# Table des matières

# Liste des figures

# Liste des tableaux

# Introduction générale

Les réseaux véhiculaires peuvent contribuer à accroître la sécurité et l'efficacité des transports en offrant des applications commerciales, de gestion de trafic et de sécurité. Cependant, ils soulèvent des problèmes de confidentialité et de sécurité, qui menacent considérablement les opérations du réseau et les données des utilisateurs.

Une des menaces les plus dangereuses est lorsque l'ordinateur de bord d'un véhicule est compromis et exploité par un attaquant à distance. Cette cybermenace est connue sous le nom de "vehicular botnet". Contrairement aux autres menaces de sécurité, il peut être utilisé pour exécuter à distance des tâches malveillantes à plusieurs niveaux : (1) attaques de Déni de service distribuées (DDoS) ; (2) violation des données personnelles du conducteur (suivi GPS illégal et écoute des conversations du conducteur et des passagers) ; (3) contrôler les véhicules bots à distance (ouvrir la porte, démarrer le moteur, allumer les feux, éloigner le véhicule ou désactiver les freins) ; (4) tromper le conducteur en lui donnant de fausses informations sur l'état du véhicule (falsifier le niveau de carburant, changer la lecture du compteur de vitesse et afficher les informations de panne sur le tableau de bord).

Malgré l'impact négatif des botnets véhiculaires sur la vie privée et la sécurité du conducteur, il existe dans la litérature une seule recherche [1], à notre connaissance, qui a examiné cette vulnérabilité. Garip et al. ont proposé GHOST [1], un protocole de communication pour les botnets véhiculaires, et qui utilise des messages de sécurité de base (BSMs) pour dissimuler sa communication sur le canal de contrôle. Ensuite, ils ont proposé SHIELDNET, un système de détection d'intrusion dédie pour le protocole Ghost, et qui détecte ses activités en recherchant les valeurs anormales de champs BSM spécifiques. Bien que son efficacité, SHIELDNET repose sur un protocole de

communication spécifique, il ne serait donc pas efficace si le botmaster du réseaux botnet change le protocole de communication.

Dans cette thèse, nous proposons une approche de détection des botnets véhiculaires qui ne suppose pas un protocole de communication particulier. Étant donné que le botnet véhiculaire est une cybermenace opérant sur plusieurs niveaux (intérieur et extérieure des véhicules), nous proposons un framework basé sur la détection comportementale à plusieurs niveaux (en anglais, A Multilevel Behaviour-based Framework for Botnet Detection in Vehicular Networks, AntibotV). Le framework proposé surveille l'interaction du véhicule avec l'extérieur en analysant le trafic du réseau. Il surveille également les activités intérieures du véhicule afin de détecter les opérations suspectes qui peuvent être liées à l'activité des logiciels malveillants des botnets. En outre, nous examinons de nouvelles attaques "zero-day" qui pourraient être menées exclusivement contre la pile de communication des réseaux véhiculaires : "Wave Short Message Protocol flood" (WSMP flood) ; 2) et "Geographic Wave Short Message Protocol flood" (Geo WSMP flood). Les deux attaques ciblent le protocole Wave Short Message, qui est un protocole utilisé dans les réseaux véhiculaires pour le transfert de paquets de sécurité et de gestion de trafic. Pour être plus précis, cette première contribution porte sur deux parties :

— Premièrement, nous identifiant un ensemble des attaques "zero-day" qui peuvent être exécuté par un cyber-attaquant en compromettant l'ordinateur de bord d'un véhicule, en utilisant des bots malveillants. Nous fournissons une description détaillée de deux zero-day DDoS attaques, ainsi que des attaques de vol d'information spécifique au contexte des réseaux véhiculaires.

— Deuxièmement, nous proposons un framework pour la détection des botnets véhiculaires basé sur une technique de détection comportemental sur plusieurs niveaux. Un framework qui se base sur les algorithmes des arbres de décision pour la détection des attaques "zero-day" proposées et d'autres attaques existantes, en surveillant les activités réseaux et intérieures du véhicule.

Étant savoir que l'efficacité de l'IDS dépend du modèle de classification, il est primordiale de faire l'apprentissage du classifieur sur un ensemble de données qui contient des traces de trafic réaliste, représentatif et qui couvre une variété d'échantillons normaux et d'attaques. Cependant, à notre connaissance, un tel ensemble de données n'a pas encore été produit pour les réseaux VANETs. Les ensembles de données existants [2, 3, 4, 5] contiennent des traces de trafic réseau générées et capturées par des simulateurs (comme celle que nous avons généré pour faire l'apprentissage de notre modèle AntibotV), ce qui ne prennent pas en valeur l'impact de l'environnement. Par conséquent, les modèles des systèmes de détection d'intrusion basés sur des données générer à partir des simulateurs peuvent ne pas être réaliste, représentatif, et ne fonctionne pas correctement dans un environnement réel.

Notre deuxième contribution porte sur la génération d'un prototype d'une nouvelle base de données de traces du trafic réseaux véhiculaire réelle, VDoS-LRS. Une base de données qui peut être utilisé pour ajouter une dimension de réalité sur les modèles formels générés pour les réseaux véhiculaires (comme le nôtre, AntibotV). Notre base de données contient un trafic réseau réel et de divers types de cyberattaques déni de service (DoS). Pour la génération de VDoS-LRS, nous faisons l'implémentation d'un environnement de test réaliste qui tient en compte de différents types d'environnement (urbain, routier et rural). En outre, nous explorons un large éventail de caractéristiques de trafic (en anglais, features) pour détecter et classer le trafic réseau véhiculaire. Nous évaluons la fiabilité de VDoS-LRS en utilisant différents algorithmes d'apprentissage automatique à des fins de cyber-criminalistique.

Le mémoire est organisé comme suit :

— Le premier chapitre est consacré à la présentation des réseaux véhiculaires. Ainsi, nous étudions leurs modes de communication, caractéristiques, normes et standards utilisées, et leurs applications les plus utilisées.

— Le deuxième chapitre est consacré aux problèmes de sécurité informatique les plus connus dans les réseaux véhiculaires, les défis soulevés due aux caractéristiques spécifiques de ce types de réseaux, ainsi les

contraintes a prendre en considération pour le développement des mécanismes de défense.

— Le troisième chapitre est dédié aux botnets véhiculaires. nous décrivons au départ les botnets en générale (architecture, cycle de vie, et leurs méthodes de communications), ensuite les botnets véhiculaires en spécifique. Nous discutons les méthodes de détection des botnets en générale, et ensuite nous discutons les techniques proposées pour les botnets véhiculaires en spécifiques, ainsi que leurs limites.

— Dans le quatrième chapitre nous présentons notre framework (AntibotV) pour la détection des botnets véhiculaires basé sur une technique de détection comportemental sur plusieurs niveaux.

— Dans le cinquième chapitre nous travaillons sur un problème que nous avons rencontré lors de la génération du modèle formel utilisé pour l'apprentissage de notre framework (AntibotV), et qui le manque d'une base de données réelles. Nous proposons un prototype d'une nouvelle base de données de traces du trafic réseaux véhiculaire réelle, VDoS-LRS. Une base de données qui peut être utilisé pour ajouter une dimension de réalité sur les systèmes de détection d'intrusion dans les systèmes de transport intelligent.

— Enfin, dans le dernier chapitre, nous concluons ce mémoire par un récapitulatif des résultats obtenus et les perspectives pour de futurs travaux.

# Les réseaux ad-hoc véhiculaires (VANET)

Sommaire

## 1.1 Introduction

Les systèmes de transports intelligents (STI) représentent la combinaison de technologies avancées de l'information et de la communication dans le domaine du transport et de la logistique. Ils sont utilisés afin d'améliorer la sécurité routière, réduire la congestion du trafic et améliorer l'expérience de conduite.

Historiquement, les STIs sont nés de la reconnaissance des limites des systèmes de transports traditionnels, avec l'accroissement des phénomènes de congestion et d'accidents de la route (chaque année, une moyenne de 1,3 million de personnes meurent dans des accidents de la route à travers le monde, et entre 20 et 50 millions de personnes subissent des traumatismes non mortels [6]). D'une autre part, l'apparition des technologies de gérance de transport en utilisant des infrastructures routières (les feux tricolores aux États-Unis en 1914, les parcmètres en 1935), l'incrémentation du nombre de véhicules (75 millions de véhicules dans les années 60), et l'apparition de nouvelles technologies orientée-véhicule (ceintures de sécurité, tableaux de bord rembourrés, standardisation des pare-chocs).

A la fin des années 60 et au début des années 70, les STIs ont vu le jour avec le système CACS (Comprehensive Automobile Traffic Control System) au Japon [7], et le système l'ERGS (Electronic Route Guidance System) en Allemagne et aux États-Unis [8], où sont lancés les premiers programmes de guidage autoroutier, dont la mise en place des premiers capteurs de vitesse, de débit autoroutier et d'occupation des surfaces routières. Seront ainsi généralisés les premiers signaux électroniques de trafic (DMS, dynamic message signs) et les premiers algorithmes de localisation avec leur représentation sur des cartes digitalisées. L'ensemble de ces innovations a été capitalisé par la création des premiers centres de gestion du trafic (TMS, traffic management centers) qui intègrent les données liées à la météorologie, à la vitesse des véhicules, à la congestion et à l'accidentologie [9].

Les technologies utilisées dans les STIs varient, allant de systèmes de gestion basiques (tels que les systèmes de gestion des feux de circulation, les ra-

dars ou la vidéo-surveillance) aux applications plus avancées qui intègrent des données en temps-réel avec retours d'informations de nombreuses sources (comme les systèmes de navigation embarqués en temps réel). Pour assurer ces applications, les STIs doivent assurer la transmission des informations sur le trafic aux conducteurs et aux autorités de transport avec précision et dans les plus brefs délais. Cette transmission est assurée par un réseau sans fil et mobile (figure 1.1), connu sous le nom de **Vehicular Ad-hoc Network** (**VANET**).

## 1.2 Définition des VANETs

Plusieurs définitions académiques des VANETs ont été proposées dans la littérature. Dans [10], les auteurs ont défini les VANETs comme une sous-classe des réseaux ad-hoc mobile (MANET), seulement que les nœuds sont des véhicules en mouvement, contrairement aux MANETs dont les nœuds sont des équipements mobiles connectés sans fil.

Dans [11], les auteurs ont défini VANET comme un ensemble de nœuds mobiles composés de véhicules, ainsi que de nœuds fixes appelés RSU déployés à des endroits critiques tels que des routes glissantes, des stations de service, des bâtiments administratifs, des intersections dangereuses ou des endroits bien connus pour des conditions météorologiques dangereuses.

## 1.3 Les modes de communication

VANET assure l'échange des informations inter-véhiculaire via le mode de communication Véhicule-à-Véhicule (V2V), entre les véhicules et les RSUs via le mode véhicule-à-infrastructure (V2I), et entre les autres véhicules et l'infrastructure en même temps avec le mode de communication hybride.

### 1.3.1 Communication Inter-véhicule (V2V)

La communication inter-véhicule utilise le multicast/broadcast en multi-sots pour transmettre les informations relatives aux trafics à un groupe de

récepteurs. Dans les systèmes de transport intelligents, les véhicules doivent être intéressés uniquement par l'activité sur la route qui est en avant et ignorer ce qui est derrière (exemple de diffusion de message ALERT au sujet d'une collision imminente). Il y a deux types d'expédition de message dans des communications inter-véhicule : la diffusion naïve et la diffusion intelligente.

Dans la diffusion naïve, les véhicules envoient des messages à diffusion générale périodiquement et sur un intervalle régulier. À la réception du message, le véhicule ignore le message s'il est venu d'un véhicule qui est derrière lui. Si le message vient d'un véhicule de l'avant, le véhicule récepteur envoie à son tour le message en diffusion générale vers les véhicules qui sont derrière lui. Ceci assure que tous les véhicules qui sont en déplacement vers l'avant reçoivent tous les messages à diffusion générale. Les limitations de cette méthode c'est qu'on va avoir un grand nombre de messages à diffusion générale, ce qui augmente le risque de collision de message dans le réseau.

La diffusion intelligente avec une reconnaissance implicite des adresses corrige le problème qu'on trouve dans la diffusion naïve en limitant le nombre de diffusion d'un message donné. Si le véhicule reçoit le même message venant de derrière, il présume qu'au moins un véhicule derrière l'a déjà reçu. L'idée est que chaque véhicule qui est derrière est responsable de transmettre le message pour ses prochains qui sont derrière lui. Si un véhicule reçoit un message de plus d'une source il agira sur le premier message seulement [12].

### 1.3.2 Communication entre Véhicules et Infrastructure (V2I)

La communication entre Véhicules et le Bord de la route représente une émission en un seul sot, où l'unité de Bord de la route envoie un message en diffusion à tous les véhicules à proximité. Ce type de communication fournie une bonne bande passante entre les véhicules et les Points d'accès de Bord de route. Les unités de bord de route peuvent être placées à chaque kilomètre, ainsi elles permettent d'avoir des débits de transmission élevés même dans un environnement de dense circulation [12].

### 1.3.3 Communications hybride

La combinaison de ces deux types de communications permet d'obtenir une communication hybride très intéressante. En effet, les portées des infrastructures étant limitées, l'utilisation de véhicules comme relais permet d'étendre cette distance. Dans un but économique et en évitant de multiplier les bornes à chaque coin de rue, l'utilisation de sauts par véhicules intermédiaires prend toute son importance [13].

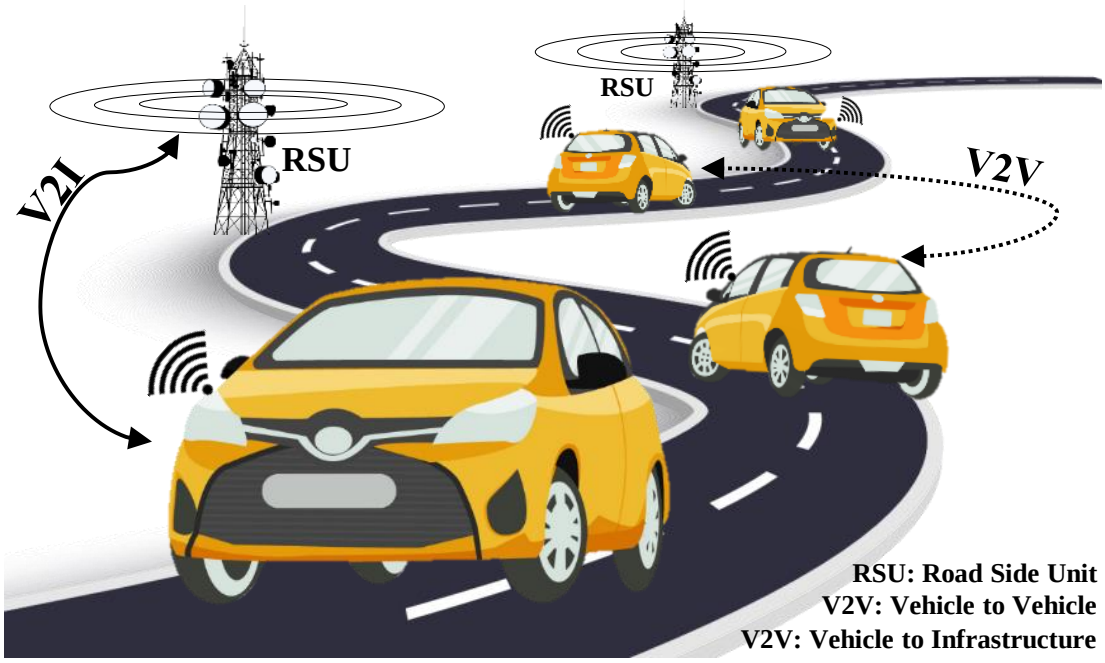


FIGURE 1.1 – *Réseau Ad-hoc Véhiculaire (VANET)*

## 1.4 Les Caractéristiques des VANETs

Contrairement aux autres types de réseaux sans-fil, Les VANET sont caractérisés par un ensemble de propriétés particulières qui les rendent très distincts, nous citons parmi eux [14] :

### 1.4.1 Déconnexion fréquente

Les véhicules se déplacent tout en échangeant des informations. En raison des changements rapides de topologie, les connexions entre deux véhicules

sont facilement déconnectées. Habituellement, les déconnexions se produisent dans des réseaux peu fréquents [15].

### 1.4.2 Le potentiel énergétique

À la différence des réseaux sans fil traditionnels où la contrainte d'énergie représente un facteur important, les entités des réseaux véhiculaires disposent de grandes capacités énergétiques qu'elles tirent du système d'alimentation des véhicules.

### 1.4.3 L'environnement de communication et le modèle de mobilité

Les réseaux véhiculaires imposent la prise en compte d'une plus grande diversité environnementale. Du fait de la mobilité des véhicules, il est en effet possible de passer d'un environnement urbain caractérisé par de nombreux obstacles à la propagation des signaux, à un environnement périurbain ou autoroutier présentant des caractéristiques différentes. En plus de cette diversité environnementale, les réseaux véhiculaires se distinguent également des réseaux sans fil ordinaires par un modèle de mobilité dont une des traductions les plus évidentes est l'importante vitesse des nœuds qui réduit considérablement les durées de temps pendant lesquelles les nœuds peuvent communiquer.

### 1.4.4 Le modèle de communication

Les réseaux véhiculaires ont été imaginés principalement pour les applications liées à la sécurité routière (ex. diffusion de messages d'alerte). Dans ce type d'application, les communications se font presque exclusivement par reliages successifs d'une source vers une multiplicité de destinataires. Le modèle de transmission en broadcast ou en multicast est donc appelé à dominer largement dans les réseaux véhiculaires, ce qui n'est par exemple pas sans conséquence sur la charge du réseau et le modèle de sécurité à mettre en œuvre.

### 1.4.5 La taille du réseau

Étant donné les avancées importantes réalisées dans le domaine des communications sans fil et les bas coûts des équipements associés, les véhicules qui intègrent déjà massivement des systèmes GPS et des équipements Bluetooth, seront très probablement équipés et ce, tout aussi massivement, de plateformes de communication leur permettant de constituer de véritables réseaux. Ce faisant, et compte tenu de l'importance sans cesse grandissante de la densité et du parc des véhicules, on peut s'attendre à ce que la taille des réseaux véhiculaires dont les déploiements restent encore très confidentiels, soit d'une tout autre ampleur. L'importance potentielle de la taille des réseaux véhiculaires constitue donc une caractéristique majeure à prendre en compte dans la conception de ces réseaux.

## 1.5 Normes et Standardisation : Dedicated Short Range Communication (DSRC)

Afin d'assurer les différents modes de communication (V2V, V2I et hybride), l'industrie des automobiles à développé la norme de communication dédiées à courte portée, DSRC (Dedicated Short-Range Communication standard) (figure 1.2). À l'origine, la technologie DSRC a été conçue pour répondre au besoin de transactions financières électroniques (télépéage). C'était un modèle de communication à courte portée (4 à 10 mètres) avec des débits inférieurs à 1 Mbit/s. Ensuite, le standard DSRC a évolué à partir du IEEE 802.11a [16] vers la norme IEEE 802.11p ou WAVE (Wireless Access for Vehicular Environments) [17] afin de répondre aux caractéristiques des VANETs.

Le DSRC propose un canal de communication spécialement conçu pour transmettre des messages de très haute priorité à l'instar de certains messages critiques liés à la sécurité routière. Le WAVE présente aussi des caractéristiques beaucoup plus adaptées à la mobilité (comme des temps d'établissement de connexion plus courts) qui permettent l'envoi à la volée d'informations à des véhicules roulants à grande vitesse. Il présente une bonne fiabilité

avec un taux d'erreur de 10-6 à 160 km/h. La technologie IEEE 802.11p est particulièrement adaptée pour les applications à portée moyenne et sensibles au délai [18]. Dans cette section, nous allons décrire en détaille les différents standard déployé dans la technologie DSRC : IEEE 802.11p, IEEE 1609.2, 1609.3, 1609.4, "SAE J2735 Message Set Dictionary", et "the emerging SAE J2945.1 Communication Minimum Performance Requirements standard". [19].

