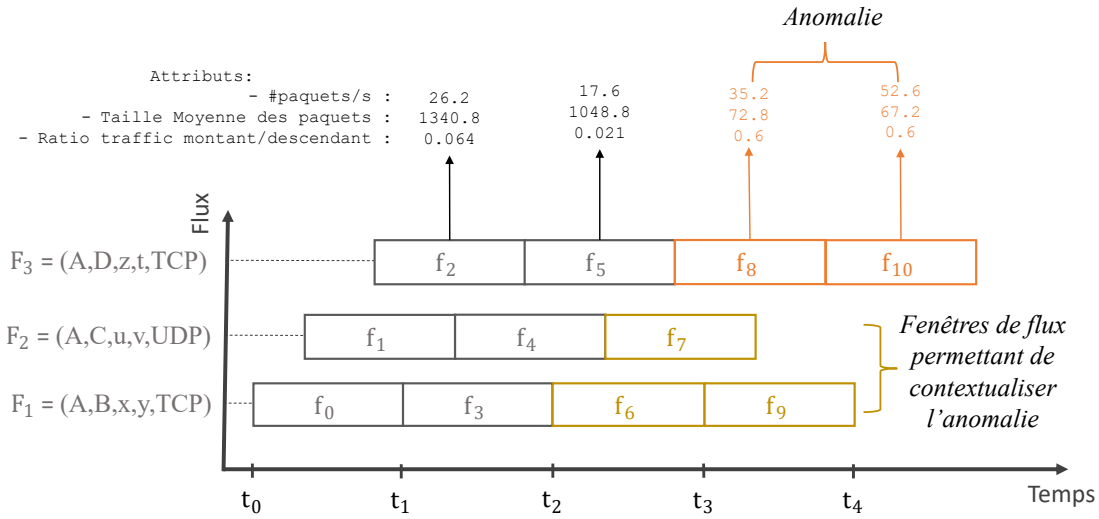


FIGURE 2.12 – Illustration d'une anomalie dans le service de télémétrie du véhicule.

— Le nombre de paquets par seconde capturés par *fenêtre*.
— La taille moyenne des paquets capturés par *fenêtre*.
— Le ratio de trafic descendant/montant par *fenêtre*.

Le service de télémétrie communique par le *flux* F_3. Les *flux* F_1 et F_2 représentent d'autres communications du véhicule qui se déroulent en même temps. Le *flux* F_3 est composé de 4 *fenêtres* dont les 2 dernières sont anormales. En effet, nous observons que les 2 *fenêtres* représentées en orange, se distinguent par des attributs très différents, notamment la taille des paquets qui est très faible. Ainsi, lorsque le processus de détection reconnaît cette anomalie, nous désirons générer un rapport qui inclut :

— les deux *fenêtres de description instantanée* anormales et tous les paquets qui en font partie.
— les dernières *fenêtres* des *flux* F_1 et F_2 et tous les paquets qui en font partie.

En effet, les *flux* F_1 et F_2 étaient actifs au moment de l'anomalie et peuvent donc permettre d'optimiser le traitement celle-ci par les opérateurs.

2.5.2 L'ontologie

Nous devons donc être capable de construire une représentation de toute les communications du véhicule en regroupant chaque *fenêtre de description instantanée* en fonction des hôtes impliqués, des *flux*, des *liens* ou des *conversations*. En conservant les relations entre les *fenêtres* créées au fur et à mesure de la capture des paquets sur le réseaux et les hôtes impliqués dans les communications nous pourrons ainsi extraire les échanges pertinents pour l'analyse et l'historisation d'une anomalie lorsqu'elle est détectée.

Par exemple, l'ensemble des *flux*, *conversations* ou *liens* que le véhicule a pu entretenir avec l'hôte impliqué dans un échange anormal, ou encore l'ensemble des

autres communications qui avaient lieu lorsque l'anomalie a été détectée.

Nous nous sommes donc intéressés aux moyens de représentation des données que nous pourrions utiliser, le choix s'est principalement orienté entre deux méthodes : les bases de données relationnelles ou les ontologies.

Les bases relationnelles sont extrêmement répandues pour la gestion de l'information, cependant elles sont assez rigides et ne permettent pas de faire évoluer le modèle de données facilement.

Au contraire, les ontologies [50] se démarquent depuis quelques années pour leurs capacités à représenter des informations compréhensibles par les machines ainsi que les humains tout en permettant une grande liberté dans la conception des modèles de données.

Les ontologies sont des descriptions formelles de connaissances ou d'un ensemble de concepts. Celles-ci sont définies à partir d'un ensemble de classes, d'attributs et de relations ainsi que d'un ensemble de restrictions, règles et axiomes. Par exemple, nous pouvons utiliser une ontologie pour définir les relations entre un paquet et des hôtes en indiquant que l'un en est la destination et l'autre la source. En décrivant ainsi les instances des objets sauvegardés dans la base de connaissances de l'ontologie, il est possible non seulement de partager les données enregistrées dans la base sous la forme de graphes de connaissance, mais aussi d'extraire de nouvelles connaissances de l'ontologie à partir de règles et d'axiomes.

Ces règles peuvent ainsi nous permettre d'extraire le contexte durant lequel une anomalie a été détectée, mais aussi de regrouper tous les paquets qui pourraient être impliqués dans celle-ci.

En définitive, les ontologies permettent [90] :

— De partager la structure d'une information.

— De définir de façon explicite les particularités d'un domaine étudié.

— D'analyser les connaissances d'un domaine particulier.

De plus, contrairement aux bases relationnelles, les ontologies sont faciles à comprendre grâce à leur représentation sous forme de graphe, elles sont aussi extensibles par le simple ajout de nouvelles relations et concepts aux ontologies existantes.

2.5.2.1 Les classes et les relations

Par conséquent, nous avons fait le choix de créer une ontologie pour représenter de façon simple les relations entre des paquets, des hôtes et les différents types de *fenêtres de description instantanée* que nous avons définis (*lien*, *flux*, *conversation*). Cependant, il existe plusieurs moyens de représenter la même réalité dans une ontologie. Nous avons décidé de structurer notre ontologie autour des échanges entre le véhicule et des hôtes du réseau puisque la plus simple des mesures de sécurité en réaction à une anomalie pourrait être de bloquer les communications en provenance ou à destination de certains ports du véhicule en fonction de l'hôte impliqué grâce à la création d'une simple règle de pare-feu.

Les relations entre les classes de notre ontologie ont pour seul rôle de conserver le contexte lié à chacun des paquets reçus et émis par le véhicule. Nous utilisons ensuite

ces relations afin de générer le rapport des anomalies détectées grâce aux règles d'inférences que nous définissons dans la fin de cette section (sous-section 2.5.4).

Une représentation simplifiée de notre ontologie est illustrée par la Figure 2.13. Un paquet est en relation avec deux membres de la classe Hôte, l'un en est la source, l'autre la destination (*SourceDe* et *DestinationDe*). Ce paquet est associé aux trois différents types de *fenêtres de description instantanée* i.e., *fenêtre* de *flux*, de *conversation* et de *lien* par la relation *FaitPartieDeFenêtre*. Ces *fenêtres* font elles-mêmes partie de collections en fonction du type de *fenêtre* i.e. collection de *fenêtres* de *flux*, de *conversation* ou de *lien*. Enfin, les différentes collections avec un même hôte sont regroupées au sein d'un même échange avec la relation *AppartientÀEchange*.

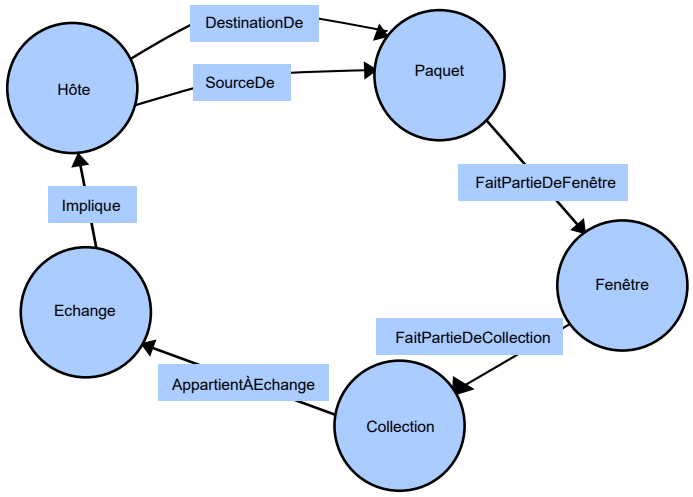


FIGURE 2.13 – Classes et leurs relations à l'intérieur l'ontologie.

2.5.3 La classe Anomalie et ses relations

La classe *Anomalie* s'intègre au reste de l'ontologie comme indiqué par la Figure 2.14. Pour ce faire, nous avons défini plusieurs relations entre les anomalies et les *fenêtres de description instantanée*. Cela nous permet de réunir les paquets impliqués dans les anomalies et les informations contextuelles concernant les échanges qui avaient lieu lors de la détection grâce à des règles d'inférence.

Ainsi, lorsqu'une *fenêtre* est considérée comme anormale, une instance de la classe *Anomalie* est créée au sein de l'ontologie, et la *fenêtre* l'ayant causée y est reliée par la relation *ResponsableDe*.

– **Relation ContaminéPar** – La relation *ContaminéPar* a pour objectif d'identifier les autres paquets pouvant être impliqués dans l'anomalie. Pour ce faire, avant d'ajouter une nouvelle *fenêtre* à une collection nous vérifions si cette collection comporte une *fenêtre* anormale. Si tel est le cas, alors, la nouvelle *fenêtre* est, elle aussi, liée à la même anomalie par la relation *ContaminéPar*.

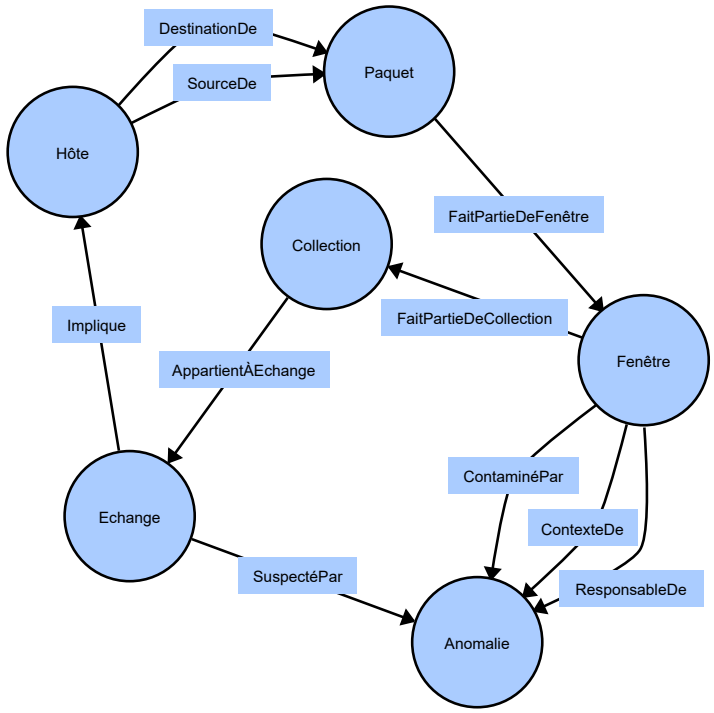


FIGURE 2.14 – Illustration des relations de la classe Anomalie avec le reste de l'ontologie.

– **Relation SuspectéPar** – La relation *SuspectéPar* a pour objectif d'associer à une anomalie les autres échanges que l'hôte responsable de l'anomalie entretient avec le véhicule. Pour ce faire, lors de la détection d'une anomalie, nous interrogeons l'ontologie afin de récupérer l'ensemble des échanges reliés à l'hôte par la relation *Implique*. Nous ajoutons ensuite ces échanges à l'anomalie par la relation *SuspectéPar*.

– **Relation ContexteDe** – Enfin, la relation *ContexteDe* a pour objectif d'associer à une anomalie les *fenêtres* qui étaient actives lors de la détection d'une *fenêtre* anormale. Pour ce faire, à la détection d'une anomalie, nous interrogeons l'ontologie afin de récupérer l'ensemble des *fenêtres* dont la date d'expiration est supérieure à la date de l'anomalie. Ces *fenêtres* sont ensuite ajoutées à l'anomalie par la relation *ContexteDe*.

2.5.4 Les règles d'inférence

Grâce à ces relations, nous pouvons définir des règles d'inférence qui nous permettent de récupérer l'ensemble des paquets concernés par les anomalies détectées par HTM.

Ces règles sont définies selon le principe de transitivité suivant : Si A implique B et B implique C alors A implique C que nous pouvons résumer formellement par

la syntaxe suivante :

— $(A \to B) \wedge (B \to C) \to A \to C$

Ainsi, les paquets impliqués dans les anomalies sont définis par les règles suivantes :

1. Les paquets responsables d'une anomalie sont les paquets appartenant à la *fenêtre* responsable de l'anomalie. Ils sont définis par la règle suivante :
   — Anomalie(A) ∧ ResponsableDe(A, Fenêtre(F))
   ∧ FaitPartieDeFenêtre(F, Paquet(P)) → ResponsableDe(A, P)
2. Les paquets contaminés par une anomalie sont les paquets appartenant aux *fenêtres* contaminées par l'anomalie et sont définis par :
   — Anomalie(A) ∧ ContaminéPar(A, Fenêtre(F))
   ∧ FaitPartieDeFenêtre(F, Paquet(P)) → ContaminéPar(A, P)
3. Les paquets apportant du contexte à l'anomalie sont les paquets appartenant aux *fenêtres* liées à l'anomalie par la relation *ContexteDe*. Ils sont définis par :
   — Anomalie(A) ∧ ContexteDe(A, Fenêtre(F))
   ∧ FaitPartieDeFenêtre(F, Paquet(P)) → ContexteDe(A, P)

Ainsi, les relations et règles d'inférence que nous avons définies nous permettent de dresser un portrait de chaque anomalie détectée.

2.5.4.1 Exemple

Afin d'illustrer ces règles nous reprenons l'exemple de l'anomalie de télémétrie que nous évoquons au début de cette section (Figure 2.12). Nous avons donc les *fenêtres de description instantanée* affectées aux *flux* F_1, F_2, et F_3 de la façon suivante :

— $F_1(A, B, x, y, TCP) = [f_0, f_3, f_6, f_9]$
— $F_2(A, C, u, v, UDP) = [f_1, f_4, f_7]$
— $F_3(A, D, z, t, TCP) = [f_2, f_5, f_8, f_{10}]$

L'anomalie A_0 est détectée dans la *fenêtre* f_8, les *fenêtres* f_6, f_7, f_9 étaient actives en même temps que f_8 et f_{10} appartient au même *flux* que f_8. Le rapport de l'anomalie peut donc être représenté par la Figure 2.15.

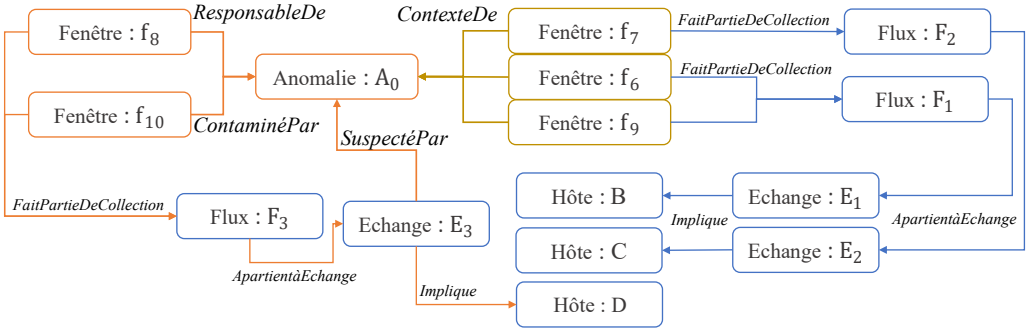


FIGURE 2.15 – Illustration d'un rapport d'anomalie.

2.6 Dispositif expérimental

Afin de mettre en application **Svalinn**, nous avons développé un prototype expérimental pour le processus de détection d'anomalies comprenant l'ontologie, HTM et le traitement des anomalies. Nous présentons dans cette section son architecture.

2.6.1 Architecture de la détection

L'architecture de **Svalinn** est représentée par la Figure 2.16. Une sonde est en charge de capter les communications sur le lien cellulaire au niveau du TCU de la voiture et transmet chaque paquet au moniteur. Celui-ci construit et enregistre les *fenêtres de description instantanée* dans la base de connaissances et transmet chaque *fenêtre* à l'analyseur. L'analyseur, quant à lui, va effectuer la détection d'anomalies en s'appuyant sur l'ensemble des *fenêtres* précédentes et de l'algorithme de détection. Enfin, pour chaque anomalie détectée, la *fenêtre* responsable est identifiée dans la base de connaissance et transmise au planificateur. Celui-ci est en charge de procéder au traitement de l'anomalie et de générer le rapport.

Cette architecture est inspirée de l'architecture MAPE-K (pour *Monitor-Plan-Analyse-Execute over a Knowledge base*). Elle a été présentée par IBM en 2001 notamment dans [62, 30] comme un moyen de structurer et de faciliter le développement de l'informatique autonome. L'objectif était notamment de permettre à des systèmes informatiques de s'auto-superviser afin qu'ils puissent s'adapter à des événements imprévus facilitant ainsi le travail des opérateurs et les interactions du système avec les utilisateurs. Le choix de l'architecture MAPE-K est en effet adapté à notre situation, puisque notre système est voué à être installé directement à l'intérieur du véhicule sans interaction directe possible avec un opérateur.

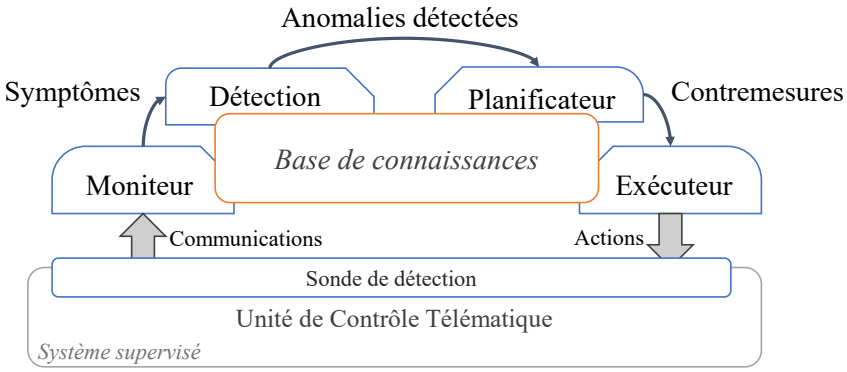


FIGURE 2.16 – Architecture de **Svalinn** (basée sur l'architecture MAPE-K).

Le fonctionnement de l'architecture MAPE-K commence tout d'abord par le système que l'on souhaite superviser et à partir duquel on extrait des informations pour permettre cette supervision. Ces informations sont transmises au Moniteur chargé d'en extraire les connaissances et de générer des symptômes à traiter. Ainsi, l'analyseur traite chacun des symptômes. Il est de ce fait responsable de la prise

de décision sur les actions et changements à opérer sur le système. Il transmet ces requêtes de changement (Request-For-Change, ou RFC) au planificateur. Celui-ci est en charge de la procédure de mise en application des changements demandés par l'analyseur. Il génère donc des plans d'exécution à destination de l'exécuteur. Ce dernier va effectuer les changements sur le système supervisé. Chacun de ces composants effectue ses tâches en se basant sur la base de connaissances notamment pour l'analyse des symptômes.

2.6.2 Implémentation

L'implémentation de la détection a été réalisée entièrement en python (versions 2.7 et 3). Ce langage est très répandu dans le domaine de l'apprentissage automatique car il dispose de nombreuses librairies comme sklearn, tensorflow, ou pandas qui accélèrent grandement le développement de prototypes en mettant à disposition des programmeurs des algorithmes déjà implémentés et éprouvés par les membres de la communauté.

Afin de réduire au maximum l'impact de notre solution en matière de ressources, mais aussi d'isoler la détection d'anomalies du reste des applications exécutées sur l'hôte, nous avons conçu un prototype utilisant une architecture en micro-services exécutés dans des conteneurs grâce au logiciel `docker`[9].

2.6.2.1 Conteneur Logiciel

Le conteneur logiciel est une méthode de virtualisation opérant au niveau du système d'exploitation de la machine hôte et particulièrement de son noyau. Ils permettent la création de plusieurs instances d'espaces utilisateurs dans lesquels des processus peuvent s'exécuter dans leur propre espace virtuel protégeant ainsi la mémoire des processus d'activités malveillantes.

Les processus lancés dans des conteneurs logiciels ont uniquement accès aux ressources qui leur ont été affectées. Ainsi, chaque conteneur dispose de son propre dossier racine accessible uniquement au processus lancé en son sein. De plus, le noyau permet aussi de limiter l'accès aux ressources CPU de la machine hôte de façon à contrôler l'impact de l'activité des conteneurs sur le reste du système.

Un conteneur logiciel exécuté sur un noyau Linux est donc un ensemble de processus :

— Isolés du reste de la machine, empêchant ainsi le conteneur d'accéder ou de voir les ressources ou fichiers de l'hôte ou d'un autre conteneur.
— Utilisant les espaces utilisateurs afin d'avoir une vision privée des interfaces réseaux, des identifiants de processus et des points de montage de la machine hôte.
— Utilisant les groupes de contrôle afin de limiter l'utilisation des ressources CPU, mémoire, et réseau.

9. https://docs.docker.com/

`Docker` est un outil qui permet une configuration simple de ces conteneurs logiciels comparé aux alternatives telles que *chroot*[10] ou encore *lxc*[11]. Il permet de configurer des images au moyen de fichiers (*Dockerfile*) dans lesquels les utilisateurs spécifient la distribution sur laquelle elle doit se baser (par exemple `Ubuntu`, ou `Alpine`) ainsi que les librairies et dépendances que le programme nécessite pour son exécution.

2.6.2.2 Implémentation des composants

Ainsi, chaque composant de l'architecture MAPE-K est exécuté isolément des autres grâce à ces conteneurs. Nous détaillons ci-après le fonctionnement de chacun des composants de l'architecture.

Base de Connaissances Le point central de notre architecture est la base de connaissances. Celle-ci est implémentée grâce à un conteneur basé sur l'image officielle de `MongoDB`[12] et sur la distribution `Ubuntu Xenial`. `MongoDB` est un logiciel de base de données orienté document grâce auquel nous enregistrons les attributs des communications du véhicule que nous souhaitons superviser pour la détection d'anomalies.

Sonde La sonde de détection est en charge de capturer les paquets sur l'interface réseau de l'hôte sur laquelle elle est installée. Pour nos expérimentations, celle-ci est aussi capable de traiter des fichiers au format pcap afin d'effectuer des analyses en mode hors-ligne. Cette sonde est basée sur une distribution minimale nommée Alpine Linux, sur laquelle est installé Python3, la librairie Scapy[13] et zmq[14] pour une taille totale de 201MB. Scapy est une librairie simple d'utilisation qui permet d'envoyer, de capturer et de décoder des paquets réseaux de nombreux protocoles. ZMQ (ou *Zero-MQ*) est une librairie de messagerie permettant la communication inter-processus de manière asynchrone. Ainsi, lorsqu'un paquet est capturé sur une interface ou lu dans un fichier, il est décodé avant d'être transmis à la file de traitement des paquets du moniteur grâce à ZMQ.

Moniteur Le moniteur est composé de plusieurs conteneurs eux aussi basés sur Alpine Linux, python3 et l'interface de programmation MongoEngine. À la réception des paquets qui lui sont transmis par la sonde, il extrait les composants des *flux* supervisés. Il traduit le formalisme des paquets réseaux comme les adresses IP les ports ou les protocoles dans les classes de notre ontologie. Les *fenêtres de description instantanée* sont donc créées par le moniteur. Celui-ci utilise l'interface de programmation MongoEngine afin de peupler la base de connaissances avec les

10. https://linux.die.net/man/1/chroot
11. https://linuxcontainers.org/lxc/manpages/man5/lxc.container.conf.5.html
12. https://www.mongodb.com/
13. https://scapy.net/
14. https://zeromq.org/

éléments extraits du trafic. Chaque insertion d'une nouvelle *fenêtre* déclenche le traitement de cette dernière afin de construire les attributs destinés à l'analyseur comme le nombre de paquets par seconde ou la taille moyenne des paquets contenus dans une *fenêtre*.

Analyseur Enfin, à chaque fois que ces attributs des *fenêtres de description instantanées* sont créés, l'analyseur récupère directement les valeurs nécessaires à l'exécution d'une itération de l'algorithme de détection d'anomalies. L'analyseur est basé sur l'image officielle de Nupic [15], la plateforme d'apprentissage automatique qui implémente l'algorithme HTM.

Planificateur et Exécuteur Le fonctionnement de ces deux composants a été pensé dans le cadre de ces travaux, mais leur réalisation n'a pas abouti à la création du prototype complet capable d'analyser le trafic en ligne. Le rôle du planificateur est de recevoir les résultats positifs de détection et de déclencher leur traitement. Celui-ci se déroule en deux phases qui consistent d'une part, en la consolidation de l'anomalie et d'autre part, la création du rapport sur cette anomalie au moyen de règles d'inférence présentées dans la section 2.5. L'exécuteur quant à lui pourrait réaliser deux types d'opérations. La première consisterait à prendre une action sur le *flux* jugé anormal en ajoutant une règle de pare-feu pour couper le *flux* anormal par exemple. La seconde consisterait à transmettre le rapport d'anomalie à un service distant de gestion de l'information et des événements de sécurité (SIEM).

2.7 Résumé

Dans ce chapitre nous avons présenté l'architecture et le fonctionnement de notre système de détection d'anomalies et d'intrusions baptisé **Svalinn**.

Dans un premier temps, le trafic du véhicule est caractérisé par ce que nous appelons des *fenêtres de description instantanée* des communications. **Svalinn** procède à deux traitements discrets. Le premier concerne la génération des *fenêtres de description instantanée*. Il est déclenché par la capture d'un paquet. Le second concerne le traitement des *fenêtres* à l'expiration du délai de celles-ci.

Ces *fenêtres* décrivent la nature des communications grâce à un ensemble d'attributs temporels issus de l'analyse des propriétés des paquets capturés en continu sur le réseau et affectés aux *fenêtres* avant leur expiration.

À l'expiration du délai des *fenêtres*, celles-ci sont soumises au processus de détection d'anomalies et d'intrusions. Ce dernier utilise l'algorithme à mémoire temporelle hiérarchique (HTM) qui, pour chaque entrée, produit un score permettant d'identifier si une *fenêtre* est anormale, et de lever une alerte si nécessaire.

Dans le cas où une *fenêtre* est considérée comme anormale par l'algorithme, les autres *fenêtres* liées à celle-ci sont regroupées au sein d'un rapport permettant de faciliter le diagnostic de l'anomalie ainsi que son historisation. Ce regroupement est

15. https://hub.docker.com/r/numenta/nupic/

réalisé grâce à un moteur d'inférence. Celui-ci permet d'étendre la couverture d'une anomalie détectée à l'ensemble des paquets liés à cette dernière.

Nous avons également présenté les détails de l'implémentation de **Svalinn** grâce à l'utilisation de conteneurs logiciel et l'utilisation de l'architecture MAPE-K.

Chapitre 3

Evaluation de la détection

Sommaire

L'évaluation des algorithmes et des méthodes utilisés pour la détection d'anomalies est une tâche difficile. Comme souligné par Sommer et Paxson [108] : «la construction d'une méthode d'évaluation peut s'avérer plus difficile que la conception même du détecteur d'anomalies». La difficulté résiderait ainsi dans deux problèmes, à savoir, l'obtention d'un jeu de données d'entraînement suffisamment représentatif pour la validation des algorithmes et l'interprétation des résultats de la détection.

Dans ce chapitre, nous présentons les méthodes qui nous ont permis de pallier ces problèmes. Dans un premier temps, nous présentons le corpus de données que nous avons utilisé pour la validation de **Svalinn**. Ce corpus a été créé à partir d'un environnement d'émulation de réseaux cellulaires de véhicules. Cet outil, que nous avons baptisé **Autobot**, est destiné à produire des jeux de données contenant des communications et des anomalies réalistes pour la validation de méthodes de

détection d'anomalies spécifiques aux réseaux mobiles. Cet outil ainsi que le corpus sont présentés respectivement dans la section 3.1 et la section 3.2.

La section 3.4 présente l'évaluation de **Svalinn** selon les attributs de l'ontologie, du type et de la durée des fenêtres de description. Les résultats de la détection sont présentés dans la section 3.5, laquelle détaille également une comparaison des performances de HTM par rapport à l'algorithme LSTM et l'impact du moteur d'inférence sur la couverture des anomalies.

3.1 Processus de génération du corpus

L'accès à un corpus de données capable de représenter de façon fidèle chaque réseau et ses spécificités, est un problème récurrent dans l'évaluation des méthodes de détection d'anomalies. Nous avons présenté dans notre état de l'art un certain nombre de ces corpus issus d'initiatives comme celle de la compétition KDD99 de la DARPA ou, plus récemment, celle de l'institut Canadien pour la Cybersecurité (CIC), lesquelles ont engendré la publication de corpus [61, 104].

Pour autant, ces jeux de données présentent de nombreux défauts, comme le manque de réalisme des attaques. Par exemple, dans le corpus de KDD99, les anomalies et attaques ont été synthétiquement ajoutées à des communications capturées sur une base militaire en environnement clos. Le corpus publié par le CIC, quant à lui, comporte des attaques qui ont été générées dynamiquement par les chercheurs. Cependant, les communications normales sont créées par le biais de profils de comportement de manière automatisée. Ainsi, il est difficile de trouver des corpus dans lesquels les communications et les attaques ont été capturées de manière naturelle, et ce pour plusieurs raisons. La première, d'ordre légal, concerne l'origine des communications du corpus. Si celles-ci sont issues de véritables systèmes d'information, elles contiennent souvent des données sensibles ou à caractère personnel qu'il est difficile d'anonymiser pour en permettre la publication. La seconde, d'ordre technique, concerne les classes des communications du corpus. En effet, l'utilisation d'algorithmes d'apprentissage suppose que les chercheurs puissent connaître précisément les classes du trafic qu'ils manipulent. Dans le cas présent, il s'agit de savoir différencier le trafic normal de l'anormal, sans quoi il est difficile d'évaluer les résultats des méthodes d'apprentissage. Or, ce processus d'identification du trafic anormal dans de grands jeux de données est une tâche difficile et coûteuse puisqu'elle requiert l'intervention d'opérateurs expérimentés capables de reconnaître par eux-mêmes les anomalies. Enfin, les réseaux cellulaires de véhicules sont encore très peu répandus et il n'existe donc à notre connaissance aucun corpus public.

Jusqu'alors, l'étude des réseaux véhiculaires par la communauté scientifique a principalement été portée vers trois méthodes : la simulation, l'expérimentation grandeur nature ou l'émulation.