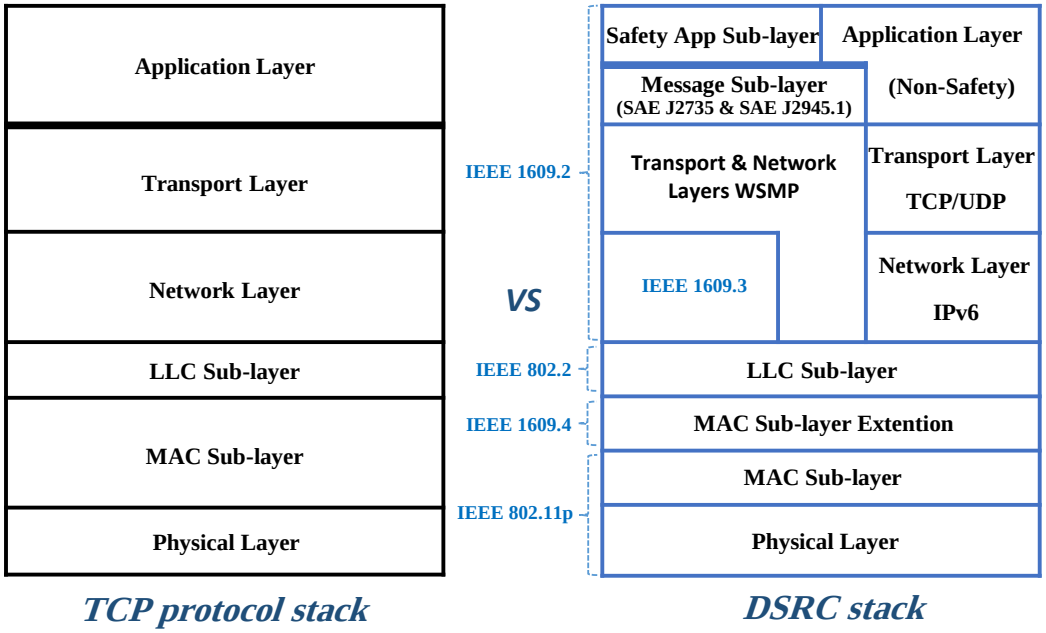


FIGURE 1.2 – *Comparaison entre la pile protocolaire DSRC et la pile TCP/IP*

### 1.5.1 IEEE 802.11p

Au niveau des couches PHY et MAC, la DSRC utilise le standard IEEE 802.11p, ou connu aussi sous le nom de Wireless Access for Vehicular Environments (WAVE). WAVE est un amendement approuvé de la norme IEEE 802.11 qui permet une communication sans fil sécurisée et un accès physique pour une vitesse élevée (jusqu'à 27 Mo/s), une courte portée (jusqu'à 1000 m) et une faible latence. Le spectre qui lui est alloué va de 5,850 à 5,925 GHz, divisé en sept canaux de 10 MHz [20]. Le canal 178 est réservé aux informations de contrôle et les six autres canaux sont utilisés pour les applications de services (figure 1.3).

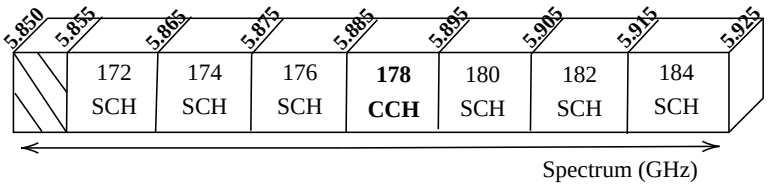


FIGURE 1.3 – *Les bandes de fréquences de la DSRC*

### 1.5.2 IEEE 1609.4

Au niveau de l'extension des sous-couches MAC du DSRC, la norme IEEE 1609.4 est déployée. IEEE 1609.4 est utilisé afin de gérer les priorités d'accès, faire la gestion des service de permutation entre les canaux et le routage [21] afin d'assurer un passage efficace et fluide entre les sept canaux de transmission [22].

### 1.5.3 IEEE 1609.3

Est la norme utilisé par la DSRC afin de gérer les services réseaux. IEEE 1609.3 prend en charge deux piles protocolaires, Wave Short Message Protocol (WSMP) et IPv6. Le choix de l'utilisation d'une pile (WSMP ou IPv6 + UDP/TCP) dépends principalement sur les exigences de l'application . Par exemple, pour les applications qui dépendent des capacités de routage pour transmettre des paquets multisauts comme les applications commerciales, la pile IPv6+ UDP/TCP est utilisé. Cependant, les applications qui nécessitent le transfert de messages à un seul saut comme les applications de sécurité et de gestion de trafic, la pile protocolaire WSMP est utilisé. Contrairement aux protocoles IP, UDP et TCP, le WSMP est un protocole unique de la couche réseau WAVE et ne peut être trouvé que dans un trafic du réseau VANET. Il est utilisé uniquement pour prendre en charge une communication hautement prioritaire et sensible au temps.

### 1.5.4 IEEE 1609.2

Afin de définir les formats des messages sécurisés, la DSRC utilise la norme IEEE 1609.2. IEEE 1609.2 comprend des techniques utilisées pour la

sécurisation des messages et pour la description des fonctions administratives nécessaires pour prendre en charge les fonctions de sécurité de base [23].

### 1.5.5 Le standard "SAE J2735 Message Set Dictionary"

Est le standard utilisé pour spécifié l'ensemble de formats des messages échangés via la pile protocolaire DSRC. Il représente une base de données qui contient la syntaxe des messages suivants : (1) Le "Basic Safety Message (BSM)", le message le plus important. Il est transmis périodiquement par les véhicules pour fournir des informations sur l'état du véhicule émetteur (position, dynamique du véhicule, etc.) ; (2) Le message "A la Carte" ; (3) Le "Common Safety Request" ; (4) Le message d'alerte "Emergency Vehicle" ; (5) Le "Intersection collision avoidance" message ; (6) Le message "Map data" ; (7) Les deux messages "NMEA et RTCM Corrections" ; (8) Le "Probe data Management" message ; (9) Le "Probe Vehicle Data" message ; (10) Le "Roadside Alert" message ; (11) Le "Signal phase and Timing" message ; (12) Le message "Signal Request" ; (13) Le message "Signal Status" ; et finalement, (14) Traveler information [24]. Le tableau 1.1 présente chaque message et sa description brièvement.

### 1.5.6 Le standard "SAE J2945.1"

Il prend en charge les exigences de performance des applications de sécurité V2V, comme le taux et la puissance de transmission des messages BSMs, ainsi le contrôle de congestion des canaux afin d'éviter la saturation [25].

## 1.6 Les applications

L'objectif principal des VANETs est d'assurer une panoplie d'applications qui vise a amélioré la sécurité routière, réduire la congestion du trafic et amélioré l'expérience de conduite. Ces applications peuvent être divisée en trois catégories [26] : orienté-sécurité, orienté-gestion de trafic et orienté-commerciale. Dans cette section nous allons décrire en détaille chaque ca-

| Type de message | Usage |
|---|---|
| "Basic Safety Message" | transmis périodiquement par les véhicules pour fournir des informations sur l'état du véhicule émetteur (position, dynamique du véhicule, etc.) |
| Le message "A la Carte" | message générique avec contenu flexible, utilisé pour personnaliser les éléments de données entre les véhicules |
| Le "Common Safety Request" | utilisé pour collecter des informations supplémentaires sur l'état d'un autre véhicule. |
| Le message d'alerte "Emergency Vehicle" | utilisé pour avertir les conducteurs qu'un véhicule d'urgence est dans la zone. |
| "Intersection collision avoidance" | utilisé pour fournir des informations sur l'emplacement du véhicule par rapport à une intersection spécifique. |
| Le message "Map data" | envoyé par les RSUs pour transmettre la description géographique d'une intersection. |
| "NMEA et RTCM Corrections" | deux messages qui en-capsulent des styles de correction des coordonnés GPS, NMEA 183 et RTCM respectivement. |
| "Probe data Management" | utilisé par les RSUs afin de gérer la collecte des données récupérer depuis les capteurs des véhicules ("probe data"). |
| Le "Probe Vehicle Data" message | contient des informations concernant le statut du véhicule dans une partie de la route. Ces informations peuvent être utilisées afin de former une image sur les conditions de conduite sur cette route. |
| Le "Roadside Alert" message | envoyé par les RSUs afin d'avertir les véhicules aux conditions de conduites dangereuses. |
| Le "Signal phase and Timing" | envoyé par les RSUs dans les intersections avec les feux de signalisation afin de transmettre aux véhicules l'état du signal et le temps restant. |
| Le message "Signal Request" | envoyé par un véhicule pour demander la priorité d'un feu de signalisation. |
| Le message "Signal Status" | envoyé par le RSU pour répondre aux messages "Signal Request" envoyés par les véhicules. |
| "Traveler information" | des messages envoyés par les RSUs pour transmettre des conseils concernant la signalisation routière. |

TABLE 1.1 – *Les messages définis par le standard SAE J2735, dans la DSRC*

tégorie d'applications, ses caractéristiques (protocoles utilisés, les canaux, ..), ainsi que donner des exemples.

### 1.6.1 Orienté-Sécurité

Les applications de cette catégorie visent à surveillé activement l'environnement du véhicule (les autres véhicules et les routes) via des échanges de messages en mode V2V. Leurs objectifs principaux sont d'aider les conducteurs à gérer les événements à venir ou les dangers potentiels bien à l'avance (zones de construction, limites de vitesse, les virages dangereux et les véhicules en panne). Certaines applications peuvent automatiquement prendre les mesures appropriées (comme le freinage automatique) pour éviter les accidents potentiels, tandis que d'autres ne fournissent que des informations de conseil ou d'avertissement au conducteur.

En termes de performances, par rapport aux véhicules traditionnels qui se basent seulement sur les capteurs, les applications de sécurité basée sur la technologie de communication DSRC sont capable de détecter des événements distants qui se produisent en dehors de la plage de détection des capteurs traditionnels. Ce préavis peut donner aux conducteurs suffisamment de temps pour planifier les manœuvres autour d'obstacles ou pour planifier des changements sur leurs itinéraires.

Concernant les propriétés fondamentales et les exigences de fonctionnalité des applications de sécurité, ils sont différents par rapport aux autres catégories. La majorité d'entre eux repose sur **(1) un mode de communication V2V**. **(2) Une région d'intérêt "Region of Interest : RoI" courte ou moyenne**. Par exemple, les véhicules doivent être conscients de l'état cinématique d'autres véhicules dans leur voisinage direct (c'est-à-dire à quelques centaines de mètres), alors que dans d'autres applications de sécurité, les véhicules doivent connaître la situation de danger d'un tronçon de route qui se trouve devant (c'est-à-dire jusqu'à 1 kilomètre). **(3) Des méthodes de communication périodique ou événementiel**, comme les applications de détection de collision qui sont diffusée périodiquement, tandis que les messages d'avertissement pour des événements tels que le freinage d'urgence sont gé-

néralement déclenchés par les événements.**(4) Un modèle des destinataires potentiels : "one-to-many" ou bien "one-to-a-zone"**, dont on trouve quelques applications de sécurité diffusent les alertes a tous les véhicules voisins (one-to-many) afin d'éviter les collisions potentielles. Tandis que d'autres applications diffusent l'information seulement aux véhicules voisins (one-to-a-zone) parce que seuls les véhicules de la région sont concernés.

D'un point de vue réseau, les applications de sécurité reposent sur **(1) le protocole WSMP**, un protocole introduit dans la DSRC par la norme IEEE 1609.3, et qui fournit un service de diffusion efficace avec un faible latence (5 $\mu$s). (2) Dans la couche physique, les messages sont transmis sur **le canal de contrôle CCH "178"**, le canal dédies spécialement aux applications de sécurité pour ne pas être saturé par le trafic réseau des autres catégories d'applications non-vitale.

De nombreux applications de sécurité ont été définie dans la littérature [27]. Dans notre travail, nous considérons les sept applications suivantes :

1. "Cooperative Collision Warning" (CCW) : une application qui vise à aider le conducteur à éviter les collisions en utilisant les messages d'état cinématique collectés auprès des véhicules alentour.
2. "Cooperative Violation Warning" (CVW) : les unités routières (RSUs) envoient aux conducteurs les informations nécessaires à l'approche d'une phase de signalisation routière ou d'un feu rouge.
3. "Emergency Electronic Brake Light" (EEBL) : Si un véhicule fait un freinage brusque, il diffuse un message pour informer les autres véhicules.
4. "Post Crash Notification" (PCN) : si un véhicule est impliqué dans un accident, il diffuse un message pour informer les autres véhicules.
5. "Road Feature Notification" (RFN) : si un véhicule détecte que la route contient une caractéristique dangereuse (ex : virage serré), il diffuse un message pour informer les autres véhicules.
6. "Road Hazard Condition Notification" (RHCN) : si un véhicule détecte un danger routier naturel (roches, animaux, glace), il diffuse un message pour informer les autres véhicules.

7. "Stopped or Slow Vehicle Advisor" (SVA) : si un véhicule réduit sa vitesse ou s'arrête, il diffuse un message pour informer les autres véhicules [26].

### 1.6.2 Orienté-Gestion de trafic

Ces applications sont utilisées pour partager des informations sur le trafic routier entre les infrastructures installés au bord de la route, les véhicules, et les systèmes de contrôle de la circulation centralisés. Ce partage permettre d'avoir un contrôle du flux de trafic plus efficace, maximiser le débit des véhicules sur la route et amélioré le degré de commodité pour les conducteurs.

Une réduction de la congestion routière a un avantage direct sur la société en termes de consommation de carburant et d'émissions de $CO_2$, et d'autres avantages supplémentaires tels que la réduction du temps perdu pendant les embouteillages. Cependant, ses avantages sont réalisés à un niveau qui n'est pas aussi perceptible pour le client.

La majorité des applications de gestion de trafic reposent sur **le mode de communication V2I**. Des **régions d'intérêts "RoI" longues généralement**, par exemple, les conducteurs des véhicules peuvent vouloir connaître l'état de la congestion routière plusieurs kilomètres à l'avance pour planifier leurs voyages. Des méthodes de communication **événementiel ou initié par l'utilisateur**. Tandis que le modèle des destinataires potentiellement utilisée est le **"one-to-one"**, véhicule-infrastructure, infrastructure-serveur distant. Concernant les caractéristiques réseaux, ils utilisent le protocole **WSMP**, avec les canaux de service **SCH**, et ils se basent généralement sur des protocoles de routages unicast.

Parmi les applications de gestion de trafic, nous considérons :

1. "Congested Road Notification" (CRN) : si un véhicule détecte une collision, il envoie un avertissement aux autres véhicules afin qu'ils puissent emprunter une autre voie.
2. "Parking Availability Notification" (PAN) : cette application est utilisée pour demander aux RSUs de fournir le parking le plus proche et si il contient des places de stationnement vides.

3. "Parking Spot Locator" (PSL) : Il s'agit d'un échange d'informations entre un véhicule et l'unité routière d'allocation des places dans un parking, afin de savoir les places de stationnement disponibles.
4. "Free Flow Tolling" (TOLL) : il permet d'appliquer un e-paiement en passant par un péage autoroutier.
5. "Trafic Probe" (TP) : l'objectif de cette application est de collecter les messages d'état cinématique des véhicules et de les envoyés au centre de gestion de trafic a travers les RSUs [26].

### 1.6.3 Orienté-Commerciale

Ces applications fournissent aux conducteurs divers types de services de communication pour assurer le divertissement et la satisfaction des conducteurs, tels que : l'accès web, le streaming audio et vidéo, les actualités, les informations sur le trafic et la météo, e-mailing, téléchargement de la musique et paiements en voiture.

Contrairement aux applications de sécurité et de gestion de trafic, les applications commerciales se basent sur le protocole IP (TCP/UDP). Ce sont des applications similaire a celle qu'on trouve dans tous les autres réseaux IP, pour cela, au niveau de la couche physique il peuvent utiliser soit la norme DSRC-SCH, soit d'autres standard (Wifi, Wimax, Lte, ...). Plusieurs applications ont été définie, nous considérons :

1. "Content, Map or Database Download" (CMDD) : un véhicule se connecte à un RSU pour télécharger du contenu (cartes, multimédia ou pages Web).
2. "Real-Time Video Relay" (RTVR) : un véhicule peut lancer une video de la route en streaming, ce qui peut être utile aux autres conducteurs de la région.
3. "Remote Vehicle Personalization/Diagnostics" (RVP/D) : les conducteurs peuvent se connecter à un point d'accès pour télécharger une mise a jour des paramètres du véhicule, ou pour faire un diagnostic à distance.

4. "Service Announcement" (SA) : une station de service utilise un RSU qui envoie périodiquement des messages Hello pour annoncer aux véhicules sa présence [26].

## 1.7 L'architecture intérieure des noeuds

Pour qu'un véhicule qui fait partie d'un réseau VANET puisse collecter des informations, les enregistrées, les traitées, les exploitées, les relayées vers d'autres véhicules, ou bien les envoyées vers l'infrastructure routière, elle doit être équipé par un ensemble des équipements et des technologies qui lui permettre de faire tout ces actions.

### 1.7.1 Les composants

Contrairement aux véhicules traditionnels, les véhicules modernes et connectés (figure 1.4) contiennent :

**Une plateforme de calcule, "Computing platform"** : un module qui représente le cerveau du véhicule. Il est capable de prendre en charge tout, des mises à jour des logicielles à la technologie des voitures sans conducteur. Ainsi la supervision de l'exécution des protocoles, y compris ceux liés à la sécurité [28].

**"On-Board Unit" (OBU)** : un dispositif de communication monté sur les véhicules. Il permet les communications DSRC avec les autres OBUs et les RSUs [29].

**Mobile wireless gateway** : est une passerelle cellulaire, utilisée pour assurer une communication fiable, sécurisée et à large bande passante pour les applications mobiles avec les infrastructures cellulaire (3G, 4G, ou wifi ).

**Un système télématique** : qui est un boîtier électronique embarqué, utilisé par la voiture pour émettre de nombreuses informations à distance : kilométrage, géo-localisation, consommation ou indications concernant la maintenance [30]. Ce système offre plusieurs types des technologies de communication, tel que : GPS, Radio, Bluetooth, etc.

**Un système de sécurité** : a pour but la protection du conducteur et des

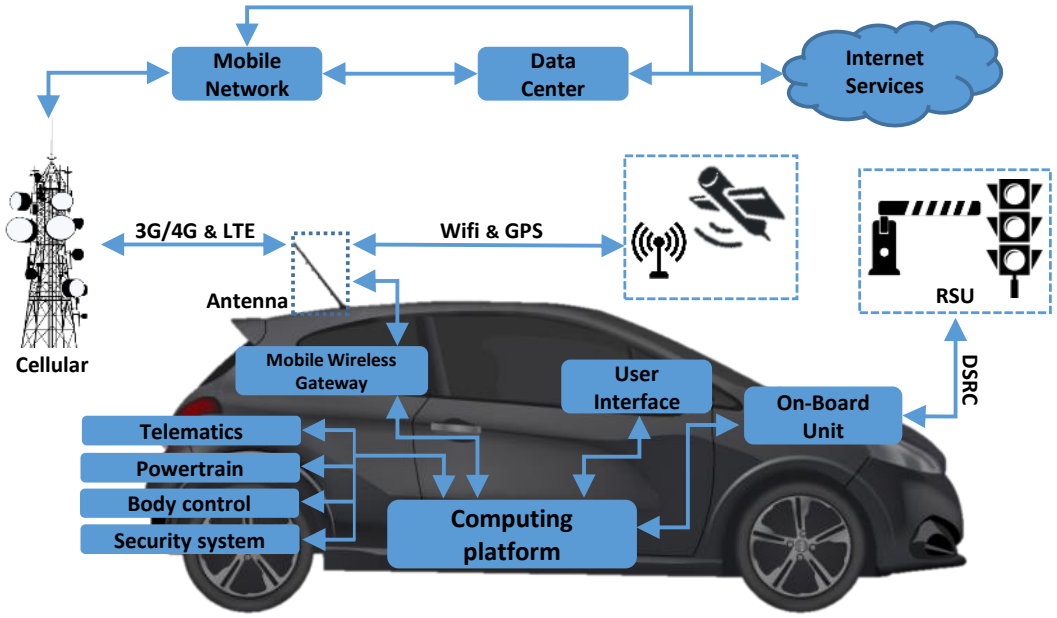


FIGURE 1.4 – *Véhicule moderne (inspiré de [31])*

passagers en cas d'accident. Il est composé de plusieurs technologies, nous citons parmi eux : L'Anti-Blocage de Sécurité (ABS), qui est utilisé pour éviter le blocage des roues lors du freinage notamment sur un sol où l'adhérence est faible. "L'Electronic Stability Program" (ESP), un système qui corrige la trajectoire en agissant sur le freinage et le couple moteur. "L'Acceleration Slip Regulation" (ASR), utilisé pour éviter le patinage des roues lors d'une accélération violente ou sur un sol humide [32].

**Le système motopropulseur "powertrain"** : est au cœur de la conception des véhicules, le moteur. Il fournit la puissance motrice, qui est ensuite gérée et contrôlée par les composants de transmission et le conducteur. Le système motopropulseur définit donc les performances dynamiques et le caractère du véhicule [33].

**Le système de contrôle du corps "body control"** : c'est le système qui permet la communication, le contrôle, et la coordination entre les modules électroniques embarqués. Parmi ses taches : éclairage intérieur et extérieur, verrouillage des portes, le contrôle des fenêtres et des miroirs [34].

### 1.7.2 Le réseau embarqué

Ces systèmes et technologies embarquées repose sur des composants électroniques libellés les calculateurs embarqués, ou ECU (Electronic Control Unit). Contrairement aux équipements mécaniques utilisées dans les véhicules traditionnels, les ECUs ont simplifié l'architecture intérieure des véhicules, ainsi la réparation et le diagnostic même pour ceux qui ne connaissent rien aux véhicules. Chaque ECU contient ses capteurs et ses actionneurs, il reçoit les entrées de ses capteurs et met en œuvre des fonctions spécifiques par ses actionneurs. La communication entre ces calculateurs est assurée par un type de bus dédié spécialement pour les véhicules ; appelé Controller Area Network (CAN). Les ECUs et les bus CAN forment ensemble le réseau embarqué du véhicule (figure 1.5).

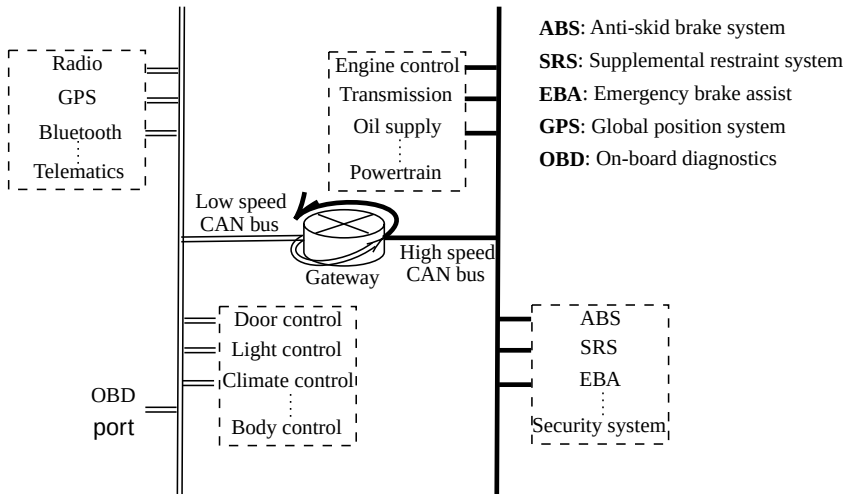


FIGURE 1.5 – *Le réseau de communication intérieur d'un véhicule*

Dans ce réseau embarqué, il existe deux types de CAN bus (basse vitesse et haute vitesse) connectés par une passerelle [35]. Pour la communication des modules critiques (groupe motopropulseur, frein, etc.), le CAN bus à haute vitesse est utilisé. Pour les autres types de modules (télématique, contrôle de carrosserie, etc.), le CAN bus à basse vitesse est utilisé. La transmission sur le CAN bus se fait d'une manière séquentielle. Cependant, si plus d'un ECU transmet en même temps sur le même CAN bus, la méthodes d'accès par compétition CSMA-CR "Carrier Sense Multiple Access with Collision Resolution" est utilisé [36]. Avant de commencer la transmission, chaque ECU doit

tester l'état du support et il ne peut transmettre que si le support est libre. Pour éviter les collisions en chaîne, le ECU qui transmet une trame (sachant qu'une trame commence par un identificateur unique), cesse d'émettre s'il reçoit un bit différent du sien. Ainsi, un noeud qui émet un bit à 1 s'arrête s'il voit passer sur le support un bit à 0. En revanche un ECU qui reçoit un bit identique à celui qu'il a émis continue de transmettre. Comme les adresses diffèrent au moins par un bit, un seul noeud poursuit la transmission de sa trame jusqu'à sa fin.

## 1.8 Conclusion

Dans ce chapitre nous avons présenté les réseaux VANETs qui sont apparus comme solution aux besoins des applications de sécurités routières ; mais actuellement ils permettent aussi de développer de nouveaux services aux usagers de la route comme (la localisation des stations d'essence, emplacements de parking libre et l'accès à Internet).

Afin de mieux comprendre les réseaux véhiculaires, nous avons présenté leurs caractéristiques principales et les différentes entités communicantes ainsi que les différents modes de communications existants.

Dans le chapitre suivant nous nous intéresserons aux problèmes de sécurité dans les VANETs, les défis, les contraintes, ainsi que les différents types d'attaques qui ont été étudié jusqu'à aujourd'hui.

# La sécurité dans les réseaux VANETs

Sommaire

## 2.1 Introduction

Vu l'importance des informations échangées entre les véhicules eux-mêmes et avec les infrastructures, la sécurité des réseaux véhiculaires est donc un sujet très important à traiter.

Dans ce chapitre, nous présentons en premier lieu les caractéristiques et services de sécurité spécifiques aux VANETs, les contraintes, et les challenges. Nous détaillons ensuite les modèles d'attaquants existants, les attaques et leurs classifications, ainsi que les mécanismes et éléments de base de la sécurité.

## 2.2 La sécurité informatique

La sécurité est l'état d'être exempt de danger ou de menace, elle signifie la sûreté, ainsi que les mesures prises pour être sûr et protégé.

## 2.3 Les sécurité des VANETs

Dans les VANETs, il est essentiel de se prémunir contre les activités abusives et de bien définir l'architecture de sécurité, car il s'agit d'une communication sans fil spécifique, et qui est plus difficile à sécuriser par rapport aux autres types de réseaux : des caractéristiques différents, de nouvelles contraintes, de nouveaux challenges de sécurité, une architecture de communication (DSRC) différente avec des protocoles et standard spécifique, etc.

La sécurité dans les VANETs affectent la sécurité des personnes d'une manière directe, ce qui a rendu la recherche dans cette filière très active. Il y a quelques années, de nombreux chercheurs ont exploré les attaques de sécurité et tenté de trouver les solutions associées. D'autres ont tenté de définir des infrastructures de sécurité ou de formaliser des normes et des protocoles. Malgré cela, de nouvelles vulnérabilités n'ont pas cessé d'apparaître. [37].

## 2.4 Les services de sécurité

La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'un système sont uniquement utilisées dans le cadre prévu. Elle vise généralement cinq principaux objectifs qui doivent être prises en compte lors de la conception des protocoles, des systèmes de sécurité, des algorithmes cryptographiques, ainsi que dans leur mise en œuvre dans les VANETs :

### 2.4.1 Authentification

Ce concept de sécurité permet aux entités du réseau de s'assurer de la bonne identité des entités avec lesquelles elles communiquent. L'authenticité permet aux différentes entités du réseau de se fier aux données et messages diffusés. Elle est la seule exigence qui permet la coopération entre les différents participants ; leur identification permet ainsi d'assurer le bon contrôle de l'authenticité des messages échangés [38]. Il convient de préciser ici qu'il existe deux types d'authentification : une authentification des messages qui permet d'en retracer la source et une authentification des entités qui permet, elle, d'identifier les nœuds du réseau [39].

### 2.4.2 Confidentialité

Ce concept de sécurité permet de garantir la non-divulgation des données transmises dans le réseau à des parties non autorisées. Seules les parties habilitées peuvent y accéder à travers le réseau [40]. La confidentialité consiste ainsi à préserver les informations vitales liées aux véhicules par l'application des algorithmes de cryptographie asymétrique et symétrique, ce qui empêche les entités malveillantes de suivre et d'écouter les messages concernant un véhicule ciblé dans le réseau.

### 2.4.3 Disponibilité

Ce concept de sécurité permet de garantir que toute entité autorisée puisse accéder aux ressources du réseau en tout temps avec une qualité de service (QoS) adéquate [40]. En effet, tous les participants dans le réseau doivent avoir un accès effectif et rapide aux différents services de la gestion du trafic, aux applications de sécurité et de confort sollicités. Pour atteindre un bon niveau de disponibilité, il est indispensable d'installer du matériel et mettre en œuvre des protocoles de sécurité de hautes performances.

### 2.4.4 Intégrité

Ce concept de sécurité permet d'assurer que les messages diffusés ne seront pas modifiés ou altérés volontairement ou accidentellement entre la phase d'émission et de réception par des entités non autorisées (malveillantes). Cet objectif de sécurité vise ainsi à doter les destinataires d'un pouvoir permettant de détecter les manipulations de données effectuées durant leur transmission par les entités malveillantes et rejeter les paquets correspondants. L'intégrité peut être réalisée principalement par l'utilisation des fonctions de hachage et de la cryptographie sur des champs spécifiques des paquets. Cependant, dans les réseaux sans fil, se pose toujours la contrainte de l'intégrité qui n'est pas toujours forcément liée au terme de manipulation. En effet, bien des altérations sont le fait des conditions de propagation radio.

### 2.4.5 Non-répudiation

Ce concept de sécurité permet de démontrer et localiser avec certitude l'origine des données. Grâce à ce principe, chaque entité diffusant un message sur le réseau ne peut le nier ou se rétracter de l'avoir émis. Ainsi, la non-répudiation permet d'identifier les entités malveillantes qui tentent de commettre des actes illégaux, ce qui permet d'écarter toute possibilité pour qu'un attaquant injecte des données erronées sans qu'elles ne soient immédiatement identifiées. Le concept de non-répudiation est essentiel dans les transactions commerciales en lignes et financières, ainsi que dans les opéra-

tions électroniques de facturation. Dans le contexte des VANETs, la signature numérique est utilisée pour garantir la non-répudiation des messages concernant les applications de sécurité et de gestion du trafic [40].

## 2.5 Les défis de sécurité dans les VANETs

Pour dire qu'un réseau VANET est sécurisé, les cinq services de sécurité mentionnés dans la section précédente doivent être assurer. Cependant, les différentes caractéristiques des VANETs soulèvent plusieurs défis de sécurité qui n'apparaître pas dans les autres types de réseaux sans-fil, nous citons parmi eux :

1. **Absence d'une autorité globale** : la taille d'un réseau VANET peut être géographiquement illimitée et très évolutive. Due a cette croissance rapide, aucune autorité globale ne pourra gérer ce genre de réseau.
2. **La confiance et l'intégrité** : la confiance est requise car la nature ad-hoc du VANET incite les nœuds à collecter des informations auprès d'autres véhicules et RSUs. Par conséquent, cet échange d'informations est fréquent, il doit être fiable et vérifiée (intégrité ).
3. **La gestion des clés** : La gestion des clés sécurisée est fondamentale pour fournir la sécurité réseau. Cependant, elle est difficile a assuré dans les VANETs due a (1) la nature sans fil du médium de communication des réseaux ad hoc, le trafic est fréquemment sujet à des écoutes clandestines ; (2) le manque d'un mécanisme de contrôle d'accès permet à un noeud quelconque de prendre partie dans le réseau ; (3) la faible protection physique des noeuds qui signifie que certains des noeuds du réseau peuvent potentiellement être compromis.
4. **Les algorithmes de routages** : Pour appliquer n'importe quel protocole de routage des réseaux sans-fil à VANET, des modifications substantielles doivent être effectuées pour surmonter les problèmes d'inondation et de la grande mobilité des véhicules, la topologie dynamique, la courte durée de connexion et les déconnexions fréquentes [37].

## 2.6 Les contraintes de sécurité dans les VANETs

Due aux caractéristiques spécifiques des VANETs, de nombreuses contraintes doivent être prise en considération lors du développement des solutions de sécurité. Nous énumérons ci-dessous :

1. **L'impact de l'environnement** : une solution pour les VANETs doivent prendre en considération les différentes caractéristiques des environnements urbain, autoroutier et rural. Chaque environnement a ses propres caractéristiques. Par exemple, la disponibilité de la couverture réseau en milieu rural n'est pas aussi bonne que l'environnement urbain, la durée de connexion entre un véhicule et un RSUs dans une autoroute n'est pas comme dans un environnement urbain, etc. Lors du développement des solutions, il faut proposer des solutions adaptable aux trois environnements.
2. **Faible latence** : lorsque nous parlons d'échange des messages de sécurité dans les VANETs, le délai du message ne doit pas dépasser les 5 micro seconds, sinon, il sera inutile. Donc lors du développement d'un algorithme, ou d'un protocole pour les VANETs, il faut qu'il prend en considération ce critère.
3. **Taux d'erreur diminuer** : certains protocoles sont basés sur les probabilités et toute erreur peut affecter la vie des gens. Pour cela, les protocoles développes doivent assurer un taux d'erreur diminuer.

## 2.7 Les attaquants

Les attaquants dans les VANET sont l'un des intérêts fondamentaux des chercheurs. Ils ont obtenu de nombreux noms canoniques listés ci-dessous en fonction de leurs actions et de leurs objectifs :

1. **Selfish driver** : L'idée générale de la confiance dans les réseaux véhiculaires est que tous les véhicules doivent faire confiance au départ. Les véhicules sont autorisés à suivre les protocoles spécifiés par l'application, certains conducteurs tentent de maximiser leur profit du réseau,

quel que soit le coût du système en profitant des ressources du réseau illégalement. Un selfish driver (en français, conducteur égoïste) peut dire aux autres véhicules qu'il y a de la congestion sur la route, ils doivent donc choisir un itinéraire alternatif, de sorte que la route soit dégagée [41].

2. **Malicious attacker** : (en français, attaquant malicieux) il a des objectifs spécifiques. Il cause des dommages et des préjudices via des applications dans les VANETs [42].
3. **Pranksters** : l'attaquant fait les choses pour son propre plaisir ; tels qu'une DoS attaque ou altération des messages (avertissement de danger) pour provoquer une circulation routière par exemple [42].
4. **Greedy drivers** : ils essaient d'attaquer pour leur propre bénéfice. Par exemple : l'envoi d'un message d'accident peut provoquer des embouteillages sur la route ou l'envoi de faux messages pour libérer la route [42].
5. **Snoops/eavesdropper** : l'attaquant tente de collecter des informations sur d'autres ressources [42].

Ces attaquants peuvent être classé en attaquant interne ou externe, malveillant ou rationnel, passif ou actif.

### 2.7.1 Attaquant interne vs attaquant externe

L'attaquant interne est une entité dans le réseau, elle est les mêmes privilèges et les mêmes caractéristiques que les autres entités du réseau qui peut communiquer avec les autres membres du réseau. A contrario, L'attaquant externe nest pas un nœud dans le réseau. Généralement : (1) les attaques internes sont difficiles à détecter para port à les attaques externes. (2) l'attaquant externe est plus limité quant à la diversité des attaques que l'attaquant interne [43].

### 2.7.2 Attaquant malveillant vs attaquant rationnel

L'attaquant malveillant (ou malicieux) cherche à prouver une prouesse personnelle à travers la détection des points faible dans le réseau pour blesser des nœuds du réseau ou attaquer le système, il n' utilise aucune structure. A contrario, L'attaquant rationnel est professionnel, il a un objectif précis. L'attaquant malveillant est facile identifier, tandis que l'attaquant rationnel est très difficile à identifier [43].

### 2.7.3 Attaquant passif vs attaquant actif

L'attaquant passif écoute les messages transmis et attend jusqu'à ce qu'il capte une information utile pour son attaque. A contrario, L'attaquant actif agit sur les messages et les informations qui sont transmissent entre les noeuds. il peut les modifier, supprimer,... etc [43].

## 2.8 Les différents types d'attaques dans les VANETs

Les VANET sont vulnérables à différentes types d'attaques. Il est très important de classer ses attaques de sécurité par catégories ou selon leurs objectifs afin d'extraire des caractéristiques communs, qui peuvent être utile pour la compréhension des attaques et pour la proposition des solutions. Dans la littérature, il existe plusieurs types de classification. Nous allons présenter les attaques d'un point de vue service visé (Figure 2.1), par exemple : attaques liée a la disponibilité, la confidentialité, l'authentification, l'intégrité, et la non-répudiation.

### 2.8.1 Sur l'authentification

Afin d'avoir également un réseau VANET sécurisé, les réactions des véhicules aux événements doivent être basées sur des messages légitimes générés par des expéditeurs légitimes. L'authentification des expéditeurs et des messages est exposée à de nombreux types d'attaques, nous citons :

1. **'Sybil attack'** : dans l'attaque Sybil, le noeud attaquant est capable de revendiquer des entités multiples appelées noeuds sybils ou faux noeuds. Il envoie des messages avec des identités multiples aux autres noeuds du réseau. Ainsi, en se faisant passer pour ces différentes identités, le noeud malveillant pourra compromettre plus facilement le fonctionnement général du réseau de véhicules. L'attaque peut consister à donner l'illusion d'un embouteillage ou d'accident afin que les autres véhicules changent de chemin ou quittent la route pour le bénéfice de l'attaquant. Le noeud malveillant peut également injecter de fausses informations dans les réseaux via de faux noeuds fabriqués [44].
2. **'Falsified entities'** : Dans l'attaque des entités falsifiées, l'attaquant obtient un identifiant valide et passe pour un autre nœud légitime. Cela constitue une violation du processus d'authentification. Chaque entité ITS a un identifiant de réseau, ce qui permet de la distinguer des autres nœuds du système ITS. Par exemple, des points d'accès non autorisés (AP) peuvent être déployés le long des routes pour imiter des RSU légitimes et pour lancer des attaques contre les utilisateurs et les véhicules associés [45].
3. **'Replication'** : Lors d'une attaque de réplication, un nœud malveillant tente d'ajouter des nœuds dans le réseau. Ces types d'attaques utilisent l'identité d'un autre nœud présent légalement sur le réseau pour transmettre de faux messages sur le réseau [46].
4. **'GNSS spoofing'** : l'attaquant simule ou modifie les vrais signaux GNSS et les rediffuse. En faisant cela, l'attaquant peut modifier la solution PVT à sa guise [47].
5. **'Timing'** : sans altérer le contenu du message, l'attaquant applique cette attaque en ajoutant un délai qui amène l'utilisateur à recevoir le message en retard [48].
6. **'GPS Spoofing'** : en français, l'usurpation GPS, est une technique plus sophistiquée que le Jamming, dont les signaux envoyés par l'attaquant imite les vrais signaux de navigation provenant des satellites GPS. Sans aucune fonction de protection, un récepteur GPS est vulnérable à l'usur-

pation d'identité. Les signaux émis par le spoofer (dispositif de spoofing) non seulement bloquent les vrais signaux, mais provoquent également une estimation incorrectes de valeurs de position, de vitesse et de temps [49].

### 2.8.2 Sur disponibilité

Pour avoir un réseau VANET sécurisé, la disponibilité doit être assurer, surtout lorsqu'un nœud partage une information critique. La disponibilité d'un réseau VANET est exposée à de nombreux types d'attaques, nous citons :

1. **Déni de Service (DoS)** : est une famille d'attaques qui visent la disponibilité des services réseau, ce qui peut avoir des conséquences graves en particulier pour les applications VANET. En raison de leurs impacts, les attaques DoS sont classées comme une classe d'attaques dangereuses. Ils peuvent être effectués par des nœuds malveillants internes ou externes au réseau. Dans ces attaques, l'attaquant tente de bloquer les principaux moyens de communication et vise à interrompre les services, afin qu'ils ne soient pas accessibles aux utilisateurs légitimes. À titre d'exemple, inonder le canal de contrôle avec des volumes élevés de messages pour que les autres noeuds sur le réseau ne peuvent pas recevoir les messages de sécurité [42].
2. **DDoS** : 'Distributed Denial of Service', est une DoS attaque distribuée, commandée par un attaquant principal qui joue le rôle de «gestionnaire d'attaque» auprès d'autres agents qui peuvent également être victimes sans le savoir. DDoS inondent dans la plupart des cas le réseau et ses résultats sont toujours désastreux [42].
3. **'Jamming'** : en français brouillage, ou encombrement. C'est une attaque qui interfère avec les fréquences radios du réseau en émettant des signaux à une certaine fréquence, afin de provoquer un déni de service [50].
4. **Spamming** : cette attaque vise a augmenté la latence de la transmission et d'exhauster la bande passante du réseau, en envoyant des messages

de spam aux autres noeuds légitimes. Par exemple, un attaquant X diffuse des messages de spam (qui sont souvent des publicités) à un groupe particulier de véhicules [50].

5. **'Wormhole'** : (en français : 'trou de ver'), dans cette attaque, il est nécessaire d'insérer au moins deux noeuds malicieux. Le but de cette attaque est de tromper les noeuds voisins sur les distances. Généralement le protocole de routage cherche le chemin le plus court en nombre de sauts (hop). Dans le cas d'une attaque du trou de ver, les deux noeuds malicieux permettent d'atteindre un lieu éloigné avec un saut unique. Cette possibilité va tromper les autres noeuds sur les distances réelles qui séparent les deux noeuds, mais va surtout obliger les noeuds voisins à passer par les noeuds malicieux pour faire circuler les informations. Ainsi les noeuds malicieux qui forment le trou de ver vont se trouver dans une position privilégiée qui va leur permettre d'avoir une priorité sur l'information circulant à travers leurs noeuds proches [51].
6. **Black hole** : la notion de trou noir ('Trou noir' en français) correspond aux points d'un réseau (noeuds) qui font discrètement disparaître le trafic sans informer la source du trafic. Lors de l'examen d'un réseau informatique les trous noirs ne peuvent pas être détectés. Pour les trouver il faut analyser le trafic perdu.

### 2.8.3 Sur la confidentialité

1. **'Eavesdropping'** : Il s'agit d'une attaque passive sur la confidentialité du réseau. Les attaquants observent et recueillent silencieusement le trafic du réseau ou la position actuelle et les activités d'un véhicule particulier. La détection de tels attaquants est très difficile car ils ne donnent aucune réaction dans le réseau actuel [52].
2. **'Data interception'** : cette attaque consiste à écouter le réseau pendant une certaine durée puis à tenter d'analyser les données pour en extraire les informations utiles [53].
3. **Brute force** : Le véhicule de l'expéditeur doit envoyer ses données au

véhicule de destination en prenant l'aide d'un autre véhicule si le nœud de destination est loin de sa portée. Pour maintenir le secret, le véhicule expéditeur peut crypter ses données et les envoyer à destination via n'importe quel véhicule médiateur [52]. Si le véhicule médiateur est un véhicule malveillant, il tentera de décrypter les informations qu'il a reçu en utilisant une attaque brute force. L'attaque par force brute est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé de cryptage des informations. Il s'agit de tester, une à une, toutes les combinaisons possibles. Cette méthode est en général considérée comme la plus simple concevable. Elle permet de casser tout mot de passe en un temps fini indépendamment de la protection utilisée, mais le temps augmente avec la longueur du mot de passe.