La simulation est un moyen efficace de valider de nouveaux concepts et protocoles avant leur déploiement. L'état de l'art en matière de simulateurs dédiés aux

réseaux véhiculaires laisse apparaître deux solutions : `Ns3` [101] et `omnetpp` [119]. Ils sont souvent utilisés en combinaison avec une simulation de mobilité de véhicules grâce à l'outil `Sumo` [66]. Cependant ces solutions ne sont pas prévues pour être exécutées en temps réel et ne peuvent donc pas être utilisées pour tester de réelles applications ou anomalies. De plus, ces outils ne se concentrent pas sur les réseaux cellulaires de véhicules, mais principalement vers les VANETs.

L'autre solution est l'expérimentation grandeur nature. Il est ainsi possible d'expérimenter en temps réel, mais à condition d'acquérir du matériel onéreux et de disposer d'un lieu pour procéder aux expériences, notamment à cause des interférences avec les réseaux cellulaires existants. En effet, il est primordial de s'assurer que les interactions entre les réseaux commerciaux et le réseau expérimental sont impossibles. En particulier pour envisager la génération d'intrusions ou d'anomalies.

La solution que nous avons choisie est l'émulation car celle-ci présente l'avantage de permettre l'utilisation de protocoles et d'applications très réalistes. Elle permet aussi d'avoir la capacité d'émuler un large nombre d'entités sans avoir à en payer le coût comme dans le cadre d'une expérimentation grandeur nature. De plus, l'utilisation d'un environnement isolé permet aussi de générer des attaques de façon sûre, ce qui serait difficile à assurer dans le cadre d'une étude avec de véritables véhicules. Nous avons décidé de créer dynamiquement le corpus au moyen d'un environnement d'émulation des communications.

À cet effet, nous avons conçu **Autobot** en suivant les recommandations présentées par Shiravi *et al.* [105] lesquelles sont résumées comme suit :

— Perception : les effets des attaques et anomalies sur le réseau doivent être perceptibles. Ainsi, elles ne doivent pas être injectées a posteriori, mais générées dynamiquement.
— Évaluation : l'évaluation des résultats requiert l'accès aux informations de classification des anomalies.
— Réalisme : Les activités normales et anormales doivent être suffisamment réalistes.

3.1.1 Vue d'ensemble

Autobot est un environnement d'émulation d'un réseau véhiculaire que nous utilisons pour la création de notre corpus de données. L'émulation a pour objectif de produire des communications réalistes entre des véhicules et un serveur telles qu'elles pourraient avoir lieu dans la réalité. Son fonctionnement est illustré par la Figure 3.1 dont la première partie représente la réalité que nous souhaitons émuler et dans laquelle des applications de véhicules connectés échangent grâce aux communications mobiles avec des serveurs sur Internet. La seconde partie de la figure représente le mécanisme d'émulation que nous avons mis en place afin de reproduire cette réalité. Celui-ci est basé sur l'intercommunication de conteneurs logiciels à l'intérieur de la machine hôte de l'environnement d'émulation.

Nous avons conçu plusieurs applications communicantes réparties en deux familles en fonction des services qu'elles procurent au véhicule.

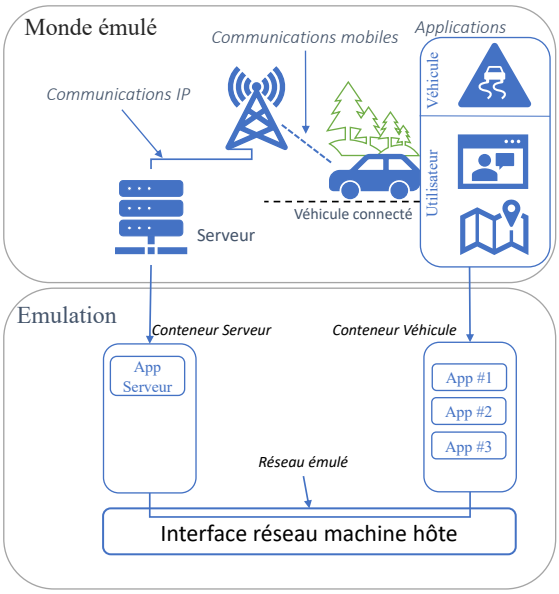


FIGURE 3.1 – Illustration du fonctionnement d'**Autobot**

1. *App-Utilisateur* : pour les applications générant du trafic lié aux services de divertissement disponibles dans les véhicules (email, musique en ligne, ou la navigation)
2. *App-Véhiculaire* : pour les applications liées au fonctionnement du véhicule comme la télémétrie ou les mises à jour.

Pour ce faire, **Autobot** s'appuie sur les mécanismes de conteneurisation du logiciel libre `Docker` [1]. Chaque véhicule ou serveur est exécuté dans un environnement isolé. De plus, nous sommes capables d'émuler un nombre conséquent d'applications à l'intérieur de l'environnement pour autant que la machine hôte soit capable de maintenir les attributs du réseau et la bonne exécution des applications. En effet, le rôle des applications est de produire des communications entre les véhicules et des serveurs ou directement entre les véhicules de façon réaliste. **Autobot** peut ainsi être utilisé afin de tester des applications dans des conditions réseau spécifiques.

3.1.2 Fonctionnement d'Autobot

Nous revenons dans cette sous-section sur le fonctionnement détaillé d'**Autobot** et le processus de création du corpus utilisé pour la détection d'anomalies.

3.1.2.1 Objectifs

Tout d'abord, **Autobot** a pour objectif de permettre la création de corpus de données réalistes pour la détection d'anomalies dans les communications cellulaires

1. https://www.docker.com/

de véhicules grâce à de l'émulation. À ce titre, il convient d'assurer que les échanges émulés sont réalistes quant à leur contenu et leurs propriétés.

Le réalisme du contenu des échanges tient compte des applications déployées dans les véhicules connectés dans le cadre des systèmes de transports intelligents. Il est donc nécessaire que les communications émulées dans **Autobot** soient similaires à celles-ci et que les échanges engendrés par les applications de télémétrie ou de divertissement des véhicules se rapprochent au maximum de la réalité.

Le réalisme dans les propriétés des échanges implique qu'**Autobot** puisse être capable de reproduire les conditions de communications que les véhicules rencontrent dans la réalité. Ainsi, il faut imiter les propriétés relevant de la qualité de service constatée sur les réseaux mobiles. L'environnement d'émulation doit donc être capable de manipuler de façon précise les échanges qui ont lieu en son sein.

3.1.2.2 Virtualisation et Conteneurisation

L'émulation de réseaux consiste donc à tester et évaluer les performances de véritables applications dans des environnements réseaux virtuels. Pour ce faire, des outils d'émulation et de génération de trafic existent. Par exemple, `GNS3` [2] [86] est un outil permettant aux utilisateurs de créer des réseaux virtuels auxquels peuvent être connectés des machines virtuelles dans lesquelles sont embarquées les applications à tester. Cependant, `GNS3` n'est pas capable de reproduire des réseaux LTE ou 5G ; nous n'avons donc pas pu l'utiliser.

Un outil célèbre pour la génération de trafic est `Iperf` [3] [46], il est conçu pour tester les capacités d'un réseau. Il n'engendre donc que des *«dummy»* paquets qui ne sont en lien avec aucune application réelle. Il était donc exclu de l'utiliser pour obtenir un corpus de données réaliste pour la détection d'anomalies.

Par conséquent, nous avons créé **Autobot** qui se base sur la conteneurisation des applications pour produire les communications.

Contrairement aux solutions de virtualisation dont l'objectif est d'exécuter dans un environnement isolé un système d'exploitation, les conteneurs exécutent uniquement des processus de manière isolée en embarquant les librairies de dépendances et les configurations nécessaires à cette exécution. L'outil de conteneurisation s'appuie directement sur les fonctionnalités du système d'exploitation fournies par la machine hôte. Cette solution est avantageuse par rapport à la virtualisation, notamment au regard de la consommation des ressources CPU et de stockage. De plus, le déploiement des solutions de conteneurisation est rapide et hautement paramétrable.

L'architecture de l'environnement est décrite dans la Figure 3.2. Chaque conteneur agit comme un nœud à l'intérieur du réseau émulé. Chaque conteneur accède au réseau par le biais d'une interface virtuelle. L'environnement d'émulation est hébergé sur un seul hôte, les communications entre les différents conteneurs passent

2. https://www.gns3.com
3. https://iperf.fr/

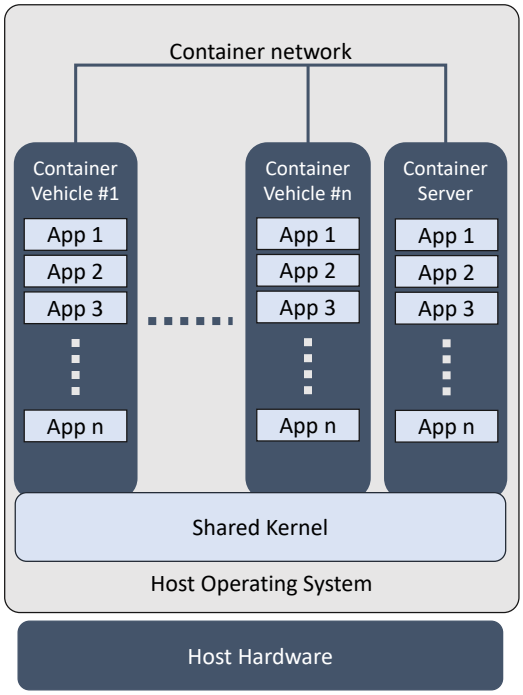


FIGURE 3.2 – Architecture de l'environnement d'émulation **Autobot**.

directement par l'interface de bouclage (*loopback*) de l'hôte. Par conséquent la latence, le débit et le taux de perte des paquets ne sont pas réalistes par rapport à ce qui est observé sur les réseaux cellulaires actuels.

Dès lors, nous définissons des règles de filtrage grâce à `traffic-control` qui nous permettent de manipuler les propriétés du trafic des interfaces de chaque conteneur. Cet outil, ou plus spécifiquement `netem`, permet de définir un comportement personnalisé pour l'interface réseau afin d'en manipuler les propriétés, en particulier le débit, la latence et la perte de paquets.

D'autres outils d'émulation existent. Nous avons choisi `netem` pour ses capacités à être employé sur des interfaces virtuelles et pour sa compatibilité avec Linux. En effet, il fait partie de la suite logicielle `iproute2` disponible par défaut sur le système `Ubuntu 18.04`. De plus, `netem` a fait l'objet de nombreuses études [58, 106, 91, 72] qui ont démontré les capacités de l'outil à émuler de façon correcte, efficace et économique les attributs désirés du réseau par les utilisateurs. Les limitations de l'outil sont principalement liées à la variation du délai des paquets affectant les applications multimédia en continu ainsi que la qualité d'expérience (QoE). Or, ces limitations ne sont pas préjudiciables à nos expérimentations compte-tenu de la faible intensité à laquelle est soumis l'environnement.

Le module `netem` intégré à `traffic-control` permet de manipuler le planificateur de paquets du noyau Linux responsable des tampons de réception et d'émission des interfaces réseau.

En définissant des filtres, les utilisateurs peuvent ainsi manipuler les paquets

suivant différentes politiques de la manière suivante :

— *Shaping* : afin de contrôler la fréquence de transmission des paquets.
— *Scheduling* : afin d'ordonner les paquets.
— *Dropping* : pour simuler la perte de paquets.

En conséquence, nous sommes capables de manipuler les communications des conteneurs de véhicules et serveurs à l'intérieur de l'environnement d'émulation en termes de latence, de bande passante et de perte de paquets.

3.1.2.3 Sélection des attributs du réseau

Afin d'assurer que les communications dans l'environnement d'émulation **Autobot** soient les plus réalistes, nous avons cherché à définir les propriétés des réseaux 4G (Long-Term-Evolution) en nous basant sur des études et une expérience sur le terrain. Cela nous a permis de définir des paramètres pour la bande passante, la latence et la perte de paquets lesquels sont ensuite reportés dans les propriétés des communications réseau d'**Autobot**.

– **Bande Passante** – Nous nous sommes basés sur l'étude publiée par l'autorité de régulation des communications électroniques des postes (ARCEP). Dans sa publication d'octobre 2018 [4], elle a enregistré la bande passante moyenne de quatre opérateurs en suivant la densité de population des zones étudiées. Nous avons utilisé ses résultats pour définir trois profils différents pour l'environnement d'émulation :

— Large zones urbaines (plus de 400 habitants/km^2) : 43 Mbps descendant /12 Mbps montant (Méga-bits par seconde)
— Petites zones urbaines (entre 10 et 400 habitants/km^2) : 32 Mbps descendant /9.8 Mbps montant(Méga-bits par seconde)
— zones rurales (moins de 10 habitants/km^2) : 14 Mbps descendant /4.25 Mbps montant(Méga-bits par seconde)

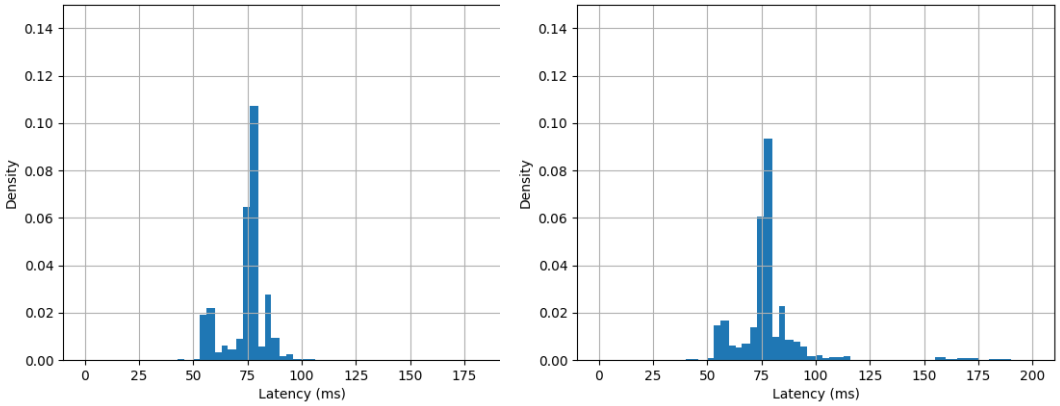
La même étude a été réalisée en Octobre 2019. Elle présente une augmentation du débit moyen de 44% pour les zones urbaines et de 100% dans les zones rurales. Une règle de filtrage `netem` est utilisée afin de fixer une limite à la bande passante à l'intérieur d'**Autobot**.

– **Latence et perte de paquets** – Il n'existe pas à notre connaissance de corpus contenant des mesures détaillées des caractéristiques de latence et de perte de paquets. Aussi, nous avons mis en place une expérimentation visant à définir des profils de distribution de la latence observée dans divers scénarios de mobilité.

Pour ce faire, nous avons utilisé un rasberryPi2-B [5] relié à une antenne de réception de signal 4G équipé d'une carte SIM d'un opérateur français. La latence a été mesurée entre le dispositif et un serveur DNS adressé par `8.8.8.8`. Les scénarios de mobilité ont été définis ainsi :

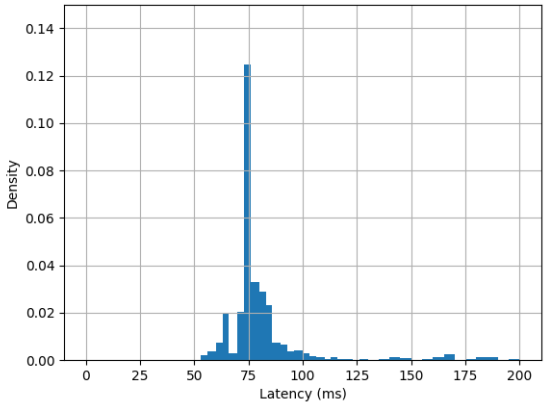
4. https://www.arcep.fr/actualites/les-communiques-de-presse/detail/n/qualite-des-services-mobiles-1.html
5. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

— **Statique** : capture dans les locaux de Continental Digital Services France à proximité (800m) d'une antenne relais 4G (EnodeB).
— **Zone Urbaine** : capture en zone urbaine en mouvement entre les locaux de Continental et du LAAS, elle comporte des zones à vitesse modérée (30km/h) et moyenne (90km/h).
— **Zone Rurale** : capture sur autoroute dans des zones à faible densité de population.



(a) Histogramme du scénario statique en zone urbaine.

(b) Histogramme du scénario mobile en zone urbaine.



(c) Histogramme du scénario mobile en zone rurale.

FIGURE 3.3 – Profils de latence en fonction des scénarios de mobilité représentés sous la forme d'histogrammes.

La Figure 3.3 présente les résultats de cette expérimentation sous la forme d'histogrammes. Il est intéressant de noter les différences entre les 3 profils, notamment la stabilité de la latence entre le scénario statique en zone urbaine et les autres. De plus, le scénario en zone rurale et à grande vitesse montre une légère augmentation

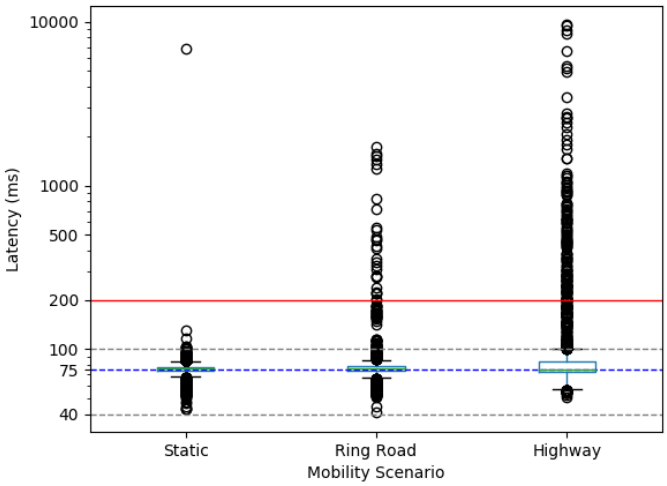


FIGURE 3.4 – Boîtes à moustaches représentant les pics de latence observés à l'intérieur de notre jeu de données.

du nombre de paquets dont la latence est supérieure à 100ms. Ce constat est accentué lorsque l'on observe les boîtes à moustaches présentées dans la Figure 3.4. En effet, on remarque une large différence dans les distributions et surtout dans le nombre de pics de latence représentés par des cercles noirs, notamment ceux supérieurs à 200ms qui représentent :

— Pour le scénario mobile en zone rurale 0.05% des paquets.
— Pour le scénario mobile en zone urbaine 0.02% des paquets.
— Pour le scénario statique en zone urbaine 3.5×10^{-4}% des paquets.

De plus, on remarque que ces pics sont bien plus épars lorsque le scénario de mobilité est en zone rurale.

Enfin, nous avons constaté que chaque scénario présente le même taux de perte de paquets (inférieur à 5×10^{-3}%). Nous avons donc défini un seul ratio de perte pour tous les cas d'utilisation de notre évaluation sous la forme d'une autre règle `netem`. De même, nous avons défini une règle permettant de nous assurer que la latence constatée à l'intérieur d'**Autobot** est similaire à celle que nous avons constaté dans notre évaluation sur le terrain. Nous utilisons `maketable` pour construire une représentation de la distribution de la latence suivant les différents scénarios. `Maketable` est un outil de `netem` permettant cette extraction et ainsi la configuration des règles de filtrage en accord avec le scénario de mobilité.

3.1.3 Évaluation d'Autobot

Nous avons évalué **Autobot** sous deux aspects :

— La consommation des ressources du processeur.
— Les caractéristiques du réseau `docker`.

Notre objectif est de vérifier qu'**Autobot** est capable de maintenir les propriétés des communications que nous désirons émuler. Nous souhaitons donc un maximum de véhicules tout en maintenant :

— Les propriétés du réseau émulé.
— Les propriétés comportementales des applications, c'est à dire assurer à tous les conteneurs les ressources nécessaires à leur bonne exécution.

Cette évaluation a été réalisée sur deux plateformes différentes. La première consiste en une machine virtuelle qui exécute le système d'exploitation `Ubuntu (18.01)` à laquelle sont alloués 6 processeurs sur les 8 de la machine hôte ainsi que 6 giga-octets de mémoire vive (RAM) sur les 16 giga-octets de la machine hôte. La seconde plateforme est un serveur `Ubuntu (16.04)` de 32 processeurs et 64 giga-octets de mémoire vive.

Nous vérifions les propriétés de l'émulation grâce aux communications d'une application ITS de télémétrie.

3.1.3.1 Application de télémétrie ITS

L'application utilise le protocole MQTT (pour *Message Queuing Telemetry Transport*) pour échanger des messages entre le client exécuté sur le véhicule et le serveur situé dans un autre conteneur de l'environnement. MQTT est un protocole de messagerie basé sur le mécanisme de publication-abonnement conçu pour des connexions distantes où les capacités de bande passante ou de calculs du client sont limitées.

Les messages sont au format *Sensoris*[6] pour *Sensor Interface Specification*. Il s'agit d'un format d'échange d'informations entre les capteurs d'un véhicule et un service dédié cloud ainsi qu'intercloud. Ce format est le résultat des travaux du consortium *Sensoris* dont Continental fait partie avec 41 autres acteurs industriels tel que Bosch, TomTom, ou encore IBM. L'objectif de ces travaux coordonnés par ERTICO, une organisation de recherche et standardisation des systèmes de transports intelligents, est de créer un format de message qui puisse être traité par tous les acteurs du consortium.

Dans cette thèse, nous avons utilisé la spécification de la version 2.0.2. Afin d'apporter un maximum de réalisme aux messages échangés, nous avons utilisé un enregistrement du bus CAN d'un véhicule effectuant un trajet de 1h30. Cet enregistrement a été traduit en messages *Sensoris* grâce à un outil développé par les équipes de Continental Digital Services France (`can2sensoris`). Le client MQTT ouvre une session chiffrée avec le serveur et transmet à intervalles réguliers un message contenu dans l'enregistrement vers le serveur. Dans le but de limiter le coût d'envoi sur le réseau, les messages au format JSON sont sérialisés grâce à protobuf[7]. Protobuf est un mécanisme de sérialisation de données structurées. Il permet de produire facilement, à partir d'un fichier de description de la structure des données, le code de lecture et d'écriture permettant de manipuler ces données.

6. https://sensor-is.org/
7. https://developers.google.com/protocol-buffers

Celles-ci sont contenues dans ces messages et représentent donc l'état du véhicule à un instant donné, chaque message ayant en commun une donnée de géolocalisation essentielle aux services ITS.

Chaque session MQTT représente donc l'émulation des communications d'un véhicule dans l'environnement **Autobot**.

3.1.3.2 Configuration du réseau

Le réseau est configuré de la façon suivante pour tous les scénarios de tests :
- Chaque conteneur de véhicule est connecté au même sous-réseau `docker` que le conteneur du serveur (10.0.0.0/8).
- Les interfaces virtuelles des véhicules sont paramétrées en fonction du scénario de mobilité en zone rurale :
  - Bande passante de 14 Mbps descendant et 4.25 Mbps montant.
  - Profil de latence du scénario en zone rurale (Figure 3.3c).
  - Perte de paquets fixée à 0.05%.
- Les communications des véhicules avec le serveur sont générées grâce à l'application de télémétrie ITS MQTT.

3.1.3.3 Caractéristiques du réseau

En premier lieu, nous souhaitons vérifier si les caractéristiques du réseau émulé sont respectées par **Autobot**. Nous utilisons `tcpdump` [8] afin de collecter le trafic sur l'interface virtuelle d'un des conteneurs de véhicule. Nous analysons cette trace afin de vérifier que les communications du véhicule respectent :
- Le délai entre l'envoi de chaque message sur la session MQTT.
- Le profil de latence défini dans le scénario de mobilité en zone rurale (Figure 3.3c).

Cela nous permet de nous assurer que tout au long de l'exécution de l'émulation, chaque conteneur dispose des ressources suffisantes pour envoyer les messages suivant la fréquence prévue. En outre, nous nous assurons également que les règles de filtrage fixées sur les interfaces virtuelles des conteneurs grâce à `netem` sont elles aussi respectées. Le scénario est le suivant :

– **Scénario** – L'environnement est exécuté deux fois pendant 30 minutes. Le nombre de véhicules est fixé à 400 et la fréquence d'envoi des messages *Sensoris* dans l'application ITS est fixée à 1Hz et 2 Hz.

– **Latence** – Nous collectons la latence de chaque paquet de la trace grâce à l'outil d'analyse de protocoles réseaux Tshark [9] via la commande suivante :
- `tshark -Y "tcp.stream eq 0" -r fichier.pcap -Tfields -e "frame.time_epoch" -e "tcp.analysis.ack_rtt"`

8. https ://www.tcpdump.org/manpages/tcpdump.1.html
9. https://www.wireshark.org/docs/man-pages/tshark.html

Cette commande isole le flux MQTT grâce au filtre `"tcp.stream eq 0"` et collecte la latence de chaque paquet du flux MQTT. Nous représentons le profil de latence observé durant le scénario d'exécution de l'émulation de 400 véhicules avec une fréquence d'émission des messages fixée à 2 Hz dans la Figure 3.5. Nous constatons que le profil de communication est correctement respecté, considérant le nombre de véhicules émulés sur la plateforme. Cela signifie que `netem` est en capacité d'assurer les propriétés réseaux que nous désirons pour **Autobot**.

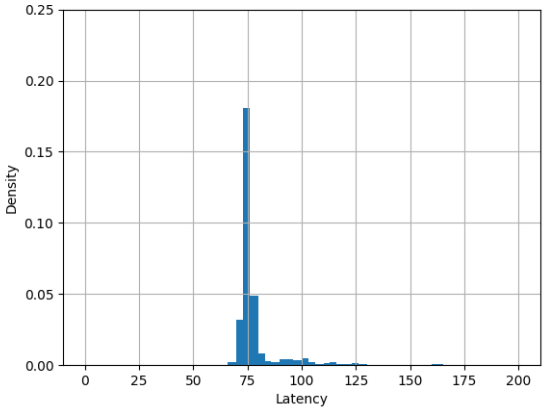


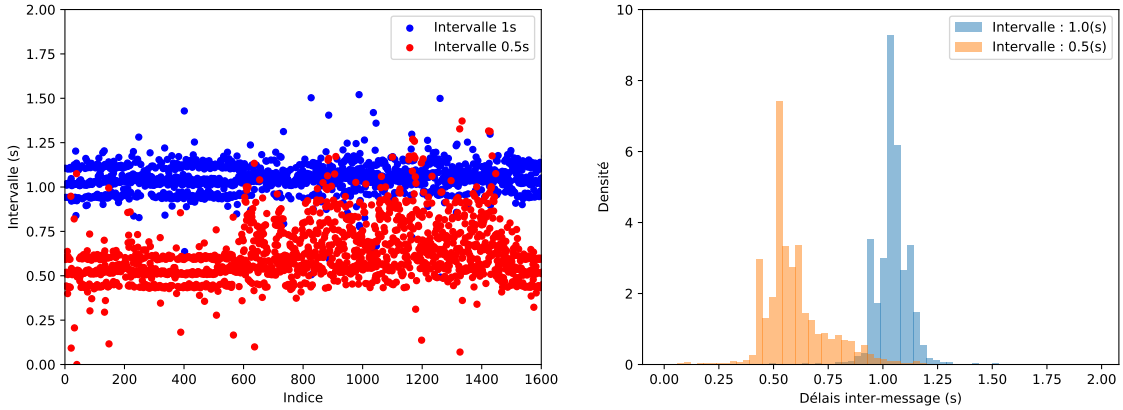
FIGURE 3.5 – Profil de latence observé durant l'exécution de l'émulation de 400 véhicules.

– **Délais inter-messages** – À partir du même scénario, nous extrayons des traces réseaux le délai entre chaque message MQTT envoyé par un seul des 400 véhicules en fonction des deux fréquences différentes (1 et 2 Hz) du scénario. Pour ce faire, nous utilisons la commande suivante :

— `tshark -Y "tcp.stream eq 0 && mqtt.msg" -r fichier.pcap -Tfields -e "frame.time_epoch"`

Cette commande isole le premier paquet des messages MQTT émis par le véhicule grâce au filtre `tcp.stream eq 0 && mqtt.msg`. La date d'envoi du message est extraite grâce au filtre `-Tfields -e "frame.time_epoch"`. Il est important de noter que plusieurs paquets peuvent être envoyés pour un seul message. Toutefois, nous souhaitons vérifier que les conteneurs des véhicules peuvent envoyer les messages à la fréquence qui leur est donnée. C'est pourquoi nous ne conservons que l'instant précis du premier paquet émis par message. La Figure 3.6 présente la distribution des intervalles observés dans les deux situations. On constate que l'intervalle est assez bien respecté pendant les dix premières minutes de l'exécution d'**Autobot** indifféremment de la fréquence de message employée. Nous remarquons toutefois que passé 10 minutes l'intervalle entre chaque envoi de message devient moins précis (Figure 3.6a). Toutefois, les histogrammes représentant la distribution des délais inter-messages illustrés dans la Figure 3.6b montrent que la grande ma-

jorité des messages sont envoyés dans les délais, indépendamment de la fréquence d'envoi.



(a) Intervalles observés par message pour le scénario à 1Hz et 2Hz.

(b) Histogramme de la distribution des intervalles entre messages pour une émission à 1Hz et 2Hz.

FIGURE 3.6 – Représentation des intervalles observés entre les messages pour une émission à 1Hz et 2Hz (resp. 1 seconde et 0.5 seconde d'intervalle).

3.1.3.4 Consommation CPU et mémoire vive

Il nous faut désormais déterminer les limites d'**Autobot** en termes de consommation des ressources CPU et de mémoire de la machine hôte. Pour ce faire, nous évaluons **Autobot** en fonction du nombre de véhicules et de la fréquence d'envoi de messages de l'application de télémétrie. L'évaluation de la consommation est réalisée suivant le scénario suivant :

– **Scénario** – L'environnement est exécuté pendant 15 minutes. Tout au long de l'exécution d'**Autobot** nous mesurons la consommation globale des ressources CPU et de la mémoire de la machine hôte à un intervalle de 2 secondes.

Pour chaque exécution, nous définissons un nombre de véhicules émulés ainsi que la fréquence d'envoi de messages vers le serveur de télémétrie par les applications MQTT des véhicules. Ces paramètres sont reportés dans le Tableau 3.1. Les colonnes correspondent aux valeurs suivantes :

— min et max #veh représentent le nombre minimum et maximum de véhicules dans l'émulation ;
— min f et max f représentent la fréquence minimale et maximale à laquelle chaque véhicule envoie des messages MQTT ;
— vincr et fincr représentent respectivement les incréments du nombre de véhicules et fréquences utilisées pour chaque étape de l'évaluation ;

Installation	min #veh	max #veh	vincr	min f	max f	fincr
Serveur	10	40	5	0.5	2.5	0.5
Serveur	10	40	5	3	8	1
Serveur	40	200	20	3	8	1
Serveur	200	400	50	1	11	0
Machine Virtuelle	10	100	10	1	8	1

TABLE 3.1 – Paramètres utilisés lors de l'évaluation en fonction de la plateforme utilisée, du nombre de véhicules et de la fréquence des messages.

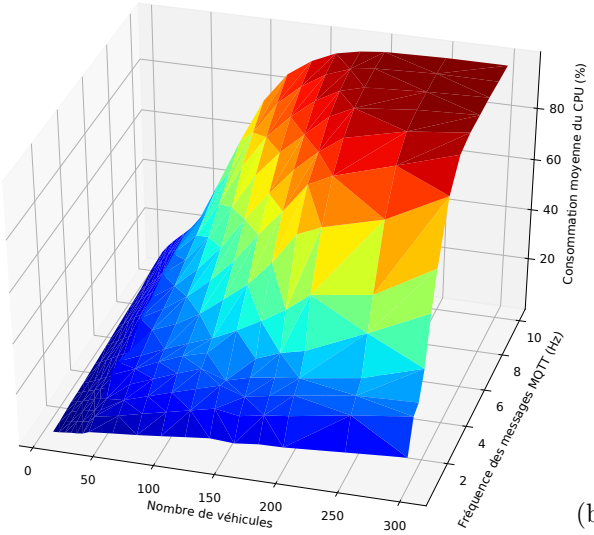
Les résultats de cette évaluation sont rapportés dans la Figure 3.7. Les consommations moyennes en CPU et en mémoire y sont rapportées pour chaque itération d'**Autobot**.

– **Consommation CPU** – Nous constatons que la combinaison de l'augmentation du nombre de véhicules couplée avec l'augmentation de la fréquence d'envoi de messages est responsable de l'augmentation de la consommation des ressources du processeur. Nous notons cependant que la fréquence a un impact plus important dans cette augmentation. En effet, lorsque 300 véhicules communiquent à une fréquence de 4Hz, la consommation atteint 87% des ressources du processeur. Le même seuil de consommation est atteint par 120 véhicules communiquant à une fréquence de 10 Hz, alors que ce même nombre de véhicules avec une fréquence de 4Hz ne consomme pas plus de 20% des ressources CPU. Enfin, il est aussi important de noter qu'au-delà de 150 véhicules la consommation CPU atteint les 100% pour une fréquence de 10Hz.

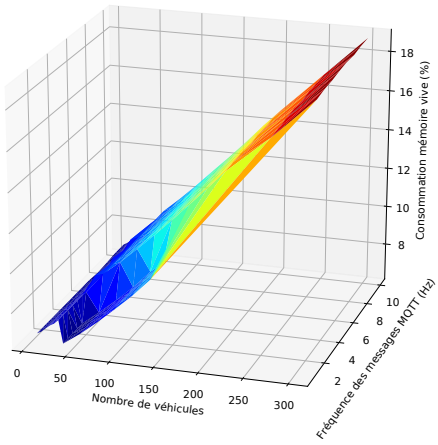
– **Consommation mémoire vive** – L'analyse des résultats de la consommation de mémoire vive révèle que la fréquence d'émission des messages n'a pas d'impact sur la consommation de mémoire vive. Celle-ci tend à croître de façon linéaire avec le nombre de véhicules émulés due au chargement du fichier contenant les messages que les conteneurs des véhicules envoient vers le serveur en mémoire par chaque conteneur. On constate cependant une chute dans la consommation de mémoire vive sur la machine virtuelle au-delà de 60 véhicules émulés. Il s'agit là de la conséquence du déclenchement du mécanisme de *swap* du système d'exploitation Linux lequel décharge une partie de la mémoire vive sur un espace disque réservé lorsque celle-ci est saturée.

3.1.3.5 Bilan de l'évaluation d'Autobot

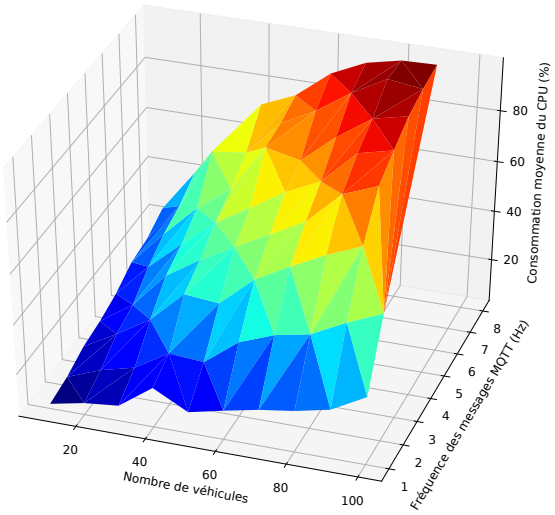
Ainsi, nous avons démontré qu'**Autobot** est capable d'émuler un grand nombre de véhicules tout en maintenant les propriétés des communications du réseau grâce à `netem`. De plus, **Autobot** est capable de fournir les ressources nécessaires aux conteneurs de véhicules pour la bonne exécution de l'application MQTT. Nous



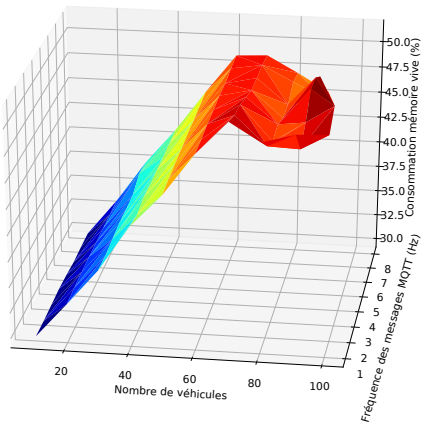
(a) Consommation moyenne de CPU sur le serveur.



(b) Consommation moyenne de mémoire vive sur le serveur.



(c) Consommation moyenne de CPU sur la machine virtuelle.



(d) Consommation moyenne de mémoire vive sur la machine virtuelle.

FIGURE 3.7 – Consommation moyenne des ressources du CPU et de la mémoire vive lors de l'exécution de l'environnement **Autobot** en fonction du nombre de véhicules et de la fréquence d'envoi de messages sur les différentes plateformes.

notons toutefois que la fréquence d'émission des messages a un impact négatif sur la consommation CPU de la machine hôte au-delà de :

— 140 véhicules pour la plateforme du serveur.
— 60 véhicules dans le cas de la machine virtuelle.

Autobot est donc capable de garantir que le trafic engendré par l'environnement d'émulation respecte les propriétés qui lui sont fixées au préalable. Ainsi, nous pouvons l'utiliser pour la création de notre corpus de données pour la détection d'anomalies dans les réseaux cellulaires de véhicules.

3.2 Contenu du corpus

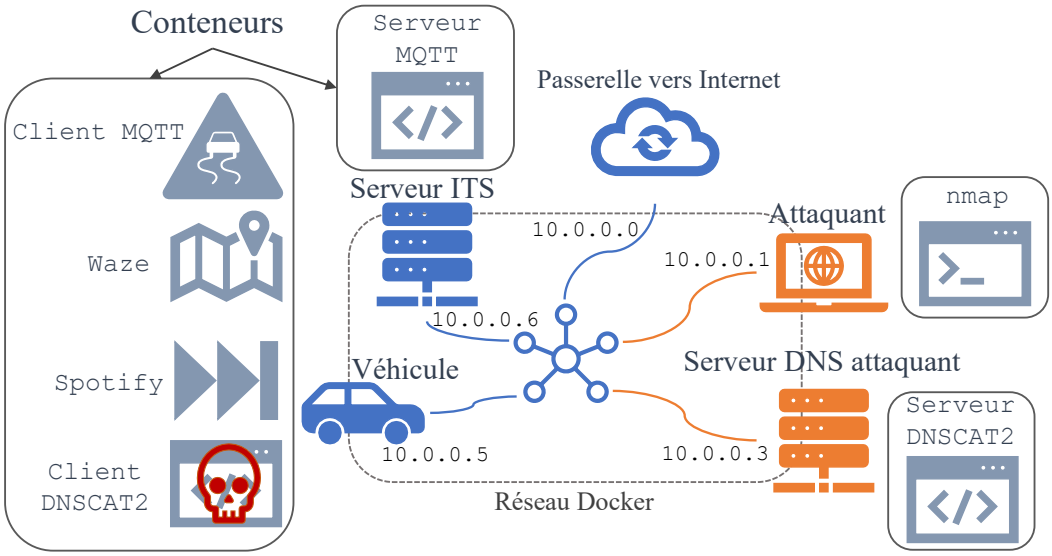


FIGURE 3.8 – Autobot pour la génération du corpus de détection d'anomalies.

Le corpus est généré de manière dynamique à l'intérieur d'un environnement dont les propriétés du réseau sont en adéquation avec celles qu'il est possible d'observer sur les réseaux 4G actuels. Nous avons créé un conteneur dans lequel sont installées les applications chargées de produire les communications dites normales à l'intérieur de l'environnement. Ce conteneur représente ainsi un comportement réseau normal d'un véhicule connecté. Nous avons créé deux autres conteneurs chargés d'engendrer les anomalies et attaques contre le conteneur du véhicule, représentant ainsi les attaquants. La Figure 3.8 représente l'environnement Autobot avec les conteneurs chargés de générer le trafic normal ainsi que les conteneurs destinés à produire le trafic anormal. Dans cette section nous revenons sur le détail du fonctionnement de chacun de ces conteneurs.

3.2.1 Trafic applicatif normal

Nous avons développé deux applications embarquées dans le conteneur du véhicule qui engendrent du trafic normal. La première imite un service de télémétrie véhiculaire et la seconde des communications d'applications d'infotainment.

3.2.1.1 Application ITS

Nous réutilisons la même application de télémétrie basée sur MQTT que celle utilisée durant la phase de vérification du comportement réseau d'**Autobot** (sous-sous-section 3.1.3.1).

3.2.1.2 Applications d'infotainment

Toujours dans le but de produire un corpus le plus réaliste possible, nous avons aussi intégré au conteneur du véhicule deux applications d'infotainment. La première propose un service d'écoute de musique en ligne et la seconde est une application de navigation GPS capable d'informer les utilisateurs de l'état des routes en temps réel. Nous avons décidé d'utiliser des applications populaires et de les intégrer à `docker`. Ainsi, les communications de l'application musicale sont générées grâce au démon spotifyd lancé dans le conteneur. Le démon se connecte aux serveurs de Spotify et lance une playlist entraînant ainsi des communications. L'application de navigation est WAZE, elle engendre des communications en récupérant les informations de l'état des routes à intervalles réguliers.

3.2.1.3 Configuration du réseau

Le réseau est configuré de la même manière que dans l'évaluation d'**Autobot** à savoir :

- Chaque conteneur de véhicule est connecté au même sous-réseau `docker` que le conteneur du serveur (10.0.0.0/8).
- Les interfaces virtuelles des véhicules sont paramétrées en fonction du scénario de mobilité en zone rurale car celui-ci montre une plus grande distribution des latences :
  - Bande passante de 14 Mbps descendant et 4.25 Mbps montant.
  - Profil de latence du scénario en zone rurale (Figure 3.3c).
  - Perte de paquets fixée à 0.05%.
- Les communications des véhicules avec le serveur sont produites grâce à l'application de télémétrie ITS MQTT.
- Les communications des applications d'infotainment avec Internet se font par l'intermédiaire de la machine hôte.

3.2.2 Génération des anomalies et attaques

Nous avons démontré dans la section précédente qu'**Autobot** est capable d'imiter le comportement réseau de véhicules connectés au réseau cellulaire. Nous cher-

chons désormais à générer des attaques réalistes et représentatives de l'état de l'art actuel des attaques sur les véhicules connectés. Nous nous sommes inspirés des scénarios d'attaques évoqués dans notre état de l'art (chapitre 1). Afin d'évaluer les capacités de l'algorithme à détecter différents types d'anomalies, nous avons choisi de modéliser des anomalies aux propriétés temporelles différentes. Nous avons modélisé trois anomalies :

— Un scan réseau.
— Une attaque par tunnel-DNS.
— Une anomalie dans les données de télémétrie.

3.2.2.1 Scan réseau

La première attaque que nous avons émulée, est souvent considérée comme la première étape d'une attaque informatique, à savoir le repérage. En effet, les attaquants doivent avant tout choisir leur cible. À cet effet, des outils populaires comme `Shodan` [10] ou `Censys` [11] peuvent être utilisés pour tenter de découvrir les adresses IP de systèmes particuliers. Dans le contexte du véhicule connecté, ces outils pourraient être utilisés par les attaquants afin de déduire les IP des véhicules connectés leur permettant ainsi d'étudier leurs vulnérabilités. Par exemple, dans le cas d'une des attaques présentées dans l'état de l'art (sous-section 1.3.6), les attaquants ont été capables de déduire les plages d'adresses IP qui étaient affectées aux véhicules [78].

Les scans permettent aussi aux attaquants de découvrir de potentielles vulnérabilités dans les services réseaux exécutés sur les hôtes scannés. En particulier, en extrayant des informations comme les versions des systèmes d'exploitation, ou des applications exécutées sur l'hôte. L'attaquant peut ensuite se documenter sur des sites, comme `ExploitDb` [12], qui répertorient les moyens d'exploiter les vulnérabilités découvertes sur des applications ou des systèmes afin d'en prendre le contrôle ou d'en perturber le fonctionnement.

Les scans ont donc pour objectif de déterminer si les ports d'un hôte sont ouverts, fermés ou filtrés. Pour ce faire, l'attaquant envoie des paquets réseau particuliers en direction des ports de l'hôte. Un port ouvert indique à l'attaquant que le port est prêt à recevoir des communications. Un port fermé indique qu'aucun service de l'hôte n'écoute sur ce port et qu'il est donc impossible d'établir un échange avec celui-ci. Enfin, un port filtré indique qu'un port est peut-être ouvert, mais qu'un mécanisme de pare-feu ou de filtre a entraîné la perte du paquet et qu'aucune réponse n'a été reçue.

Plusieurs méthodes de scan existent : les scans UDP ou TCP. Les scans TCP comprennent le SYN-SCAN, Connect-SCAN, le NULL-SCAN, FIN-SCAN ou XMAS-SCAN. Les scans TCP reposent sur le comportement espéré des hôtes lorsqu'ils reçoivent des paquets particuliers.

10. https://www.shodan.io/
11. https://censys.io/
12. https://www.exploit-db.com/

Par exemple, un SYN-SCAN repose sur l'envoi par l'attaquant de paquets TCP dont le bit SYN, correspondant à l'établissement d'une connexion, est activé. L'hôte recevant ce type de paquet sur un port ouvert répondra avec un paquet avec le bit SYN-ACK [13]. L'attaquant sera ainsi en mesure de déduire que le port sur lequel il a envoyé son paquet est ouvert. Le SYN-SCAN est la méthode la plus souvent employée pour sa rapidité d'exécution.

Le Connect-SCAN est plus lent que le SYN-SCAN puisqu'il repose sur l'établissement complet d'une connexion avec l'hôte, l'attaquant procède au handshake TCP complet avant de fermer la connexion. Si le handshake n'a pas pu être complété, le port est alors considéré comme clos.

Le NULL-SCAN, FIN-SCAN ou XMAS-SCAN reposent sur la manipulation des flags TCP. Par exemple, un scan de type XMAS [14] envoie des paquets TCP où les flags FIN, PSH et URG sont activés et pour lesquels l'hôte est censé répondre par un paquet RST ou ICMP si le port est clos, ou ne pas répondre si celui-ci est ouvert.

Enfin, les scans UDP quant à eux reposent sur l'envoi de paquets UDP vers les ports de l'hôte. Celui-ci répondra par des paquets UDPs si le port est ouvert ou par des paquets ICMP s'il est clos.

De nombreux moyens existent pour détecter plus ou moins efficacement ce genre d'attaques. Par exemple, l'étude de simples seuils comme le ratio du nombre de tentatives de handshake contre le nombre de connexions établies peuvent suffire à détecter des SYN-SCAN. L'analyse de la durée des connexions est requise pour les Connect-SCANs. Les NULL-SCAN, FIN-SCAN, ou XMAS-SCAN peuvent, quant à eux, être détectés en étudiant la proportion de paquets reçus avec certains flags TCP activés.

Cependant, ces méthodes peuvent se retrouver inefficaces face à la diversité des scans et l'intensité variable à laquelle ils peuvent être effectués [19]. Bien que les attaquants aient besoin d'effectuer ces scans pour obtenir les informations dont ils ont besoin afin de réaliser les attaques, le scan reste sans impact direct quant à la sécurité de l'hôte. Nous avons donc décidé de générer deux types de scans afin de démontrer la capacité de l'algorithme à ce type d'attaque.

L'outil le plus populaire permettant d'exécuter ce genre d'analyse est `nmap`. Il permet d'effectuer de nombreux types de scans différents. Nous avons réalisé un XMAS-SCAN ainsi qu'un SYN-SCAN avec `nmap` dirigé vers les 1024 «well-known» ports [15]. Ce scan génère un échange entre l'attaquant et le véhicule sur une période inférieure à 3 secondes aux propriétés suivantes :

— Taille totale des paquets échangés : 0.1Mo ;
— Taille moyenne des paquets échangés : 54o ;
— 960 paquets par seconde en moyenne pendant la durée du scan ;

13. Ce comportement est défini dans la RFC 793 de TCP https://tools.ietf.org/html/rfc793
14. «Light them up like a chrismas tree» https://nmap.org/book/scan-methods-null-fin-xmas-scan.html
15. Il s'agit de ports réservés par l'IANA pour des applications historiques particulières

3.2.2.2 DNS tunelling

Nous considérons le même scénario d'attaque que celui présenté dans la sous-section 2.3.4 concernant le trafic dissimulé d'un logiciel malveillant. Ce scénario représente un réel danger pour les véhicules connectés puisque, sans aucun mécanisme de vérification, il serait alors possible d'installer des applications provenant de marchés secondaires ou de catalogues. Or, comme sur les marchés d'applications mobiles, nous pourrions alors voir apparaître des applications malveillantes. De plus, si un attaquant parvient à installer ce genre d'applications, il serait alors important d'être en mesure de détecter le trafic issu de cette application malveillante.

Notre objectif étant de reproduire les actions qu'un attaquant pourrait prendre lors d'une attaque par tunnel-DNS, nous avons donc utilisé un outil open-source `dnscat2` [16] qui nous permet de réaliser un tunnel-DNS entre le véhicule et un serveur contrôlé par l'attaquant.

Dans notre cas, le serveur de l'attaquant est lui aussi exécuté à l'intérieur de l'environnement **Autobot**. Une fois la communication établie entre les deux entités, nous disposons d'un terminal distant à l'intérieur du véhicule. Grâce à ce terminal nous explorons le contenu des répertoires du véhicule et téléchargeons des fichiers qui s'y trouvent comme pourrait le faire un véritable attaquant.

L'échange entre le véhicule et le serveur est établi pendant une durée de 3 minutes et présente les propriétés suivantes :

— Taille totale des paquets échangés : 0.43Mo ;
— Taille moyenne des paquets échangés : 259.19o ;
— 8.56 paquets par seconde en moyenne pendant la durée de l'attaque ;

3.2.2.3 Anomalie Télémétrique

Nous souhaitons être capables de détecter des dysfonctionnements dans les applications ITS du véhicule connecté qui peuvent résulter d'une défaillance technique de la part des capteurs du véhicule ou du système d'extraction des données captées sur le bus-CAN.

Pour cela, nous avons engendré une anomalie dans les données télémétriques partagées par le véhicule. Pendant quelques instants, les échanges du véhicule avec le serveur de télémétrie sont interrompus de manière à imiter un tel dysfonctionnement. Cette interruption se traduit en pratique par l'arrêt de l'envoi de message de la part du véhicule dans la session MQTT tout en maintenant la session active grâce au mécanisme de *Keep-alive* prévu par le protocole. On observe ainsi une chute du débit du flux MQTT représenté dans la Figure 3.9 avant une reprise normale du trafic. Il est important d'être capable de détecter une telle anomalie afin d'assurer le bon fonctionnement des services ITS qui reposent sur l'obtention des informations des véhicules en circulation sur les routes.

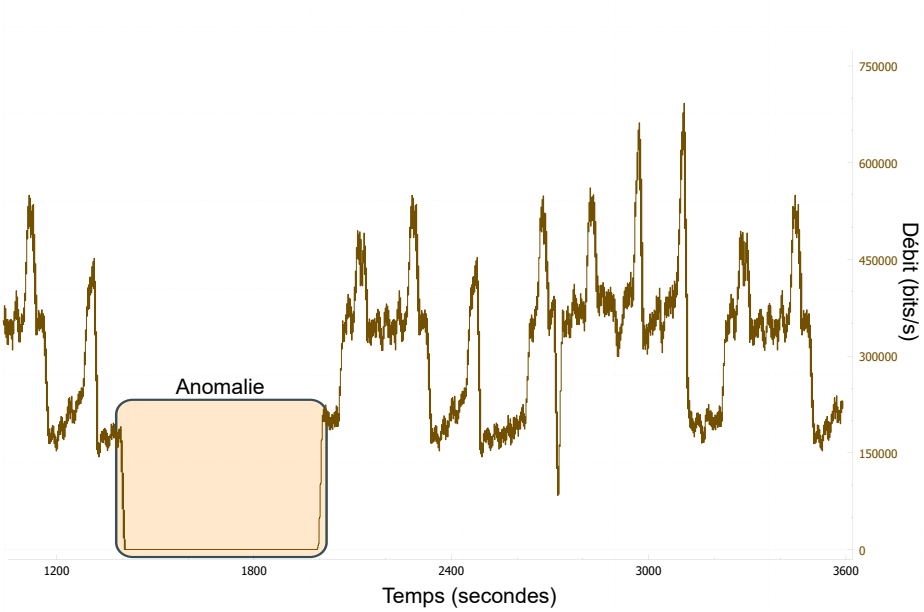


FIGURE 3.9 – Variation du débit sur le flux MQTT lors de l'anomalie de télémétrie.

3.2.3 Propriétés générales du corpus

Notre corpus de communication contient donc le trafic normal d'exécution d'une application de télémétrie, deux applications tierces de musique et de navigation ainsi que trois anomalies. Nous avons capturé ces communications au niveau de l'interface du conteneur d'un véhicule émulé dans l'environnement et en avons extrait les propriétés principales présentées dans le Tableau 3.2. Nous y indiquons les plages d'IP du véhicule et du serveur MQTT, ainsi que celles des différents serveurs avec lesquelles les applications de divertissement communiquent. Nous indiquons également les différentes IP des attaquants. Enfin, nous présentons les principaux protocoles observés dans le corpus en pourcentage du nombre d'octets qu'ils représentent par rapport à la totalité des communications du corpus. On observe ainsi :

— 541190 paquets
— 4h03 de capture
— Une moyenne 0.6 paquets par seconde
— Une taille des paquets moyenne de 1774 octets
— Pour un total de 960Mo de trafic
— un débit moyen de 1100 (octets/s)

La représentation dans l'ontologie du corpus généré par `Autobot` contient **45** *Liens*, **124** *Conversations* et **2278** *Flux*. Parmi chaque type de *fenêtre*, **3** *Liens*, **6** *Conversations* et **2054** *Flux* sont composés de paquets responsables d'anomalies répartis de la façon suivante entre les différents types d'anomalies :

— Scans : 1 *Lien*, 4 *Conversations* (2 par scan) et 2052 *Flux*

16. https://github.com/iagox86/dnscat2

Véhicule	Serveur	Serveurs Infotainment	Attaquants	Principaux protocoles
10.0.0.6	10.0.0.5	104.199.64.136/23 130.211.34.59 130.211.9.172 140.93.5.46 157.240.195.35 172.217.0.0/16 179.60.192.36 216.58.198.67 35.201.119.145 66.102.1.154 74.125.140.155	10.0.0.1 10.0.0.3	mqtt (85.4%) https (13.6%) dns (0.6%) http (0.1%)

TABLE 3.2 – Répartition des IPs et principaux protocoles présents à l'intérieur du corpus

— Tunnel-DNS : 1 *Lien*, 1 *Conversation* et 1 *Flux*
— Anomalie dans les données de télémétrie : 1 *Lien*, 1 *Conversation* et 1 *Flux*

3.3 Vue d'ensemble des paramètres du système

Afin d'évaluer les capacités de détection de **Svalinn**, nous utilisons le corpus de données produit grâce à **Autobot**, mais il nous faut aussi prendre en compte l'ensemble des paramètres du système.

En effet, **Svalinn** repose sur une représentation ontologique des communications existantes entre un véhicule et son environnement. Les attributs des *fenêtres* qui peuplent l'ontologie sont utilisés par un algorithme en ligne de détection d'anomalies (HTM). Des règles d'inférence sont ensuite utilisées pour le traitement de chaque anomalie détectée. Nous récapitulons dans cette section tous les paramètres qui entrent en compte dans l'exécution de ce système. Ces derniers sont illustrés dans la Figure 3.10.

3.3.1 Paramètres de l'algorithme HTM

L'algorithme repose sur un très grand nombre de paramètres (Appendice B), et leur optimisation est longue et fastidieuse. Dans nos travaux, les hyper-paramètres de l'algorithme ont été fixés à partir des résultats des travaux de Ahmad *et al.* [6] dans lesquels ils ont défini les paramètres de HTM grâce à une méthode de recouvrement : l'optimisation par essaims particulaires [40]. Ils ont appliqué cet algorithme à différents corpus de données temporelles. Le détail des paramètres est présenté dans la section B.6.

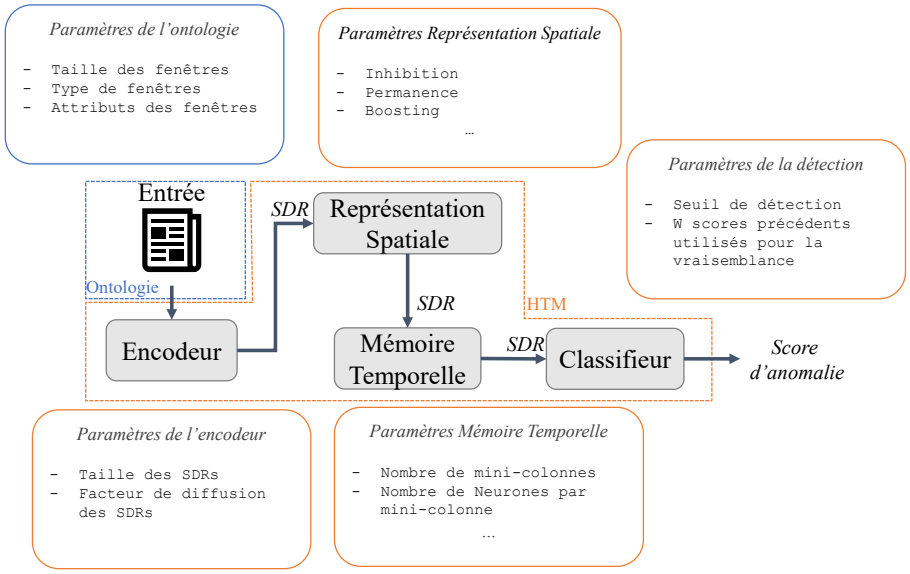


FIGURE 3.10 – Représentation des paramètres de chacun des composants du système de détection d'anomalies.

Les paramètres liés à l'encodage des attributs de l'ontologie en représentations distribuée éparses (SDR) sont : la taille totale de la SDR (n) ainsi que le nombre de bits actifs (w). Ces paramètres influent directement sur la complexité du réseau de neurones car, pour chaque bit de l'espace des entrées, des connexions seront créées avec les neurones des mini-colonnes. Plus ce nombre de connexions est grand, plus l'algorithme devra passer de temps à mettre à jour les persistances de ces connexions à chaque itération. Nous avons donc choisi de fixer ce paramètre au minimum possible défini par la librairie python que nous utilisons de façon à accélérer au maximum le processus de détection. Les dimensions sont : $n = 1080$ et $w = 21$ par attribut encodé. Cette dimension permet tout de même d'encoder $\binom{n}{w}$=8.10×10^{43} SDRs différentes par attribut.

3.3.1.1 Classification

À chaque itération, l'algorithme de la mémoire temporelle produit des prédictions représentant les mini-colonnes les plus probables d'être activées à l'instant suivant. La détection d'anomalies consiste alors à calculer à quel point l'algorithme s'est trompé en fonction des mini-colonnes actives à l'instant t et celles prédites à l'instant $t-1$

Plusieurs mesures d'anormalité ont été imaginées par Ahmad *et al.* [6]. Nous revenons ici sur celles que nous avons utilisées dans nos travaux :

— L'erreur de prédiction
— Le score de vraisemblance

– **Erreur de prédiction** – La première représente une mesure instantanée de l'anormalité d'une entrée x au temps t introduite par la notion d'erreur de prédiction. Celle-ci est calculée en suivant le nombre de bits en commun entre la SDR représentant les mini-colonnes actives à instant t et la SDR des mini-colonnes prédite pour cette entrée à l'instant $(t-1)$. Cette mesure $(S(x_t))$ est calculée comme suit :

$$S(x_t) = 1 - \frac{\pi(x_{t-1}).a(x_t)}{|a(x_t)|} \tag{3.1}$$

Où :

— $a(x_t)$ correspond à la représentation de l'entrée x de l'algorithme à l'instant t,

— $\pi(x_{t-1})$ correspond à la représentation de la prédiction de l'algorithme à l'instant $t-1$ de l'entrée x

Cette mesure représente donc une valeur scalaire inversement proportionnelle au nombre de bits partagés entre les deux variables $a(x_t)$ et $\pi(x_{t-1})$. Ainsi, lorsque les deux valeurs sont identiques, le score sera de 0. Il sera de 1 si elles sont complètement différentes.

– **Score de vraisemblance** – Cependant, cette mesure instantanée de l'anormalité se révèle être très sensible au bruit [6]. Une autre mesure basée sur une modélisation de la distribution des erreurs de prédiction a donc été proposée afin de déterminer la vraisemblance de l'anomalie en se basant sur la moyenne et la variance des dernières W valeurs calculées ainsi :

$$\mu_t = \frac{\sum_{i=0}^{i=W-1} S_{t-i}}{W} \tag{3.2}$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} (S_{t-i} - \mu_t)^2}{W-1} \tag{3.3}$$

Où,

— μ_t représente l'erreur moyenne pour les W précédentes prédictions,

— σ_t^2 représente la variance des W précédentes prédictions.

Enfin, la vraisemblance est définie comme le complément de la fonction-Q Gaussienne [60] et représentée comme suit :

$$L(x_t) = 1 - Q(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t}) \tag{3.4}$$

Où,

— $\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1} S_{t-i}}{W'}$,

— et $W' << W$ représente une fenêtre glissante de la moyenne des erreurs de prédiction.

La vraisemblance d'une anomalie $(L(x_t))$ est enfin définie suivant un seuil ε tel que :

$$L(x_t) \geq 1 - \varepsilon \tag{3.5}$$

3.3.2 Paramètres de l'ontologie

Tout d'abord, les attributs du trafic manipulés par l'algorithme (Appendice A) sont les paramètres les plus importants du système.