### 2.8.4 Sur l'intégrité

1. **'Masquarading'** : (en francais, déguisement), est une attaque qui consiste à tromper les mécanismes d'authentification pour se faire passer pour un utilisateur autorisé, de façon à obtenir des droits d'accès illégitimes et ainsi compromettre la confidentialité et l'intégrité [54]. Par exemple, un attaquant peut prendre l'identité d'un véhicule de police, et tente de frauder un autre véhicule pour ralentir ou s'arrêter.
2. **'Replay'** : (en francais, attaque par rejeu), est une forme d'attaque réseau dans laquelle le noeud malveillante intercepte puis rejoue une transmission de données valide en passant par un réseau sans même pas les déchiffrer. En raison de la validité des données d'origine (qui proviennent généralement d'un véhicule légitime), les protocoles de sécurité du réseau traitent l'attaque comme s'il s'agissait d'une transmission de données normale.
3. **'Data alteration'** : ce type d'attaque brise l'intégrité des données en modifiant, en supprimant ou en altérant le contenu des messages. Autrement dit, c'est l'injection de faux messages [53].
4. **'Data temparing'** : (en français, La falsification de données) est l'acte de

modifier (détruire, manipuler ou éditer) délibérément des données via des canaux non autorisés.

### 2.8.5 Sur la non-répudiation

1. **'Loss of event traceability'** : c'est une attaque basée principalement sur l'effacement de l'action de trace (traçabilité) et crée une confusion pour l'entité d'audit [55].

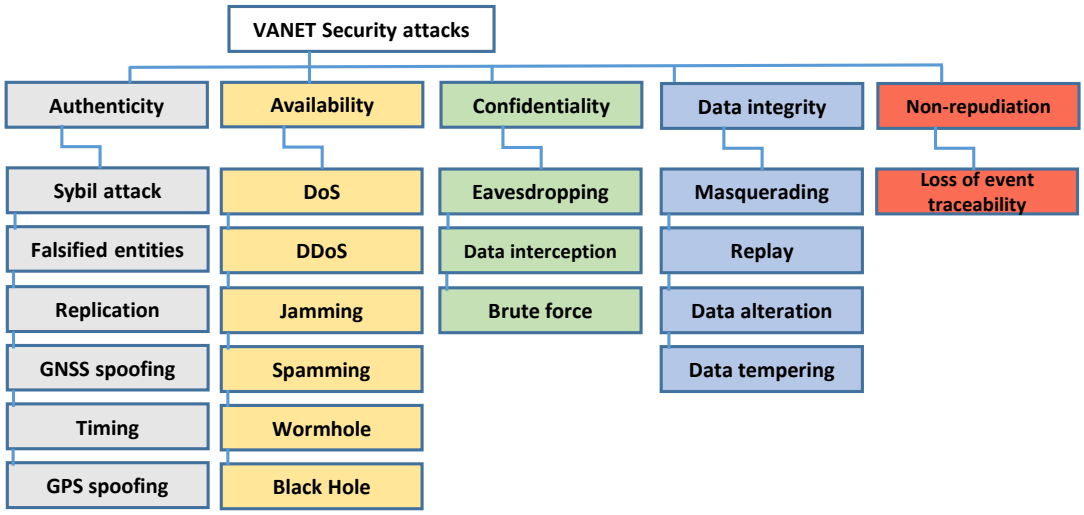


FIGURE 2.1 – *Classification des attaques de sécurité dans les VANETs [53]*

## 2.9 Conclusion

Dans ce chapitre, nous avons abordé le problème de la sécurité dans les VANETs au sens général. Afin de comprendre comment sécuriser ce genre de réseau, nous en avons présenté les principaux concepts et les aspects de sécurité : l'authenticité, la non-répudiation, la confidentialité, l'intégrité et la disponibilité. Ensuite, nous avons abordé et énuméré les différentes exigences et défis de sécurité généré due aux caractérestqiue spécifiques des VANETs. Après cela, nous avons présenté les différentes attaques de sécurité sur chaque services, ainsi les types des attaquants. Enfin, nous avons conclu en présentant une vue d'ensemble sur les mécanismes appropriés et les techniques crypto-

graphiques existantes qui peuvent apporter des solutions aux problèmes liés à la sécurité dans les réseaux VANETs.

Bien que la sécurité dans les réseaux VANETs ait attiré déjà beaucoup d'attention et suscité autant de travaux au cours de ces dernières années, elle reste toutefois l'un des principaux sujets controversés dans les applications des systèmes de transport intelligents. En effet, parmi les menaces les plus dangereuses, nous trouvons les botnets véhiculaires. Une menaces qui peut causer des scénarios catastrophiques, cependant, elle n'a été abordé qu'une seule fois dans la littérature. Pour cela, dans le chapitre suivant, nous allons présenter l'état de l'art à travers quelques travaux existant liée a la lutte contre les botnets en générale, et les botnets véhiculaires en spécifique, afin d'avoir une idée sur les solutions proposées.

# Les Botnets Véhiculaire et la Littérature

Sommaire

## 3.1 Introduction

Dans ce chapitre, nous allons présenté les botnets en générale (architecture, cycle de vie, et méthodes de communications). Ensuite, nous allons cités les différents méthodes de détections proposées. Après cela, nous présentons les botnets véhiculaires en spécifiques, et nous allons clarifié pourquoi les méthodes de détection standards ne peuvent pas être appliqué directement dans le contexte des botnets véhiculaires. Finalement, nous présentons en détailles le seul travail sur la détection des botnets véhiculaires, ainsi ses lacunes.

## 3.2 Les botnets

Depuis quelques années, les botnets sont de plus en plus utilisés par les cybercriminels à tel point qu'ils sont devenus l'une des plus grandes menaces de cybersécurité. Un Botnet est constitué d'un ensemble d'appareils connectés à internet, infectés et contrôlés par un même malware. Ces appareils peuvent être des PC, des serveurs, des smartphones et tablettes ou même des objets connectés. Une fois infectés par le malware, les appareils peuvent être contrôlés à distance par le cybercriminel à l'origine de l'attaque. Ils peuvent ensuite être utilisés pour des tâches spécifiques tels que les attaques DDoS sur des sites ou services concurrents, les campagnes de fraude au clic, l'envoi de spams, le vol des informations privées des utilisateurs/équipements, et l'application des activités intérieures sur les appareils infectés. Les Botnets peuvent aussi être utilisés pour la propagation de Fake News sur les réseaux sociaux. Par exemple, le botnet Mirai [56] en 2016, a pu mener une attaque DDoS massive qui a fait tomber des sites majeurs comme Amazon, Netflix, Paypal, et Reddit [57]. Les botnets se compose (1) de plusieurs bots, (2) d'un serveur de Command et Control (C&C), et (3) un botmaster.

— **Les Bots** (également appelés zombies) sont les machines infectées, et qui exécute les codes malveillants a l'insu de l'utilisateur. Ces machines traitent les tâches fourni par le serveur de C&C.

— **Le serveur C&C** est le point de ralliement central du botnet, et qui

assure la communication entre le botmaster et les bots. Comme ce serveur est capable de contrôler chaque bot, il doit être protégé contre les prises de contrôle par des tiers, comme par exemple les forces de l'ordre, les chercheurs ou les botmasters rivaux.

— **Le botmaster** est la personne qui contrôle les machines bots via le serveur C&C. Les botmasters essaient de rester aussi furtifs que possible, car démasquer l'identité du botmaster peut conduire à des poursuites potentielles.

### 3.2.1 Architecture

D'un point de vue typologique, l'échange des commandes entre le serveur C&C et les nœuds infectés peut être regroupé en trois catégories : centralisée, décentralisée (P2P), ou Hybride (centralisée et P2P a la fois).

#### *3.2.1.1 Centralisé*

Dans cette architecture, plusieurs serveurs C&C sont utilisés pour contrôler le réseau des botnets. Les noeuds infectés (bots) se communiques directement avec le serveur C&C pour avoir les instructions, ou bien rendre les rapports des attaques accomplies ; et le serveur C&C est contrôlé par son tour par l'opérateur (botmaster). Comme l'envoie des instructions est direct, le botmaster est capable de déclencher des attaques d'une manière rapide. En parallèle, le serveur C&C représente aussi le point faible de tout le réseau. Par exemple, une panne a son niveau pourra arrêter la connexion et l'échange des commandes avec tous les noeuds, ce qui implique une interruption du botnet, et aucune commandes ne pourra être envoyée ou reçue.

L'une des techniques les plus utilisées pour rendre la détection du communication entre le botmaster et les bots difficile, est d'utiliser des protocoles populaires au niveau réseau. Vu que ces protocoles sont également utilisés par les applications légitimes, le trafic réseau de la communication botnet sera similaire a celle du trafic réseau légitimes. Cette similarité ajoute des challenges de détection au système de détection d'intrusion (IDS) basé sur les signatures, et augmente le taux des faux positifs. [58].

Lorsque la structure est centralisée, le serveur C&C doit assurer l'échange des commandes avec tous les noeuds infectés du botnet. Cependant, si le nombre des noeuds est très grand, il se peut que le serveur C&C ne peut pas supporter une telle charge. Pour faire face a ce genre de challenge de gestion des botnets, les opérateurs utilisent plusieurs serveurs C&C au lieu qu'un seul. De cette façon, la charge des commandes échangées sera répartie entre eux, se qui rend le réseau du botnet plus scalable a l'échelle. En plus, le démantèlement du botnet sera difficile, car il faudra arrêter tous les serveurs.

Afin de rendre la détection de ce genre de structure des botnets plus difficile, les attaquants cachent les adresses des serveurs C&C utilisés, pour ne pas pouvoir appliquer des contre-mesures contre eux [58].

#### 3.2.1.2 *Peer-to-Peer (P2P)*

Dans la structure décentralisée ou pair-à-pair (P2P), l'échange des commandes ne s'effectue pas directement entre le serveur C&C et les noeuds infectés, mais a travers l'utilisation d'autres noeuds intermédiaire du botnet. Donc pour envoyer une commande, le serveur se connecte à n'importe quel nœud infecté du réseau. Contrairement à la structure centralisée, il n'y a pas de point de défaillance spécifique. La destruction de ce type de structure est très difficile, car même si certains noeuds infectés du botnet ne sont pas accessibles, d'autres parties peuvent toujours fonctionner normalement [58].

Cependant, cette structure est vulnérable aux infiltrations, comme le botnet Storm, qui a été infiltré par les chercheurs. Tout d'abord, ils ont travaillé sur la récupération de la clé de chiffrement, qui est nécessaire pour le déchiffrement et l'analyse des messages échangés entre le botmaster et les noeuds infectés. Dans une deuxième étape, ils se sont fait passer pour les opérateurs de botnet et ont envoyé des commandes aux noeuds infectés leur demandant de désinstaller le malware. Bien que l'opération d'infiltration de tels botnets puisse être menée à bien. Cependant, elle est conservée seulement pour lutter contre les botnets avec un très grand nombre de machines, qui représente une vrai menace, et qui mènent des attaques gênantes. En plus de ça, il est coûteux de récupérer les clés de chiffrement du botnet pour analyser la com-

munication et pouvoir reproduire les commandes, car les fichiers binaires du malware du botnet doivent faire l'objet d'une ingénierie inverse.

Puisque il n'y a pas d'entité centralisée, les noeuds infectées dans ce type de botnets doivent se connecter aux autres membres du réseau pair-à-pair pour rejoindre le botnet. Ils le font en se connectant à des points d'entrée appelés «super nœuds». Ceux-ci font partie du botnet et fournissent les adresses des autres membres aux ordinateurs nouvellement infectés. Cette opération s'appelle le ralliement dans les réseaux p2p.

La disponibilité en permanence des super noeuds est primordiale pour l'opération de ralliement de nouveaux machines infectées avec le botnet. C'est pour cela, que les noeuds nouvellement infectées doivent avoir leurs adresses IP ou leurs noms de domaines. Cette information pourra être transmit dans le logiciel malveillant du botnet. Par conséquent, si ces super noeuds ne sont pas accessibles, le botnet ne pourra pas accueillir de nouveaux membres. Les messages de contrôles et commandes envoyées par le botmaster dans une structure P2P mettent plus de temps à atteindre les noeuds infecté que dans une structure centralisée, car elles doivent passer par plusieurs nœuds intermédiaires avant d'atteindre leur destination.

Dans cette structure, les opérateurs du botnet contrôlent conjointement les nœuds infectés, et les commandes qu'il génère sont applicables à tous les noeuds infectés. Cependant, dans certains cas, il est facile de détecter les communications des botnets P2P, comme les réseaux d'entreprise qui interdisent l'utilisation du P2P [58].

#### 3.2.1.3 *Hybrid*

Vu que dans les botnets avec une structure centralisé, il est facile de bloquer la communication entre le serveur et les noeuds infectées (bots), seulement en attaquant le serveur, qui représente le point le défaillance de cette structure. Et de l'autre part, les botnets décentralisés sont exposés aux infiltrations, et difficile à contrôlé. Les opérateurs de botnet utilisent une structure hybride afin de contourner ces blocages potentiels [58].

Si aucune tentative n'est faite pour empêcher ou détruire le botnet, cette

structure hybride agira comme une structure centralisée. En revanche, si la machine infectée ne peut plus accéder au serveur de contrôle, le botnet passera en mode peer-to-peer. Les machines infectées sont connectées les unes aux autres pour créer un réseau peer-to-peer, auquel le botmaster peut accéder. Un botnet peut également modifier périodiquement sa structure pour réduire le risque d'être détecté ou améliorer la structure de son réseau peer-to-peer. Cette structure hybride est adoptée par le botnet Zeus [58].

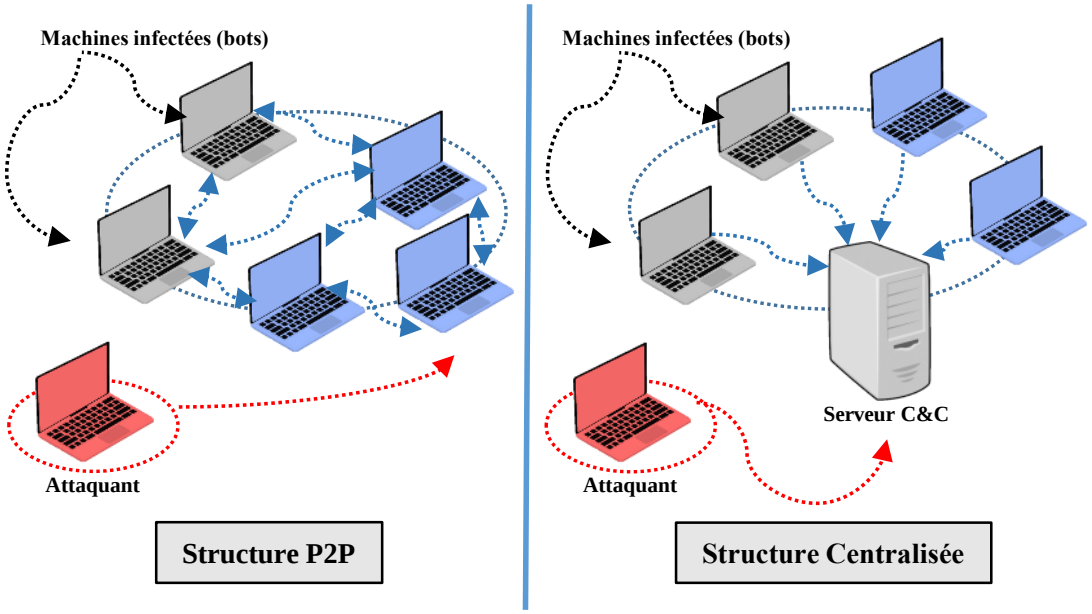


FIGURE 3.1 – *Structures de botnets : centralisée et pair-à-pair [58]*

### 3.2.2 Cycle de vie

Pour qu'un botnet soit profitable à son opérateur, il doit contenir des machines infectées à contrôler qui exécutent le logiciel malveillant de ce botnet. Ce logiciel permet de corrompre et de passer la machine qui l'exécute sous le contrôle de l'attaquant pour déclencher des attaques ou pour intercepter les données de la victime. Les machines infectées sont connectées en permanence avec le C&C du botnet, afin de recevoir les commandes d'attaques, et pour envoyer les données dérobées et les comptes-rendus des attaques accomplies. De plus, l'opérateur du botnet met à jour en permanence son réseau et son logiciel malveillant, afin d'ajouter de nouvelles fonctionnalités à son botnet ou pour contrecarrer les opérations de remédiation [58].

Le cycle de vie ordinaire d'un botnet est constitué de : la création, l'infection, la communication avec le serveur de C&C, les attaques, et la mise à jour. Ceci est illustré dans la figure 3.2.

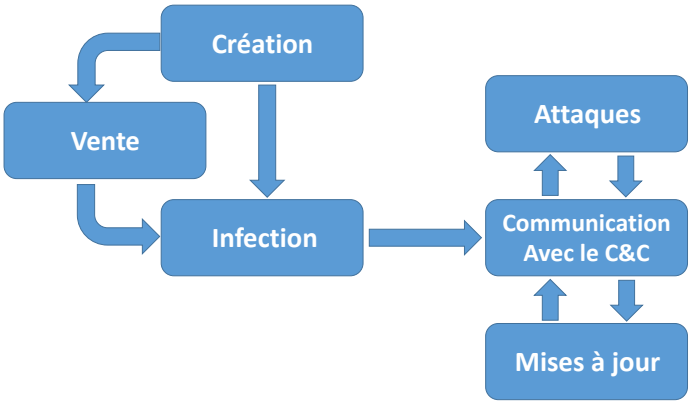


FIGURE 3.2 – *Les différentes étapes du cycle de vie d'un botnet [58]*

#### 3.2.2.1 *Création*

L'histoire des botnets remonte à Eggdrop [59], en 1993. Ce bot n'avait pas d'intention malveillante. Il a été créé afin d'aider à la gestion des canaux de discussion dans les serveurs IRC [60]. Les bots Eggdrop reçoivent et interprètent des commandes afin de fournir des services aux utilisateurs du canal [58].

Plusieurs botnets basés sur le protocole IRC sont apparus à la fin des années 1990. Toutefois, le tournant majeur dans le développement des botnets est la diffusion du code source de logiciels malveillants tel que Agobot [61], SDBot [62], et GT-Bot [63]. La particularité du code source de ces logiciels est qu'il était possible et sans grande difficulté d'ajouter de nouvelles fonctionnalités au botnet initial.

Les attaquants se sont basés sur ces codes sources afin de développer de nouveaux botnets. Ainsi, des milliers de variantes du botnet Agobot ont été détectées par les solutions d'antivirus moins d'une année après la diffusion de son code source. De plus, les créateurs de botnets de type IRC se sont basés directement sur le code des clients et des serveurs IRC légitimes [58].

Avec la popularité du web, les attaquants se sont tournés vers le protocole HTTP pour le canal de contrôle du botnet. Ils se sont basés sur l'architecture

des serveurs web pour leur serveur de contrôle et commande C&C. L'utilisation des serveurs web leur permet d'assimiler leur trafic de contrôle au trafic légitime et de ne pas attirer l'attention des systèmes de détection. En effet, ils génèrent le même type de trafic réseau que les utilisateurs légitimes [58].

Une autre étape importante dans le développement des botnets est la divulgation du code source du botnet bancaire Zeus en mai 2011 [64]. Comme la divulgation du code source de botnets IRC a engendré plusieurs variantes, plusieurs botnets se sont basés sur son code source telles que Citadel [65] et GameOver [66]. La divulgation du botnet Carberp [67], plus évolué que Zeus, ferra vraisemblablement l'objet de plusieurs variantes dans le futur [58].

Les développeurs de botnets sont de moins en moins impliqués dans la gestion journalière des machines infectées. Ils développent leur botnet avec un haut niveau d'abstraction, afin de pouvoir revendre leur code à un nombre important de clients. En effet, un même code peut engendrer un nombre important de botnets avec des ensembles de machines infectées disjoints et des serveurs de contrôle différents [58].

#### 3.2.2.2 *La Vente*

L'émergence des kits de création de botnets prêts à l'emploi a introduit l'étape de vente dans le cycle de vie d'un botnet. Ainsi, le premier kit de création de botnets de type Zeus était vendu à plusieurs milliers de dollars à la fin des années 2000. Pour ce prix, l'acquéreur disposait d'un générateur permettant de créer des variantes du logiciel malveillant servant à infecter les machines, et un site web configurable permettant de contrôler le botnet [58].

Afin d'empêcher la génération frauduleuse du botnet, les développeurs des kits de création de botnets ont intégré des mécanismes pour empêcher la distribution ou la revente de leurs kits. Les développeurs du botnet Zeus ont mis en oeuvre le même procédé pour protéger leur produit que celui utilisé pour l'activation des produits Microsoft. Ainsi, l'acquéreur de ce kit doit fournir au préalable la configuration matérielle et logicielle de sa machine au vendeur. Par conséquent, il est le seul à pouvoir utiliser le kit et créer un botnet [58].

Après l'apparition de Zeus, plusieurs autres kits de création de botnets sont apparus, dont le plus connu est SpyEye [68], concurrent direct de Zeus. Il intégrait dans son code une fonctionnalité pour supprimer Zeus de la machine de la victime avant de s'installer lui-même [58].

En parallèle des kits de création de botnets destinés à un large public, certains sont développés sur mesure pour certains clients. Par exemple, le botnet Mariposa [69] a été développé par un Slovène, mais était opéré par des Espagnols. L'avantage de ne pas passer par des kits de création de botnets est d'éviter de partager des signatures, sur le logiciel malveillant ou sur le trafic réseau, avec les autres botnets créés avec le même kit. En effet, le canal de contrôle du botnet Mariposa était basé sur un protocole réseau propriétaire [58].