Utiliser le maximum de paramètres possible pour l'apprentissage n'est pas forcément synonyme d'efficacité. En effet, la gestion d'un trop grand nombre de paramètres peut ralentir considérablement l'algorithme utilisé. Ensuite, le fléau de la dimensionnalité peut entraîner une réduction des performances lorsqu'un trop grand nombre de variables est considéré par l'algorithme. De plus, le mauvais dimensionnement des attributs peut entraîner le surapprentissage (ou *overfitting*[53]) de l'algorithme. Par conséquent, nous cherchons à limiter le nombre des attributs que nous utilisons lors de la phase de détection.

Enfin, la durée et le type des *fenêtres de description* constituent aussi des paramètres importants de l'ontologie. En effet, la durée des *fenêtres* a un impact direct sur le délai introduit entre l'apparition de l'anomalie et sa détection. Par exemple, si une anomalie est contenue dans les premiers paquets d'une *fenêtre* d'une durée de 15 secondes, il est nécessaire d'attendre la fin de la *fenêtre* avant de pouvoir procéder à la détection. Le type de *fenêtre* utilisé influe directement sur les attributs des *fenêtres* utilisés pour la détection, il est donc nécessaire d'en évaluer l'impact.

3.4 Évaluation de la détection

Afin d'évaluer Svalinn, nous avons procédé de manière itérative pour déterminer parmi un ensemble de paramètres, ceux produisant les meilleurs résultats de détection tout en minimisant autant que possible les fausses alertes.

L'avantage de *Svalinn* par rapport aux autres systèmes de détection d'intrusion réside dans le fait que l'ontologie lie entre eux les paquets des échanges du véhicule. Ces liens sont ensuite utilisés par les règles d'inférence afin de regrouper au sein d'un rapport d'anomalie les paquets responsables de l'anomalie.

À ce titre, il n'est donc pas nécessaire que l'algorithme détecte chacune des *fenêtres* impliquées dans une anomalie puisqu'une seule suffit à classifier le *lien*, la *conversation* ou le *flux* comme anormal. C'est pourquoi nous nous intéressons en particulier à évaluer les capacités de l'algorithme à détecter au plus tôt les anomalies. De cette façon les règles d'inférence peuvent regrouper le plus grand ensemble de paquets constitutifs de l'anomalie.

Notre évaluation peut donc être résumée ainsi :

1. Isoler les meilleurs attributs de l'ontologie pour la détection en fonction du type et de la durée des *fenêtres de description*.
2. À partir de ces ensembles d'attributs, définir le nombre optimal d'attributs à utiliser en fonction des capacités de détection que l'algorithme obtient.
3. En déduire la couverture maximale de Svalinn des anomalies présentes dans le corpus.

3.4.1 Sélection des attributs de l'ontologie

Dans le but d'obtenir les meilleurs résultats, tout en minimisant le nombre d'attributs, nous avons cherché à sélectionner le meilleur sous-ensemble d'attributs parmi ceux présents dans l'ontologie.

De nombreuses méthodes de sélection des attributs existent [26]. De manière générale, elles se séparent en deux catégories : les méthodes de filtre (ou *Filter Methods*) et les méthodes de recouvrement (ou *Wrapper Methods*).

Les méthodes de filtre classent les attributs des variables en fonction d'un score. Ce score peut représenter le degré de corrélation entre l'attribut et la classe ou encore l'information mutuelle qui mesure la dépendance entre l'attribut et la classe de la variable.

Les méthodes de recouvrement, elles, sélectionnent un sous-ensemble des attributs disponibles et évaluent les résultats de l'algorithme en fonction de ce sous-ensemble. L'opération est répétée jusqu'à la découverte d'un sous-ensemble obtenant les meilleurs résultats. Plusieurs méthodes, séquentielles ou heuristiques, existent pour construire ce sous-ensemble.

Nous étudions ici plusieurs méthodes de sélection d'attributs en utilisant des méthodes de filtre. En effet, celles-ci sont souvent plus rapides que les méthodes de recouvrement qui requièrent l'exécution répétée de l'algorithme sur un sous-ensemble du corpus de données.

Nous comparons donc 4 méthodes : la F-Valeur, le test $\mathcal{X}^2$, les Arbres extrêmement randomisés [47] et le score de superposition.

– Le test $\mathcal{X}^2$ – Le test $\mathcal{X}^2$ est utilisé en statistiques pour tester l'indépendance de deux variables. Dans notre cas, nous cherchons à déterminer si un attribut influe grandement sur la classe de trafic de la *fenêtre* qu'il représente (normal, ou anormal). Le test consiste à déterminer si l'attribut et la classe de trafic ne sont pas indépendants.

Pour ce faire, le test construit un score à partir de la distribution des valeurs de l'attribut et des probabilités qu'ont chaque valeur d'appartenir à l'une des deux classes. Ce score est exprimé de la façon suivante :

$$\mathcal{X}^2 = \sum_{i=0}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.6}$$

où :

- O_i représente le nombre de fois que la classe (normale ou anormale) a été observée pour la valeur i de l'attribut.
- E_i représente la fréquence à laquelle cette même valeur i est affectée à la même classe.

Plus ce score est petit, plus la classe est dépendante de cet attribut et donc jugé intéressant pour la détection.

– **La F-valeur** – La F-valeur est calculée de manière similaire au $\mathcal{X}^2$. Elle se base sur la moyenne d'un attribut en fonction de la classe à laquelle une variable appartient. Dans notre cas, la f-valeur étudie la différence entre la moyenne d'un attribut pour toutes les *fenêtres* «anormales» et la moyenne du même attribut pour les *fenêtres* «normales». Plus ces moyennes sont proches, moins l'attribut est jugé intéressant pour la détection.

– **Arbres extrêmement randomisés** – Les arbres extrêmement randomisés (ou *Extra-Tree* pour ***ext****remely* ***ra****ndomized* ***tree****s*) constituent une méthode semblable à celle des forêts aléatoires. Les forêts aléatoires construisent les arbres à partir d'échantillons de l'ensemble des attributs disponibles pour une variable puis ajoutent et retirent des attributs de façon aléatoire (bagging). Les Extra-trees sélectionnent le sous-ensemble d'attributs de façon complètement aléatoire. Les attributs sélectionnés sont ensuite évalués en fonction du coefficient de Gini. Celui-ci mesure la probabilité que les différentes valeurs d'un attribut appartiennent toutes à une seule classe ou au contraire soient distribuées de façon aléatoire entre les différentes classes. Ainsi, on préfèrera sélectionner les attributs pour lesquels chaque valeur n'appartient qu'à une seule classe.

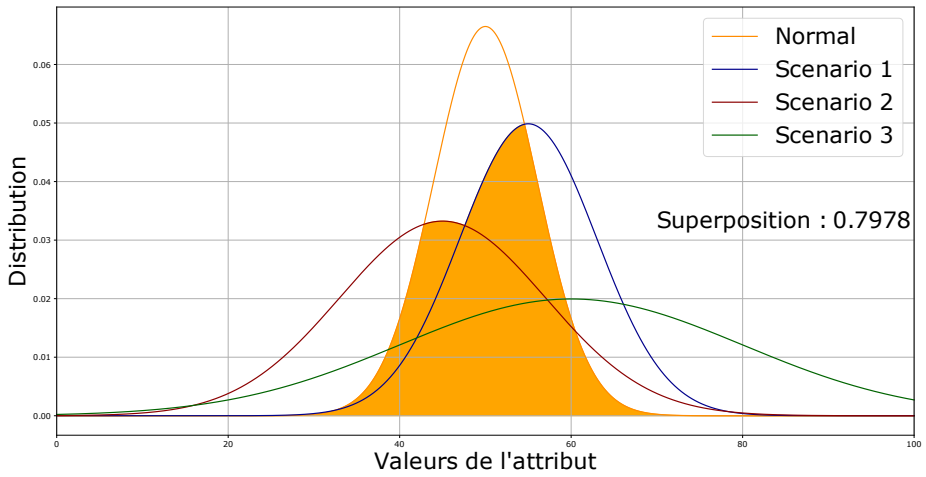
– **Score de Superposition** – Le score de superposition est une mesure que nous avons conçue. Elle est basée sur l'étude des distributions de l'ensemble des attributs des *fenêtres* de l'ontologie. Nous cherchons à déterminer ceux qui présentent la plus grande dissimilarité en cas d'anomalie par rapport à la situation normale. Nous détaillons dans la sous-sous-section 3.4.1.1 le fonctionnement de cette méthode.

3.4.1.1 Score de superposition

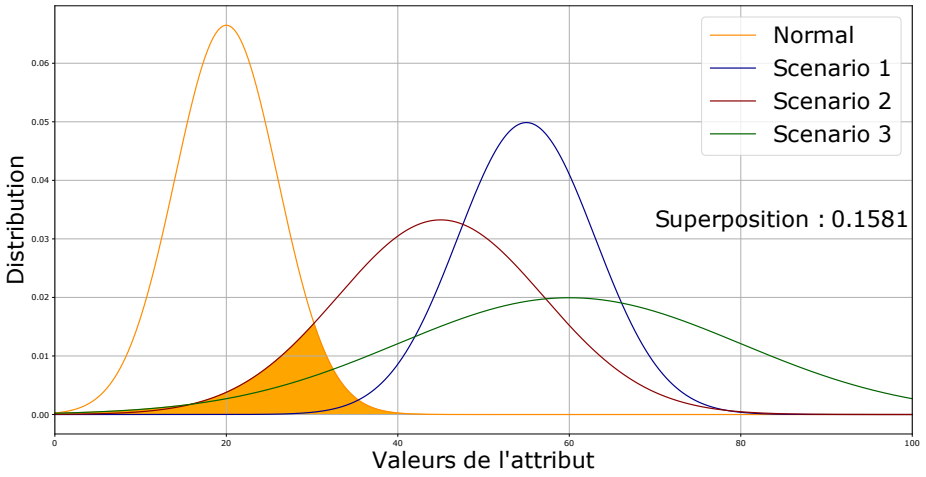
Le score de superposition est basé sur le calcul de l'intégrale résultant de l'union des distributions d'un même attribut. Nous illustrons cette notion dans la Figure 3.11. Pour cet exemple nous étudions deux attributs différents pour lesquels les distributions des valeurs suivent une distribution de loi gaussienne aux paramètres différents suivant les situations.

En pratique, les distributions que nous observons sont de taille variable. En effet la situation normale est présente majoritairement à l'intérieur de notre corpus, et donc de l'ontologie. De plus, les différents scénarios anormaux produisent un nombre différent de *fenêtres*. Ainsi, nous procédons à une phase d'estimation statistique des distributions afin de manipuler des représentations de taille identique pour chacun des attributs.

– **Construction des histogrammes des caractéristiques globales** – Nous définissons l'histogramme des caractéristiques globales $\mathcal{H}$ à n classes dans lequel nous répartissons l'ensemble E_s des valeurs d'un attribut pour un scénario donné $s \in S$ dans des intervalles de Δ valeurs. À cet effet, nous utilisons la fonction $R(.)$



(a) Attribut avec un fort taux de superposition.



(b) Attribut avec un taux de superposition faible.

FIGURE 3.11 – Illustration du score de superpositions pour deux attributs différents

indicatrice de l'appartenance d'un échantillon x à l'intervalle $i \in n$, telle que :

$$R(x,i) = \begin{cases} 1, \text{ si } \Delta \times i < x <= \Delta \times (i+1) \\ 0, \text{ sinon} \end{cases} \tag{3.7}$$

Où Δ représente la largeur des intervalles des classes de l'histogramme :

$$\Delta = \frac{max(\mathcal{E}) - min(\mathcal{E})}{n} \tag{3.8}$$

avec : $\mathcal{E}$ l'ensemble des valeurs de l'attribut tout scénario confondu, $\mathcal{E} = \bigcup_{s}^{S} E_s$.

Puisque nous désirons comparer les histogrammes entre eux, nous conservons la proportion d'échantillons de E_s appartenant à un intervalle donné. Ainsi, l'histo-

gramme $\mathcal{H}$ est défini tel que pour chaque classe $i \in n$ et chaque échantillon $j \in m$ de E_s :

$$\mathcal{H}(i) = \frac{1}{m} \sum_{j=0}^{m} R(x_j, i) \tag{3.9}$$

Avec donc,

$$\sum_{i=0}^{n} \mathcal{H}(i) = 1 \tag{3.10}$$

– **Construction du score de superposition** – La distribution des échantillons des attributs pour chaque scénario est représentée dans des histogrammes de même dimension et sur les mêmes intervalles (Équation 3.4.1.1). Nous pouvons désormais estimer le score de superposition de ces distributions comme illustré dans la Figure 3.11. Pour ce faire, nous cherchons à estimer l'aire sous les courbes des différents scénarios en vue de les comparer.

Nous avons donc, $\mathcal{H}_n$ l'histogramme défini entre a et b d'un attribut dans le scénario normal, et $\mathcal{H}_a$ représentant les valeurs maximales des histogrammes du même attribut dans les scénarios anormaux. Le score $\mathcal{S}$ de superposition est défini tel que :

$$\mathcal{S} = 1 - \frac{\int_a^b min(\mathcal{H}_n(x), \mathcal{H}_a(x))dx}{\int_a^b \mathcal{H}_n(x)dx} \tag{3.11}$$

Puisque nous calculons l'intégrale de fonctions non continues nous utilisons la méthode des trapèzes successifs (Équation 3.4.1.1) pour effectuer le calcul :

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta \tag{3.12}$$

Ainsi, le score est compris entre 0 et 1 ; il est interprété de la façon suivante :

— Un score proche de 0 indique que les distributions des valeurs d'un attribut sont similaires dans le cas normal et anormal.

— Inversement un score proche de 1 indique que les distributions sont différentes dans les scénarios normaux et anormaux.

3.4.2 Métriques d'évaluation

La classification des événements détectés par notre système est binaire. Le détecteur est chargé de classifier une *fenêtre* comme anormale ou normale en lui attribuant un score d'anomalie. Il est courant d'utiliser la matrice de confusion définie par le Tableau 3.3 afin d'évaluer les capacités de détection d'un algorithme.

En conséquence, pour évaluer la qualité du détecteur, nous comparons les classes réelles et les classes que l'algorithme affecte à chacune des *fenêtres* analysées. Nous considérons donc :

— Un vrai positif (noté VP) comme étant une *fenêtre* anormale classifiée comme une anomalie par l'algorithme ;

		Classe réelle	
		Anormale	Normale
Classification de l'algorithme	Anormale	VP	FP
	Normale	FN	VN

TABLE 3.3 – Matrice de confusion pour la classification des *fenêtres*.

— Un faux positif (noté FP) comme étant une *fenêtre* normale classifiée comme une anomalie par l'algorithme ;
— Un faux négatif (noté FN) comme étant une *fenêtre* anormale classifiée comme normale par l'algorithme ;
— un vrai négatif (noté VN) comme étant une *fenêtre* normale classifiée comme normale par l'algorithme ;

Ces indicateurs permettent ainsi d'évaluer les capacités du détecteur en fonction de plusieurs mesures. Les plus couramment utilisées comme la précision, le rappel, la courbe d'opération et l'indice de Sørensen-Dice, sont présentées et discutées ci-après.

3.4.2.1 Métriques usuelles

– **Exactitude** – L'exactitude, notée ACC, estime la dextérité avec laquelle l'algorithme effectue la classification indifféremment de la classe associée. Ainsi, elle ne différencie pas le nombre de classifications correctes en fonction du cas normal ou anormal. Elle est représentée par la formule suivante (Équation 3.4.2.1) :

$$ACC = \frac{VP + VN}{VP + FP + FN + VN} = \frac{\textit{Nombre total de classifications correctes}}{\textit{Nombre total d'éléments}} \tag{3.13}$$

Deux autres mesures permettent d'évaluer séparément la capacité de l'algorithme à prédire la bonne classe.

– **Sensibilité** – La sensibilité, ou rappel, correspond au taux de vrais positifs noté TVP. Elle représente la capacité du détecteur à lever une anomalie pour un événement réellement anormal. Elle prend en compte le nombre d'éléments correctement classifiés comme anormaux (VP) par rapport au nombre d'anomalies n'ayant pas été considérées comme anormales par la méthode de détection (FN). Ce taux est exprimé par la formule suivante (Équation 3.4.2.1) :

$$TVP = \frac{VP}{VP + FN} = \frac{\textit{Vrais positifs}}{\textit{Nombre d'attaques réelles}} \tag{3.14}$$

– **Taux de faux positifs** – Le taux de faux positifs, noté TFP, représente quant à lui la probabilité qu'un détecteur lève une alerte alors que l'événement observé ne constitue pas une anomalie. Il est donc construit par le nombre d'éléments incorrectement classifiés comme anormaux, noté FP, par rapport au nombre d'éléments bénins, noté VN, considérés comme normaux par le détecteur. Ce taux est exprimé par la formule suivante (Équation 3.4.2.1) :

$$TFP = \frac{FP}{FP + VN} = \frac{FP}{\textit{Nombre d'événements normaux}} \tag{3.15}$$

– **Précision** – La précision, ou valeur prédictive positive, noté VPP, correspond à la probabilité qu'une anomalie détectée corresponde véritablement à une anomalie. Pour cela, et contrairement à la sensibilité, la précision prend en compte les événements normaux considérés comme anormaux (FP). Dans le contexte de la détection d'anomalie, elle évalue donc l'efficacité globale de l'algorithme. La précision est exprimée par la formule suivante (Équation 3.4.2.1) :

$$VPP = \frac{VP}{VP + FP} = \frac{VP}{\textit{Nombre anomalies détectées}} \tag{3.16}$$

– **Indice de Sørensen-Dice** – L'indice de Sørensen-Dice plus couramment appelé *f1-score* est une mesure prenant en compte à la fois la précision et la sensibilité pour ne fournir qu'un seul indicateur. Elle est exprimée en fonction d'un facteur $\mathcal{B}$ (Équation 3.4.2.1) :

$$\text{f1-score} = \frac{(\mathcal{B}+1) \times VPP \times TVP}{\mathcal{B} \times VPP + TVP} \tag{3.17}$$

Lorsque $\mathcal{B} > 1$, la précision est favorisée dans l'évaluation, dans le cas contraire c'est la sensibilité qui est favorisée, elle ne prend cependant pas en compte les vrais négatifs (VN) dans l'évaluation.

– **Courbes ROC** – La méthode classique de l'évaluation des détecteurs d'anomalies est basée sur l'analyse des courbes ROC (receiver operating characteristic). Celles-ci évaluent le détecteur en fonction de l'aire sous la courbe du taux de vrais positifs (sensibilité) en abscisse et de faux positifs en ordonnée.

Chaque point de la courbe est déterminé en faisant varier un paramètre utilisé dans la classification qui représente habituellement un seuil de détection. Le détecteur est jugé en fonction de l'aire de la courbe ROC ; plus celle-ci est élevée, plus la détection est considérée comme bonne.

L'inconvénient de cette mesure, mais aussi de l'indice de Sørensen-Dice, est qu'ils souffrent de l'oubli de la fréquence de base [102, 107]. En effet, dans notre cadre

d'application, les anomalies sont très inférieures en nombre par rapport au nombre d'événements bénins du trafic. Dès lors nous avons :

$$\textit{Nombre d'attaques} << \textit{Nombre d'événement normaux} \tag{3.18}$$

En conséquence, un détecteur générant plus de faux positifs, mais aussi détectant plus d'anomalies serait considéré meilleur qu'un détecteur ne générant pas de faux positifs, mais détectant moins d'anomalies.

Pour ces raisons, nous utilisons d'autres métriques pour l'évaluation de la détection : les courbes d'opération ainsi que le coefficient de corrélation de Matthews (MCC).

3.4.2.2 Courbes d'opération et coefficient de corrélation de Matthews

– **Taux d'anomalies** – La modélisation de l'ontologie et la durée des *fenêtres* influent directement sur la pondération des anomalies à l'intérieur du corpus d'entraînement. En effet, suivant que l'on se concentre sur les *Flux*, *Conversations* ou *Liens*, mais aussi en fonction de la durée des *fenêtres*, le ratio des éléments anormaux contenus dans le corpus ne sera pas le même. Nous représentons par la Figure 3.12 cette différence en fonction du type et de la durée des *fenêtres*.

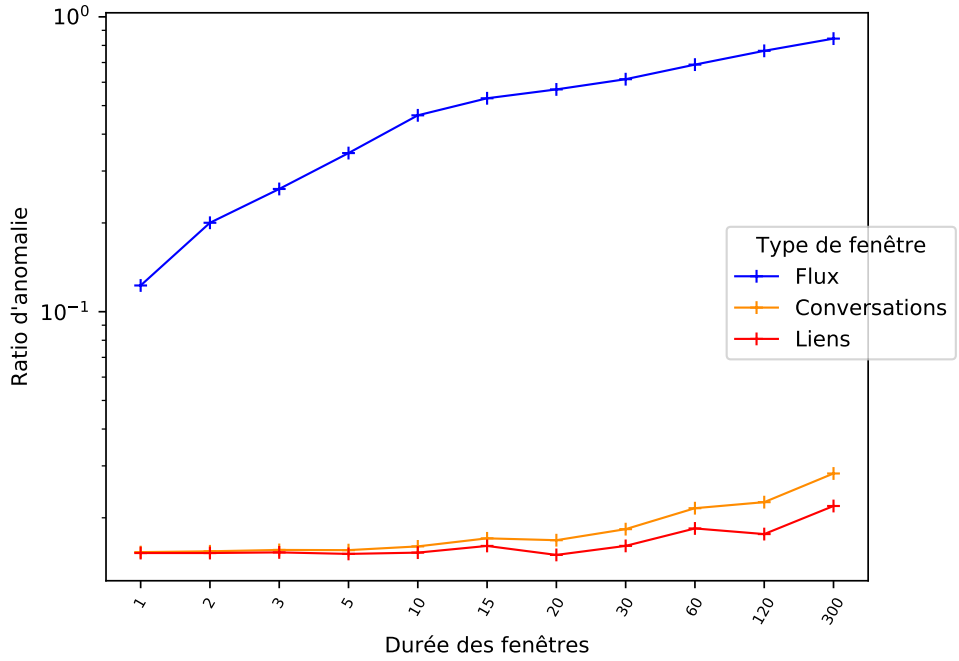


FIGURE 3.12 – Facteur d'anomalies à l'intérieur du corpus d'entraînement en fonction de la durée et du type de *fenêtre*.

Nous constatons donc que le ratio de *fenêtres de Flux* anormales est très diffé-

rent des autres types de *fenêtres*. Cette différence est la conséquence des scans qui vont engendrer la création d'un très grand nombre de *fenêtres de Flux* différentes puisque l'attaquant envoie des paquets à de nombreux ports différents. Le nombre de *Conversations* et de *Liens* n'est pas affecté par cette attaque puisque les ports du véhicule ne sont utilisés ni dans la discrimination des différentes *Conversations*, ni dans celle des *Liens*. Il est donc important de prendre en compte cette pondération dans l'évaluation du détecteur. Le Tableau C.1 présenté dans l'Appendice C détaille le nombre de *fenêtres* créées par chacune des anomalies et illustre ainsi cette différence de pondération entre le trafic normal et les anomalies. On constate également un très grand écart entre le nombre de *fenêtres* de chacune des anomalies en particulier pour les *fenêtres de Flux*. Il nous faudra donc également prendre en compte cet écart dans l'évaluation de notre dispositif de détection.

– **Courbes d'opération** – Pour cela les courbes d'opération se basent sur la précision (VPP) que nous avons définie plus haut. Cependant, la VPP ne prend pas en compte la fréquence des anomalies par rapport au trafic normal. Signalons que Nasr *et al.* [85] exprime la précision en prenant en compte cette fréquence de la manière suivante :

$$VPP_{eid} = \frac{pi \times TVP}{pi \times TVP + (1 - pi) \times TFP} \tag{3.19}$$

Dès lors, la précision n'est plus évaluée simplement selon le TVP et le TFP, mais prend en compte la différence de pondération entre le trafic normal et les anomalies dans le corpus. Cette pondération pi représente simplement le nombre d'anomalies (NA) par rapport à la taille totale du corpus (N) tel que $pi = \frac{NA}{N}$. Le *score d'efficacité* de la détection est alors calculé en fonction de l'aire entre la courbe d'opération optimale (ZRC) et la courbe d'opération du détecteur évalué comme représentée par la zone orange dans la Figure 3.13.

– **Coefficient de corrélation de Matthews** – Le coefficient de corrélation de Matthews [74] est une autre mesure capable de prendre en compte la différence de pondération dans le jeu de données d'apprentissage ainsi que tous les autres ratios de la matrice de confusion (vrais et faux positifs, ainsi que vrais et faux négatifs) [28]. Cette mesure (notée MCC) est exprimée tel que (Équation 3.4.2.2) :

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}} \tag{3.20}$$

Le résultat de cette mesure, compris entre -1 dans le pire des cas et $+1$ pour la meilleure classification, tend ainsi à récompenser un détecteur capable de correctement classifier à la fois les éléments normaux et anormaux.

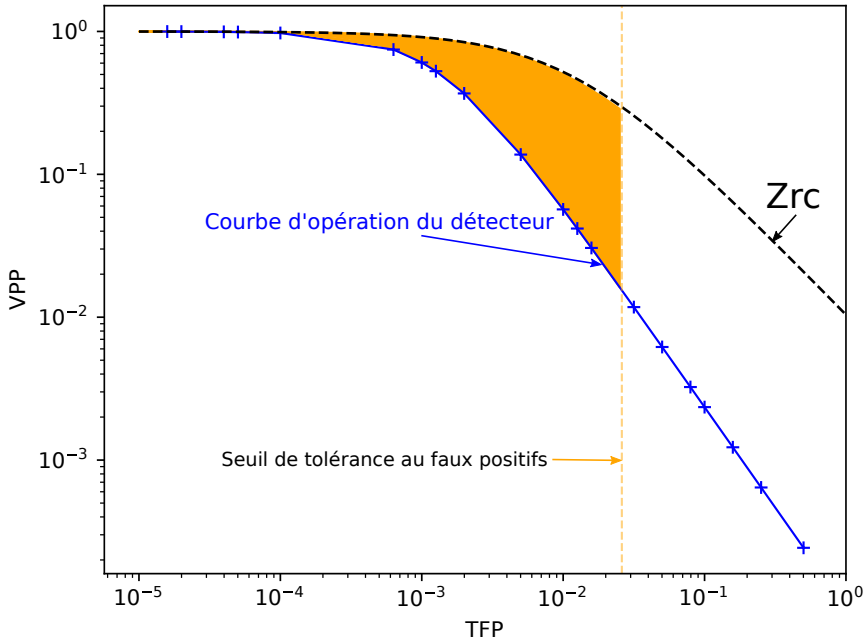


FIGURE 3.13 – Représentation du *score d'efficacité* (en orange).

3.4.3 Couverture des anomalies

Nous présentions dans la Figure 3.12 le taux d'anomalies à l'intérieur du jeu de données d'entraînement selon la durée et du type de *fenêtre*. Nous soulignions l'importance de la prise en compte de cette pondération dans l'évaluation du détecteur, mais nous cherchons également à déterminer si notre détecteur est capable de détecter des anomalies différentes. Or, l'écart entre le nombre de *fenêtres* de chacune des anomalies (Tableau C.1) peut directement impacter la qualité de la détection en récompensant les détecteurs si nous nous intéressons uniquement à la classification binaire (Anormal/Normal). Pour ce faire, nous définissons donc le *taux de couverture* des anomalies. Ce dernier consiste à évaluer les capacités de l'algorithme à détecter au plus tôt les différentes anomalies de façon à ce que les règles d'inférence puissent regrouper le plus grand ensemble de paquets constitutifs de ces anomalies. Nous cherchons ainsi à définir les paramètres pour lesquels le dispositif est capable de détecter toutes les anomalies avec la plus grande couverture possible.

Par exemple, la figure Figure 3.14 représente une anomalie dont la durée s'étend sur 10 fenêtres de descriptions. Les flèches indiquent les fenêtres qui ont déclenchées une détection par l'algorithme HTM. Dans cet exemple, on constate que l'anomalie est détectée à partir de la 5ème fenêtre, ce qui correspond à un taux de couverture de l'anomalie de 60%.

– **Conclusion** – Le coefficient de corrélation de Matthews est capable de prendre en compte à la fois la différence de pondération entre le nombre de cas normaux et de cas anormaux présents dans le corpus, mais aussi la capacité du détecteur à

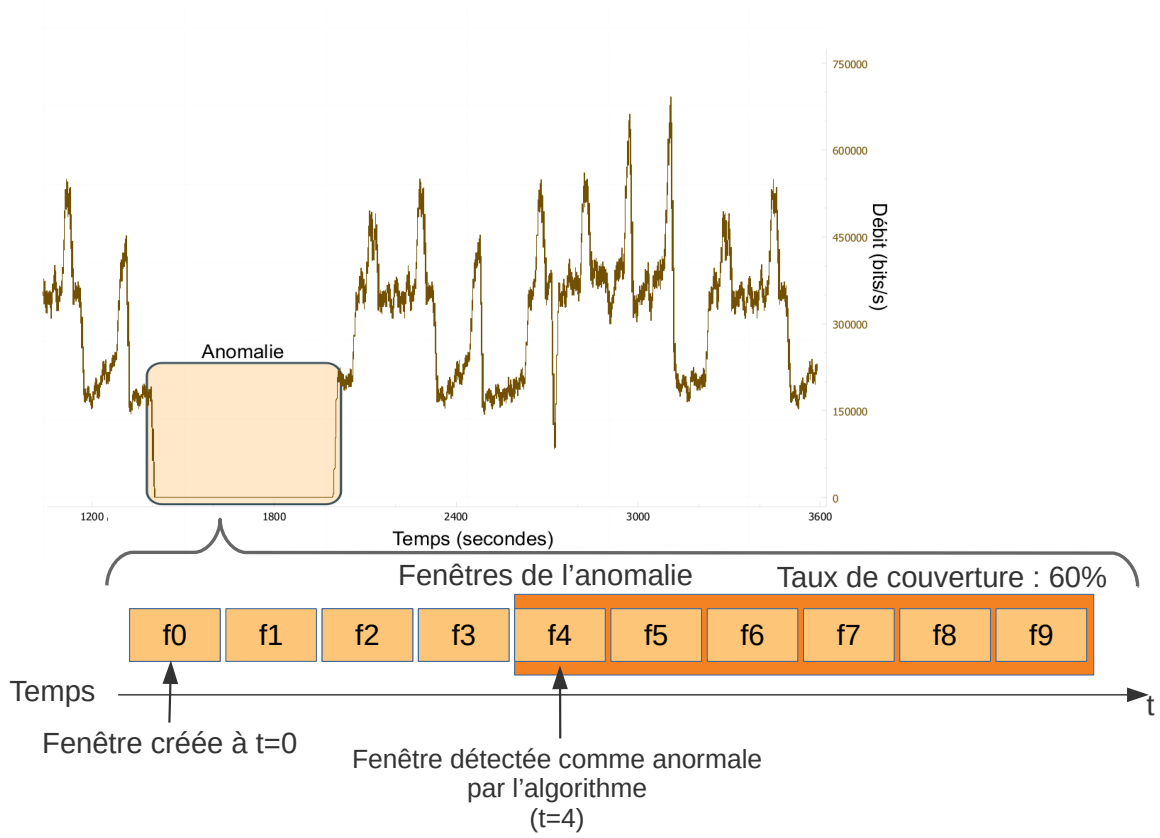


FIGURE 3.14 – Illustration du *taux de couverture* d'une anomalie.

correctement classifier les négatifs. *Le score d'efficacité* quant à lui, récompense les détecteurs dont la précision est plus élevée. Nous évaluons les capacités du détecteur en fonction des paramètres qui donnent le meilleur score suivant ces deux métriques en nous intéressant au délai entre la détection d'une anomalie et le premier paquet qui compose l'anomalie. En d'autres termes, nous nous préoccupons des capacités de l'algorithme à détecter au plus tôt une anomalie en cours.

3.5 Résultats de la détection de Svalinn

Nous étudions l'impact des différentes méthodes de sélection des attributs sur la qualité de la détection. Pour ce faire, nous avons effectué une itération de l'algorithme HTM selon le rang de chacun des attributs que nous évaluons en fonction du *score d'efficacité* et du coefficient de corrélation de Matthews. Les attributs sont sélectionnés suivant la durée et le type de *fenêtre*. Nous évaluons aussi l'impact du nombre d'attributs sur la détection. L'ensemble des paramètres des itérations est résumé dans le Tableau 3.4.

Paramètres	**Détails**
Nombre d'attributs	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35
Méthode de sélection	F-valeur, Superposition, Forêts aléatoires, $\mathcal{X}^2$
Type de *fenêtre*	Flux, Lien, Conversation
Durée des *fenêtres* (en secondes)	1, 2, 3, 5, 10, 15, 20, 30, 60, 120, 300

TABLE 3.4 – Paramètres utilisés lors de l'évaluation.

En règle générale, nous notons aussi que les fonctions de sélection F-valeur et $\mathcal{X}^2$ tendent à s'accorder sur le classement des attributs selon le type de *fenêtre*. Par exemple, les classements de min_fpktl en fonction du type de *fenêtre* sont très similaires ; il en est de même pour $mean\_fpktl$ ou $bpkts\_ps$. Au contraire, nous constatons également que la méthode de sélection par *score de superposition* tend à donner des résultats très différents des autres méthodes. Les attributs max_fpktl, $total\_fpktl$ ou encore $var\_flowpktl$ en sont de bons exemples.

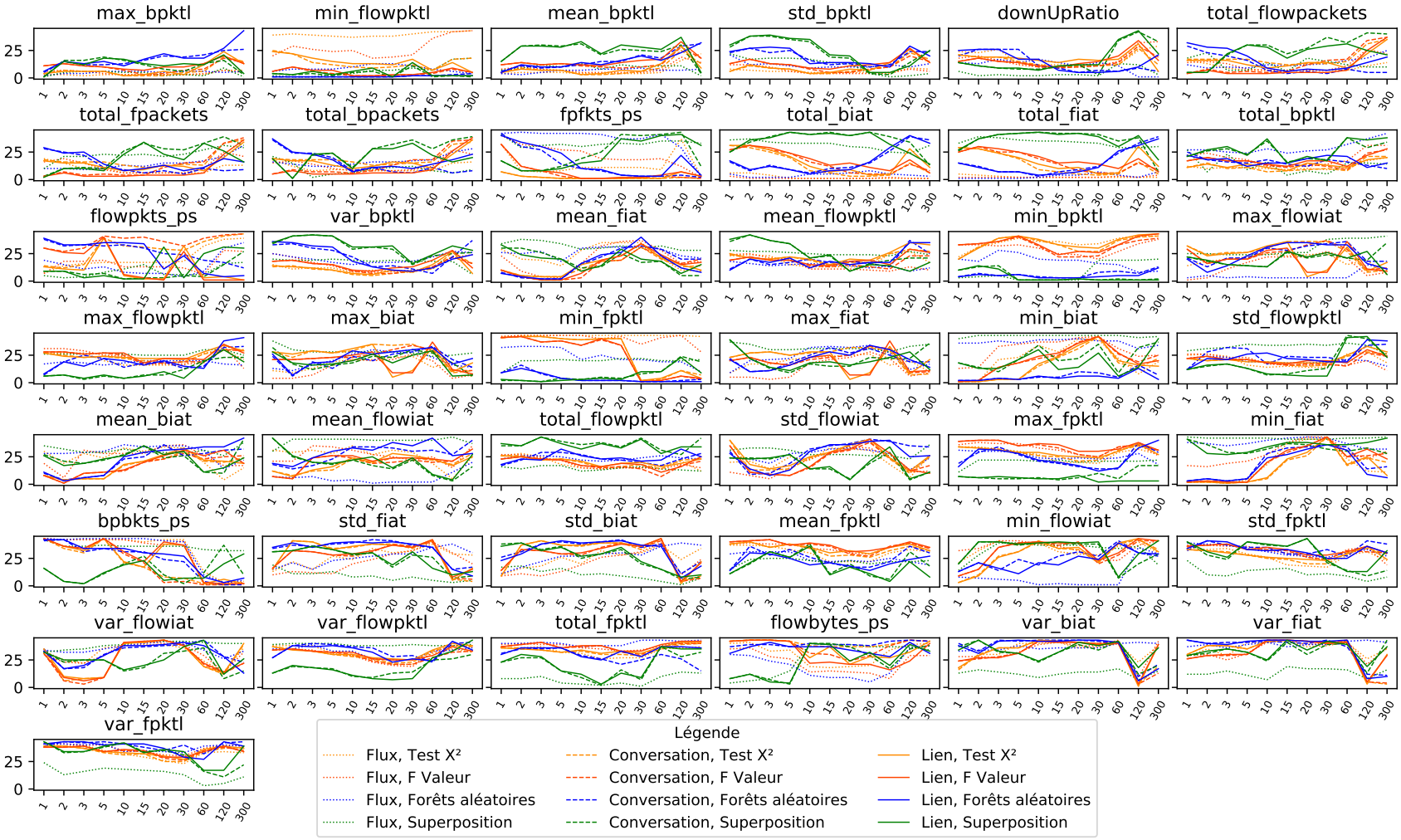


FIGURE 3.15 – Rang des Attributs suivant la méthode de sélection.

3.5.2 Résultats obtenus par les différents paramètres

Pour chaque itération nous sélectionnons donc les attributs utilisés par le dispositif de détection selon leur rang. Nous évaluons ensuite la qualité de la détection en fonction :

— Du *score d'efficacité* qui représente l'aire entre la courbe d'opération optimale de référence (ZRC) et la courbe d'opération du détecteur pour un taux maximal de 3% de faux positifs.

— Du coefficient de corrélation de Matthews de la détection calculé en fonction de l'aire sous la courbe du MCC selon un taux de faux positifs maximal de 3%.

Ces courbes sont tracées en manipulant un seuil à partir duquel une fenêtre est considérée anormale si son score dépasse cette valeur. En pratique les différents seuils correspondent à des centiles de l'ensemble des scores obtenus par les fenêtres lors de l'exécution de l'algorithme. Dans notre étude nous avons sélectionné les centiles suivants :

— 90.0, 90.53, 91.05, 91.58, 92.10, 92.63, 93.15, 93.68, 94.21, 94.73, 95.26, 95.79, 96.32, 96.84, 97.37, 97.89, 98.42, 98.95, 99.47, 1.

Le *score d'efficacité* de chaque itération est représenté par la Figure 3.16. Chaque sous-figure représente le score des différentes méthodes de sélection des attributs utilisées pour une durée de *fenêtre* donnée. Les scores sont obtenus en fonction du type de *fenêtre* et du nombre d'attributs utilisés pour la détection.

Nous pouvons établir que la sélection des attributs par des méthodes de filtre n'est pas systématiquement efficace pour choisir les meilleurs attributs pour l'algorithme HTM. En effet, aucune méthode ne semble être plus efficace que les autres. De plus, il ne semble pas y avoir de corrélation entre le nombre d'attributs ou la durée des *fenêtres* avec l'efficacité de la détection. Toutefois, en observant les distributions des différents scores de détection représentés sous la forme de boîtes à moustaches dans la figure, nous constatons que les méthodes de sélection par la F-valeur et le Test $\mathcal{X}^2$ sont en règle générale plus efficaces. Nous observons également que les distributions des scores des *fenêtres de Flux* s'étendent des pires au meilleurs scores. Il reste tout de même difficile de conclure sur l'efficacité d'une méthode particulière tant le comportement des distributions ne semble pas présenter une quelconque logique.

Nous avons donc utilisé l'autre métrique à notre disposition afin de tenter d'interpréter les résultats de l'algorithme HTM. La Figure 3.17 présente donc les scores du MCC. Nous constatons que contrairement à ce que nous pouvions observer sur la figure précédente, l'utilisation des *fenêtres de Flux* présente systématiquement de moins bons résultats que les autres types de *fenêtres*. Toutefois, nous notons aussi que les différentes fonctions de sélection des attributs ne semblent pas avoir un impact significatif bien qu'il apparaît que les distributions des valeurs de MCC pour la F valeur et le test $\mathcal{X}^2$ semblent donner de meilleurs résultats comme le laissait suggérer la Figure 3.16.

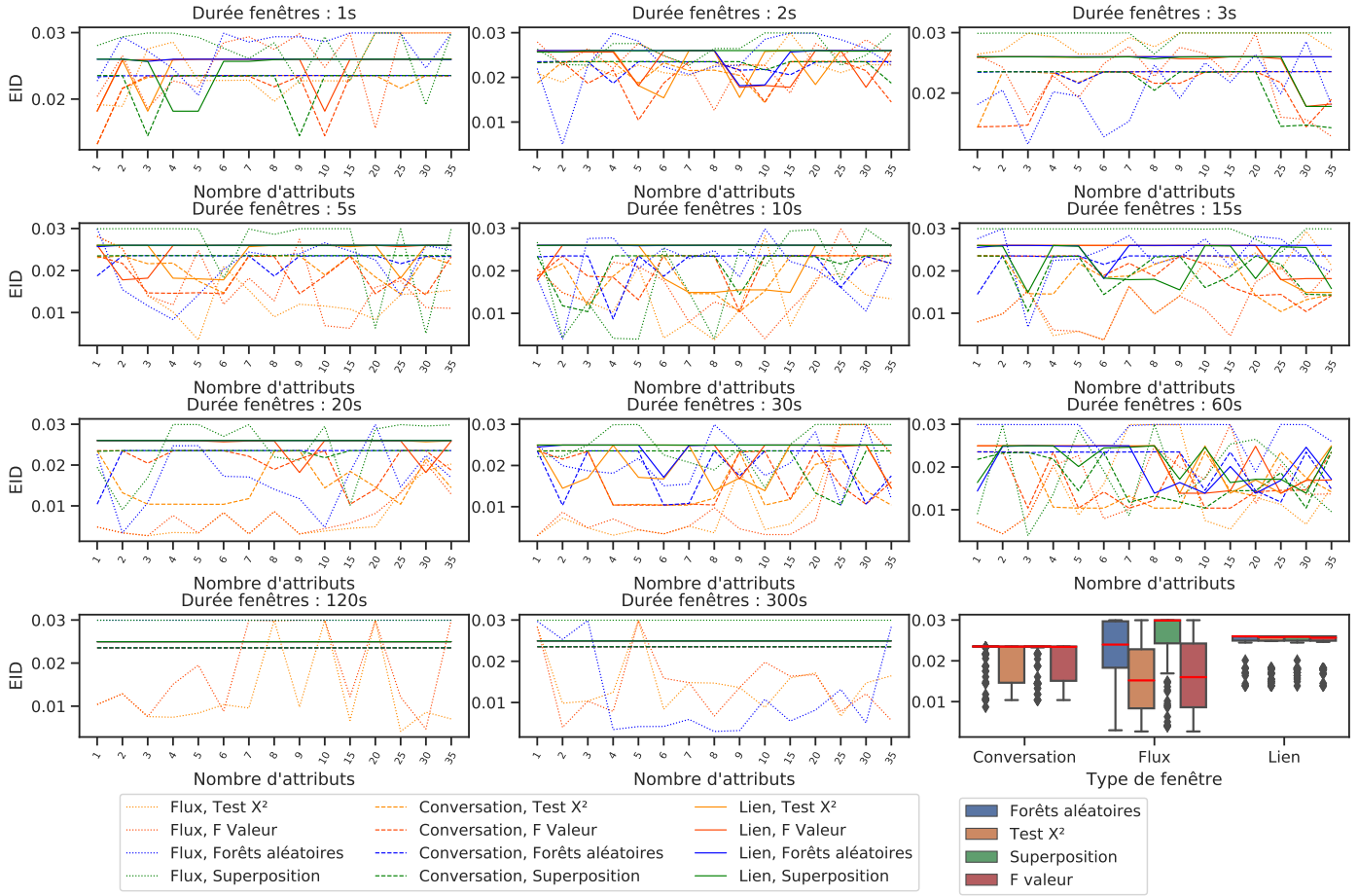


FIGURE 3.16 – Score d'efficacité de la détection (EID) par durée des *fenêtres* pour l'algorithme HTM.

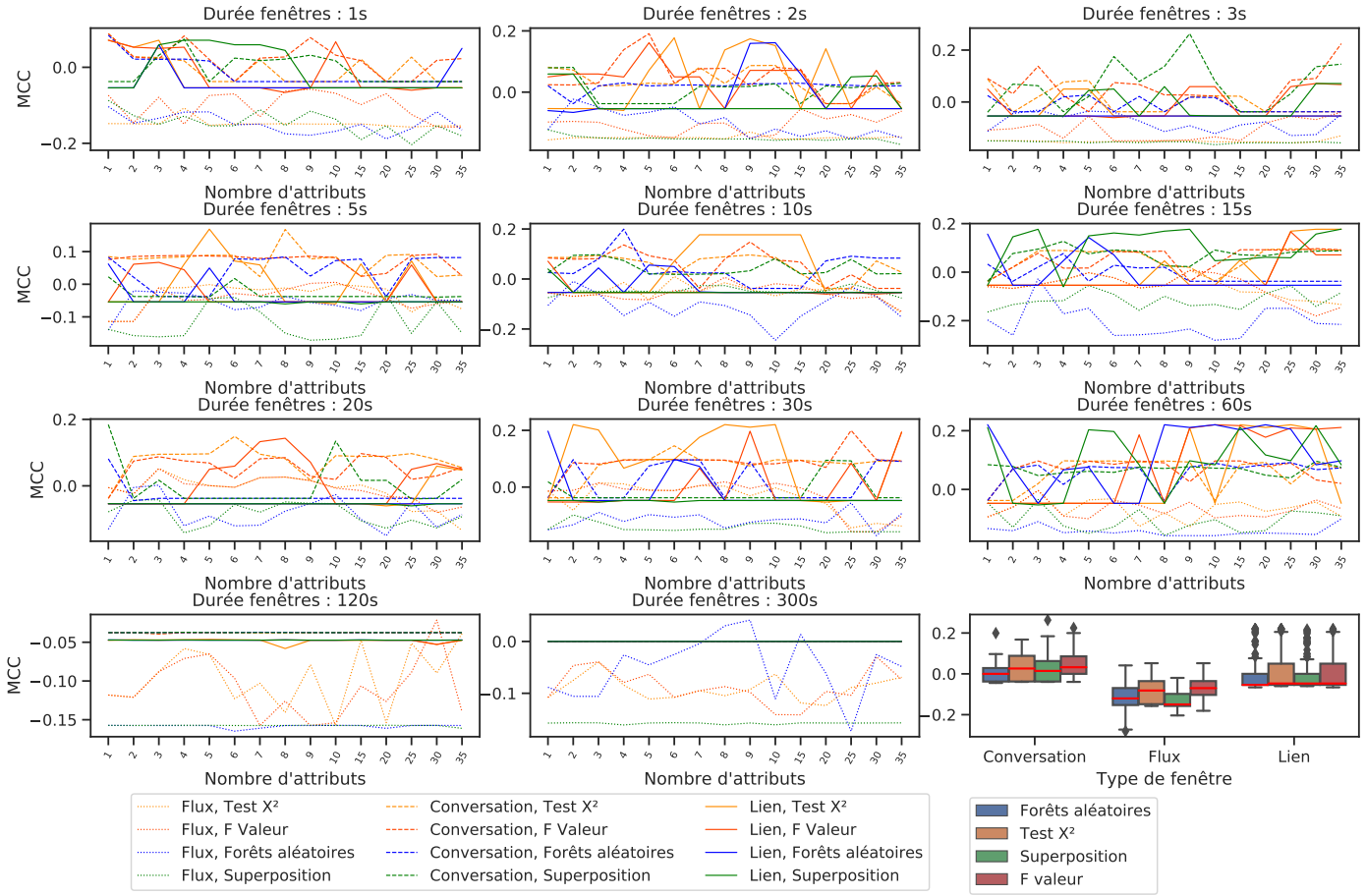


FIGURE 3.17 – Score de MCC par durée des *fenêtres* pour l'algorithme HTM.

L'observation d'un moins bon score de MCC pour les *fenêtres de Flux* pourrait être la cause du taux d'anomalies présent à l'intérieur du corpus (Figure 3.12). En effet, les différentes anomalies n'ont pas le même poids dans le corpus d'entraînement. Par exemple, lorsque nous utilisons la représentation par *fenêtres de Flux* d'une durée de 1 seconde, le nombre de *fenêtres* nécessaires à la représentation des différentes anomalies sera (Appendice C, Tableau C.1) :

— 235 pour l'attaque par tunnel-DNS.
— 1030 et 1022 pour les différents scans (XMAS et Syn).
— 10 pour l'anomalie de télémétrie.

Les scans sont donc très majoritairement représentés dans le corpus de données si nous utilisons la représentation des *Flux*. De plus, les scans entraînent la création des *fenêtres* dans une durée très courte d'environ 2 secondes par Scan. Or, l'algorithme HTM sera amené à traiter ces *fenêtres* au fur et à mesure qu'elles sont créées, et, compte tenu de leur nombre, il va donc s'habituer aux attributs qu'elles présentent : il ne les considèrera donc plus comme anormales.

Nous nous sommes ensuite intéressés au *taux de couverture* des anomalies du détecteur en fonction des *scores d'efficacité* et de *MCC* et du type de *fenêtre* utilisé. À cet effet, nous avons sélectionné pour chaque durée de *fenêtre* les couples formés d'une méthode de sélection et d'un nombre d'attributs donnant les meilleurs scores d'efficacité et de MCC. La Figure 3.18 et la Figure 3.19 représentent ce *taux de couverture* par durée des *fenêtres*. Chaque courbe illustre le *taux de couverture* moyen d'un type d'anomalie suivant le type de *fenêtre*.

Ainsi, chaque courbe représente le *taux de couverture* des anomalies en fonction du taux de faux positifs pour :

— Une durée et un type de fenêtre fixe (flux, lien, conversation) ainsi qu'un nombre d'attributs.

De la même manière que pour les figures précédentes, le taux de faux positifs dépend du seuil de détection d'une anomalie fixé en fonction du score obtenu par l'ensemble des fenêtres lors de l'exécution de l'algorithme. Pour tracer les courbes nous sélectionnons les centiles suivants comme seuil de détection :

— 90.0, 90.53, 91.05, 91.58, 92.10, 92.63, 93.15, 93.68, 94.21, 94.73, 95.26, 95.79, 96.32, 96.84, 97.37, 97.89, 98.42, 98.95, 99.47, 1.

Cela nous permet d'évaluer les différents résultats de l'ensemble de nos paramètres lorsque l'algorithme a été capable de détecter toute les anomalies.

Nous cherchons donc à déterminer parmis les meilleurs scores de MCC et de EID quels étaient les paramètres pour lesquels HTM détecte toutes les anomalies avec la plus grande couverture possible (i.e. le plus tôt possible).

Nous constatons qu'en règle générale, l'anomalie de télémétrie et l'attaque par tunnel DNS sont mieux détectées que les scans en utilisant les *fenêtres de Conversations* ou de *Liens*. Nous remarquons également que c'est uniquement lorsque les *fenêtres de Flux* sont utilisées que l'ensemble des anomalies sont détectées. En particulier lorsque les durées des *fenêtres* sont supérieures à 10 secondes. Par exemple, nous présentons dans le Tableau 3.5 les taux de couverture des anomalies des *fenêtres de Flux* en fonction des deux métriques d'évaluation. Nous y présentons les

taux de faux positifs (inférieurs à 3%) lorsque toutes les anomalies sont détectées.

Dur.	Type	#Attr.	Métr.	Cent.	Anomalies				TFP
					SYN	XMAS	DNS	Télé.	
10.0	Flux	6	MCC	98.95	0.019	0.036	0.750	0.889	0.013
15.0	Flux	6	EID	100	0.001	0.012	0.750	0.667	0.004
20.0	Flux	3	MCC	99.47	0.090	0.043	0.833	0.889	0.009
20.0	Flux	3	EID	99.47	0.090	0.043	0.833	0.889	0.009
60.0	Flux	3	EID	99.47	0.009	0.012	0.500	0.750	0.009

TABLE 3.5 – Taux de couverture maximal des anomalies suivant le plus bas taux de faux positifs et de la métrique d'évaluation pour différentes durées de *fenêtres*.

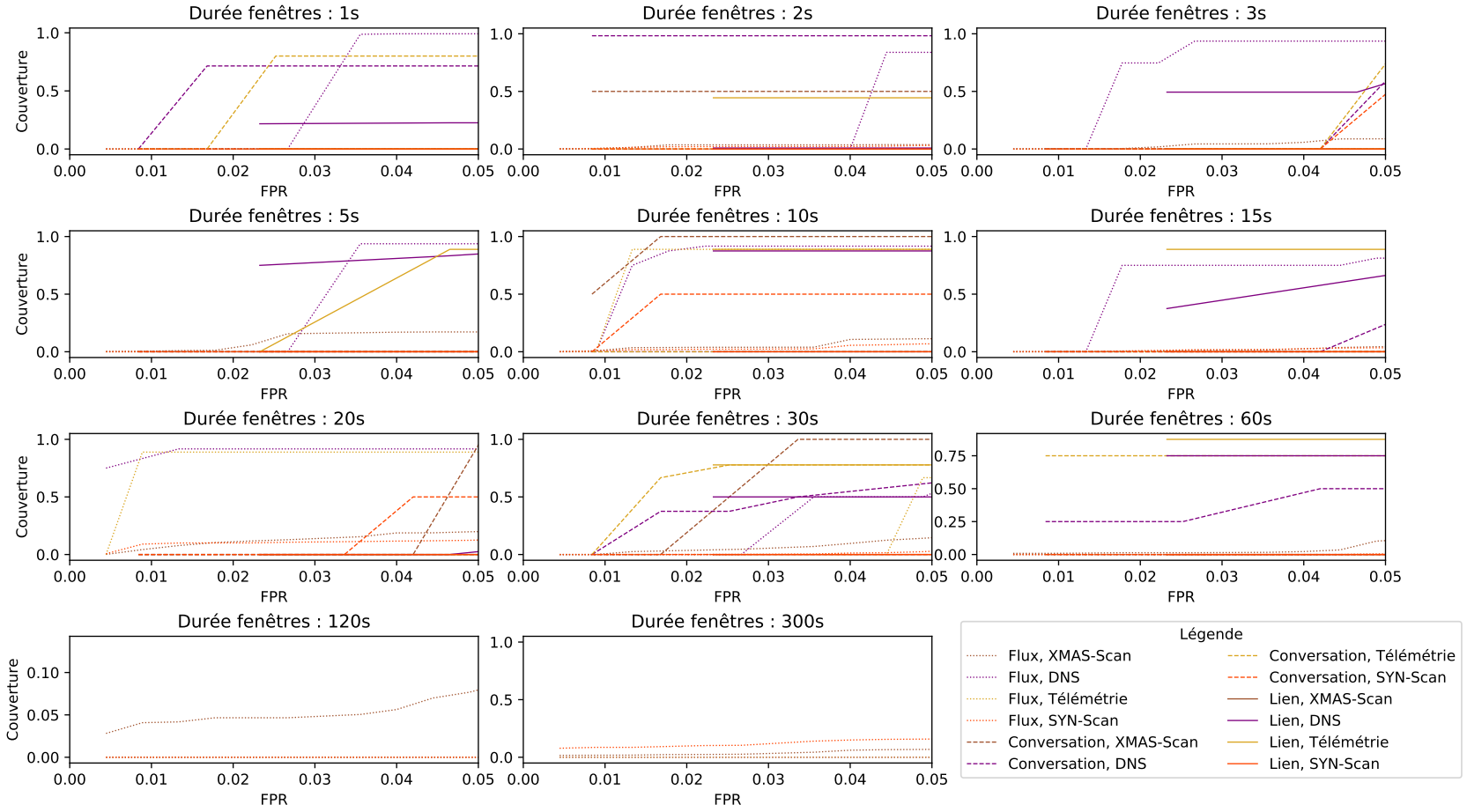


FIGURE 3.18 – Couverture des anomalies par durée des *fenêtres* en suivant le score MCC.

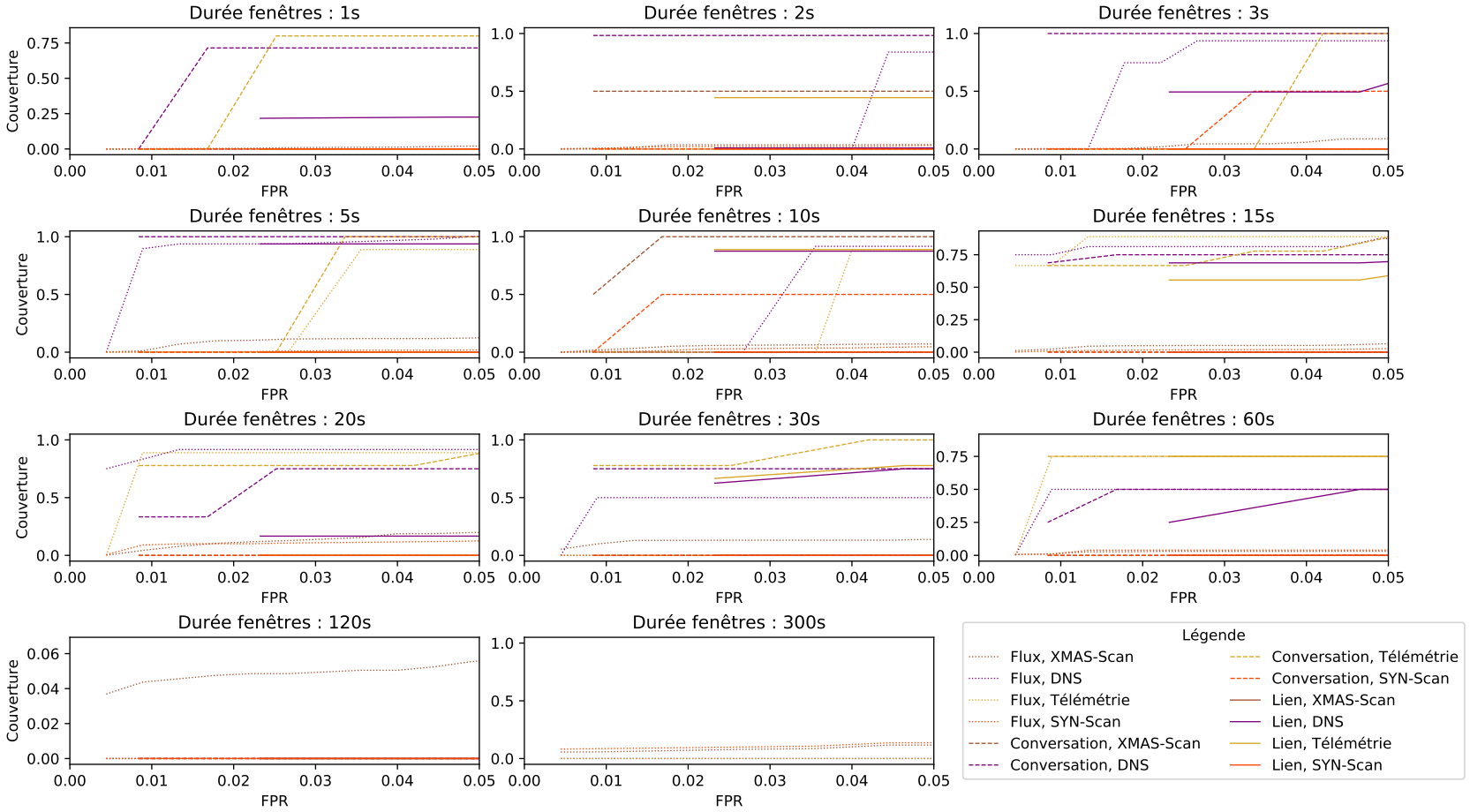


FIGURE 3.19 – Couverture des anomalies par durée des *fenêtres* selon le score d'efficacité pour l'algorithme HTM.

3.5.3 Comparatif avec LSTM

Nous avons procédé à la même évaluation en remplaçant l'algorithme HTM par un Autoencodeur LSTM. Ce réseau de neurones non supervisé est composé de 7 couches de neurones dont les 3 premières servent à créer une représentation compressée de l'entrée et constitue la phase d'encodage. La couche centrale sert à répéter l'entrée compressée à travers les différentes couches du décodeur représenté par les trois dernières couches. La dernière couche de neurones a pour fonction de produire une version reconstruite de l'entrée, conformément à la Figure 3.20 qui sert de prédiction. Ce réseau a été créé grâce à la librairie python `Keras` [17] qui permet la création de différents types de réseaux de neurones. Pour chaque itération l'algorithme est entraîné grâce aux *fenêtres* des 50 premières minutes du corpus durant lesquelles aucune anomalie n'est présente.