#### 3.2.2.3 *L'infection*

Pour avoir des machines sous leur contrôle, les opérateurs de botnets doivent exécuter leur logiciel malveillant sur les machines des victimes. Ce logiciel malveillant permet à l'opérateur du botnet de prendre le contrôle de la machine, autrement dit, de l'infecter. Ainsi, cette dernière rejoint le botnet et attend la réception de commandes d'attaque.

L'infection d'une machine peut s'effectuer de manière directe ou indirecte. Dans la méthode indirecte, un programme d'infection achemine le logiciel malveillant et lance son exécution dans la machine de la victime. Ce programme d'infection est appelé "dropper" puisqu'il permet de déposer le logiciel malveillant. Par contre, l'infection directe n'utilise pas de "dropper". La victime exécute directement le programme malveillant dans sa machine et rejoint le botnet [58].

Afin d'éviter que le programme malveillant ne soit détecté lors de l'infection par les solutions de sécurité qui se basent sur la signature des fichiers, les opérateurs de botnet chiffrent le contenu du fichier binaire de leur logiciel malveillant. Ceci permet d'empêcher les solutions d'antivirus et les systèmes d'intrusion de lire les instructions contenues, et ainsi d'inférer le comportement du logiciel malveillant. Les solutions de chiffrements les plus utilisées

sont appelées "Packer" [70]. Elles permettent de retarder le déchiffrement des instructions du programme malveillant jusqu'à son chargement dans la mémoire de la machine de la victime [58].

Une autre parade pour contourner les mécanismes de détection des programmes indésirables, consiste à signer ces programmes malveillants avec des certificats légitimes. En effet, des opérateurs de botnets ont signé leur programme malveillant avec des certificats de programmes légitimes volés.

Les techniques d'infection varient considérablement. On décrit ci-dessous les principales techniques :

— *Vulnérabilité logicielle* : Lors de cette infection, la machine ciblée est introspectée à distance afin d'obtenir des informations sur les services en cours d'exécution, ainsi que les différents ports réseau ouverts. Si elle détecte un service vulnérable est en cours d'utilisation, le programme d'infection essaie de l'exploiter et d'exécuter le programme malveillant du botnet [58].

— *Drive-by-Download* : le programme d'infection est téléchargé et exécuté par la victime lors de la consultation d'une page web compromise. Elle consulte cette page, soit parce que cette dernière est légitime, mais a été piratée, ou parce qu'elle a été redirigée vers cette page par le biais d'un spam ou d'ingénierie sociale. Le programme d'infection est téléchargé soit de manière automatique soit de manière manuelle. Il est téléchargé et exécuté automatiquement et sans informer la victime en exploitant une vulnérabilité dans le navigateur ou dans une des extensions installées. Le programme d'infection est téléchargé et exécuté manuellement par la victime parce que l'attaquant l'a fait passer pour un programme légitime. Par exemple, le programme d'infection peut se faire passer auprès de la victime pour un lecteur multimédia pour lire une vidéo proposée sur la page compromise [58]. Afin d'automatiser le processus de recherche et d'exploitation des vulnérabilités des navigateurs des victimes potentielles, les attaquants utilisent des kits d'exploits tels que Blackhole ou Eleonore.

— *Infection par média externe* : Dans cette infection, le programme mal-

veillant est transporté dans un média externe. Lorsque ce média est connecté à une machine, le programme malveillant est exécuté automatiquement et le système devient corrompu et rejoint le botnet [58]. La particularité de cette infection est que les médias qui seront branchés à l'avenir au système corrompu transporteront à leur tour ce programme malveillant et pourront infecter d'autres machines.

— *Cheval de Troie* : L'attaquant dissimule un code d'infection dans un programme ou un fichier légitime. Lorsqu'une machine exécute le programme légitime, le code d'infection est exécuté en parallèle. Par conséquent, le logiciel malveillant est installé et la machine rejoint le botnet . En effet, le programme d'infection du botnet CryptoLocker de type scareware est dissimulé dans des programmes légitimes partagés dans les réseaux d'échange de fichier pair-à-pair [58].

— *Surinfection* : Certains attaquants se sont spécialisés dans l'infection des machines. Ils vendent les machines infectées aux opérateurs d'autres botnets. Ce service est appelé Pay Per Install (PPI). Le prix des machines infectées dépend de leur position géographique. Une machine dans un pays riche est vendue plus cher qu'une machine d'un pays en développement puisque l'attaquant peut espérer en retirer plus de profits [58].

#### 3.2.2.4 *Communication avec le serveur de C&C*

Dès que la machine est infectée, elle essaye de communiquer avec son serveur de contrôle afin de le notifier de son infection et pour recevoir la dernière configuration. Les machines infectées sont en contact permanent avec leur serveur de C&C, afin de recevoir les dernières commandes d'attaque ou pour envoyer le résultat des attaques en cours. Le canal utilisé par les machines infectées pour communiquer avec leur serveur de contrôle est appelé canal de contrôle [58].

#### 3.2.2.5 *Attaques*

Le but de l'attaquant disposant du contrôle d'un botnet est d'utiliser les machines infectées pour déclencher différentes attaques. Les machines infectées reçoivent les commandes d'attaques de la part du serveur de C&C, qui est contrôlé à son tour par l'attaquant [58]. On peut regrouper les attaques générées par les botnets en deux catégories principales :

— *Attaques bruyantes* : Le trafic réseau généré lors de ces attaques peut facilement être identifié par un système d'analyse de trafic réseau. En effet, les machines infectées génèrent un trafic réseau malveillant important. On peut citer par exemple : le scan ou le balayage du réseau à la recherche de machines vulnérables, l'envoi massif de courriers indésirables, et la participation dans des attaques de déni de service [58].

— *Attaques furtives* : Contrairement aux attaques précédentes, les attaques furtives ne génèrent pas de trafic réseau important. Ainsi, ces attaques sont difficilement détectables par une solution d'analyse de trafic réseau. Elles surviennent essentiellement à l'intérieur du système de la victime. On peut citer par exemple : le vol de données des utilisateurs, et la fraude à la publicité en ligne [58].

#### 3.2.2.6 *Mise à jour*

L'opérateur du botnet entretient son réseau en envoyant continuellement aux machines infectées des mises à jour du programme malveillant. Ces mises à jour peuvent contenir de nouvelles fonctionnalités, des corrections de vulnérabilités, ou une amélioration de la méthode de communication du botnet [58].

### 3.2.3 Méthodes de communication

Les dispositifs infectés établissent une connexion avec un serveur de contrôle et de commande (C&C), cette connexion sert à la réception d'instructions à exécuter et à l'envoi de rapports sur les attaques en cours ou accomplies. Afin d'éviter la détection par les systèmes de détection d'anomalies

dans le trafic réseau, les communications sont faites en utilisant des protocoles réseau ordinaire, ainsi, le trafic génère ne soit pas très différent du trafic légitime.

Les machines infectées utilisent majoritairement le protocole DNS pour récupérer l'adresse IP du serveur de contrôle et commande du botnet. Ainsi, avant d'établir la connexion avec le serveur de contrôle, elles envoient des requêtes vers le serveur DNS. L'utilisation du protocole DNS permet d'ajouter plus de robustesse au botnet afin de contourner les dispositifs de blocage mis en œuvre au niveau du réseau [58]. Après avoir obtenu l'adresse IP de leur serveur de C&C, les machines infectées se connectent au serveur en utilisant des protocoles ordinaires.

Les premiers botnets utilisaient les protocoles IRC (Internet relay Chat) pour monter le canal de contrôle. Cependant, avec l'émergence du web et la facilite du développement d'un serveur web ont motivé les concepteurs de botnets à utiliser les protocoles HTTP.

Finalement, les opérateurs de botnet chiffrent les communications entre les serveurs de C$C et les botnet pour dévier les méthodes de détection basées signature.

#### *3.2.3.1 Détermination de l'adresse du serveur de C&C (Protocole DNS)*

Dans les premiers botnet, l'adresse IP du serveur de C&C était codée d'une manière statique dans le code source du logiciel malveillant. Cette méthode représentait plusieurs inconvénients. Ainsi, les machines infectées ne peuvent plus se connecter au serveur de C&C si ce dernier tombe en panne ou soit inaccessible a cause d'une opération de blocage. De plus, l'utilisation d'adresse IP statique ne permet pas à l'opérateur du botnet de changer aisément l'emplacement du serveur de contrôle [58].

Afin de résoudre ce problème, les développeurs de botnets ont utilisé le protocole DNS afin de permettre aux machines infectées de localiser leurs serveurs de C&C d'une manière dynamique. Il suffit juste d'avoir le nom du serveur de C&C et d'envoyer une requête au serveur DNS afin de récupérer l'adresse IP du serveur de C&C. L'utilisation du protocole DNS permet à

l'opérateur du botnet d'ajouter plus de résilience à son serveur de C&C. En effet, il peut changer l'emplacement du serveur de C&C ou rajouter d'autres serveurs afin de contrôler ses botnets sans mise à jour du logiciel malveillant.

L'opérateur du botnet intervient uniquement au niveau du serveur DNS en modifiant le mapping entre le nom de domaine du serveur et son adresse IP. Ainsi, la machine infectée continue de communiquer avec le serveur DNS pour obtenir l'adresse IP du serveur de C&C [58].

#### 3.2.3.2 *Le canal de contrôle*

Après l'obtention de l'adresse IP du serveur de C&C, les machines infectées établissent un canal de communication avec ce dernier, ce canal est appelé le canal de contrôle du botnet, ainsi, les machines infectées peuvent commencer à recevoir des commandes et envoyer des rapports sur les attaques exécutées. Plusieurs protocoles de communications sont utilisés :

— **Protocole IRC :** c'est protocole de messagerie instantanée très utilises au moment de l'apparition des premiers botnets (fin des années 90). Dans ce scénario, les machines infectées reçoivent les commandes de l'opérateur botnet à travers un serveur de messagerie IRC. En partant du principe que le protocole IRC est basé sur des salons de communication ou des canaux, lorsque l'opérateur botnet envoie une commande à travers un canal de communication, toutes machines infectées connectées à ce canal reçoivent cette commande. Les botnets IRC peuvent enclencher des attaques rapides par ce qu'ils sont connectés en permanence au serveur de C&C et reçoivent instantanément les commandes envoyer par l'opérateur botnet à travers de canal de communication [58].

— **Protocole HTTP :** c'est l'un des protocoles les plus répandus et les plus utilisé au cours des dernières années, ce qui a motiver les développeurs de botnets pour utiliser protocole HTTP comme alternative au protocole IRC. Contrairement aux botnets de type IRC, quand les botnets de type HTTP evoie un rapport à l'opérateur botnet, le rapport et reçu uniquement par ce dernier, et ne sont pas reçu par les autres machines

infectées. Ainsi, dans un botnet de type HTTP, seul l'attaquant peut connaître le nombre de machines infectées sous son contrôle [58].

— **Autres Protocoles :** À la différence de la plupart des botnet utilisant les protocoles IRC et HTTP, d'autres botnets utilisent des protocoles personnalisés. Par exemple, le botnet *Mariposa* utilise le protocole Iserdo Transport Protocole situé au-dessus de la couche UDP, et les botnets P2P qui utilisent aussi des protocoles situés au-dessus de la couche UDP. La raison principale derrière l'adoption de protocoles personnalisés est de s'échapper aux systèmes de détection de botnet basés sur des signatures spécifiques aux protocoles IRC et HTTP. Toutefois, l'utilisation de protocoles personnalisés peut avoir des inconvénients, par ce que les pare-feu peuvent être configurés pour bloquer les protocoles inconnus, ce qui rend les botnet utilisant ce genre de protocoles inutilisables[58].

#### 3.2.3.3 *Chiffrement des communications*

Les premiers botnets communiquaient avec les serveur C&C en texte clair sans utilisation de chiffrement, ce qui fait que les communications pourraient être interceptées et analysées par d'autres entités. De plus, elles peuvent être détectées et bloquées facilement par les systèmes de détection d'anomalies dans le trafic internet ainsi que les systèmes de détection de botnets. Ces derniers, basent leur détection sur les signatures des messages échangées entre les serveur de C&C et les botnets (Commandes envoyées et rapports sur les attaques accomplies). Les développeurs de systèmes de détection de botnets génèrent les signatures de ces derniers en exécutant des botnets dans des environnements contrôlés. Ceci leurs permet aussi d'examiner les différents types des messages non chiffres échangés entre les botnets et les serveurs de C&C, communiquer avec ces derniers en se faisant passer par des botnets ou bien communiquer avec les autres botnets en se faisant passer par des serveurs de C&C afin d'obtenir plus de renseignements sur les attaques planifiées et les attaques accomplies.

La raison principale pour laquelle les chercheurs en sécurités se font pas-

ser par des serveurs de C&C est la possibilité d'envoyer des commandes aux botnets afin de désinstaller les logiciels malveillants dans les machines infectées.

Pour empêcher l'analyse des communications, les développeurs de botnets ont commencé à utiliser des algorithmes de cryptage afin de chiffrer les échanges entre les serveurs de C&C et les botnets. Ces deux derniers partagent une clé de chiffrement qu'ils utilisent pour chiffrer la totalité du canal de communication ou bien les messages échangés seulement selon le choix du développeur de botnets [58].

## 3.3 Les botnets dans la littérature

Dans cette section, nous allons présenté en détailles les différents techniques de détection des réseaux de botnets.

### 3.3.1 Détection basée sur les Honeypots

Un honeypot (pot de miel) est une ressource du système d'information dont la valeur réside dans l'utilisation non autorisée ou illicite de cette ressource (autrement dit, des systèmes vulnérables en attente d'attaques). L'idée derrière cette méthodologie est d'attirer les attaquants tels que les logiciels malveillants automatisés et de les étudier ensuite en détail. L'objectif de cette étude est de générer des signatures de trafic réseau malveillant pour des systèmes de détection d'intrusion. Les pots de miel se sont révélés être des outils très efficaces pour en apprendre davantage sur la criminalité sur Internet comme les réseaux de botnets. Il existe deux types de pots de miel :

1. ***Honeypots à faible interaction*** : ce type ne simule qu'une partie applicative sur une machine physique ou virtuelle, par exemple une pile TCP/IP, et des services réseaux. La mise en œuvre de ce type de honeypots est généralement peu risquée, l'intention principale est de capturer des échantillons de code nuisibles. Un exemple populaire de ce type de honeypots est le Nepenthes [71].

2. ***Honeypots à forte interaction*** : ce type de honeypots simule un système d'exploitation complet ainsi que les applications souhaitant être soumis aux pirates de passage ; ce qui signifié que le pot de miel peut être compromis entièrement. Et pour le récupéré, l'administrateur n'aurait plus vraiment de solutions que de réinstaller la machine. Le risque de déploiement tend à être plus élevé, il est donc nécessaire de mettre en place des précautions et des dispositions spéciales pour prévenir les attaques contre le système. La mise en place de ce type est plus complexe ainsi que sa maintenance. L'intention principale est de comprendre la scène de l'attaque. La configuration la plus courante pour ce type de honeypots est Honeynet GenII [72].

De nombreux articles ont utilisé les honeypots pour étudier les différentes technologies de logiciels malveillants (bot malwares), par exemple, dans [73, 74] les auteurs ont utilisé des honeypots à faible interaction pour étudier comment les bots malwares entrent en contact avec les appareils domestiques IoT par leurs services d'accès à distance et les activités et logiciels malveillants qu'ils tentent d'introduire dans ces systèmes. Ils ont aussi essayé d'obtenir des enregistrements des activités des bots malveillants dans le monde entier, dans le but d'obtenir et l'analyse des informations sur les réseaux de bots les plus actifs et sur le type des logiciels malveillants qu'ils utilisent.

Cependant, en raison de la simplicité des pots de miel à faible interaction, le principal inconvénient est la probabilité plus élevée d'être détecté comme artificiel par l'attaquant, surtout si celui-ci est un humain. L'attaquant peut constater que les services réels sont imités et que les services ne sont que partiellement mis en œuvre, et donc perdre tout intérêt. Les honeypots à faible interaction ne sont pas, par construction, optimaux pour capturer les vulnérabilités de type "zero-day", car de par leur nature, ces vulnérabilités sont inconnues au moment du déploiement du pot de miel, et ne peuvent donc pas être simulées [75].

D'autre part, plusieurs recherches ont été mené sur l'utilisation des honeypots a forte interaction, tel que [76, 77, 78, 79]. Dans [76] par exemple, les

auteurs ont conçu un pot de miel distribué et évolutif à haute interaction (SIPHON). Leur conception permet de représenter les dispositifs IoT physiques dans un seul laboratoire comme étant répartis géographiquement en établissant des tunnels entre les adresses IP publiques et les dispositifs physiques. Il permet également de collecter le trafic pour un traitement et une analyse ultérieurs. Ils ont mis en œuvre un prototype de SIPHON utilisant cinq caméras IP, un NVR, mettant ainsi 85 dispositifs IoT à la disposition des attaquants dans le monde entier. Leur mise en œuvre leur a permis de recueillir plusieurs gigaoctets de données de trafic qu'ils ont préalablement analysées, notamment du point de vue des lieux les plus attractifs pour l'attaquant.

Cependant, Les pots de miel peuvent générer des signatures du trafic réseau malveillant uniquement pour les botnets qui ne chiffrent pas leur canal de contrôle et qui sont réactifs à la réception d'une commande. La réaction d'un bot doit être identifiable au niveau de son trafic réseau. Nous pouvons citer comme réaction identifiable l'envoi de spam ou la participation à des attaques par déni de service. Si le botnet n'est pas réactif, le système peut ne pas associer des attaques aux commandes reçues par les machines infectées et ainsi ne pas générer de signatures [58].

### 3.3.2 Détection basée sur le trafic réseau

La détection basée sur le réseau est l'une des techniques de détection les plus efficaces qui a attiré l'attention de nombreux chercheurs [80, 81, 82, 83, 84]. En analysant le taux d'échec des connexions et les caractéristiques du flux réseau, la technique de détection basée sur le réseau identifie le trafic échangé entre le serveur C&C et les bots. Cette technique de détection peut être divisé en deux sous-catégories : (i) détection basée sur les signatures et (ii) détection basée sur les anomalies. dans cette section, nous allons discuté en détails les techniques de détection mentionnée ci-dessus. Ainsi, nous allons cité de différentes recherches qui ont été mené sur ces techniques.

#### 3.3.2.1 *Détection basée sur la signature*

Les privilèges d'utiliser des systèmes de détection d'intrusion de botnets qui se reposent sur des signatures est le taux de faux positifs qui est très diminuer ; ceci est due a l'utilisation d'une base de signatures de détection qui peut être adaptée facilement au trafic collecté. Par exemple, Snort [85] surveille le trafic pour déterminer la signature d'un bot existant, ce qui permet de déterminer immédiatement les bots, mais seulemnts les connus. On trouve aussi l'approche de liste noire basée sur le DNS (DNSBL) [86] qui surveille le trafic DNS afin de découvrir l'identité des bots, en se basant sur le fait que les botmasters doivent effectuer des recherches de "reconnaissance" pour déterminer le statut de leurs bots. Cependant, dû au grand nombre d'attaques réseau découvertes régulièrement et qui peuvent être causé par un bot malware, dû à la difficulté d'analyser rapidement ces attaques avant d'en générer des signatures de détection, et aux multiples formes qu'une attaque peut avoir, il est quasiment impossible de garder la base des signatures a jour. En plus de ça, les systèmes de détection d'intrusion basé sur les signatures ne peuvent détecter que les attaques dont leurs signatures est inclus dans la base des signatures, et laissera donc passer celles qu'il ne connaît pas.

#### 3.3.2.2 *Détection comportementale*

Vu de la difficulté de mettre en place un système de détection de botnets qui reposent sur les signatures, et le nombre grandissant d'attaques sur les réseaux qui peuvent les causés, les systèmes de détection d'intrusion qui identifient les anomalies, ont fait l'objet de beaucoup de recherches [87, 88, 89, 90, 91]. Ces systèmes ont embarqué des mécanismes d'apprentissage automatique supervisé et non supervisé. Pour les systèmes basés sur de l'apprentissage supervisé, le principe général est de définir en premier lieu le comportement "normal" d'un réseau avec un modèle formel. Lors de la phase de détection, toute activité réseau sera identifiée comme étant anormale si la déviation par rapport au modèle dépasse un certain seuil. Pour les systèmes basés sur l'apprentissage non supervisé, ils améliorent la fiabilité de détection face à de nouvelles menaces non répertoriées. Dans [88] [89], les auteurs

ont travaillé sur la detection des botnets qui visent le système d'exploitation Android. Ils ont proposé des méthodes de détection comportementales qui visent à détecter les applications malveillantes en surveillant les différentes couches du système d'exploitation Android (appels API, ...). Nous trouvons aussi d'autres recherches qui ont été mené sur la détection des botnets IoT en utilisant des techniques de détection d'anomalies non supervisées afin de détecter les attaques de botnets invisibles [90, 91].

Cependant, l'application des techniques de détection comportementale ou de signature directement sur les botnets véhiculaires n'est pas possible, vu les différences entre l'architecture des botnets véhiculaires et les autres types de réseaux de zombies.