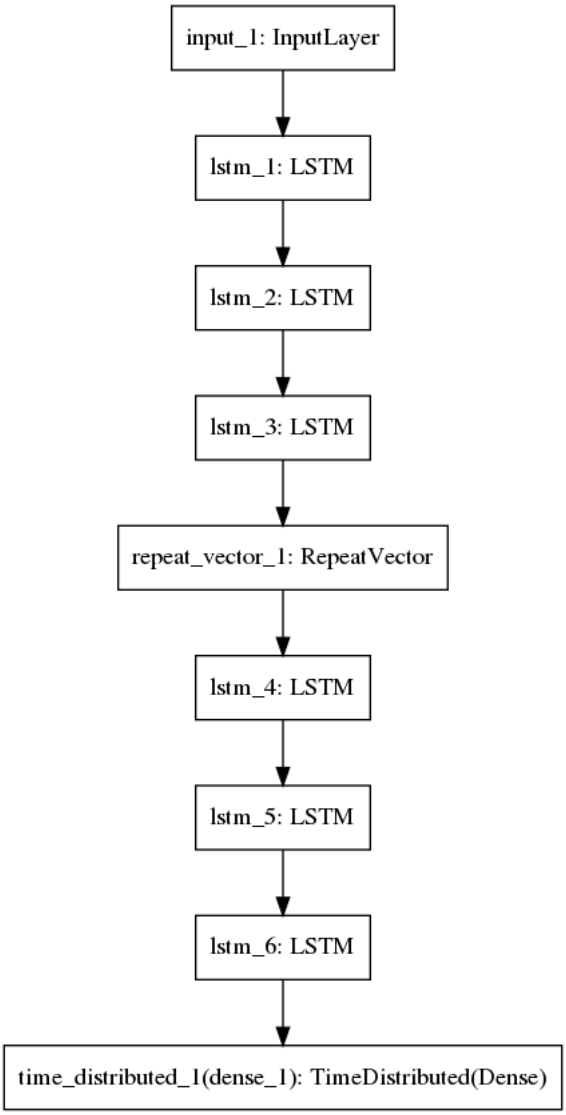


FIGURE 3.20 – Représentation générée par `Keras` du réseau LSTM que nous avons utilisé pour l'algorithme HTM.

Nous traçons les mêmes courbes que pour HTM et constatons en premier lieu que, contrairement à HTM, les scores d'efficacité et de MCC qu'obtient LSTM sont grandement impactés par le nombre d'attributs utilisés. En effet, les Figure 3.21 et Figure 3.22 montrent bien une augmentation des capacités de détection en fonction du nombre d'attributs. De plus, nous constatons également que l'utilisation des *fenêtres de Flux* est moins efficace que les *fenêtres de Liens* ou de *Conversation* lorsque le nombre d'attributs utilisés est bas. Cependant, les *fenêtres de Flux* sont presque systématiquement plus efficaces lorsque le nombre d'attributs utilisés est supérieur à 15. Toutefois, nous constatons également que contrairement à HTM, LSTM est particulièrement efficace sur les *fenêtres* de durée supérieure à 30 secondes.

De la même façon que pour HTM nous présentons également les taux de couver-

17. https://keras.io/

ture des anomalies des meilleurs scores d'efficacité et de MCC pour chaque durée de *fenêtre*. De manière générale, il apparaît que les résultats de LSTM semblent en règle générale meilleurs que ceux de HTM, notamment au niveau de la couverture des scans (Figure 3.21 et Figure 3.22). Il faut cependant noter que LSTM n'apprend pas en continu et qu'il n'est donc pas capable de s'adapter à l'évolution du trafic. Enfin, comme le montre le Tableau 3.6, LSTM est capable de détecter toutes les anomalies pour chaque durée de *fenêtre*. Cependant nous pouvons remarquer que les taux de faux positifs sont supérieurs à ceux qu'obtient HTM lorsque nous nous focalisons sur la détection de chaque anomalie sans chercher à maximiser le taux de couverture.

Dur.	**Type**	**#Attr.**	**Métr.**	**Cent.**	**Anomalies**				**TFP**
					SYN	**XMAS**	**DNS**	**Télé.**	
1.0	Conv.	30	MCC	99.47	0.25	0.167	0.996	0.40	0.017
1.0	Conv.	10	EID	98.95	0.25	0.167	0.996	0.40	0.017
2.0	Conv.	35	MCC	100	0.50	0.250	0.051	1.00	0.008
3.0	Conv.	2	MCC	98.95	0.50	0.500	0.987	1.00	0.017
3.0	Conv.	2	EID	98.95	0.50	0.500	0.987	1.00	0.017
5.0	Conv.	7	MCC	98.95	0.50	0.500	1.000	1.00	0.017
5.0	Conv.	5	EID	98.95	0.50	0.500	0.979	1.00	0.017
10.0	Conv.	3	EID	98.95	0.50	0.500	0.958	1.00	0.017
15.0	Conv.	2	EID	100	0.50	0.500	1.000	1.00	0.008
20.0	Conv.	2	EID	98.95	0.50	0.500	1.000	1.00	0.017
30.0	Conv.	5	MCC	98.95	0.50	0.500	1.000	1.00	0.017
30.0	Conv.	2	EID	98.95	0.50	0.500	1.000	1.00	0.017
60.0	Conv.	6	MCC	98.95	1.00	1.000	1.000	0.25	0.017
60.0	Conv.	6	EID	98.95	1.00	1.000	1.000	0.25	0.017
120.0	Conv.	10	MCC	98.95	1.00	1.000	1.000	1.00	0.017
120.0	Conv.	6	EID	98.95	1.00	1.000	1.000	1.00	0.017

TABLE 3.6 – Taux de couverture maximal des anomalies en fonction du plus bas taux de faux positifs et de la métrique d'évaluation pour différentes durées de *fenêtres*.

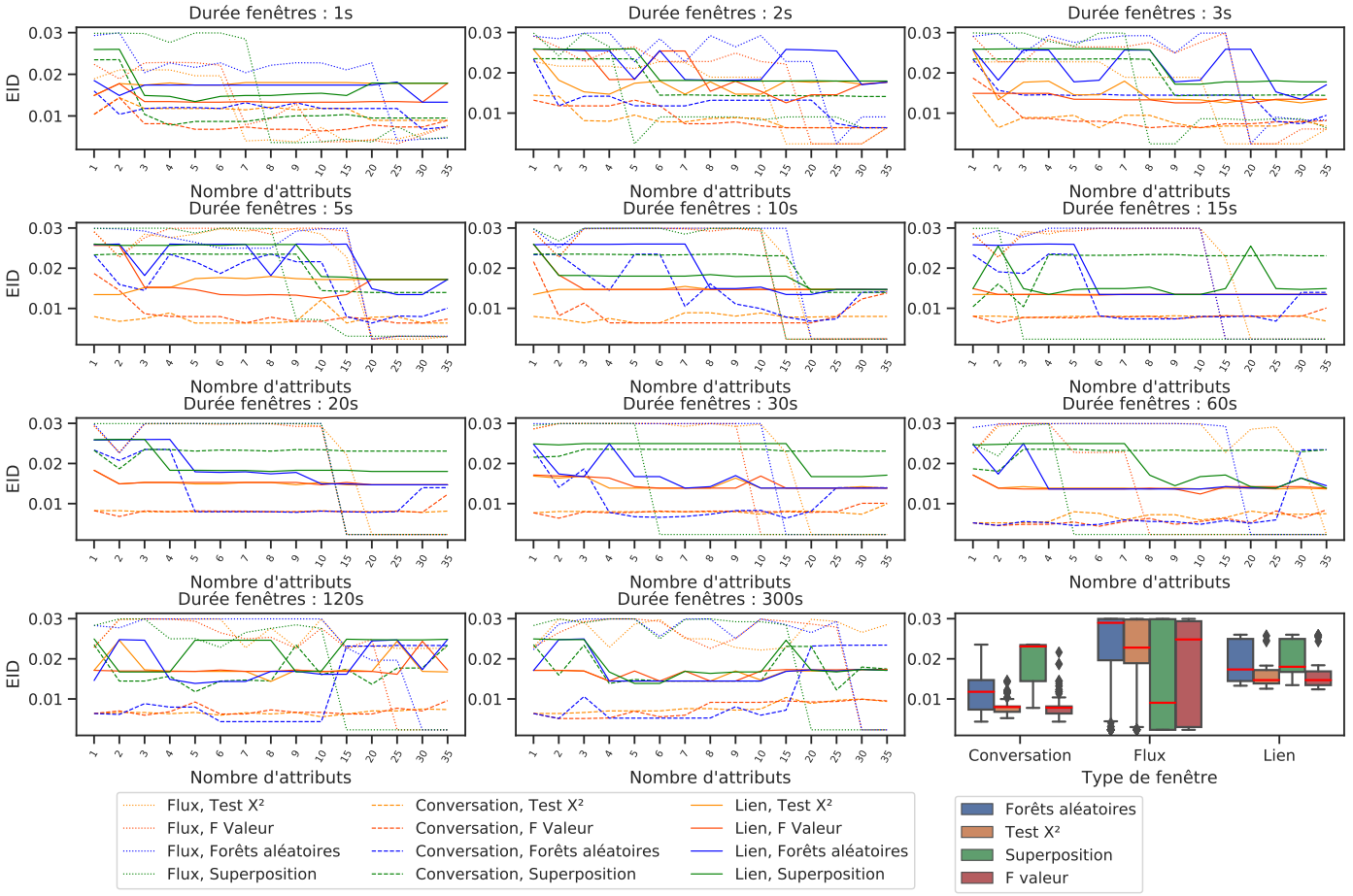


FIGURE 3.21 – Score d'efficacité de la détection (EID) par durée des *fenêtres* pour l'algorithme LSTM.

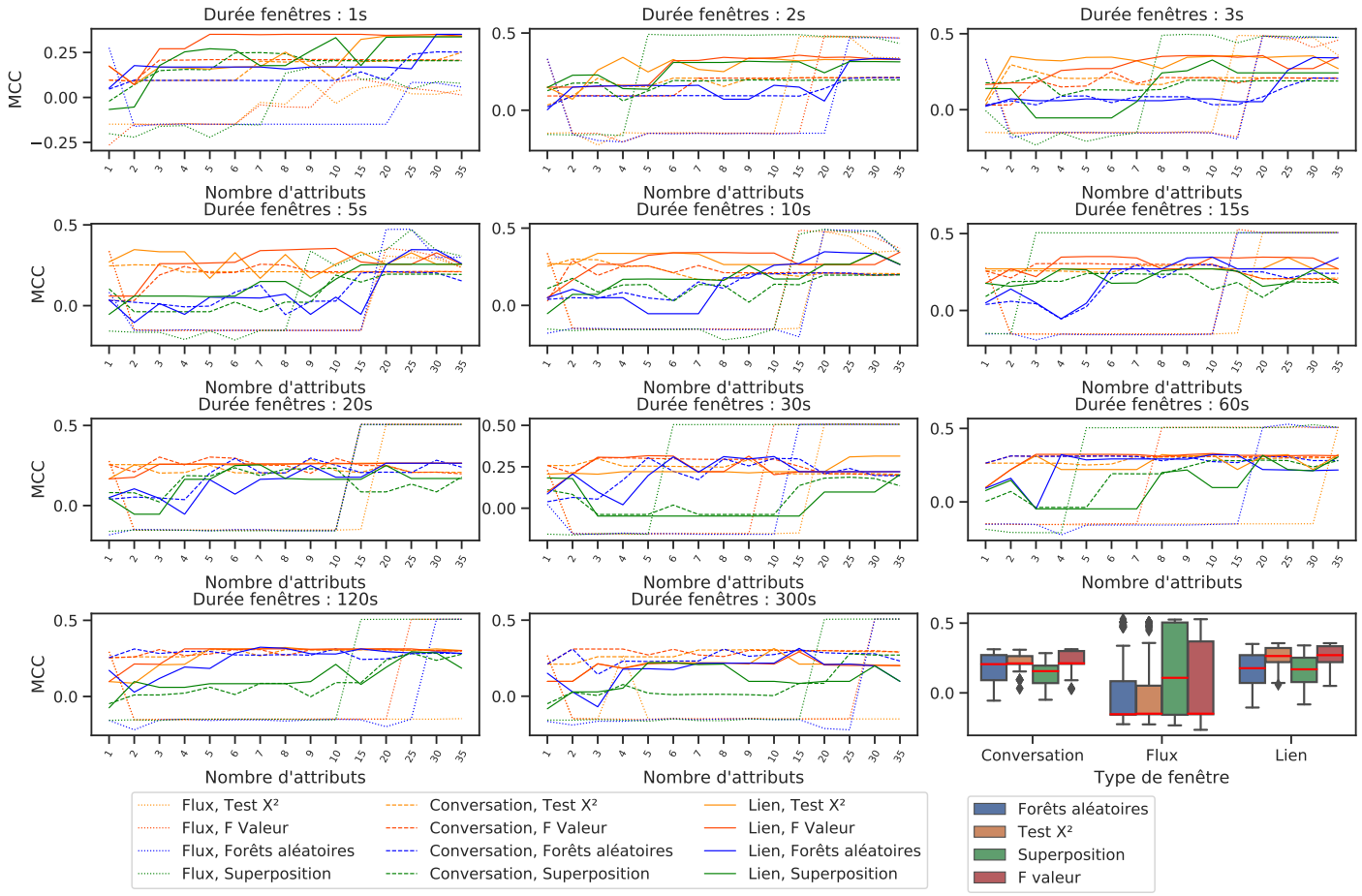


FIGURE 3.22 – Score de MCC par durée des *fenêtres* pour l'algorithme LSTM.

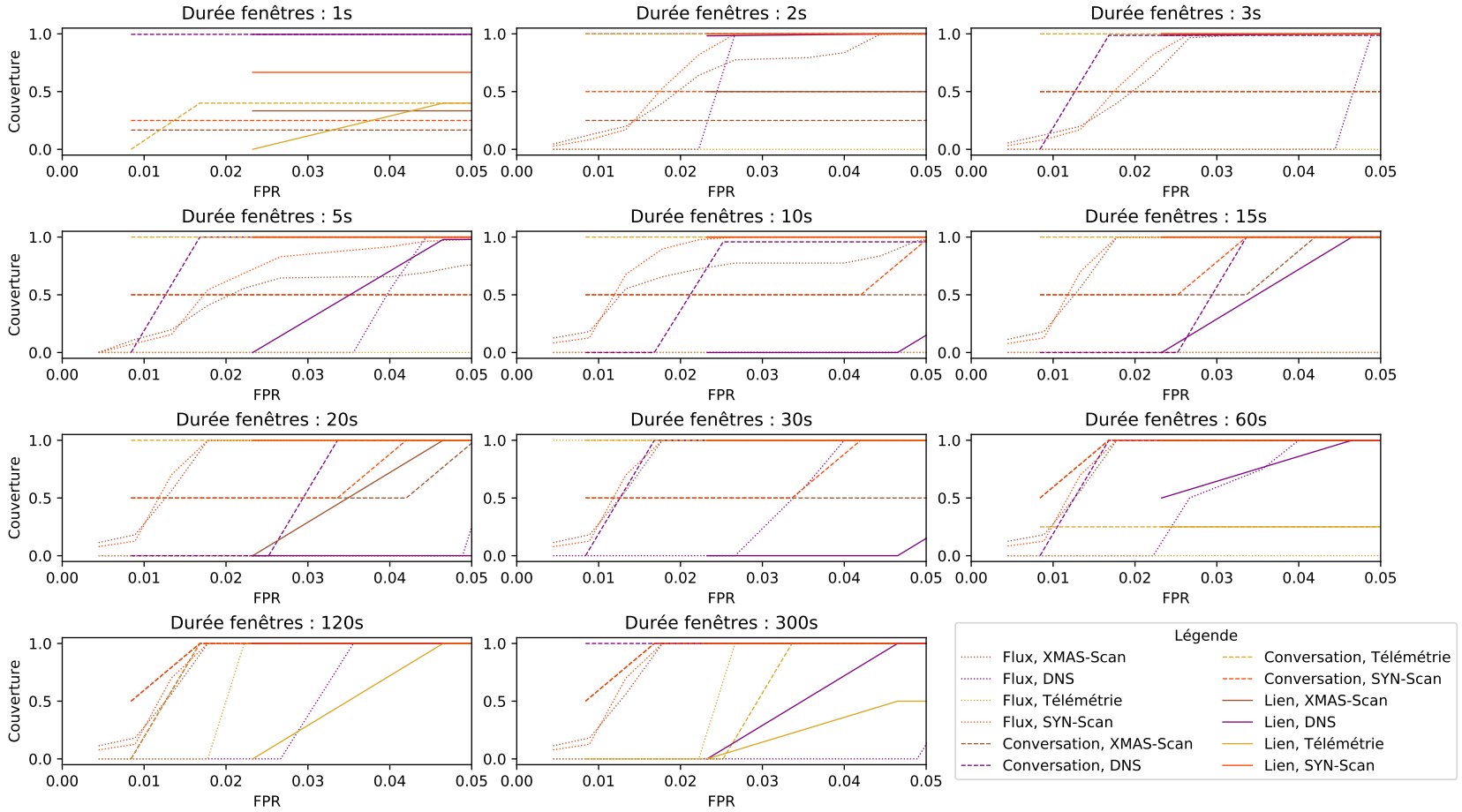


FIGURE 3.23 – Couverture des anomalies par durée des *fenêtres* en fonction du MCC pour l'algorithme LSTM.

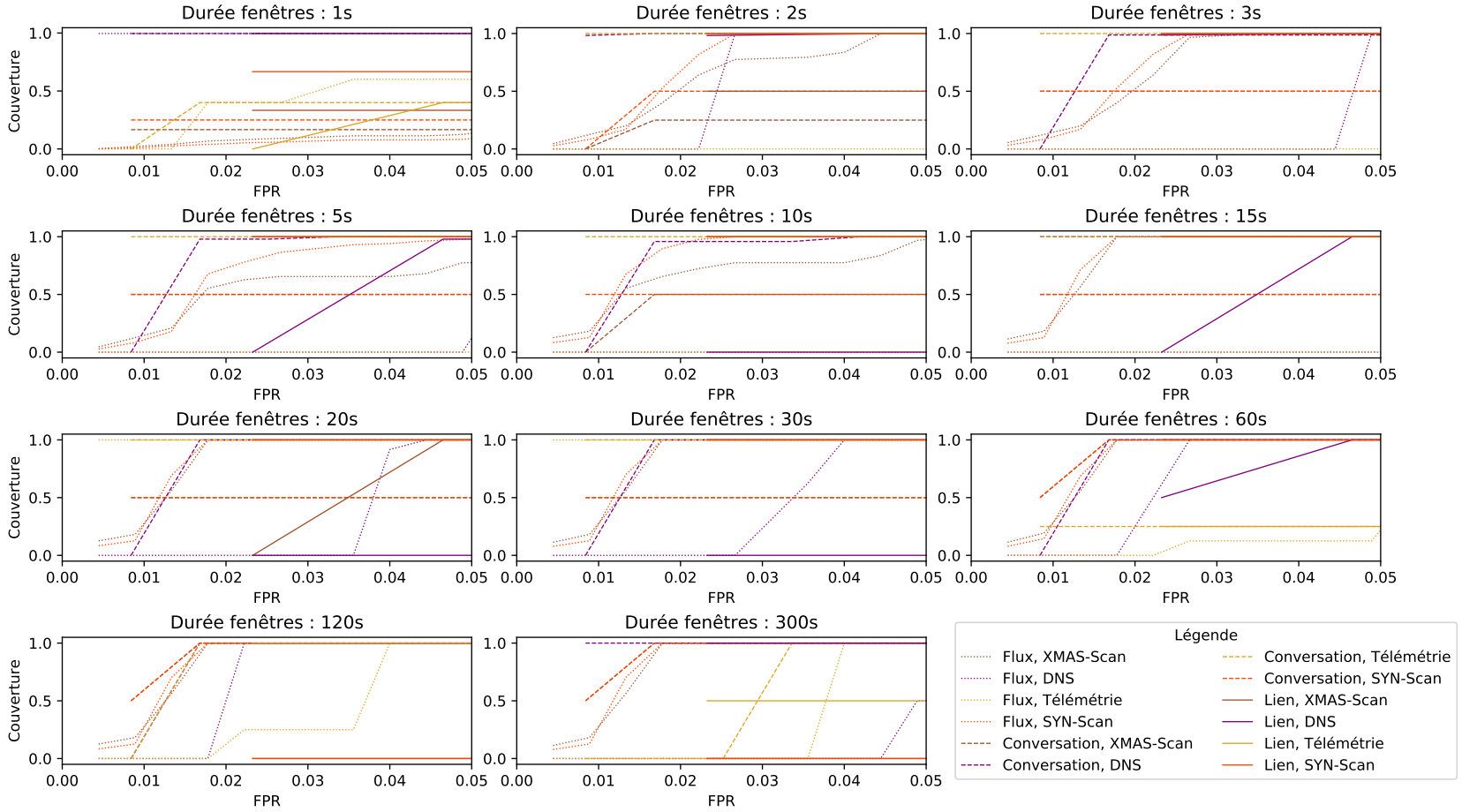


FIGURE 3.24 – Couverture des anomalies par durée des *fenêtres* en fonction du score d'efficacité pour l'algorithme LSTM.

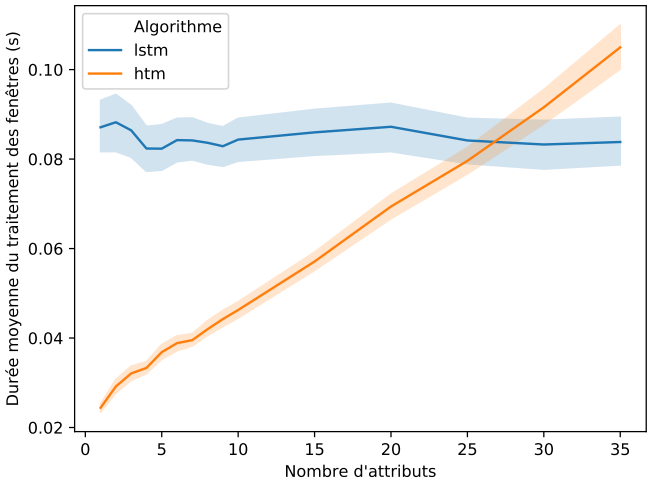


FIGURE 3.25 – Durée moyenne du traitement d'une *fenêtre de description* par les deux algorithmes.

– **Vitesse de traitement** – Nous avons également évalué la durée moyenne de traitement des *fenêtres de description* par les deux algorithmes. Ces résultats présentés dans la Figure 3.25 résultent de l'exécution isolée des algorithmes sur un serveur Ubuntu Bionic 18.04 doté de 32 cœurs Intel Xeon (2.60Ghz) et 62GB de ram. Nous constatons que le traitement des *fenêtres* par l'algorithme HTM croît de façon linéaire avec le nombre d'attributs utilisés pour la détection alors que l'autoencoder-LSTM reste constant. De plus, les meilleurs résultats de HTM sont obtenus avec très peu d'attributs (3 ou 6 attributs), en conséquence sont fonctionnement est environ 200% plus rapide que l'autoencoder-LSTM.

3.6 Résumé

Dans ce chapitre, nous avons présenté l'évaluation de notre dispositif de détection. Celui-ci repose sur la génération du jeu de données à partir de l'environnement d'émulation `Autobot`. Nous l'avons conçu en adéquation avec les besoins de réalisme et de diversité des anomalies que nous cherchons à détecter. Notre méthodologie d'évaluation de notre dispositif se déroule en 2 étapes : La sélection des attributs et l'évaluation de la détection. Nous avons utilisé 4 méthodes différentes de sélection des attributs pour le processus de détection. Ensuite nous évaluons la qualité de la détection suivant 2 métriques : Le score d'efficacité et le score du coefficient de corrélation de Matthews.

Nous avons ainsi démontré les capacités de l'algorithme HTM à détecter les anomalies du jeu de données en fonction de la durée et du type des *fenêtres de description instantanée*. Enfin, nous avons comparé les résultats de HTM avec ceux d'un réseau de neurones non-supervisé Autoencodeur-LSTM.

Conclusion

Sommaire

La détection d'anomalies dans les réseaux est un domaine de recherche qui a été l'objet de nombreuses contributions au cours des dernières décennies. Cependant, les cas d'applications se sont principalement concentrés sur les réseaux informatiques classiques ou les réseaux industriels. Or, l'avènement des véhicules connectés et les nombreuses démonstrations de piratages dont ils ont été victimes, incitent la communauté scientifique et les acteurs de l'industrie à concevoir des moyens de sécurisation de ces nouveaux composants du réseau. À cet effet, nous avons présenté les différents dispositifs de communications intégrés aux véhicules modernes comme le bus-CAN et les ECUs, le port OBD-II ainsi que les TCUs et les composants permettant aux véhicules de communiquer de proche en proche par le biais des réseaux VANETS.

Nous avons par la suite dressé un état de l'art des vecteurs d'attaques qui rendent vulnérables les véhicules connectés notamment par l'exploitation par les pirates de ces différents dispositifs. Ce constat nous a conduit à présenter de nombreux travaux ayant pour objectif de permettre la détection d'intrusions et d'anomalies dans les communications des véhicules. Pour autant, nous avons réalisé qu'une grande majorité de ces travaux se concentrait principalement sur les communications internes aux véhicules au détriment des communications externes et en particulier celles utilisant les réseaux cellulaires.

Or, les réseaux cellulaires de véhicules constituent un vecteur majeur d'attaques contre les véhicules connectés. En effet, des exemples d'attaques à distance contre des véhicules actuellement commercialisés, prouvent que les communications cellulaires constituent une vulnérabilité pour le véhicule connecté. De surcroît, cette vulnérabilité est accentuée par le fait que la découverte d'un moyen d'exploitation d'une faille de sécurité sur un modèle de véhicule pouvait permettre à un attaquant de lancer des attaques à grande échelle si le modèle avait été vendu à de nombreux exemplaires.

Nous avons ainsi décidé d'explorer la voie de la détection d'anomalies dans les réseaux cellulaires de véhicules. Pour cela notre problématique s'est divisée selon les points suivants :

— Proposer une détection efficace compte tenu des caractéristiques du trafic véhiculaire. Celui-ci est composé de deux types de services : ceux liés aux systèmes de transport intelligent et ceux destinés aux systèmes de divertissement des utilisateurs du véhicule.
— Proposer une détection capable de couvrir un grand nombre d'anomalies. Les véhicules sont soumis à de nombreux types d'attaques allant du déni-de-service à l'exfiltration de données, lesquelles ont un impact différent sur les propriétés du réseau.
— Proposer une détection capable de s'adapter à l'évolution du trafic. Le dispositif de détection étant destiné à intégrer le véhicule sur toute sa période d'utilisation, il doit donc être capable de s'adapter à la nature changeante du trafic comme l'amélioration des conditions réseaux ou le changement de comportement des utilisateurs laquelle entraîne une modification des propriétés des communications.
— Proposer une détection autonome, capable de classifier et d'archiver les anomalies sans avoir recours à une assistance humaine. Il convient de rappeler que la nature de l'environnement de l'exécution du dispositif de détection limite les possibilités d'une supervision directe par un opérateur. De plus, une supervision directe risquerait d'exposer le véhicule à des attaques via le système de supervision.

Nous avons donc conçu un dispositif de détection baptisé **Svalinn** qui repose sur les principes suivants :

— La modélisation sous forme d'une ontologie des communications du véhicule afin de permettre la discrimination précise d'une anomalie grâce à l'utilisation de *fenêtres de description instantanée* du trafic.
— Une détection autonome par l'algorithme à mémoire temporelle hiérarchique (HTM) reposant sur l'analyse des attributs des *fenêtres de description instantanée*.
— Un archivage des alertes générées grâce à l'utilisation de règles d'inférences qui permettent de regrouper l'ensemble des paquets responsables d'une anomalie.

Les performances de notre système ont été évaluées selon le critère de la qualité de la détection. Cela a nécessité la construction d'un jeu de données grâce à notre outil **Autobot**. Cette évaluation a été complétée par un comparatif des capacités de détection de l'algorithme HTM avec un réseau de neurones Autoencodeur-LSTM.

Nous allons maintenant résumer l'ensemble de nos contributions puis présenter les limites de notre approche ainsi que des pistes d'améliorations possibles à nos travaux.

4.1 Contributions

4.1.1 Svalinn

`Svalinn` est un système de détection d'anomalies destiné à opérer en autonomie à l'intérieur de véhicules connectés. Il doit donc pouvoir être exécuté dans un environnement où les ressources en mémoire et en capacité de calculs sont limitées. `Svalinn` repose sur la représentation des communications véhiculaires à l'intérieur d'une ontologie et l'utilisation d'un algorithme de détection d'anomalies non-supervisé.

Le traitement du trafic est réalisé en plusieurs étapes. La première consiste à en extraire des paquets observés à l'entrée du véhicule et les modéliser dans l'ontologie. Tel que nous l'avons présenté, les paquets sont répartis dans des *fenêtres de description instantanée* desquelles sont extraits des indicateurs du comportement du trafic. Nous avons imaginé plusieurs types de *fenêtres* afin de représenter avec différents degrés de précision le trafic : directement au niveau des *Flux* ou à plus grande échelle au niveau des *Liens*. Les attributs des fenêtres sont utilisés par l'algorithme de détection pour la classification de chaque *fenêtre*.

Pour toutes les *fenêtres* qui seront considérées comme anormales, des règles d'inférences sont appliquées. Cela permet de générer un rapport d'alerte afin d'assurer l'archivage des anomalies, mais aussi de regrouper l'ensemble des paquets responsables.

4.1.2 Autobot

Afin d'évaluer notre système, nous avons conçu un environnement de génération dynamique de communications véhiculaires réalistes nommé `Autobot`. Cet environnement émule le comportement d'un réseau de véhicules par le biais de différentes applications telles qu'un service de télémétrie véhiculaire ainsi que des applications de divertissement et de navigation. Afin d'assurer le réalisme des caractéristiques réseau de l'environnement, nous avons reproduit les propriétés d'un réseau 4G en termes de latence, de débit et de perte de paquets à partir des résultats d'enquêtes de l'autorité de régulation des communications électroniques et des postes (ARCEP) accompagnés d'une étude de terrain.

Nous avons évalué le comportement d'`Autobot` en termes de consommation des ressources de l'hôte sur lequel est exécuté l'environnement. Cette évaluation nous a permis de démontrer qu'`Autobot` est capable de générer un grand nombre de communications de véhicules tout en maintenant les propriétés attendues du réseau.

Nous utilisons cet environnement pour la génération d'attaques et anomalies basées sur des cas réels de piratages de véhicules connectés. Le corpus que nous avons généré contient plus de 4h de communications et d'anomalies durant lesquelles un processus de reconnaissance des ports ouverts du véhicule est effectué (scan). De plus, une attaque par DNS-tuneling destinée à extraire des informations personnelles du véhicule est réalisée. Enfin, une anomalie dans l'application de télémétrie du véhicule est également réalisée.

4.1.3 Évaluation de Svalinn

Nous avons évalué les capacités de détection de **Svalinn** grâce au jeu de données que nous avons créé avec `Autobot`. Cette évaluation s'est déroulée en trois étapes :

— Sélection des attributs grâce à des méthodes de filtres.
— Évaluation de la qualité de la détection en fonction de deux métriques.
— Interprétation des résultats en fonction de la couverture des anomalies par l'algorithme de détection basé sur notre ontologie des communications.

Nous avons démontré que l'algorithme HTM était capable de détecter toutes les anomalies avec un taux de faux positifs minimal de 0.04%. Nous avons comparé ces résultats avec ceux obtenus par un réseau de neurones non-supervisé autoencodeur-LSTM qui a démontré de meilleures capacités de détection en termes de couverture des anomalies avec un taux de faux positifs minimal de 0.08%.

4.2 Limites et Perspectives

Nous présentons dans cette section : les limites de notre méthode de détection, des améliorations possibles et enfin les conséquences de l'utilisation d'un système de détection d'anomalies dans le contexte des véhicules connectés.

4.2.1 Le corpus de données

Nous avons présenté un corpus de données issu d'une étude approfondie des services et applications pouvant être utilisés par les véhicules connectés. Le secteur automobile étant en constante évolution, il est nécessaire que d'autres jeux de données issus de partenariats avec l'industrie voient le jour afin que des travaux tels que ceux présentés ici puissent contribuer à la sécurité des véhicules connectés. Par exemple, l'ajout d'applications de mise à jour à distance, ou OTA-update (pour *over-the-air*-update) et d'autres services commerciaux existant pourraient contribuer au réalisme de notre corpus.

4.2.2 Le problème de la sélection des paramètres

Nous avons évalué **Svalinn** en fonction de nombreux paramètres comme la durée des *fenêtres de description instantanée* et les attributs de celles-ci. À cet effet, nous avons cherché à isoler les paramètres permettant la meilleure détection. Pour autant, les attributs ont été sélectionnés en fonction des anomalies que nous cherchions à détecter. Or, d'autres anomalies pourraient avoir un impact sur des attributs dont nous avons privé l'algorithme. Il serait donc intéressant non seulement de concevoir la détection sur l'ensemble des attributs disponibles, et aussi d'évaluer dynamiquement le poids que chaque attribut dans la classification par un mécanisme de renforcement. Ce renforcement pourrait venir de l'action d'un opérateur qui, à la suite de l'évaluation d'un rapport d'anomalie pourrait indiquer à l'algorithme si un attribut particulier a permis de détecter une véritable anomalie où s'il s'agissait d'un faux positif.

Se posent alors plusieurs problèmes. Premièrement, concernant la question de l'efficacité de la détection, nous avons montré que HTM était efficace sur un petit nombre d'attributs tandis que LSTM obtient de bons résultats lorsque de nombreux attributs sont utilisés. Pour utiliser la totalité des attributs disponibles avec l'algorithme HTM, il faut repenser le traitement des *fenêtres*. Par exemple, en isolant chaque attribut dans sa propre instance de détection et en évaluant les anomalies suivant un mécanisme de vote des multiples instances. Utiliser plus d'attributs entraîne un coût de traitement supplémentaire des *fenêtres* qu'il faut alors prendre en compte dans l'exécution du dispositif et notamment dans ses capacités de détection en temps réel. Pour pallier ce problème il faudrait notamment utiliser des fenêtres d'une durée suffisamment grande.

4.2.3 Évaluation des performances en temps réel

Notre évaluation s'est limitée à une étude des capacités de détection du dispositif et de la vitesse de traitement des *fenêtres* dans un environnement. Nous avons quantifié la durée moyenne de l'analyse d'une *fenêtre de description* par les algorithmes HTM et LSTM. Pour autant, nous n'avons pas évalué les performances en temps réel de l'intégralité de l'exécution du système à savoir :

— La capture des paquets
— La création des fenêtres
— L'extraction des attributs et l'inscription dans l'ontologie
— L'analyse des fenêtres
— La génération du rapport en cas d'anomalie

Une analyse de la durée de chacun de ces traitements permettrait d'évaluer les besoins matériels du dispositif pour son intégration au sein du véhicule. De plus, il serait possible de déterminer le nombre maximum d'attributs pouvant être utilisés pour la détection.

4.2.4 De la détection à la prévention

Nous avons présenté un dispositif capable de détecter plusieurs types d'anomalies avec un taux de faux positifs acceptable. Cependant, compte-tenu de l'impact potentiel d'attaques réussies sur la sûreté des véhicules et de leurs occupants il est important d'explorer des mécanismes de prévention des attaques par le canal de communication cellulaire. La prévention d'attaques dans les réseaux informatiques classiques est généralement assurée par les pare-feux. L'intérêt des dispositifs de détection réside dans la possibilité pour les opérateurs de dresser des signatures d'attaques ou de définir des indicateurs de compromissions pouvant ensuite être intégrés aux pare-feux.

Cette réaction après coup est évidemment désavantageuse puisque les attaques sur les systèmes d'informations ont un impact direct sur les entreprises ou sur les individus touchés. Le vol d'informations confidentielles ou de propriétés intellectuelles, ainsi que les nuisances résultant des attaques par déni-de-service, entraînent donc

des coûts non seulement financiers pour les entreprises, mais ternissent aussi leur image et leur réputation. Dans le contexte des véhicules connectés ces attaques peuvent aussi entraîner des dommages corporels aux usagers de la route. À ce titre, la réaction a postériori ne peut se suffire à elle seule. Pour palier cela, les dispositifs de détection à l'intérieur des véhicules pourraient être également dotés de mécanismes de prévention. Une attaque pourrait ainsi être stoppée à l'instant où celle-ci est détectée. Cependant, les systèmes de détection étant prompt à des faux positifs, une évaluation de leur impact sur la bonne exécution des services ITS serait alors nécessaire.

4.2.5 Travaux Futurs

Par conséquent, nos travaux ont démontré qu'il était possible d'utiliser des algorithmes non-supervisés pour détecter plusieurs types d'attaques dans les communications cellulaires de véhicules. L'approche que nous avons proposée avec **Svalinn**, reposant sur une représentation ontologique des communications, a montré de bonnes capacités de détection et une réduction de la complexité du traitement des anomalies grâce à la génération des rapports d'anomalies. Nous espérons que ces travaux encourageront la communauté à expérimenter d'autres approches sur des jeux de données améliorés. Ceux-ci pourraient prendre en compte l'évolution rapide des propriétés des communications cellulaires ainsi que d'autres menaces pesant sur les véhicules connectés. Enfin, nous espérons également que ces travaux encourageront les acteurs industriels à poursuivre leurs partenariats avec le monde académique et que des approches telles que celles que nous avons présentées pourront être utilisées afin d'améliorer la sécurité des véhicules connectés.

Annexe A

Attributs des fenêtres de description instantanée

Nom	Propriété	Attribut	Sens
total fpktl	Total	Taille des paquets	Montant
min fpktl	Minimum		
max fpktl	Taille		
mean fpktl	Moyenne		
var fpktl	Variance		
std fpktl	Écart type		
total fpackets	Total	Nombre de paquets	Montant
total bpackets	Total		Descendant
total flowpackets	Total		Les deux
total bpktl	Total	Taille des paquets	Descendant
min bpktl	Minimum		
max bpktl	Maximum		
mean bpktl	Moyenne		
var bpktl	Variance		
std bpktl	Écart type		
total fiat	Total	Durée entre l'émission de deux paquets	Montant
min fiat	Minimum		
max fiat	Maximum		
mean fiat	Moyenne		
var fiat	Variance		
std fiat	Écart type		
total biat	Total	Durée entre la réception de deux paquets	Descendant
min biat	Minimum		
max biat	Maximum		
mean biat	Moyenne		
var biat	Variance		
std biat	Écart type		
total flowpktl	Total	Taille des paquets	Les deux
min flowpktl	Minimum		
max flowpktl	Maximum		
mean flowpktl	Moyenne		
var flowpktl	Variance		
std flowpktl	Écart type		
total flowiat	Total	Durée entre l'émission de deux paquets	Les deux
min flowiat	Minimum		
max flowiat	Maximum		
mean flowiat	Moyenye		
var flowiat	Variance		
std flowiat	Écart type		
fpfkts ps	Total	Nombre de paquets par seconde	Montant
bpbkts ps	Total		Descendant
flowpkts ps	Total	Taille des paquets par seconde	
flowbytes ps	Total	Octets échangés par seconde	Les deux
downUpRatio	Ratio	Octets échangé Descendant/Montant	

Table A.1 – Attributs temporels des fenêtres de description instantanée.

Annexe B

Algorithme HTM

B.1 Vue d'ensemble

Nous rappelons que l'algorithme HTM est composé des éléments suivants :

— L'encodeur (ou *Encoder*) des données présentées en entrée à l'algorithme
— Un réseau de neurones composé d'un ensemble de mini-colonnes dans lesquelles résident plusieurs neurones pyramidaux [111].
— L'algorithme de la représentation spatiale (ou *Spatial Pooler*) et de la Mémoire Temporelle (ou *Temporal Memory*) qui manipulent les liens entre les neurones du réseau en fonction des entrées de l'algorithme.
— Un classifieur (ou *Classifier*) chargé de juger si une entrée est anormale ou non.

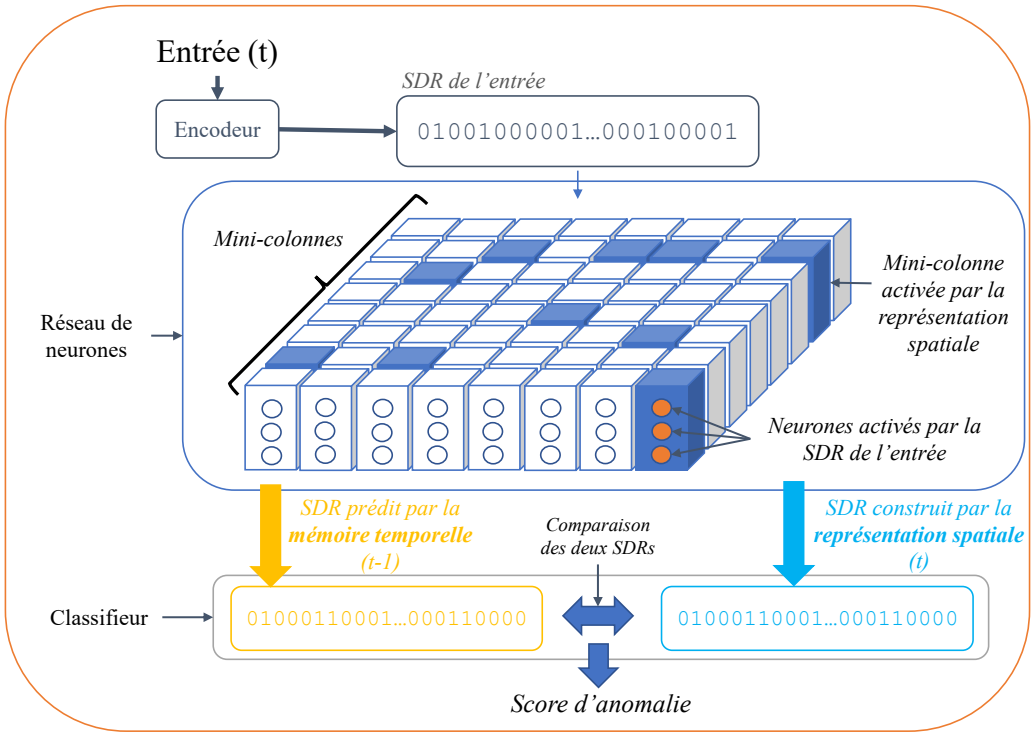


Figure B.1 – Représentation du fonctionnement de HTM

Nous rappelons également le fonctionnement général de l'algorithme à partir de la Figure B.1 déjà présentée dans la sous-section 2.4.2 :

1. À l'instant `(t)`, l'encodeur convertit les entrées présentées à l'algorithme en vecteurs binaires de taille fixe, aussi appelés **représentations distribuées éparses** (ou **SDR** pour *Sparse Distributed Representations*).
2. L'algorithme de la représentation spatiale, représenté en turquoise sur la figure, déclenche à partir de la SDR de l'entrée, l'activation d'une **petite quantité** de mini-colonnes du réseau de neurones. L'ensemble des mini-colonnes du réseau est représenté sous la forme d'une autre SDR où les bits actifs représentent les mini-colonnes actives à l'instant `(t)`.
3. L'algorithme de la mémoire temporelle, représenté en jaune sur la figure, apprend les séquences de l'activation des mini-colonnes et effectue des prédictions sur l'état futur du réseau. Il produit donc une SDR représentant les mini-colonnes dont l'activité à l'instant `(t)` a été prédite à l'instant `(t-1)`.
4. Ces prédictions sont ensuite présentées au classifieur. Dans notre cas, nous utilisons une fonction qui calcule le score d'anormalité d'une entrée en comparant la SDR produite par la représentation spatiale à l'instant `(t)` et la SDR prédite à l'instant `(t-1)` par l'algorithme de la mémoire temporelle.

Nous détaillons dans la suite de cette annexe les algorithmes de la représentation spatiale et de la mémoire temporelle ainsi que les paramètres qui les régissent.

B.2 L'algorithme de la représentation spatiale

Nous commençons donc par détailler le fonctionnement de l'algorithme de la représentation spatiale.

L'espace des entrées dans HTM est représenté par un vecteur dont la dimension est équivalente à celle des SDRs produites par l'encodeur.

L'algorithme de la représentation spatiale a pour rôle de déclencher l'activation des cellules du réseau de neurones lorsque celles-ci sont connectées à des bits actifs dans l'espace des entrées. Ce processus est décrit par la Figure B.2. Dans cette figure, l'espace des entrées est représenté sous la forme d'une matrice binaire destinée à recevoir les SDRs provenant de l'encodeur. Les bits actifs de la SDR sont représentés par des rectangles pleins. Les connexions proximales sont représentées par des flèches liant une cellule à plusieurs bits de l'espace des entrées. Les connexions proximales actives sont figurées par des flèches pleines tandis que celles connectées à des bits non actifs sont représentées en pointillés.

Lorsqu'une cellule est connectée à un nombre suffisant de bits actifs dans l'espace des données, elle est activée. Dans la figure, cela correspond à un cercle plein. Enfin, si un nombre suffisant de cellules sont actives dans une mini-colonne alors celle-ci est aussi activée, ce qui est représenté sous la forme d'une colonne pleine.

Seul un certain nombre de mini-colonnes peuvent être actives à un instant donné. En effet, un mécanisme d'inhibition est implémenté afin que les mini-colonnes ayant le plus de connexions actives avec l'espace des données soient favorisées.

L'algorithme utilise plusieurs mécanismes afin de renforcer, réduire, ou créer les connexions proximales entre les cellules et l'espace des entrées et peut être résumé

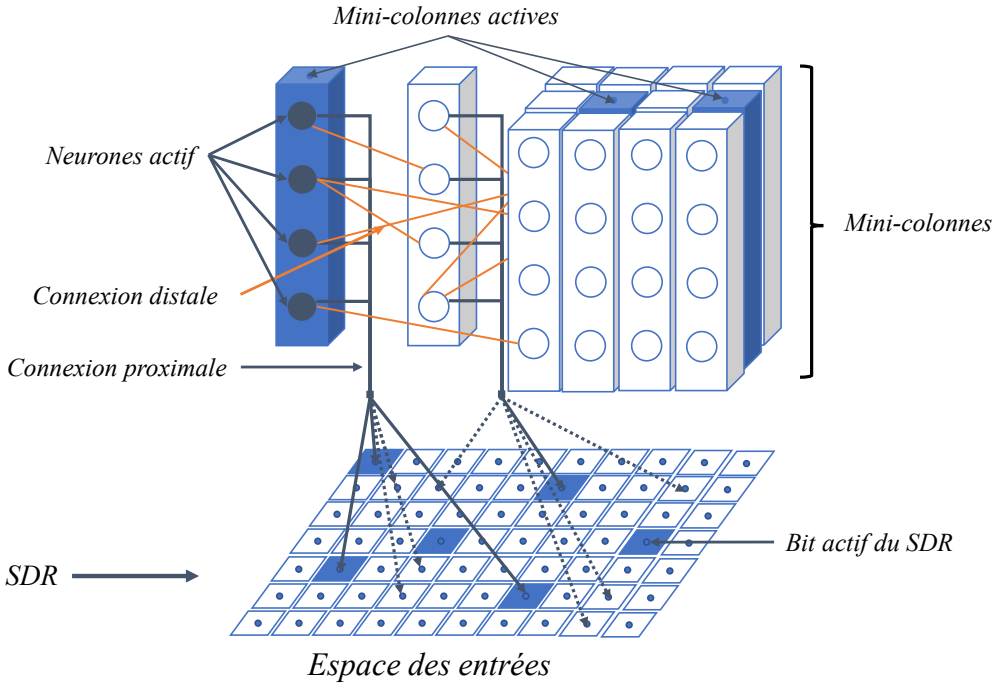


FIGURE B.2 – Représentation des composants de HTM

ainsi :

1. Sélectionner un nombre de bits pour l'espace des données.
2. Initialiser le réseau en affectant un nombre fixe de mini-colonnes à l'espace des données :
   — Chaque colonne dispose d'un ensemble de cellules.
   — Chaque cellule est connectée à un sous-ensemble aléatoire de l'espace des entrées grâce à des connexions proximales.
   — La force (ou permanence) de chacune de ces connexions est initialisée aléatoirement en fonction d'un seuil prédéterminé.
   — Si la force de ces connexions dépassent le seuil, cela signifie que certaines cellules seront déjà connectées à certains bits de l'espace des entrées.
3. Pour chaque entrée, déterminer le nombre de connexions actives par mini-colonne en fonction du nombre de connexions proximales liées à des bits actifs de l'espace des entrées.
4. Le nombre de connexions actives est multiplié par un facteur de «boosting» déterminé en fonction de la fréquence à laquelle une mini-colonne est activée par rapport aux colonnes voisines.
5. Une portion des mini-colonnes est activée si ces dernières obtiennent un score supérieur aux autres mini-colonnes dans la zone d'inhibition. Cette zone d'inhibition est déterminée en fonction de la répartition des bits actifs dans l'espace des entrées.

6. Pour chacune des mini-colonnes actives à l'issue des étapes précédentes (4 et 5) : la permanence des connexions proximales est mise à jour, les connexions établies avec un bit actif de la SDR d'entrée sont renforcées, alors que celles établies avec un bit non actif sont réduites. Si, après modification les permanences de certaines connexions dépassent le seuil prédéfini, les connexions en conséquence sont établies ; dans le cas inverse les connexions sont rompues.
7. Recommencer à l'opération 4 jusqu'à ce qu'il n'y ait plus de données à analyser.

B.3 L'algorithme de la mémoire temporelle

L'algorithme de la mémoire temporelle continue le processus d'apprentissage en manipulant les connexions distales des neurones du réseau. Celles-ci sont représentées par des traits bleus clairs sur la Figure B.2.

L'objectif de la mémoire temporelle consiste à apprendre les séquences d'activations des mini-colonnes déclenchées par la représentation spatiale. Pour ce faire, l'algorithme renforce, crée ou réduit les connexions distales des cellules, en fonction des mini-colonnes actives à un instant t et celles qu'il avait prédites à l'instant $t-1$.

L'algorithme peut donc être divisé en 4 blocs résumés ainsi :

1. Pour chaque mini-colonne :
   (a) Si la mini-colonne est active et contient des neurones dans un état prédictif (connexion distale active).
   — Traitement : bonne prédiction
   (b) Si la mini-colonne est active mais ne contient aucun neurone dans l'état prédictif.
   — Traitement : mauvaise prédiction
   (c) Si la mini-colonne est inactive mais contient au moins un neurone dans un état prédictif.
   — Traitement : prédiction de la mauvaise mini-colonne
2. Prédire les cellules actives à l'instant $t+1$.

a) Traitement bonne prédiction – Le premier cas correspond à une situation dans laquelle l'algorithme a prédit correctement à l'instant $t-1$ qu'une partie des mini-colonnes serait activée par l'entrée de l'instant t. Dans cette situation l'algorithme procède aux opérations suivantes :

1. Les neurones dans un état prédictif sont marqués en tant que gagnants de l'instant t
2. Les liens distaux de ces neurones avec les autres cellules ayant contribué à leur activation sont renforcés tandis que ceux n'y ayant pas contribué sont réduits.

3. Si le nombre de liens actifs décroît du fait que la permanence de certains d'entre eux passe en dessous du seuil, de nouveaux liens sont alors créés entre ces cellules et celles marquées comme gagnantes à l'instant $t-1$.

b) Traitement mauvaise prédiction – Le second cas correspond à une situation dans laquelle l'algorithme ne s'attendait pas à l'instant $t-1$ à ce que la mini-colonne soit activée par l'entrée de l'instant t. Dans cette situation, l'algorithme procède aux opérations suivantes :

1. Toutes les cellules de la mini-colonne sont activées.
2. S'il existe des neurones en état prédictif dans cette mini-colonne.
   — Marquer le neurone ayant le plus de connexions distales en état prédictif, comme gagnant de l'instant t.
3. Sinon :
   — Sélectionner le neurone ayant le moins de connexions distales comme gagnant de l'instant t.
4. Renforcer les connexions distales ayant contribué à l'activation partielle du neurone.
5. Réduire les connexions distales n'ayant pas contribué à l'activation du neurone.
6. En fonction du nombre de connexions déjà établies, créer de nouveaux liens entre les cellules et celles marquées comme gagnantes à l'instant $t-1$.

c) Traitement prédiction de la mauvaise mini-colonne – Le dernier cas correspond à une situation dans laquelle une mini-colonne n'a pas été activée à l'instant t par l'algorithme de la représentation spatiale alors qu'elle avait été prédite à l'instant $t-1$ par la mémoire temporelle. Dans cette situation l'algorithme se contente de réduire la permanence des connexions distales des cellules actives de la mini-colonne.

– **Prédiction à l'instant t+1** – Dans tous les cas la dernière étape de l'algorithme consiste à évaluer les neurones dont la probabilité d'être activé à l'instant $t+1$ est la plus forte. Pour ce faire, l'algorithme parcourt l'ensemble des connexions distales de chaque neurone. Si un neurone est connecté à un nombre suffisant de neurones actifs à l'instant t, alors il passe en état prédictif pour l'instant $t+1$.

B.4 Les paramètres de la représentation spatiale

L'agrégat spatial est régi par de nombreux paramètres affectant sa taille, la taille des entrées ainsi que son comportement vis à vis des connexions établies entre l'espace des entrées et les cellules des mini-colonnes du réseau. Les paramètres sont les suivants :

— Le nombre de mini-colonnes de l'agrégat spatial ;

  - Plus l'agrégat spatial est grand, plus le nombre théorique de connexions et de relations possibles entre chacune des valeurs qui lui sont présentées sera grand. Cependant, le temps de traitement de chaque entrée est lui aussi augmenté avec la taille de l'agrégat.
- La taille des entrées traitées par l'agrégat spatial ;
  - Cette valeur influe sur la taille de la SDR de l'entrée de l'algorithme. Plus celle-ci sera grande, plus la SDR sera capable de représenter un grand nombre de données différentes. En contrepartie, une taille excessive participe au ralentissement du traitement de chaque entrée. De plus, si cet espace est trop grand, il est alors possible que les SDR ne soient plus en mesure de représenter correctement les données dont la sémantique est similaire.
- L'inhibition globale ;
  - Ce paramètre est un booléen qui permet de modifier le comportement de l'algorithme dans l'activation des mini-colonnes. S'il est actif, les mini-colonnes les plus actives seront sélectionnées en fonction de l'ensemble des mini-colonnes de l'agrégat spatial. S'il ne l'est pas, alors cette même inhibition des mini-colonnes est réalisée en fonction des mini-colonnes du proche voisinage. Ce voisinage est déterminé en fonction de la taille de l'agrégat, du nombre des connexions disponibles pour chaque mini-colonne ainsi que du nombre de mini-colonnes pouvant être rendues actives dans l'agrégat spatial. Les performances de l'algorithme en termes de vitesse d'exécution sont grandement impactées par ce paramètre. Il est donc recommandé pour des cas d'applications ne manipulant pas des données spatiales telles que des images ou vidéos de conserver ce paramètre actif.
- Le nombre de mini-colonnes pouvant être rendues actives dans l'agrégat spatial :
  - Cette valeur détermine le nombre de mini-colonnes qui seront activées à chaque itération du jeu de données. Il est recommandé par les concepteurs de cet algorithme de maintenir 2% de mini-colonnes actives à chaque itération.
- Le ratio de connexions synaptiques potentielles entre les mini-colonnes et l'espace des entrées :
  - Chaque mini-colonne de l'agrégat peut être connectée potentiellement à l'ensemble des bits de l'espace des entrées. Ce ratio permet de limiter ce nombre de connexions potentielles entraînant une réduction de la superposition des activations des mini-colonnes entre elles.
- Le seuil de connexion des synapses :
  - Ce seuil permet de déterminer le nombre minimal de connexions proximales actives qu'une mini-colonne doit avoir pour être déclenchée. Ce seuil permet d'empêcher au bruit dans les SDRs d'activer les colonnes.
- L'incrément/décrément destiné à consolider/défaire les connexions proximales :

  - A chaque itération les connexions proximales actives ou inactives sont consolidées en fonction de ce facteur, ce qui permet à l'algorithme d'apprendre mais aussi d'oublier des associations entre les différents SDRs du jeu de données.
- Le renforcement (boosting) :
  - Le boosting met en compétition les mini-colonnes pour leur activation. En effet, puisque le nombre de mini-colonnes actives par itération est limité, il peut arriver que des mini-colonnes entrent peu en activité. Après la phase d'inhibition, le boosting permet à des colonnes inactives d'entrer quand même en activité en boostant la valeur des connexions synaptiques qu'elles ont établi avec certains bits de la SDR. Ce processus permet d'une part de favoriser l'utilisation de toutes les mini-colonnes de l'agrégat en évitant ainsi à des mini-colonnes de rentrer en activité trop fréquemment. D'autre part, il permet aussi à des mini-colonnes d'apprendre de nouveaux motifs dans les données en forçant leur activation sur d'autres entrées.

B.5 Les paramètres de la mémoire temporelle

- Le nombre de cellules par mini-colonne :
  - Cette valeur détermine la profondeur des mini-colonnes. Pour chaque cellule supplémentaire, un ensemble de synapses potentielles est créé, permettant ainsi à la mini-colonne de s'activer sur plus de motifs différents dans les données d'entrées.
- Le nombre maximal de synapses ajoutées lors de l'apprentissage pour une cellule :
  - À l'initialisation de l'algorithme des synapses sont créées pour chacune des cellules des mini-colonnes. Cette valeur permet d'autoriser les cellules à créer de nouvelles connexions avec d'autres motifs présentés en entrée.
- La valeur d'initialisation du seuil de permanence des nouvelles synapses :
  - Cette valeur définit la force de la connexion distale de chaque nouvelle synapse créée.
- Les valeurs d'augmentation et de réduction de la permanence :
  - Pour chaque prédiction réalisée par la mémoire temporelle, lorsque celle-ci se trouve être vérifiée, les connexions sont renforcées, dans le cas contraire elles sont fragilisées.
- Le nombre minimal de synapses actives pour l'activation d'un segment :
  - Si une cellule dispose de suffisamment de ses connexions activées alors elle est sélectionnée pour le renforcement de sa connexion distale avec les cellules activées à l'itération précédente.
- Le seuil d'activation d'un segment :
  - Le nombre requis de connexions distales activées pour considérer le segment comme actif.

B.6 Hyper-paramètres utilisés dans notre étude

```
{  'model': 'HTMPrediction',
   'modelParams': {
     'clParams': { 'alpha': 0.01962508905154251,
       'regionName': 'SDRClassifierRegion',
       'steps': '1,5'},
     'inferenceType': 'TemporalAnomaly',
     'spEnable': True,
     'spParams': { 'columnCount': 2048,
       'globalInhibition': 1,
       'inputWidth': 0,
       'boostStrength': 2.0,
       'numActiveColumnsPerInhArea': 40,
       'potentialPct': 0.8,
       'seed': 1956,
       'spVerbosity': 0,
       'spatialImp': 'cpp',
       'synPermActiveInc': 0.05,
       'synPermConnected': 0.1,
       'synPermInactiveDec': 0.08568228006654939},
     'tmEnable': True,
     'tmParams': { 'activationThreshold': 12,
       'cellsPerColumn': 32,
       'columnCount': 2048,
       'globalDecay': 0.0,
       'initialPerm': 0.21,
       'inputWidth': 2048,
       'maxAge': 0,
       'maxSegmentsPerCell': 128,
       'maxSynapsesPerSegment': 32,
       'minThreshold': 10,
       'newSynapseCount': 20,
       'outputType': 'normal',
       'pamLength': 1,
       'permanenceDec': 0.1,
       'permanenceInc': 0.1,
       'seed': 1960,
       'temporalImp': 'cpp',
       'verbosity': 0},
     'trainSPNetOnlyIfRequested': False},
}
```

Annexe C

Répartition du nombre de fenêtres

Type de fenêtre	Durée	DNS	XMAS-Scan	Syn-Scan	Télémétrie	Normal	Total
Flux	1	235	1030	1022	10	16419	18716
	2	118	1030	1022	9	8658	10837
	3	79	1030	1022	9	6053	8193
	5	48	1030	1022	9	3954	6063
	10	24	1030	1022	9	2375	4460
	15	16	1030	1022	9	1842	3919
	20	12	1030	1022	9	1579	3652
	30	8	1030	1022	9	1302	3371
	60	4	1030	1022	8	915	2979
	120	2	1030	1022	4	620	2678
	300	2	1030	1022	2	378	2434
Conversation	1	235	4	4	10	16284	16537
	2	118	3	2	9	8551	8683
	3	79	2	2	9	5945	6037
	5	48	2	2	9	3846	3907
	10	24	2	2	9	2263	2300
	15	16	2	2	9	1726	1755
	20	12	2	2	9	1462	1487
	30	8	2	2	9	1188	1209
	60	4	2	2	8	806	822
	120	2	2	2	4	513	523
	300	2	2	2	2	272	280
Lien	1	235	3	3	10	16274	16525
	2	118	2	2	9	8548	8679
	3	79	1	1	9	5945	6035
	5	48	1	2	9	3834	3894
	10	24	1	2	9	2250	2286
	15	16	1	1	9	1711	1738
	20	12	1	1	9	1448	1471
	30	8	1		9	1172	1190
	60	4	1		8	789	802
	120	2	1		4	495	502
	300	2	1		2	220	225

TABLE C.1 – Répartition du nombre de *fenêtres* en fonction de leur type et de leur durée ainsi que des classes d'anomalies (Tunnel-DNS, XMAS-Scan, Syn-Scan, télémétrie et normal).

Bibliographie

[1] Controlling vehicle features of nissan leafs across the globe via vulnerable apis. URL `https://www.troyhunt.com/controlling-vehicle-features-of-nissan/`. (Cité en page 30.)

[2] Beemer, Open Thyself! – Security vulnerabilities in BMW's Connected Drive. URL `https://www.heise.de/ct/artikel/Beemer-Open-Thyself-Security-vulnerabilities-in-BMW-s-ConnectedDrive-2540957.html`. (Cité en page 30.)

[3] M. N. Adnan et M. Z. Islam : Forest pa : Constructing a decision forest by penalizing attributes used in previous trees. *Expert Systems with Applications*, 89:389–403, 2017. (Cité en pages 24 et 25.)

[4] R. Agrawal, T. Imieliński et A. Swami : Mining association rules between sets of items in large databases. *In Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, p. 207–216, 1993. (Cité en page 21.)