## 3.4 Les botnets véhiculaire

L'histoire a montré que lorsque des capacités de calcul et de communication s'ajoutent à un nouveau environnement, une partie des nouveaux équipements serait inévitablement compromise par les attaquants, comme le cas d'ajout des capacités de calcul aux automobiles et les faires relier entre eux avec un réseau. Les chercheurs ont montré que les ordinateurs et les réseaux internes des voitures peuvent être compromis et exploités, permettant ainsi différents degrés de contrôle des véhicules. Dans certains cas, les véhicules peuvent être compromis à distance, en utilisant des connexions réseau sans licence V2V. D'autre part, ils peuvent être infectés a travers des téléchargements internet imprudents. À mesure que les capacités cybernétiques des voitures augmentent, la surface d'attaque qu'elles présentent et le nombre accru de cibles attireront certainement les cyberattaquants, et certaines attaques réussiront à compromettre les véhicules. À mesure que les ordinateurs internes prennent un contrôle de plus en plus important sur les capacités des véhicules, la récompense qu'une attaque réussie peut récolter ne fera que rendre la compromission de ces derniers une cible plus attrayante. Les recherches existantes ont déjà prouvé qu'il est possible de compromettre ces véhicules connectées. Un véhicule compromis de ce type offre non seulement des ca-

pacités de calcul et de communication, mais aussi la possibilité d'effectuer des actions importantes dans le monde réel qui pourraient avoir de graves conséquences.

Il est très probable que les constructeurs automobiles prennent des mesures pour améliorer la sécurité de ces véhicules, car leur popularité et leur vulnérabilité augmentent. Toutefois, d'après l'expérience acquise avec tous les types de systèmes, ces mesures ne suffiront pas à empêcher que les véhicules soient compromis. C'est pourquoi toute recherche sur la sécurité et la confidentialité des réseaux de communications mobiles doit être préparée en tenant compte de l'existence de ces véhicules compromis. Une certaine attention a déjà été accordée à la sécurité des communications des réseaux de communication à valeur ajoutée (VANET), en particulier pour s'assurer que les véhicules peuvent fournir une authentification forte pour leurs messages, ce qui est important pour les exigences de non-répudiation et de responsabilité des réseaux de communication à valeur ajoutée. Bien qu'une authentification correcte soit une étape nécessaire pour fournir de nombreuses solutions de sécurité, les solutions proposées par les recherches existantes ont été principalement orientées vers la prévention des fraudes et des mensonges aux autres véhicules et aux infrastructures routières. Ces recherches offrent un certain moyen de pression contre les agresseurs qui compromettent un seul véhicule. Cependant, l'histoire a également montré que les attaquants qui réalisent qu'ils peuvent compromettre une seule machine passent rapidement à la compromission de plusieurs machines, et à l'organisation de leur ensemble de machines compromises en une ressource distribuée d'une puissance totale plus grande que ses parties individuelles.

Il faut donc s'attendre à ce que l'avenir nous apporte des réseaux de zombies véhiculaires, des collections de véhicules autonomes ou largement contrôlés par ordinateur et en réseau, sous le contrôle coordonné d'attaquants à distance. Cet avenir soulève de nombreuses questions importantes qui n'ont pas encore été examinées, et encore moins répondues. Que ferait un attaquant avec un tel réseau de zombies véhiculaires ? Comment utiliserait-il ses actifs compromis pour atteindre ses objectifs ? Comment pouvons-nous déterminer

qu'un réseau de zombies véhiculaires est à l'œuvre ? Comment pouvons-nous identifier ses membres et quelles mesures pouvons-nous prendre pour annuler les dommages qu'ils peuvent causer ? Pouvons-nous nous défendre contre de tels réseaux de véhicules malveillants sans identifier et "désinfecter" les véhicules ? Notre travail est seulement le deuxième travail dans la littérature qui pour but de répondre à ces questions. Comme les chercheurs ont déjà montré que les véhicules autonomes peuvent être compromis, nous ne ferons pas d'autres recherches sur la manière de les compromettre, mais nous supposerons plutôt que nous disposons déjà d'un ensemble de ces véhicules compromis lorsque nous ferons nos expérimentations.

## 3.5 Les botnets véhiculaires dans la littérature

Les mécanismes de détection des botnets existants en générale, ne peuvent pas être appliqués directement aux réseaux véhiculaires due aux différences entre les réseaux véhiculaires et les autres types de réseaux en termes de pile de communication, de protocoles, de format de trame et d'architecture. Dans le contexte des réseaux véhiculaires, à notre connaissance, il n'existe qu'une seule recherche [1] qui s'est attaquée au problème de la détection des réseaux de botnets véhiculaires.

### 3.5.1 Ghost

Garip et al. [1] ont proposé un protocole de communication pour les botnets véhiculaires, appelé Ghost, et qui tire profit de l'infrastructure des réseaux véhiculaires existante pour contrôler les véhicules bots et facilité la collaboration entre eux. Les auteurs ont exploité le fait que l'infrastructure de communication des réseaux véhiculaires se repose sur deux canaux sans fil différents : les canaux de contrôle et de service. Les messages de sécurité de base (BSM) sont diffusés sur le canal de contrôle ; les autres applications non liées à la sécurité utilisent le canal de service. L'utilisation du canal de service pour échanger de nombreux messages de botnet entre un ensemble particulier de véhicules serait suspecte, c'est pourquoi les auteurs ont décidé de cacher

la communication du botnet véhiculaire à la vue de tous, en injectant les commandes dans les messages BSM. Ces messages sont déjà inondés sur le réseau, mais aucun véhicule, à l'exception des véhicules bots, ne connaîtra l'existence de ces messages secrets et ne pourra les décoder. L'idée des auteurs était de concevoir un mécanisme d'injection de commande, d'une manière à ce que les BSM modifiés n'aient pas un aspect suspect différent des BSM normaux. En outre, leurs mécanisme de communication n'injectent pas des commandes dans chaque BSM, et modifier constamment la fréquence d'injection afin de réduire davantage la possibilité d'éveiller les soupçons. Tous les champs d'un BSM ne peuvent pas être modifiés sans provoquer des données non plausibles et sans faire apparaître des avertissement. Pour qu'une injection reste non détectée, Ghost modifié que quatre champs dans les BSMs : "Transmission et Vitesse" pour la vitesse actuelle du véhicule, "Précision de positionnement" pour le bruit dans le calcul du cap du véhicule par rapport au nord vrai, "Latitude" et "Longitude" pour la localisation GPS du véhicule. Ghost injecte un demi-octet dans chaque champ en remplaçant les 4 bits les moins significatifs du champ, et il limite la taille de chaque injection à un demi-octet afin que les changements dus à l'injection dans les valeurs du champ ne soient pas suspects. Ghost modifié la valeur du champ de vitesse de 0,67 miles/heure au maximum, le champ de précision du positionnement de 0,08 degrés au maximum, et la position GPS de 24 centimètres au maximum, qui sont des variations tout à fait naturelles même sans injection.

On se basant sur le protocole Ghost [1], Garip et al. ont considéré trois attaques qui peuvent être appliqué par un réseau de botnets véhiculaires : une attaque de congestion [92], une attaque de surveillance coopérative adaptative contre les systèmes de changement de pseudonyme, "Botveillance" [93], et "RIoT", la première attaque dans la littérature contre les dispositifs connectés (IoT) en utilisant des véhicules [94].

### 3.5.2 Attaque sur la congestion

Dans [92], les auteurs ont proposé une attaque de botnet véhiculaire qui manipule les vulnérabilités communes des mécanismes existants d'évitement

des congestion dans les réseaux véhiculaires, afin de provoquer de lourds embouteillages sur une route choisie. Contrairement aux réseaux de botnet internet, dont les attaquants réalisent des attaques DDoS afin d'encombrer les liens internet, Garip et al. ont proposé une attaque qui a des conséquences physiques. Ces conséquences peuvent être utilisées pour des incitations économiques et politiques, par exemple : elle peut être utilisée dans des scénarios aussi graves que le retardement de l'arrivée de la police sur les lieux d'un vol ; le blocage de l'accès des véhicules d'urgence à une zone ciblée par une attaque terroriste ; ou tout simplement l'augmentation des embouteillages autour d'un magasin spécialisé pour favoriser son concurrent. Dans des scénarios graves comme ceux décrits ci-dessus, les conséquences de l'attaque proposé peuvent être dévastatrices, voire fatales. Les mécanismes d'évitement de congestion V2V reposent sur la diffusion des niveaux d'encombrement observés par les véhicules afin de prendre de meilleures décisions de navigation. Tous les systèmes V2V partagent les mêmes vulnérabilités que l'attaque du botnet exploitera : ils font entièrement confiance et transmettent les informations relatives au congestion sans essayer de détecter les valeurs extrêmes ou les informations erronées. Les voitures diffusent leurs vitesses actuelle dans leur BSM. Ces informations sont utilisés par chaque voiture dans le mécanisme d'évitement des congestions pour calculer la vitesse moyenne totale sur chaque route qu'elle emprunte. Ces mesures de vitesse moyenne sont ensuite utilisées pour le calcul des temps de parcours et la sélection des itinéraires les moins encombrés des autres véhicules. Le but de leurs attaque est de rediriger le plus grand nombre possible de voitures vers la route ciblée, en diminuant éventuellement sa vitesse moyenne. Un véhicule ne choisira pas un itinéraire contenant la route ciblée s'il entend des rapports accablants de faibles vitesses de la part des voitures qui ont effectivement emprunté cette route. Grâce au mécanisme de prévention des embouteillages, les autres voitures victimes qui envisagent d'emprunter la route ciblée commenceront à se dérouter, ce qui évitera que les embouteillages ne s'aggravent. Pour surmonter cet effet, les véhicules bots annoncent des vitesses élevées dans leurs BSM lorsqu'ils circulent sur la route ciblée. En d'autres termes, ils continueront à signaler qu'ils

se déplacent rapidement sur la route ciblée qui est, en fait, très encombrée. Cela permettra d'exploiter les mécanismes de calcul des moyennes de vitesse des voitures sur la route ciblée, ce qui fera augmenter leurs moyennes calculées afin de permettre à l'effet de l'attaque proposé de se poursuivre aussi longtemps que possible.

### 3.5.3 Botveillance

Dans [93], Garip et al. ont proposé "botveillance", une attaque de surveillance coopérative qui s'adapte aux conditions changeantes des systèmes de pseudonymes dans les réseaux véhiculaires. Elle est utilisée pour suivre des véhicules présentant un intérêt ou pour violer la vie privée des conducteurs. Botveillance se base sur Ghost pour assurer l'infrastructure de communication secrète nécessaire aux véhicules bots pour coordonner leurs attaques ; et INTERLOC [95] qui met en corrélation les diffusions de localisation des voiture. En plus, elle assure le suivi à long terme du véhicule cible sans nécessiter de matériel supplémentaire. Ils ont utilisé Swing [96] comme un système de changement de pseudonyme pour évaluer leur attaque. A travers les expérimentations, ils ont montré que Botveillance peut suivre un véhicule sur 85 % de son trajet et détecter son adresse de destination dans 90 % des cas.

### 3.5.4 RIoT

Dans [94], Garip et al. ont présenté RIoT, une attaque qui peut compromettre un pourcentage important de dispositifs IoT dans une zone d'intérêt en peu de temps, en tirant parti de la mobilité et de la portée de communication collective des véhicules bots. Il s'agit de la première attaque dans la littérature contre les dispositifs IoT en utilisant des véhicules. RIoT ne présente pas de nouveaux exploits contre les dispositifs IoT, sa nouveauté vient du mécanisme qui peut rapidement livrer les charges utiles des exploits existants aux dispositifs IoT. Chaque véhicule bot obtient la base de données d'exploitation des vulnérabilités la plus récente du botmaster. En deuxième étape, les véhicules bots commencent à chercher n'importe quel dispositif IoT dans leur

rayon de communication en se déplaçant tout au long de leurs routes. Les véhicules bots ne modifient en aucune façon leurs itinéraires d'origine dans le but de trouver d'autres dispositifs IoT, car sinon, les propriétaires légitimes de ces véhicules auraient des soupçons. Pour chaque dispositif IoT détecté dans leur portée de communication, les bots identifient son type, interrogent leur base de données d'exploitation pour toutes les charges utiles applicables, et déterminent la ou les méthodes de connexion ad-hoc sans fil les plus appropriées pour les fournir. D'après la simulation, ils ont montré que RIoT peut compromettre jusqu'à 87% des dispositifs IoT dans une zone d'intérêt en peu de temps.

### 3.5.5 Shieldnet

Pour lutter contre les trois attaques mentionnées ci-dessus, Garip et al. ont proposé SHIELDNET [97], un mécanisme de détection des botnets véhiculaires qui utilise des techniques d'apprentissage automatique pour détecter l'utilisation de GHOST, et identifier la communication du botnet. SHIELDNET détecte l'activité des botnets en recherchant les valeurs anormales de certains champs BSM. Comme décrit auparavant, Ghost divise les messages de communication du botnet de deux octets en quatre parties, et il les injectent dans les quatre bits les moins significatifs des champs de vitesse, de latitude, de longitude et de précision positionnelle. Comme il n'y a pas de comportement prévisible pour le champ de précision positionnelle, seuls les champs de vitesse, de latitude et de longitude sont utilisés pour la détection des anomalies. Selon les auteurs, la seule approche efficace pour détecter les anomalies dans les valeurs de vitesse consiste à connaître le schéma de mobilité récent d'un véhicule et à déterminer la plausibilité de ses émissions de vitesse suivantes en fonction de leur adéquation avec ce schéma. SHIELDNET utilise un algorithme d'apprentissage automatique, en particulier discrete linear convolution [98], afin de modéliser le schéma de mobilité de chaque véhicule en fonction de sa vitesse annonce, et détecte les valeurs aberrantes avec une norme fixe de déviation. Chaque valeur aberrante est ensuite signalée a un système de réputation. SHIELDNET crée un modèle de mobilité distinct

pour chaque véhicule et effectue la détection des anomalies simultanément pour tous les véhicules. En ce qui concerne la détection des anomalies dans les valeurs de latitude et de longitude par l'apprentissage de leurs modèles de changement, elle n'est pas pratique car les deux peuvent changer, et le changement dans chacune est déterminée par plusieurs facteurs, y compris l'environnement et la vitesse. Due a ces raison, les auteurs ont utilisé une approche de détection des anomalies des valeurs de latitude et de longitude, on se basant sur un pattern des erreur GPS qui peuvent être causer par une injections dans ces champs. Après l'évaluation de SHIELDNET, les auteurs ont trouvé qu'il peut identifier 77 % des messages des véhicules bots.

Bien que Ghost a prouvé son efficacité d'injection de message secret dans les messages BSMs d'une manière inaperçu et indétectable, dans [99], les auteurs ont fait allusion aux contre-mesures envisagées dans la norme BSMs pour empêcher les attaques, y compris la modification. Ces contre-mesures puisse rejeter les modifications des messages qui dépassent un seuil de variation naturelle, ce qui présente un challenge au protocole Ghost, et met son efficacité en question. En plus, le protocole Ghost est un protocole à saut unique, ce qui fait que la commande du botmaster n'est envoyée qu'à ses voisins, rendant la portée de communication très courte et inaccessible de loin aux véhicules bots. En outre, si le botnet utilise un autre protocole de communication, la solution proposée ne sera plus efficace. De plus, le scénario du botnet dans les réseaux véhiculaires n'est pas réel. En effet, la transmission de commandes par communication V2V rendra la capacité de contrôle du botmaster très limitée et il devra se trouver sur la route et à proximité d'autres nœuds du botnet. Par conséquent, la manière idéale d'imaginer un réseau de botnets dans un réseau véhiculaires est que le botmaster peut se trouver n'importe ou, et qu'il peut contrôler ses véhicules en utilisant la communication V2I et envoyer des commandes par l'intermédiaire de l'infrastructure.

## 3.6 Conclusion

Dans ce chapitre, nous avons abordé le problème des botnets en générale, leurs architectures, cycle de vie, et modes de communication. Ensuite, nous avons énuméré les différentes techniques et méthodes de détection proposé jusqu'a aujourd'hui. Après cela, nous avons étudier la possibilité des botnets véhiculaires, et leurs caractéristiques spécifiques qui soulèvent d'autres failles, qui peuvent être utilisé par les attaquants afin d'échapper aux techniques de détection standards. Après cela, nous avons analysé le seul travail (Garip, et al.) qui a abordé le sujet, et nous avons déduire qu'ils ont proposé une solution limité a un protocole de communication spécifique. Pour cela, dans le chapitre suivant, nous allons proposé une technique de détection des botnets véhiculaires comportementale, qui ne dépends d'aucun protocole de communication spécifique.

# AntibotV : un framework de détection des botnets véhiculaires, basé sur une détection comportementale à plusieurs niveaux

Sommaire

## 4.1 Introduction

Les VANET peuvent contribuer à accroître la sécurité et l'efficacité des transports en offrant des applications commerciales, de gestion de trafic et de sécurité [100]. Cependant, ils soulèvent des problèmes de confidentialité et de sécurité, qui menacent considérablement les opérations du réseau et les données des utilisateurs.

Une des menaces les plus dangereuses est lorsque l'ordinateur de bord d'un véhicule est compromis et exploité par un attaquant à distance. Cette cybermenace est connue sous le nom de "vehicular botnet". Contrairement aux autres menaces de sécurité, les botnets véhiculaires peuvent être utilisé pour exécuter à distance des tâches malveillantes à plusieurs niveaux : (1) attaques de Déni de service distribués (DDoS) [101] ; (2) violation des données personnelles du conducteur (suivi GPS illégal et écoute des conversations du conducteur et des passagers) [102] ; (3) contrôler les véhicules bots à distance (ouvrir la porte, démarrer le moteur, allumer les feux, éloigner le véhicule ou désactiver les freins) ; (4) tromper le conducteur en lui donnant de fausses informations sur l'état du véhicule (falsifier le niveau de carburant, changer la lecture du compteur de vitesse et afficher les informations de panne sur le tableau de bord) ; [35].

Malgré l'impact négatif des botnets véhiculaires sur la vie privée et la sécurité du conducteur, seules deux recherches existantes [1, 97], à notre connaissance, ont examiné cette vulnérabilité. Garip et al. [97] ont proposé SHIELDNET, un mécanisme de détection des botnets basé sur l'apprentissage automatique. En tant que botnet, ils ont considéré GHOST [1], un protocole de communication de botnet qui utilise des messages de sécurité de base BSMs pour dissimuler sa communication sur le canal de contrôle. SHIELDNET détecte l'activité du botnet en recherchant les valeurs anormales de champs BSM spécifiques. Bien que son efficacité, SHIELDNET repose sur un protocole de communication spécifique, il ne serait donc pas efficace si le botmaster du réseaux botnet change le protocole de communication.

Dans ce chapitre, nous proposons une approche de détection des botnets

véhiculaires qui ne suppose pas un protocole de communication particulier. Étant donné que le botnet véhiculaire est une cybermenace opérant sur plusieurs niveaux (intérieur et extérieure des véhicules), nous proposons un framework basé sur la détection comportementale à plusieurs niveaux (en anglais, A Multilevel Behaviour-based Framework for Botnet Detection in Vehicular Networks, AntibotV). Le framework proposé surveille l'interaction du véhicule avec l'extérieur en analysant le trafic du réseau. Il surveille également les activités intérieures du véhicule afin de détecter les opérations suspectes qui peuvent être liées à l'activité des logiciels malveillants des botnets. En outre, le présent chapitre examine de nouvelles attaques "zero-day" qui pourraient être menées exclusivement contre la pile de communication des réseaux véhiculaires : "Wave Short Message Protocol flood" (WSMP flood) ; 2) et "Geographic Wave Short Message Protocol flood" (Geo WSMP flood). Les deux attaques ciblent le protocole Wave Short Message, qui est un protocole utilisé dans les réseaux véhiculaires pour le transfert de paquets de sécurité et de gestion de trafic.

La contribution dans ce chapitre se résume en deux parties. Premièrement, nous identifiant un ensemble des attaques "zero-day" qui peuvent être exécuté par un cyber attaquant en compromettant l'ordinateur de bord d'un véhicule, en utilisant des bots malveillants. Nous fournissons une description détaillée de deux zero-day DDoS attaques, ainsi que des attaques de vol d'information spécifique au contexte des réseaux véhiculaires. Deuxièmement, nous proposons un framework pour la détection des botnets véhiculaires basé sur une technique de détection comportemental sur plusieurs niveaux. Un framework qui se base sur le protocole decision trees pour la détection des attaques "zero-day" proposées et d'autres attaques existantes, en surveillant les activités réseaux et intérieures du véhicule.

Vu que la possibilité d'application des botnets véhiculaires a déjà été discuté en détails dans la section 3.4, ainsi que les différents travaux qui ont été mené sur le sujet 3.5, nous allons commencé notre chapitre directement par les modèles de menaces qu'un botnet véhiculaire peut causé dans un réseau véhiculaire (section 4.2). La section 4.3, décrit en détail le framework pro-

posé, AntibotV. Dans la section 4.4, nous décrivons les étapes de génération du dataset utilisée pour l'apprentissage et le test de notre framework, ainsi la discussion des résultats obtenus. Nous finissons par une conclusion 4.5.

## 4.2 Les modèles de menace

Dans cette section, nous fournissons une description détaillée des trois catégories de cyberattaques qui peuvent être exécutées par un véhicule bot contre des véhicules légitimes. Tout d'abord, nous fournissons une description détaillée de deux zero-day attaques, type DDoS. Ensuite, en se basant sur les attaques de vol d'informations existants, nous envisageons de nouveaux scénarios d'attaques de vol d'information applicables seulement dans le contexte des véhicules connectés. Enfin, nous présentons les diveres types d'attaques mentionnées dans la littérature et qui peuvent etre appliquer au niveau intérieur du véhicule. La figure 4.1 montre les différentes catégories de cyber attaques qu'un attaquant peuvent appliquer en exploitant un véhicule bot.