[5] S. Ahmad et J. Hawkins : Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv :1503.07469*, 2015. (Cité en pages 60 et 63.)

[6] S. Ahmad, A. Lavin, S. Purdy et Z. Agha : Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017. (Cité en pages 56, 98, 99 et 100.)

[7] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour et H. Janicke : A novel hierarchical intrusion detection system based on decision tree and rules-based models. *In 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, p. 228–233. IEEE, 2019. (Cité en page 24.)

[8] S. A. Al Mamun et J. Valimaki : Anomaly detection and classification in cellular networks using automatic labeling technique for applying supervised learning. *Procedia Computer Science*, 140:186–195, 2018. (Cité en page 36.)

[9] I. Almomani, B. Al-Kasasbeh et M. Al-Akhras : Wsn-ds : A dataset for intrusion detection systems in wireless sensor networks. *Journal of Sensors*, 2016, 2016. (Cité en pages 19 et 26.)

[10] M. Aloqaily, S. Otoum, I. Al Ridhawi et Y. Jararweh : An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90:101842, 2019. (Cité en page 36.)

[11] D. Ariu, R. Tronci et G. Giacinto : Hmmpayl : An intrusion detection system based on hidden markov models. *computers & security*, 30(4):221–241, 2011. (Cité en page 19.)

[12] AWID : Awid-wireless security datasets project. URL `http://icsdweb.aegean.gr/awid/features.html`. (Cité en page 25.)

[13] L. E. Baum, T. Petrie, G. Soules et N. Weiss : A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970. (Cité en page 19.)

[14] A. Bazzi, B. M. Masini et A. Zanella : Performance analysis of v2v beaconing using lte in direct mode with full duplex radios. *IEEE Wireless Communications Letters*, 4(6):685–688, 2015. (Cité en page 29.)

[15] A. Bazzi, B. M. Masini, A. Zanella et I. Thibault : Beaconing from connected vehicles : Ieee 802.11 p vs. lte-v2v. *In 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, p. 1–6. IEEE, 2016. (Cité en page 29.)

[16] R. Berthier et W. H. Sanders : Specification-based intrusion detection for advanced metering infrastructures. *In 2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing*, p. 184–193. IEEE, 2011. (Cité en page 16.)

[17] P. Borazjani, C. Everett et D. McCoy : Octane : An extensible open source car security testbed. *In Proceedings of the Embedded Security in Cars Conference*, 2014. (Cité en page 34.)

[18] Bosch : Specification can. 1991. (Cité en page 13.)

[19] E. Bou-Harb, M. Debbabi et C. Assi : Cyber scanning : a comprehensive survey. *Ieee communications surveys & tutorials*, 16(3):1496–1519, 2013. (Cité en page 95.)

[20] R. Buschkes, D. Kesdogan et P. Reichl : How to increase security in mobile networks by anomaly detection. *In Proceedings 14th Annual Computer Security Applications Conference (Cat. No. 98EX217)*, p. 3–12. IEEE, 1998. (Cité en page 16.)

[21] Z. Cai, A. Wang, W. Zhang, M. Gruffke et H. Schweppe : 0-days & mitigations : Roadways to exploit and secure connected bmw cars. *Black Hat USA*, 2019:39, 2019. (Cité en page 32.)

[22] P. Casas, J. Mazel et P. Owezarski : Unada : Unsupervised network anomaly detection using sub-space outliers ranking. *In International Conference on Research in Networking*, p. 40–51. Springer, 2011. (Cité en page 23.)

[23] G.-Y. Chan, F.-F. Chua et C.-S. Lee : Intrusion detection and prevention of web service attacks for software as a service : Fuzzy association rules vs fuzzy associative patterns. *Journal of Intelligent & Fuzzy Systems*, 31(2):749–764, 2016. (Cité en page 22.)

[24] V. Chandola, A. Banerjee et V. Kumar : Anomaly detection : A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009. (Cité en page 17.)

[25] D. Chandramouli, R. Liebhart et J. Pirskanen : *5G for the Connected World*. Wiley Online Library, 2019. (Cité en page 29.)

[26] G. Chandrashekar et F. Sahin : A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014. (Cité en page 102.)

[27] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.* : Comprehensive experimental analyses of automotive attack surfaces. (Cité en page 28.)

[28] D. Chicco et G. Jurman : The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020. (Cité en page 109.)

[29] Computest : The connected car-ways to get unauthorized access and potential implications. *Online : https ://www.computest.nl/wp-content/uploads/2018/04/connected-car-rapport.pdf*, 2018. (Cité en page 32.)

[30] A. Computing *et al.* : An architectural blueprint for autonomic computing. 2006. (Cité en page 71.)

[31] G. Creech : *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.* Thèse de doctorat. (Cité en page 25.)

[32] J. Cui, L. S. Liew, G. Sabaliauskaite et F. Zhou : A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90:101823, 2019. (Cité en pages 15 et 33.)

[33] Y. Cui, S. Ahmad et J. Hawkins : Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, 28(11):2474–2504, 2016. (Cité en page 56.)

[34] Y. Cui, S. Ahmad et J. Hawkins : The htm spatial pooler—a neocortical algorithm for online sparse distributed coding. *Frontiers in computational neuroscience*, 11:111, 2017. (Cité en page 57.)

[35] F. D. Da Cunha, A. Boukerche, L. Villas, A. C. Viana et A. A. Loureiro : Data communication in vanets : a survey, challenges and applications. 2014. (Cité en page 14.)

[36] Z. Deng, X. Zhu, D. Cheng, M. Zong et S. Zhang : Efficient knn classification algorithm for big data. *Neurocomputing*, 195:143–148, 2016. (Cité en page 55.)

[37] M. Dibaei, X. Zheng, K. Jiang, S. Maric, R. Abbas, S. Liu, Y. Zhang, Y. Deng, S. Wen, J. Zhang *et al.* : An overview of attacks and defences on intelligent connected vehicles. *arXiv preprint arXiv :1907.07455*, 2019. (Cité en page 33.)

[38] J. R. Douceur : The sybil attack. *In International Workshop on Peer-to-Peer Systems*, p. 251–260. Springer, 2002. (Cité en page 29.)

[39] J. Dromard, G. Roudière et P. Owezarski : Online and scalable unsupervised network anomaly detection method. *IEEE Transactions on Network and Service Management*, 14(1):34–47, 2016. (Cité en page 23.)

[40] R. C. Eberhart et Y. Shi : Comparing inertia weights and constriction factors in particle swarm optimization. *In Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, p. 84–88. IEEE, 2000. (Cité en page 98.)

[41] M. H. Eiza et Q. Ni : Driving with sharks : Rethinking connected vehicles with vehicle cybersecurity. *IEEE Vehicular Technology Magazine*, 12(2):45–51, 2017. (Cité en page 37.)

[42] G. Fernandes Jr, L. F. Carvalho, J. J. Rodrigues et M. L. Proença Jr : Network anomaly detection using ip flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications*, 64:1–11, 2016. (Cité en page 55.)

[43] G. D. Forney : The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. (Cité en page 19.)

[44] I. Foster, A. Prudhomme, K. Koscher et S. Savage : Fast and vulnerable : A story of telematic failures. *In 9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015. (Cité en page 29.)

[45] S. R. Gaddam, V. V. Phoha et 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*, 19(3):345–354, 2007. (Cité en page 23.)

[46] M. Gates, A. Tirumala, J. Ferguson, J. Dugan, F. Qin, K. Gibbs et J. Estabrook : Iperf. *Website : http ://dast. nlanr. net/Projects/Iperf (August 2006)*, 2003. (Cité en page 81.)

[47] P. Geurts, D. Ernst et L. Wehenkel : Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006. (Cité en page 102.)

[48] I. Goodfellow, Y. Bengio et A. Courville : *Deep learning*. MIT press, 2016. (Cité en page 55.)

[49] B. Greenshields, J. Bibbins, W. Channing et H. Miller : A study of traffic capacity. *In Highway research board proceedings*, vol. 1935. National Research Council (USA), Highway Research Board, 1935. (Cité en page 35.)

[50] T. R. Gruber : A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993. (Cité en page 67.)

[51] Z. Hasani : Robust anomaly detection algorithms for real-time big data : Comparison of algorithms. *In 2017 6th Mediterranean Conference on Embedded Computing (MECO)*, p. 1–6. IEEE, 2017. (Cité en page 56.)

[52] H. Hasrouny, A. E. Samhat, C. Bassil et A. Laouiti : Vanet security challenges and solutions : A survey. *Vehicular Communications*, 7:7–20, 2017. (Cité en page 29.)

[53] D. M. Hawkins : The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004. (Cité en page 101.)

[54] J. Hawkins et S. Ahmad : Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:23, 2016. (Cité en page 61.)

[55] J. Hawkins et S. Blakeslee : *On intelligence : How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan, 2007. (Cité en pages 34 et 56.)

[56] S. Hochreiter et J. Schmidhuber : Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. (Cité en page 35.)

[57] A. Javaid, Q. Niyaz, W. Sun et M. Alam : A deep learning approach for network intrusion detection system. *In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, p. 21–26, Brussels, BEL, 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 9781631901003. URL `https://doi.org/10.4108/eai.3-12-2015.2262516`. (Cité en page 27.)

[58] A. Jurgelionis, J.-P. Laulajainen, M. Hirvonen et A. I. Wang : An empirical study of netem network emulation functionalities. *In 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, p. 1–6. IEEE, 2011. (Cité en page 82.)

[59] M.-J. Kang et J.-W. Kang : Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6), 2016. (Cité en page 34.)

[60] G. K. Karagiannidis et A. S. Lioumpas : An improved approximation for the gaussian q-function. *IEEE Communications Letters*, 11(8):644–646, 2007. (Cité en page 100.)

[61] KDD99 : KDD'99 Competition. URL `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. (Cité en pages 18, 21, 22, 24, 25, 26, 27, 48 et 78.)

[62] J. O. Kephart et D. M. Chess : The vision of autonomic computing. *Computer*, (1):41–50, 2003. (Cité en page 71.)

[63] L. Khan, M. Awad et B. Thuraisingham : A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB journal*, 16(4):507–521, 2007. (Cité en page 25.)

[64] L. Koc, T. A. Mazzuchi et S. Sarkani : A network intrusion detection system based on a hidden naïve bayes multiclass classifier. *Expert Systems with Applications*, 39(18):13492–13500, 2012. (Cité en page 20.)

[65] N. Kouiroukidis et G. Evangelidis : The effects of dimensionality curse in high dimensional knn search. *In 2011 15th Panhellenic Conference on Informatics*, p. 41–45. IEEE, 2011. (Cité en page 55.)

[66] D. Krajzewicz, J. Erdmann, M. Behrisch et L. Bieker : Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012. (Cité en pages 35 et 79.)

[67] A. Lang, J. Dittmann, S. Kiltz et T. Hoppe : Future perspectives : The car and its ip-address–a potential safety and security risk assessment. *In International Conference on Computer Safety, Reliability, and Security*, p. 40–53. Springer, 2007. (Cité en page 15.)

[68] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun et A. A. Ghorbani : Characterization of tor traffic using time based features. *In ICISSP*, p. 253–262, 2017. (Cité en pages 48 et 49.)

[69] M. Levi, Y. Allouche et A. Kontorovich : Advanced analytics for connected car cybersecurity. *In 2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, p. 1–7. IEEE, 2018. (Cité en page 35.)

[70] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba et K. Das : The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000. (Cité en page 20.)

[71] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham *et al.* : Evaluating intrusion detection systems : The 1998 darpa

off-line intrusion detection evaluation. *In Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, p. 12–26. IEEE, 2000. (Cité en page 23.)

[72] R. Lübke, P. Büschel, D. Schuster et A. Schill : Measuring accuracy and performance of network emulators. *In 2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, p. 63–65. IEEE, 2014. (Cité en page 82.)

[73] Z. MacHardy, A. Khan, K. Obana et S. Iwashina : V2x access technologies : Regulation, research, and remaining challenges. *IEEE Communications Surveys & Tutorials*, 20(3):1858–1877, 2018. (Cité en page 14.)

[74] B. W. Matthews : Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975. (Cité en page 109.)

[75] S. Mazloom, M. Rezaeirad, A. Hunter et D. McCoy : A security analysis of an in-vehicle infotainment and app platform. (Cité en page 30.)

[76] C. Miller et C. Valasek : Adventures in automotive networks and control units. *DEF CON*, 21:260–264, 2013. (Cité en page 31.)

[77] C. Miller et C. Valasek : A survey of remote automotive attack surfaces. *black hat USA*, 2014, 2014. (Cité en page 28.)

[78] C. Miller et C. Valasek : Remote exploitation of an unaltered passenger vehicle. 2015. (Cité en pages 31 et 94.)

[79] A. H. Mirza et S. Cosan : Computer network intrusion detection using sequential lstm neural networks autoencoders. *In 2018 26th Signal Processing and Communications Applications Conference (SIU)*, p. 1–4. IEEE, 2018. (Cité en page 55.)

[80] D. Moon, H. Im, I. Kim et J. H. Park : Dtb-ids : an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. *The Journal of supercomputing*, 73(7):2881–2895, 2017. (Cité en page 23.)

[81] T. K. Moon : The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996. (Cité en page 19.)

[82] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu et R. Reddi : A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88–103, 2011. (Cité en page 20.)

[83] N. Moustafa et J. Slay : Unsw-nb15 : a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *In 2015 military*

communications and information systems conference (MilCIS), p. 1–6. IEEE, 2015. (Cité en pages 19, 26 et 49.)

[84] G. Münz, S. Li et G. Carle : Traffic anomaly detection using k-means clustering. *In GI/ITG Workshop MMBnet*, p. 13–14, 2007. (Cité en page 22.)

[85] K. Nasr, A. Abou-El Kalam et C. Fraboul : Performance analysis of wireless intrusion detection systems. *In International Conference on Internet and Distributed Computing Systems*, p. 238–252. Springer, 2012. (Cité en page 109.)

[86] J. C. Neumann : *The book of GNS3 : build virtual network labs using Cisco, Juniper, and more*. No Starch Press, 2015. (Cité en page 81.)

[87] NHTSA : Cybersecurity Best Practices for Modern Vehicles. URL `https://www.nhtsa.gov/staticfiles/nvs/pdf/812333_CybersecurityForModernVehicles.pdf`. (Cité en page 33.)

[88] S. Nie, L. Liu et Y. Du : Free-fall : Hacking tesla from wireless to can bus. . (Cité en page 31.)

[89] S. Nie, L. Liu, Y. Du et W. Zhang : Over-the-air : how we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars, . (Cité en page 31.)

[90] N. F. Noy, D. L. McGuinness *et al.* : Ontology development 101 : A guide to creating your first ontology. (Cité en page 67.)

[91] L. Nussbaum et O. Richard : A comparative study of network link emulators. *In Proceedings of the 2009 Spring Simulation Multiconference*, p. 85. Society for Computer Simulation International, 2009. (Cité en page 82.)

[92] ONTIC : ONTIC project. URL `https://cordis.europa.eu/docs/projects/cnect/3/619633/080/deliverables/001-ONTICD432017021010final.pdf`. (Cité en page 23.)

[93] P. Papadimitratos : "on the road"-reflections on the security of vehicular communication systems. *In 2008 IEEE International Conference on Vehicular Electronics and Safety*, p. 359–363. IEEE, 2008. (Cité en page 15.)

[94] S. Parkinson, P. Ward, K. Wilson et J. Miller : Cyber threats facing autonomous and connected vehicles : Future challenges. *IEEE transactions on intelligent transportation systems*, 18(11):2898–2915, 2017. (Cité en page 37.)

[95] K. Peng, V. Leung, L. Zheng, S. Wang, C. Huang et T. Lin : Intrusion detection system based on decision tree over big data in fog environment. *Wireless Communications and Mobile Computing*, 2018, 2018. (Cité en page 24.)

[96] J. Petit et S. E. Shladover : Potential cyberattacks on automated vehicles. *IEEE Transactions on Intelligent transportation systems*, 16(2):546–556, 2014. (Cité en page 15.)

[97] S. Purdy : Encoding data for htm systems. *arXiv preprint arXiv :1602.05925*, 2016. (Cité en pages 58 et 59.)

[98] M. Raya et J.-P. Hubaux : Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007. (Cité en page 29.)

[99] Q. Ricard et P. Owezarski : Autobot : An emulation environment for cellular vehicular communications. *In Proceedings of the 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2019. (Cité en page 8.)

[100] Q. Ricard et P. Owezarski : Ontology based anomaly detection for cellular vehicular communications. *In 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020. (Cité en page 8.)

[101] G. F. Riley et T. R. Henderson : The ns-3 network simulator. *In Modeling and tools for network simulation*, p. 15–34. Springer, 2010. (Cité en pages 36 et 79.)

[102] G. Roudière et P. Owezarski : Evaluating the impact of traffic sampling on aatac's ddos detection. *In Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*, p. 27–32, 2018. (Cité en page 107.)

[103] S. L. Salzberg : C4. 5 : Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993, 1994. (Cité en page 25.)

[104] I. Sharafaldin, A. H. Lashkari et A. A. Ghorbani : Toward generating a new intrusion detection dataset and intrusion traffic characterization. *In ICISSP*, p. 108–116, 2018. (Cité en pages 19, 24, 25, 26, 48 et 78.)

[105] A. Shiravi, H. Shiravi, M. Tavallaee et A. A. Ghorbani : Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012. (Cité en pages 35 et 79.)

[106] J. Sliwinski, A. Beben et P. Krawiec : Empath : Tool to emulate packet transfer characteristics in ip network. *In International Workshop on Traffic Monitoring and Analysis*, p. 46–58. Springer, 2010. (Cité en page 82.)

[107] M. Sokolova, N. Japkowicz et S. Szpakowicz : Beyond accuracy, f-score and roc : a family of discriminant measures for performance evaluation. *In Australasian joint conference on artificial intelligence*, p. 1015–1021. Springer, 2006. (Cité en page 107.)

[108] R. Sommer et V. Paxson : Outside the closed world : On using machine learning for network intrusion detection. *In Security and Privacy (SP), 2010 IEEE Symposium on*, p. 305–316. IEEE, 2010. (Cité en pages 65 et 77.)

[109] H. M. Song, H. R. Kim et H. K. Kim : Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. *In 2016 international conference on information networking (ICOIN)*, p. 63–68. IEEE, 2016. (Cité en page 33.)

[110] J. Song, H. Takakura et Y. Okabe : Cooperation of intelligent honeypots to detect unknown malicious codes. *In 2008 WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, p. 31–39. IEEE, 2008. (Cité en pages 26 et 49.)

[111] N. Spruston : Pyramidal neurons : dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221, 2008. (Cité en pages 57 et 137.)

[112] K. Stefanidis et A. G. Voyiatzis : An hmm-based anomaly detection approach for scada systems. *In IFIP International Conference on Information Security Theory and Practice*, p. 85–99. Springer, 2016. (Cité en page 20.)

[113] V. Sze, Y.-H. Chen, T.-J. Yang et J. S. Emer : Efficient processing of deep neural networks : A tutorial and survey. *Proceedings of the IEEE*, 105 (12):2295–2329, 2017. (Cité en page 26.)

[114] A. Tajbakhsh, M. Rahmati et A. Mirzaei : Intrusion detection using fuzzy association rules. *Applied Soft Computing*, 9(2):462–469, 2009. (Cité en page 21.)

[115] M. Tavallaee, E. Bagheri, W. Lu et A. A. Ghorbani : A detailed analysis of the kdd cup 99 data set. *In 2009 IEEE symposium on computational intelligence for security and defense applications*, p. 1–6. IEEE, 2009. (Cité en pages 18, 26, 27, 36 et 48.)

[116] A. Theissler : Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. *Knowledge-Based Systems*, 123:163–173, 2017. (Cité en page 34.)

[117] S. Ustebay, Z. Turgut et M. A. Aydin : Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. *In 2018 international congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, p. 71–76. IEEE, 2018. (Cité en page 25.)

[118] D. Van Ravenzwaaij, P. Cassey et S. D. Brown : A simple introduction to markov chain monte–carlo sampling. *Psychonomic bulletin & review*, 25 (1):143–154, 2018. (Cité en page 19.)

[119] A. Varga et R. Hornig : An overview of the omnet++ simulation environment. *In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 60.

ICST (Institute for Computer Sciences, Social-Informatics and . . ., 2008. (Cité en pages 35 et 79.)

[120] R. Vijayanand, D. Devaraj et B. Kannapiran : Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Computers & Security*, 77: 304–314, 2018. (Cité en page 25.)

[121] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat et S. Venkatraman : Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550, 2019. (Cité en page 26.)

[122] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu et X. Cheng : A distributed anomaly detection system for in-vehicle network using htm. *IEEE Access*, 6:9091–9098, 2018. (Cité en pages 34 et 56.)

[123] K. Wevers et M. Lu : V2x communication for its-from ieee 802.11 p towards 5g. *IEEE 5G Tech Focus*, 1(2), 2017. (Cité en page 29.)

[124] S. Woo, H. J. Jo et D. H. Lee : A practical wireless attack on the connected car and security protocol for in-vehicle can. *IEEE Transactions on intelligent transportation systems*, 16(2):993–1006, 2014. (Cité en page 30.)

[125] J. Wu, W. Zeng et F. Yan : Hierarchical temporal memory method for time-series-based anomaly detection. *Neurocomputing*, 273:535–546, 2018. (Cité en page 57.)

[126] Z. Xu, X. Li, X. Zhao, M. H. Zhang et Z. Wang : Dsrc versus 4g-lte for connected vehicle applications : A study on field experiments of vehicular communication performance. *Journal of Advanced Transportation*, 2017, 2017. (Cité en page 29.)

[127] C. Yin, Y. Zhu, J. Fei et X. He : A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017. (Cité en page 26.)

[128] E. Yudkowsky *et al.* : Artificial intelligence as a positive and negative factor in global risk. *Global catastrophic risks*, 1(303):184, 2008. (Cité en page 37.)

[129] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan et M. Rajarajan : Host-based intrusion detection for vanets : a statistical approach to rogue node detection. *IEEE transactions on vehicular technology*, 65(8):6703–6714, 2015. (Cité en page 35.)

[130] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat et V. Chandrasekhar : Adversarially learned anomaly detection. *In 2018 IEEE International Conference on Data Mining (ICDM)*, p. 727–736. IEEE, 2018. (Cité en page 27.)

[131] Y. Zeng, M. Qiu, D. Zhu, Z. Xue, J. Xiong et M. Liu : Deepvcm : A deep learning based intrusion detection method in vanet. *In 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, p. 288–293. IEEE, 2019. (Cité en page 35.)

[132] B. Zhang, Z. Liu, Y. Jia, J. Ren et X. Zhao : Network intrusion detection method based on pca and bayes algorithm. *Security and Communication Networks*, 2018, 2018. (Cité en page 21.)

[133] Y.-Y. Zhou et G. Cheng : An efficient network intrusion detection system based on feature selection and ensemble classifier. *arXiv preprint arXiv :1904.01352*, 2019. (Cité en page 25.)

Glossaire

ADAS Advanced Driver-Assistance Systems. 44

APT Advanced Persistent Threat. 23

ARCEP Autorité de Régulation des Communications Électroniques et des Postes. 83, 131

C-ITS Cooperative Intelligent Transportation System. 14

C-V2X Cellular-Vehicle-to-Everything. 6

CAN Control Area Network. iii, 11–14, 28, 29, 31–35, 44, 129

CD Compact Disc. 28

CICIDS Canadian Institute for Cybersecurity, Intrusion Detection Evaluation Dataset. 19, 24–26, 48

CNIL Commission Nationale de l'Informatique et des Libertés. 7

CNN Convolutional Neural Network. 35

DARPA Defense Advanced Research Projects Agency. 20, 78

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 23

DNS Domain Name System. 44, 51–53, 83, 94, 96, 98, 117, 118, 122, 131, 146

ECU Electronic Control Unit. iii, 11–13, 28, 30, 32, 38, 129

EID Efficacité de la détection d'intrusion. 115, 118, 122, 123

FTP File Transfert Protocol. 23, 24

GPS Global Positioning System. 14, 93

HMM Hidden Markov Model. 19, 20, 56

HTM Hierarchical Temporal Memory. iv, v, 7, 41, 45, 46, 55–62, 64, 65, 69, 71, 74, 77, 78, 98, 111, 114–117, 120–122, 127, 130, 132, 133, 137–140, 142, 144, 163, 164

HTTP Hypertext Transfer Protocol. 19

ICMP Internet Control Message Protocol. 95

IEEE Institute of Electrical and Electronics Engineers. 14

IP Internet Protocol. 94, 97, 98

ITS Intelligent Transportation System. 1, 6, 42, 65, 86, 87, 93, 96, 134

IVI In-Vehicle Infotainment. 30

KDD Knowledge Discovery and Data Mining. 26, 27, 36, 37, 48

KNN K-Nearest Neighbors algorithm. 21, 24, 26

LAAS Laboratoire d'Analyse et d'Architecture des Systèmes. 84

LIN Local Interconnected Network. 13

LSTM Long Short-Term Memory. v, 35, 36, 55, 56, 77, 78, 121–127, 130, 132, 133, 163, 164

LTE Long-Term Evolution. 81

MAPE-K Monitor-Analyze-Plan-Execute over a shared Knowledge. 71, 73, 75

MCC Coefficient de Corrélation de Matthews. 108, 109, 114, 116–119, 121, 122, 124, 125

MQTT Message Queuing Telemetry Transport. 86–90, 93, 96, 97

NHTSA National Highway Traffic Safety Administration. 33, 37

OBD On-Board Diagnostic. iii, 11, 13, 29, 129

OCTANE Open Car Test-bed and Network Experiments. 34

OMNET Objective Modular Network Testbed in C++. 35

ONISR Observatoire National Interministériel de la Sécurité Routière. 3

ORUNADA Online and Real-Time Unsupervised Network Anomaly Detection Algorithm. 23

OSI Open Systems Interconnection model. 43

OTA Over-The-Air. 31, 132

RFC Request for Comments. 72

RGPD Règlement Général sur la Protection des Données. 7

RNN Recurrent Neural Network. 27

SCADA Système de Contrôle et d'Acquisition de Données. 20

SDR Sparse Distributed Representation. iv, 41, 57–60, 63, 64, 138, 140, 142, 143

SIEM Security information and event management. 36, 74

SIM Subscriber Identity Module. 32

SQL Structured Query Language. 22

SVDD Support Vector Data Description. 34

SVM Support Vector Machine algorithm. 25–27, 34, 35, 55

TCP Transmission Control Protocol. 15, 22, 49, 61, 62, 94, 95

TCU Telematic Control Unit. iii, 11, 14, 31, 44, 71, 129

TFP Taux de Faux Positifs. 109, 118, 122

TLD Top-Level Domain. 51

TPMS Tire-Pressure Monitoring System. 13, 28

TVP Taux de Vrai Positifs. 109

UDP User Datagram Protocol. 94, 95

UNADA Unsupervised Network Anomaly Detection Algorithm. 23

VANET Vehicular Ad-Hoc Network. iii, iv, 11, 14, 29, 33–35, 129

WAVE Wireless Access in Vehicular Environments. 14

XML Extensible Markup Language. 22

XSS Cross-Site Scripting. 20

ZMQ Open-Source Universal Messaging. 73

ZRC Zero Reference Curve. 109, 114

Abstract :

The growth of intelligent transport systems brings new highly connected vehicles on the roads of the world. These vehicles now embed new devices and services meant to increase road safety, reduce the environmental impact of the vehicles and improve the user experience. However, these new communication channels between vehicles and the rest of the world, especially cellular networks bring new vulnerabilities. Vehicles are now depending on the information provided by the network and are therefore subject to malfunction and anomalies due to such network. Worse, they become vulnerable to malicious actors of the cyber-space.

Mainstream information networks have been confronted with security problems for a long time. Numerous approaches have been designed in order to detect anomalies an intrusion inside such networks. However, these methods cannot be applied directly to the automotive context. In fact, the specific nature of the communications, the anomalies and the execution of intrusion detection systems inside the vehicles must be considered.

Therefore, we present a new anomaly detection system dedicated to vehicular networks and their vulnerabilities. Our detection is based on the creation of instantaneous description windows that are linked together thanks to an ontology. Thanks to these relations, the results of the detection are fed with the communication context of the vehicle during an anomaly. Consequently, the diagnostic from the administrator is made easier and we ensure the traceability of the anomaly. We evaluate the performances of our system thanks to a dataset produced by our tool named Autobot. It produces realistic communications, anomalies and attacks on cellular vehicular networks. We aim to evaluate our system based on the quality of the detection of different kinds of attacks while minimizing the number of false positives. We compare the results of two unsupervised machine learning algorithms that are used during the detection named HTM and LSTM.

Keywords :

intelligent transportation systems, intrusion detection, machine learning, emulation

Résumé :

L'avènement des Systèmes de Transport Intelligents entraîne avec lui l'apparition, sur les routes du monde entier, de véhicules toujours plus connectés à leur environnement. En effet, ils intègrent désormais des dispositifs destinés à améliorer la sûreté des routes, à réduire l'impact environnemental du véhicule et à rendre les trajets plus agréables aux utilisateurs. Cependant, l'interconnexion des véhicules avec le reste du monde et en particulier grâce à l'utilisation des réseaux cellulaires entraîne aussi son lot de vulnérabilités. Les véhicules connectés devenus tributaires des informations captées sur le réseau, s'exposent à des dysfonctionnements dus aux canaux de communication, ou pire encore, aux actions d'acteurs malveillants du cyber-espace.

Les réseaux classiques sont depuis longtemps confrontés à ces problèmes de sécurité. De nombreuses approches ont été imaginées afin de détecter les anomalies et tentatives d'intrusions dans les réseaux. Cependant, ces méthodes ne peuvent pas être directement appliquées au contexte véhiculaire. En effet, la nature des communications et des anomalies ainsi que l'exécution des dispositifs de détection à l'intérieur des véhicules sont des paramètres qui doivent être pris en compte.

Ce constat nous amène à proposer un nouveau système de détection d'anomalies spécifique aux communications des véhicules connectés et à leurs vulnérabilités. Sa détection repose sur la création de fenêtres de description instantanée du trafic liées entre elles par des relations modélisées sous la forme d'une ontologie. Grâce à ces relations, les résultats de la détection sont alimentés par le contexte des échanges du véhicule au moment de l'anomalie. Le diagnostic de l'administrateur est ainsi rendu plus aisé et nous assurons une traçabilité de la détection. Nous évaluons les performances de notre système grâce à un jeu de données issu d'un outil que nous avons conçu. Celui-ci génère des communications, anomalies et attaques conformes à ce qu'il serait observable sur les réseaux cellulaires de véhicules. Nous cherchons à évaluer notre système en fonction de ses capacités de détection de différents types d'anomalies et d'attaques en minimisant le nombre de faux positifs. Nous comparons les résultats de deux algorithmes non-supervisés utilisés pour la phase de détection HTM et LSTM.

Mots clés :

système de transport intelligent, détection d'intrusions, apprentissage automatique, émulation