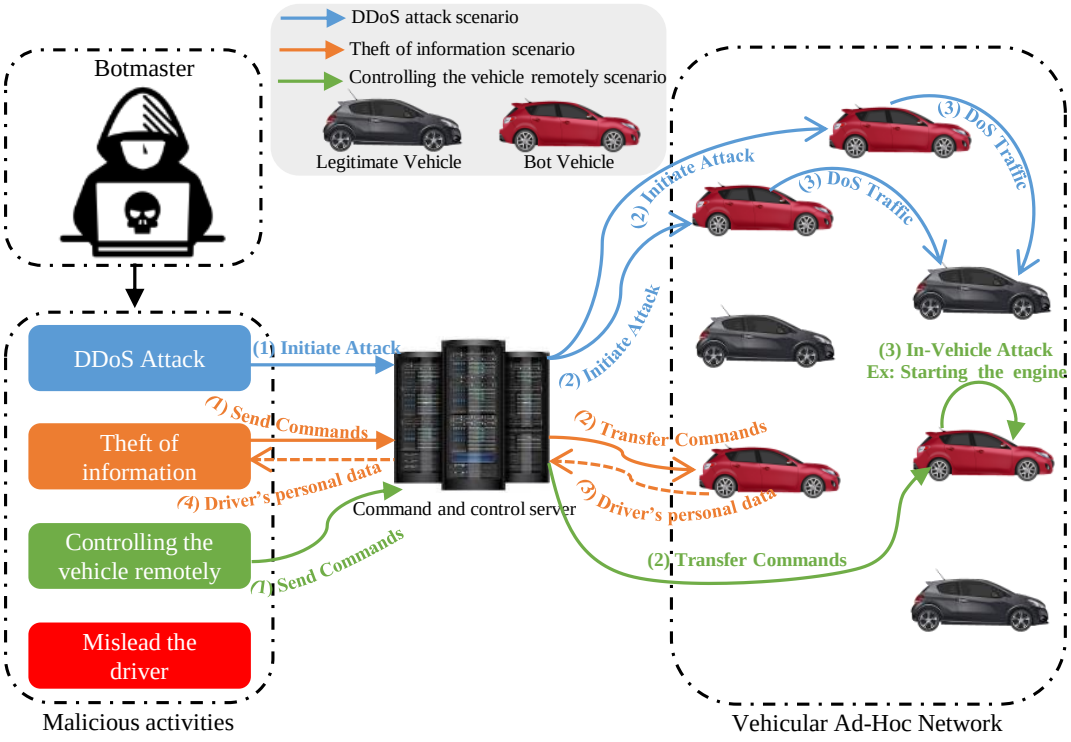


FIGURE 4.1 – *Les botnets véhiculaires*

### 4.2.1 Attaques DDoS

Due aux différences entre la pile protocolaire DSRC et celle de TCP/IP, il est important de considérer des scénarios d'attaques DDoS spécifiques aux réseaux véhiculaires, et de ne pas se limiter aux attaques DDoS communes à tous les réseaux IP. C'est pourquoi, dans cette sous-section, nous considérons deux attaques zero-day spécifiques aux réseaux véhiculaires. Les deux attaques exploitent le Wave Short Message protocole (WSMP), qui est le protocole utilisé par les applications de sécurité et de gestion du trafic pour le transfert de données critiques telles que : la vitesse du véhicule, l'état cinématique, etc. Les deux attaques peuvent empêcher le transfert de messages de sécurité entre les véhicules et même causer des dommages catastrophiques.

Les paquets WSM (figure 4.2) échangés entre véhicules sont composés des champs suivants :

1. **WSMP version** : il désigne la version du protocole.
2. **Channel number** : (en français, le numéro de canal), il spécifié quel canal doivent être utilisés pour la transmission.
3. **Data rate** : (le débit de données), il spécifié le débit qui doivent être utilisés pour la transmission.
4. **WAVE element ID** : il représente l'en-tête WSMP.
5. **WAVE Length** : il spécifier la longueur du paquet.
6. **WSM Data** : il contient les données utiles.
7. **Provider Service Identifier** : le PSID, il identifie le service auquel la charge utile WSM est associée. Par exemple, si une application tente d'accéder au service WAVE, elle doit être enregistrée avec son identifiant unique de service de fournisseur (PSID). Les dispositifs WAVE utilisent le PSID dans leurs messages d'annonce pour indiquer qu'une certaine application est fournie par ce dispositif. D'autre part, lorsque le véhicule passe devant le dispositif routier, les dispositifs utilisateurs, qui peuvent héberger une telle application, à la réception de cette annonce, comparent pour vérifier s'il y a une correspondance entre les PSID des

annonces et les PSID de ses propres tableaux, puis le véhicule établit la communication avec cette unité routière [19, 20].

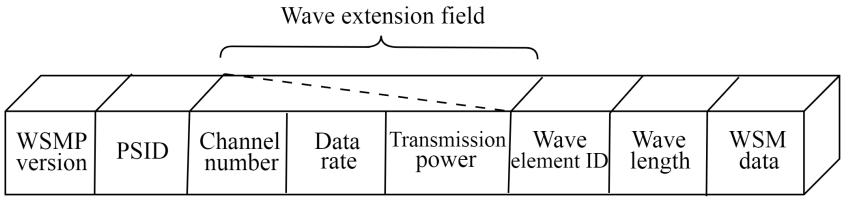


FIGURE 4.2 – *Les champs d'un paquet WSM*

Dans les deux attaques, le cyber attaquant tente d'abuser le protocole WSMP. Dans la première attaque, **WSMP flood**, l'attaquant envoie au véhicule cible des paquets WSM avec des valeurs de champ PSID inconnues (non associé à un service WAVE). À la réception des faux paquets WSM, le véhicule cible tente de vérifier (lookup) si il y'a une correspondance entre le PSID du paquet entrant avec sa table de PSID/Service. La vérification du PSID d'une trame WSM ne posera pas de problème. Cependant, la vérification du PSID d'une quantité énorme de paquets WSM en même temps épuisera les ressources du véhicule cible et le rendra incapable de répondre ou de recevoir des paquets d'applications de sécurité et de gestion de trafic légitimes.

La deuxième attaque, Geographic WSMP flood (**Geo-WSMP Flood**), a le même mode de fonctionnement que le WSMP flood, mais son impact est plus large. Le cyber attaquant diffuse les faux messages WSM à tous ses voisins dans une zone géographique spécifique. Avant de faire l'apprentissage de notre modèle AntibotV avec l'attaque WSMP flood, nous l'avons testé, et nous avons constaté qu'elle peut réduire le débit de 63 % (le nombre de trames de sécurité et de gestion de trafic arrivées), ce qui rend l'attaque très efficace comme montre la figure 4.3.

### 4.2.2 Les attaques de vol d'information

Une équipe d'universitaires de l'Université de Californie à San Diego et de l'Université de Washington ont découvert que la surveillance audio à l'intérieur des véhicules est possible [103]. Dans une telle situation, le cyber attaquant utilise indirectement des outils d'assistance virtuelle (comme Siri) en

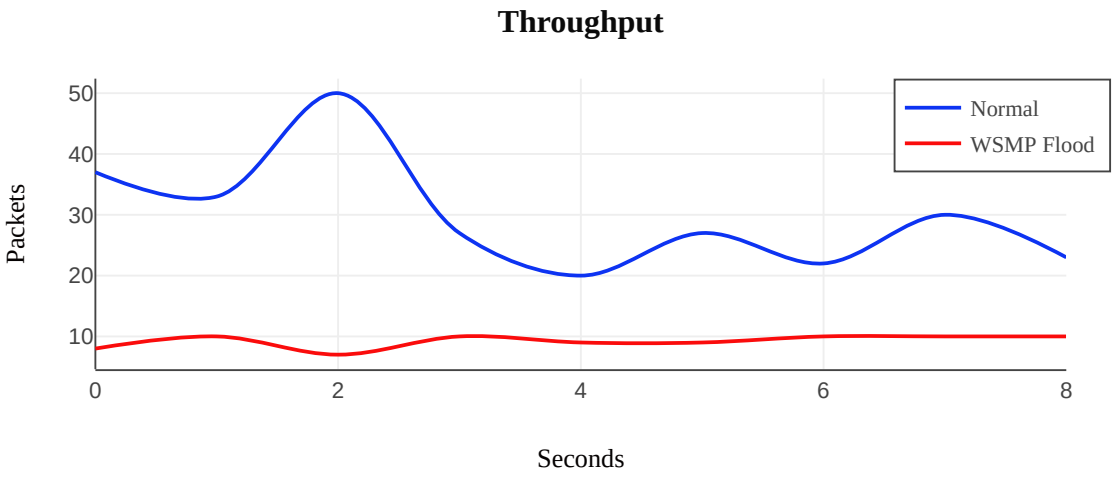


FIGURE 4.3 – *L'impacte de l'attaque WSMP Flood*

lui donnant des ordres malveillants cachés dans de la musique enregistrée, des morceaux audio inoffensives [104] ou un laser de faible puissance [105]. Siri commence à enregistrer les conversations en utilisant les microphones internes du véhicule, et les envoie à un serveur distant appartenant au cyber master toutes les 10 ou 20 secondes [106].

Dans le cas du suivi par GPS, le bot master peut tenter de suivre la position du véhicule en temps réel, de vérifier la position ou de récupérer la trajectoire complète. Pour le suivi en temps réel, les informations GPS doivent être envoyées toutes les secondes (streaming) au botmaster. Pour les deux autres types de suivi GPS, les informations sont envoyées périodiquement ou à la demande. Dans ce document, nous considérons le scénario de suivi GPS en temps réel. Le véhicule du bot envoie la latitude (4 octets), la longitude (4 octets), la vitesse (2 octets), le temps (2 octets) et les coordonnées de la direction en streaming au botmaster.

### 4.2.3 Les attaques à l'intérieur du véhicule

Le réseau intérieure du véhicule apporte un confort aux constructeurs, aux conducteurs et aux services après-vente. Cependant, ils présentent des vulnérabilités [35] qui peuvent être exploitées par un cyber attaquant pour mener les attaques suivantes :

1. **Falsification et injection de trames** : les trames CAN sont envoyées sur le CAN bus en texte clair, ce qui permet au bot malware à l'intérieur du véhicule de récupérer et d'analyser leur contenu. En utilisant les données capturées, le bot malware peut fabriquer de fausses trames qui contiennent des données erronées pour tromper les ECUs. Ensuite, les trames fabriquées sont injectées sur le CAN bus pour réaliser des activités malveillantes telles que : falsifier le niveau de carburant, modifier la lecture du compteur de vitesse ou afficher des informations de panne qui peuvent induire le conducteur en erreur.
2. **Replay Attack** : (en francais, l'attaque par rejeu) les trames sont transmises sur le CAN bus en mode broadcast et sans authentification. Ainsi, les trames CAN peuvent être facilement capturées et rejouées par le bot malware. Les trames retransmises peuvent être utilisées pour ouvrir la porte du véhicule, démarrer le moteur, allumer les lumières ou conduire le véhicule à distance.
3. **L'attaque DoS** : les trames transmis avec le plus petit identifiant sont les messages avec la plus haute priorité. Le bot malware pourrait utiliser cette vulnérabilité pour monopoliser le canal de transmission, et ainsi retarder ou empêcher la transmission de trames CAN légitimes. Par exemple, le malware bot peut empêcher la transmission des trames CAN envoyées à l'ECU des freins, ce qui pourrait conduire à un accident [35].

Le tableau 4.1 fournit un résumé des différentes menaces de sécurité évoquées précédemment.

## 4.3 Le modèle proposé : AntibotV

Dans cette section, nous décrivons en détail le modèle AntibotV, un modèle pour la détection des botnets véhiculaires (figure 4.4). De plus, nous décrivons le processus de détection basé sur des algorithmes de machine learning.

| Catégorie | Attaque | Objectif |
|---|---|---|
| DDoS | WSMP Flood | Épuiser les ressources du véhicule cible et le rendre incapable de répondre ou de recevoir des paquets d'applications légitimes de sécurité et de gestion de trafic. |
| | Geo-WSMP Flood | Épuiser les ressources de plusieurs véhicules et consommer la bande passante de toute une zone géographique. |
| Vol d'information | GPS Tracking | Suivi de la position du véhicule en temps réel. |
| | Surveillance audio | Enregistrer les conversations à l'aide des microphones internes du véhicule, et les envoyer au botmaster. |
| In-Vehicle | Falsification | Fabriquer de faux frames contenant des données erronées pour tromper le conducteur : falsification du niveau de carburant, modification de la lecture du compteur de vitesse... |
| | Replay | Rejouer les frames capturées. Peut être utilisé pour : ouvrir la porte, démarrer le moteur... |
| | DoS | Monopolisez le canal de transmission pour retarder ou empêcher la transmission des trames légitimes. Exemple : stopper les freins. |

TABLE 4.1 – *Les modèles de menaces*

### 4.3.1 Présentation du modèle AntibotV

Nous supposons qu'un trafic réseau généré par un bot malware est différent d'un trafic réseau généré par un véhicule légitime, en d'autres termes, le bot malware modifie le pattern du trafic réseau d'un véhicule légitime. Cette hypothèse vient du fait que les véhicules connectées exécutent des applications spécialisées liées à la sécurité et à la gestion du trafic, telles que : l'avertissement coopératif de collision, la notification post-collision V2V, la notification de congestion routière, etc. L'exécution d'applications spécialisées et particulières permet de régulariser le pattern du trafic réseau des véhicules connectés tant qu'elle n'est pas compromise. Ainsi, nous pensons que les bots malveillants qui compromettent un véhicule connecté devraient modifier le pattern du trafic réseau. En parallèle, au niveau intérieur du véhicule, et en raison de la régularité du modèle de communication, on peut supposer qu'il est possible d'identifier un modèle de trafic in-véhicule légitime, et donc de détecter les in-activités malveillantes des bots malwares.

Le AntibotV est un système de détection d'intrusion hôte (HIDS : Host-based Intrusion Detection System) qui s'exécute au niveau d'un véhicule plutôt qu'à un point quelconque sur réseau. Le AntibotV surveille la communication avec l'extérieur en analysant le trafic réseau, et surveille la commu-

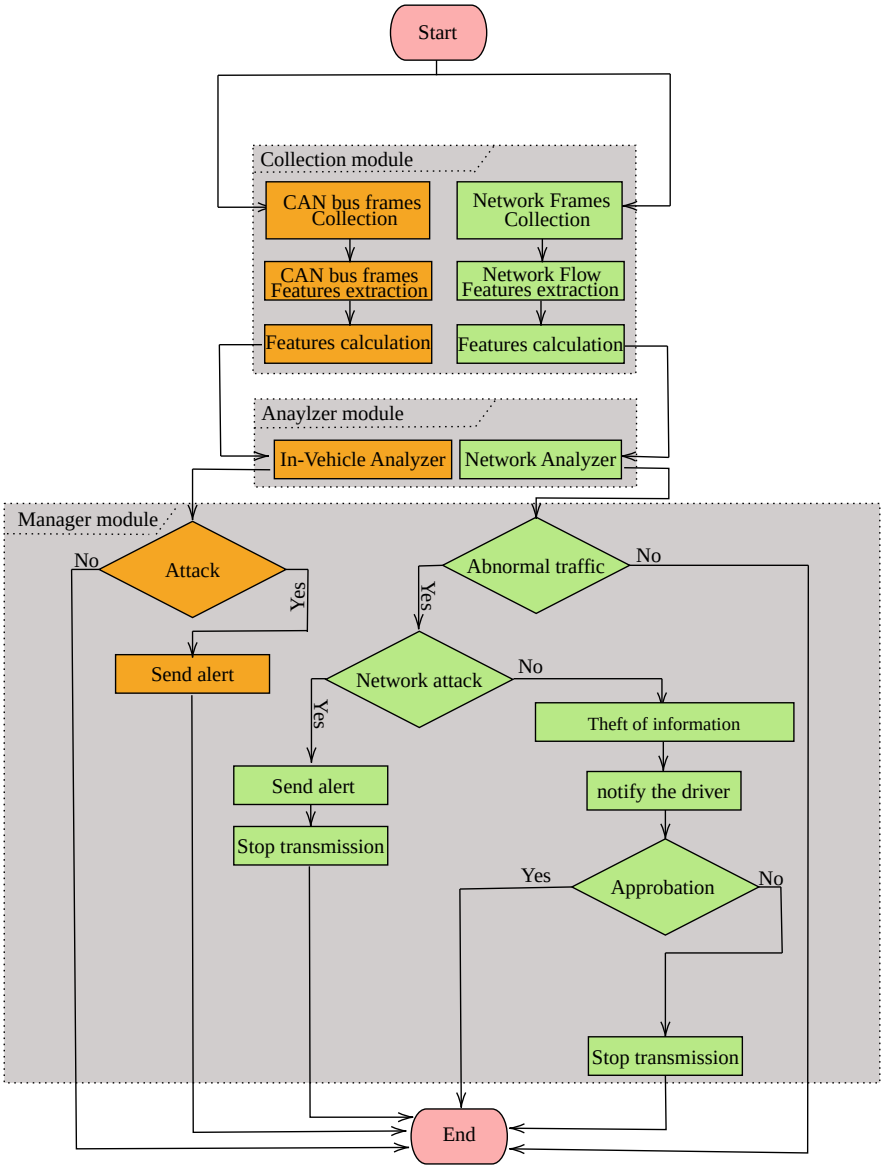


FIGURE 4.4 – *Organigramme descriptif du modèle AntibotV*

nication à l'intérieur du véhicule en analysant les trames sur le CAN bus. La surveillance à deux niveaux permet une détection efficace des activités des bots malwares qui consiste à envoyer des données et à recevoir des commandes du bot master à travers le réseau, et d'exécuter des commandes de contrôle, de falsifier le niveau de carburant, modifier la lecture du compteur de vitesse, ou afficher les informations sur les pannes au niveau intérieure du véhicule.

Le trafic du réseau est caractérisé par un ensemble de caractéristiques statistiques telles que la durée du flux, le nombre total de paquets envoyés dans le sens de l'acheminement, la longueur minimale des paquets, etc. ( une des-

cription détaillée des features utilisées est fournie dans la section suivante). Pour caractériser la communication in-véhicule, un ensemble de douze features est utilisé, dont l'horodatage (temps enregistré), CAN ID (identifiant du message CAN dans HEX), DLC (nombre d'octets de données, de 0 à 8), DATA [0 7] (valeur de la donnée en octet).

Le AntibotV étant un système basé sur l'hôte, il ne devrait pas consommer une grande partie des ressources du véhicule, car cela aurait un impact sur ses autres opérations. Pour cette raison, nous déplaçons la charge de calcul du modèle de classification de la formation vers un serveur centralisé sur le Cloud. Ainsi, le véhicule fonctionne avec un modèle de classification déjà formé, ce qui minimiserait la consommation de ressources du véhicule.

Le AntibotV a une architecture modulaire inspirée de l'architecture de l'IETF proposée par le groupe IDWG [107]. L'architecture est principalement composée de trois modules : un module de collecte du trafic, un module d'analyse et un module de gestion. Le premier module collecte le trafic du réseau et les trames CAN des véhicules. Le module analyseur est chargé d'analyser les données de trafic du véhicule. Le module gestionnaire gère les alertes, envoie des notifications et met à jour le modèle de classification pour les deux trafics. Une description détaillée de ces modules est fournie dans les sections suivantes.

### 4.3.2 Traffic collection module

Module de collecte du trafic, utilisé pour collecter et traiter le trafic de véhicules et appliquer plusieurs opérations de pré-traitement afin d'extraire a partir des données brutes un vecteur de caractéristiques (features), qui sera utilisé par le module d'analyse dans une deuxième étape. Ce module collecte deux types de trafic : le flux réseau et les trames du CAN bus.

#### *4.3.2.1 Network traffic collector*

Collecteur de trafic réseau, un module qui recueille les informations sur le trafic du réseau à partir des paquets échangés. Chaque paquet est mis en correspondance avec un flux réseau identifié par cinq attributs, à savoir l'IP

source, l'IP destination, le port source, le port destination et le protocole. La RFC 3697 [108] définit le flux de trafic comme "une séquence de paquets envoyés d'une source particulière vers une destination unicast, anycast ou multicast particulière que la source souhaite étiqueter comme un flux". Les flux TCP se terminent généralement lors du démontage de la connexion (par un paquet FIN) tandis que les flux UDP se terminent par un délai d'attente. Le tableau 5.2 montre la liste des caractéristiques extraites pour chaque flux réseau . Nous pensons que les caractéristiques basées sur le temps (Flow IAT, Fwd IAT et Idle Time) sont utiles pour détecter les attaques DoS car l'intervalle de temps entre les paquets successifs est trop court. Les fonctions basées sur le temps permettent également de détecter des événements périodiques tels que le transfert périodique des informations collectées en cas de vol d'informations. Les fonctions basées sur les octets/paquets permettent de détecter les augmentations importantes et anormales du trafic, qui sont symptomatiques des attaques par déni de service. À la fin du flux de réseau, le collecteur de trafic réseau génère un vecteur de caractéristiques calculées, puis le transfère au module d'analyse.

#### 4.3.2.2 *In-vehicle traffic collector*

Le module de collecte du trafic in-véhiculaire, collecte les trames échangées sur le CAN bus. Contrairement au trafic du réseau, qui est traité en flux, le trafic in-véhicule est analysé par la technique d'inspection des trames profondes. L'analyse est effectuée par l'observation en temps réel des trames au fur et à mesure qu'elles traversent les liaisons du CAN bus. Une trame CAN bus contient les champs suivants : le début de la trame (1b), l'ID du message (un identificateur de 11b qui représente la priorité du message), les champs de contrôle (3b), la longueur des données (nombre d'octets de données, 4b, de 0 à 8), les données [0 7] (données à transmettre, 0-64b), le CRC (15b), les champs ACK (3b), le délimiteur de fin de trame (7b) [109].

Le collecteur de trafic in-véhiculaire génère un vecteur de caractéristiques avec les champs suivants : timestamp (temps enregistré), CAN ID, DLC et les DATA, comme illustré dans le tableau 4.2. Les champs timestamp et de CAN

ID pourraient être utilisés pour détecter les attaques par Déni de Service (DoS) et par rejeu, ce qui permet d'exploiter la vulnérabilité de la priorité de l'ID comme décrit précédemment dans la section 5.4. Les champs DLC et DATA pourraient être utilisés pour détecter les trames falsifiées et injectées. Ensuite, le vecteur de caractéristique est transféré au module d'analyse.

Les vecteurs de caractéristiques du réseau et du véhicule sont prétraités avant leur transfert vers le module d'analyse. Les opérations de prétraitement consistent principalement à supprimer les valeurs manquantes et à mettre à l'échelle les valeurs des caractéristiques à l'aide de la z-transformation.

| Features | Description |
|---|---|
| Timestamp | Temps enregistré (s) |
| CAN ID | Identificateur du message CAN |
| DLC | Nombre d'octets de données, de 0 à 8 |
| DATA[0] | Valeur des données (octet) |
| DATA[1] | |
| DATA[2] | |
| DATA[3] | |
| DATA[4] | |
| DATA[5] | |
| DATA[6] | |
| DATA[7] | |

TABLE 4.2 – *Les attributs du vecteurs des features des CAN bus trames*

### 4.3.3 Analyzer module

Le module d'analyse représente le module le plus important de notre modèle. Comme il traite deux types de trafic indépendants, il utilise deux analyseurs. Le premier analyseur est chargé d'analyser le trafic du réseau, le deuxième se charge d'analyser les trames échangées sur le CAN bus. L'analyseur du réseau est un classificateur formé à l'aide des algorithmes d'apprentissage automatique et supervisé a partir d'un trafic réseau légitime et malveillant. Le trafic réseau légitime est généré en ruinant les applications liées à la sécurité spécialisées pour véhicules connectés , aux applications commerciales et de gestion de trafic. Le trafic réseau malveillant est lié aux activités typiques des bots malwares, comme les attaques DOS et le vols d'informations.

L'analyseur in-véhicule est un classificateur formé à l'aide des algorithmes d'apprentissage automatique supervisés a partir des trames de CAN bus légitimes et malveillantes (DoS, Fuzzy, RPM et Gear). Les deux classificateurs sont générés dans un serveur central puis intégrés dans notre modèle. Dans le véhicule, les deux classificateurs sont utilisés pour la surveillance continue du trafic réseau et intérieure (CAN bus trames).

L'analyseur réseau utilise les caractéristiques de réseau calculées pour classer le flux réseau reçu dans l'une des trois classes suivantes : normal, DOS ou vol d'informations. L'analyseur in-véhicule classe les trames du CAN bus en fonction des caractéristiques calculées en quatre classes : normal, DOS, injection de trames ou attaque par rejeu. Si un flux réseau ou une trame de CAN bus malveillant est détecté, le module d'analyse envoie une alerte au module de gestion pour qu'il prenne une mesure immédiate. Dans le cas contraire, elle sera ignorée.

Pour former les deux analyseurs, nous utilisons les algorithmes d'apprentissage automatique supervisés suivants. Pour chaque analyseur, nous choisissons l'algorithme qui donne les meilleures performances. Les algorithmes suivants sont sélectionnés pour leur efficacité connue et leurs performances de classification :

1. **Naive Bayes** : Il s'agit d'un classificateur probabiliste qui effectue des classifications en utilisant la règle de décision maximale a posteriori dans un cadre bayésien. Il fonctionne selon une forte hypothèse d'indépendance, ce qui signifie que la probabilité d'un attribut n'affecte pas la probabilité de l'autre [110].

$$P(c \mid x) = \frac{P(x \mid c) * P(c)}{P(x)} \tag{4.1}$$

$$P(c \mid x) = P(x1 \mid c) * P(x2 \mid c) * ... * P(xn \mid c) * P(c) \tag{4.2}$$

$P(c|x)$ est la probabilité a posteriori de la classe (cible) par rapport à un prédicteur (attribut). $P(c)$ est la probabilité de classe a priori. $P(x|c)$ (en anglais, likelihood) est la probabilité d'un prédicteur donné de classe. $P(x)$ est la probabilité antérieure du prédicteur.

2. **Support Vector Machine(SVM)** : Chaque élément de données est représenté par un point dans un espace à n dimensions (où n est le nombre d'éléments), la valeur de chaque élément étant la valeur d'une coordonnée particulière. Ensuite, la technique de classification est effectuée pour différencier les classes et définir à laquelle les points de données appartiennent [111].

3. **K-Nearest Neighbour** : L'algorithme des plus proches voisins (k-NN) est une méthode non-paramétrique utilisée pour la classification et la régression [112]. Il fonctionne sur la base d'une distance minimale entre l'instance d'interrogation et les échantillons d'entraînement pour déterminer les k voisins les plus proches. Après avoir rassemblé les K voisins les plus proches, il faut la majorité simple de ces K voisins les plus proches pour être la prédiction de l'instance de requête. Formellement :

$$score(D, Ci) = \sum Dj \in KNN(d) Sim(D, Dj) \wp(Dj, Ci) \tag{4.3}$$

Au-dessus, *KNN(d)* indique l'ensemble des K voisins les plus proches de l'instance de requête

$$D * \wp(Dj, Ci) \tag{4.4}$$

En ce qui concerne la classe *Ci*, c'est-à-dire :

$$\wp(Dj, Ci) = \begin{cases} 1, Dj \in Ci \\ 0, Dj \notin Ci \end{cases} \tag{4.5}$$

Pour le document de test *d*, il faut lui attribuer la classe qui a la somme pondérée résultante la plus élevée.

4. **Decision Trees** : Pour construire un arbre de décision, nous avons utilisé dans cet article l'algorithme ID3 (Iterative Dichotomiser 3) qui utilise l'entropie et le gain d'information comme mesures. L'entropie caractérise l'impureté d'une collection arbitraire d'exemples, elle peut être définie formellement comme suit :

$$H(s) = \sum_{c \in C} -P(c) * log_2 P(c) \tag{4.6}$$

où $S$ est l'ensemble de données actuel pour lequel l'entropie est calculée, $C$ est l'ensemble des classes dans $S$, et $P(c)$ représente la proportion d'éléments de la classe c par rapport au nombre d'éléments de l'ensemble S.

5. **Random Forest** : Comme son nom l'indique, il est constitué d'un grand nombre d'arbres de décision individuels qui fonctionnent comme un ensemble. Chaque arbre individuel de la forêt aléatoire crache une prédiction de classe et la classe ayant le plus de votes devient la prédiction de notre modèle.
6. **Neural Networks** : est l'un des algorithmes d'apprentissage automatique les plus connus. Il fonctionne sur la base de plusieurs couches afin d'analyser les données. Chaque couche essaie de détecter des modèles sur les données d'entrée. Lorsqu'un motif est détecté, la couche cachée suivante est activée, et ainsi de suite [113].

Dans cet article, nous avons utilisé les algorithmes mentionnés ci-dessus avec les paramètres par défaut, sauf avec le SVM. Pour ce dernier, nous avons utilisé un noyau non linéaire appelé le Radial Basis Function (rbf), car il est bien appliquée et relativement facile à calibrer, contrairement aux autres noyaux.

### 4.3.4 Manager module

Les modules de gestion gèrent les alertes et déclenchent des mesures de réponse adéquates en fonction des attaques détectées. Chaque fois que le module analyseur détecte une activité de botnet, il envoie une alerte au module gestionnaire. Ce dernier enregistre les traces de l'événement correspondant et en informe le conducteur. Si une attaque DoS est détectée, le module gestionnaire met fin à la session réseau avec le véhicule victime. En cas de vol d'informations, le gestionnaire enregistre les traces et en informe le conducteur. Si le conducteur n'approuve pas le transfert, la connexion à l'adresse de

destination sera bloquée. Dans le cas contraire, le flux sera ignoré. Au niveau du véhicule, lorsque l'analyseur détecte une trame CAN comme appartenant à une activité de botnet, il envoie une alerte au gestionnaire, qui en avertit le conducteur. Pour éviter d'interrompre à tort les services du véhicule, quel que soit le type d'attaques détectées, le module du gestionnaire demande l'accord du conducteur avant d'entreprendre toute mesure de riposte. L'algorithme des pseudo-codes 1 résume les différentes mesures de réponse.

**Algorithme 1 :** Mesures d'intervention du module de gestion

```
BEGIN
Input : CBF (CAN bus frame), NFW (Network flow feature vector)
if (CBF detected as malicious) then
    Send an alert to the driver
    Ask driver's approval for system reset
if (NFV detected as DoS attack) then
    Save detailed logs
    Send alert to the driver
    Terminate network session
    Ask driver's approval for system reset
else
    if (NFV detected as theft of information) then
        Save detailed logs
        Send an alert to the driver
        if (information theft confirmed) then
            Terminate network session
            Ask driver's approval for system reset
        else
            Ignore NFV ;
    Ignore NFV;
END
```

Le fait de mettre fin aux processus malveillants en cours d'exécution sur le véhicule ne signifie pas qu'il ne sera pas exécuté à nouveau. Pour se débarrasser de ce malware inapproprié qui a été détecté, le système du véhicule doit être réinitialisé. Le processus de réinitialisation supprime les applications et les fichiers installés sur le système, qui peuvent être les vecteurs du mal-

ware. Le module de gestion doit demander l'autorisation du conducteur avant l'opération de réinitialisation.

Enfin, afin de maintenir le modèle formé à jour, une nouvelle session de formation est lancée automatiquement dès que l'accuracy du modèle commence à se dégrader. Le module de gestion commence à envoyer le trafic entrant collecté par le module de collecte vers un serveur distant sur le cloud, qui est utilisé pour effectuer les opérations de calcul exhaustif. Une combinaison des nouvelles données collectées et des anciennes données de formation sont utilisées ensemble pour construire et déployer le nouveau modèle. L'avantage de cette technique de mise à jour est qu'elle est entièrement automatisée.

## 4.4 Expérimentation

Dans cette section, nous présentons les résultats de l'évaluation du modèle proposé. Tout d'abord, nous décrivons en détail les ensembles de données utilisés dans cette recherche et les opérations de prétraitement que nous avons effectuées. Comme notre framework proposé surveille la communication in-véhicule et sur le réseau, nous avons utilisé deux bases de données pour évaluer ses performances, la première contenant le trafic réseau, et la deuxième le trafic in-véhicule [114]. Une description détaillée des deux ensembles de données est fournie ci-dessous.

### 4.4.1 Base de donnés du trafic réseau

#### *4.4.1.1 Génération du trafic réseau*

À notre connaissance, aucun ensemble de données réel sur le trafic des réseaux véhiculaires, y compris le trafic des réseaux botnets véhiculaires, n'est disponible au public. Par conséquent, nous simulons les activités des réseaux de botnets véhiculaires dont il est en question dans la section 5.4. Pour générer un trafic de réseau véhiculaires réaliste et bénin, nous avons utilisé 17 applications, y compris des applications de sécurité, de gestion de trafic et commerciales. La liste des applications et leur brève description [26] est four-

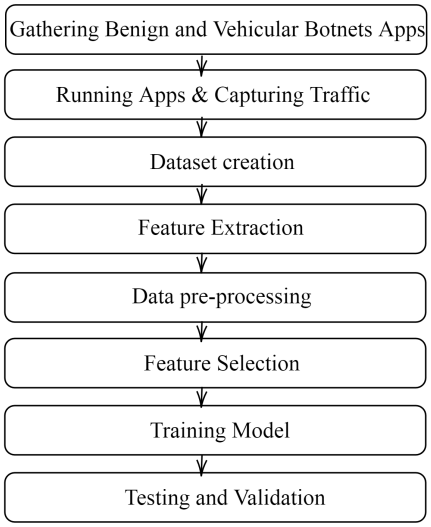


FIGURE 4.5 – *Diagramme de génération de la base de données*

nie dans le tableau **??**. Afin de s'assurer que le trafic généré est représentatif du trafic réel bénin et couvre divers types d'applications, nous avons pris en compte les facteurs suivants pour le choix des applications : 1) canal de la couche physique (CCH & SCH) ; 2) protocoles de transfert (IP & WSMP) ; 3) message TTL (single-hop & multi-hop) ; 4) protocole de routage (geocast, broadcast & unicast) ; 5) condition de déclenchement (On-demand & event-triggered) ; 6) et technologie de communication (V2V & V2I). Le tableau 4.3 fournit une catégorisation des applications bénignes basée sur les facteurs susmentionnés.

Pour générer un trafic réseau malveillant lié à des activités d'un botnet véhiculaire malveillant, nous avons simulé les scénarios d'activité suivants : 1) WSMP Flood ; 2) Geo-WSMP Flood ; 3) Suivi GPS ; 4) et vol d'informations, par l'écoute des conversations des conducteurs et des passagers. Les premier et deuxième scénarios correspondent à des attaques DDoS de type "zero-day" applicables uniquement dans l'environnement des véhicules connectés (mis en évidence plus haut dans la section 5.4).

Pour simuler les scénarios des activité des botnets véhiculaires malveillants susmentionnés, nous avons utilisé le Network Simulator version 3 (NS3) [115], et le package Simulation of Urban MObility (SUMO) [116]. SUMO est responsable de la simulation du trafic véhiculaire réaliste, tandis que NS3 est utilisé pour simuler les capacités de communication des véhicules avec

intégration du module IEEE 802.11p. La simulation se déroule pendant 500 secondes avec 40 nœuds se déplaçant selon le modèle de point de cheminement aléatoire à une vitesse de 20 m/s et sans temps de pause dans la carte de Manhattan (téléchargée sur OpenStreetMap [117]). Le WiFi est de 802.11p et le type de trafic dépend de l'application utilisée : WSMP pour les applications de sécurité et de confort (canaux CCH, SCH), et IP pour les applications commerciales. La puissance d'émission est fixée à 20 dBm et le modèle de propagation par défaut est Two-Ray Ground, tandis que la taille des paquets et le débit de données dépendent de l'application. Le tableau 4.4 résume les paramètres de simulation, et le tableau 4.5 présente la distribution des échantillons de l'ensemble de données du trafic réseau.

| Application | | Canal | | Protocole | | Message TTL | | Protocole de Routage | | | Condition de déclenchement | | | Participants | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CCH | SSH | WSMP | IP | Multi-hop | Single-Hop | Geocast | Broadcast | Unicast | Beaconing | Event-triggred | On-demand | V2V | V2I | Internet |
| Orienté sécurité | BSM | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| | CCW | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| | CVW | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| | EEBL | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | PCN | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | RFN | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | RHCN | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | SVA | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| Orienté gestion du trafic | CRN | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | PAN | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| | PSL | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| | TOLL | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ |
| | TP | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ |
| Orienté commercial | CMDD | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |
| | RTVR | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |
| | RVP/D | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| | SA | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |

TABLE 4.3 – *Applications des réseaux véhiculaires classées en fonction des attributs réseau*

#### 4.4.1.2 *Extraction des caractéristiques*

Après avoir recueilli le trafic réseau généré pendant la simulation (sous forme de fichiers PCAP), nous avons extrait un ensemble de 79 caractéristiques de réseau (voir tableau 5.2). Pour extraire les caractéristiques, nous avons utilisé CICFlowMeter [118], un générateur de flux de trafic réseau distribué par l'Institut canadien de cyber sécurité (CIC). Il génère des flux bidirectionnels, où le premier paquet détermine les directions avant (source à

destination) et arrière (destination à source). Notez que les flux TCP sont généralement terminés lors du démontage de la connexion par le paquet FIN, tandis que les flux UDP sont terminés par un délai d'attente. La valeur du délai d'attente du flux peut être attribuée arbitrairement par le schéma individuel, par exemple 600 secondes pour TCP et UDP.

| | |
|---|---|
| Simulateur réseau | NS3 |
| Générateur du trafic | SUMO |
| Carte de simulation | Manhattan Map |
| Temps | 500 seconds |
| Nombre de noeuds | 40 |
| Vitesse maximale | 20 m/s |
| MAC/PHY Standard | IEEE802.11p |
| Trafic réseau | WSMP, IP |
| Canal | CCH, SCH |
| Modèle de propagation | Two-ray ground-reflection model |
| Puissance de Transmission | 20 dBm |
| Taille des paquets | Dépend des application et du protocole |
| Débit | Dépend de l'application |

TABLE 4.4 – *Paramètres de simulation*

| Échantillons | | Nb | | | % | | |
|---|---|---|---|---|---|---|---|
| | | Apprentissage | Test | Totale | Apprentissage | Test | Totale |
| Bénigne | | 909 | 606 | 1515 | 31.14% | 20.76% | 51.90% |
| Malicieux | GPS Tracking | 295 | 197 | 492 | 10.11% | 6.74% | 16.85% |
| | Phishing Attack | 191 | 128 | 319 | 6.55% | 4.37% | 10.92% |
| | WSMP-Flood | 182 | 121 | 303 | 6.23% | 4.15% | 10.38% |
| | Geo-WSMP-Flood | 174 | 116 | 290 | 5.96% | 3.97% | 9.93% |
| Totale | | 1751 | 1168 | 2919 | 60% | 40% | 100% |

TABLE 4.5 – *Statistiques des échantillons normaux et d'attaques de l'ensemble de données du trafic réseau*

#### 4.4.1.3 *Prétraitement des données et sélection des caractéristiques*

Pour l'étape de prétraitement des données (data pre-processing), nous avons fait le nettoyage (cleaning) et la normalisation. Pour vérifier les valeurs manquantes et les traiter, nous avons utilisé une des fonctions du langage de programmation python appelée Dropna(). Dropna supprime une ligne ou une colonne, qui contient un NaN ou aucune valeur. De plus, pour traiter

les énormes différences entre l'amplitude, les unités et la plage dans le jeu de données généré, nous avons utilisé la fonction de mise à l'échelle (feature scaling). L'échelle des caractéristiques vise à placer toutes les valeurs de l'ensemble de données entre 0 et 1, afin de rendre les caractéristiques plus cohérentes entre elles et de rendre l'étape d'apprentissage moins sensible à ce problème.

Nous avons utilisé dans cette recherche deux types d'algorithmes de sélection de caractéristiques. Le premier est le Forward selection [119], qui appartient à la classe wrappers. Le Forward selection est un algorithme itératif qui commence par un ensemble vide de caractéristiques. Dans chaque itération, il ajoute la meilleure caractéristique qui améliore le modèle jusqu'à ce que l'ajout d'une nouvelle caractéristique n'améliore pas les performances du modèle. L'algorithme Forward feature selection a réduit le nombre de caractéristiques de 37 à 18, comme indiqué dans le tableau 5.3. Le deuxième algorithme de sélection des caractéristiques est le Linear Support Vector Classifier LinearSVC [120], qui fait partie des algorithmes de sélection des caractéristiques intégrées (embedded class). LinearSVC est un algorithme qui donne à chaque caractéristique un attribut coef_ ou feature_importances_. Toutes les caractéristiques sont considérées comme non importantes au départ et il leur donne des valeurs sous un paramètre de seuil. Ensuite, il utilise des heuristiques intégrées pour trouver le seuil de chaque caractéristique à l'aide d'un argument de type chaîne de caractères. LinearSVC a réduit le nombre de caractéristiques de 37 à 22 comme indiqué dans le tableau 5.3. Les meilleurs résultats ont été obtenus en utilisant le sous-ensemble de caractéristiques donné par l'algorithme Forward selection (comme indiqué dans la section des résultats).

### 4.4.2 In-Véhicule ensemble de données

Nous avons utilisé dans notre expérimentation l'ensemble de données construit par Song et al. [114]. Les auteurs ont utilisé la YF Sonata de Hyundai comme véhicule de test. Ils ont connecté un Raspberry Pi3 au CAN bus par le port OBD-II (Figure **??**), et ils ont connecté le Raspberry à un ordinateur por-

| Ensemble de caractéristiques de Forward selection | Ensemble de caractéristiques de LinearSVC |
|---|---|
| Flow Duration, Tot Fwd Pkts, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Tot, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min, Fwd Pkts/s, Bwd Pkts/s, Down/Up Ratio, Idle Mean, Idle Std, Idle Min. | Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, Flow Byts/s, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min, Fwd Header Len, Fwd Pkts/s, Bwd Pkts/s, Pkt Size Avg, Bwd Blk Rate Avg, Init Fwd Win Byts, Init Bwd Win Byts, Active Mean, Active Max. |

TABLE 4.6 – *Liste des caractéristiques réseau sélectionnées*

table par WiFi. Ils ont généré un ensemble de données du trafic in-véhicule qui contient des trames CAN normales et d'autres malveillantes (DoS frames, Fuzzy attaque, RPM et Gear). L'ensemble de données est disponible en ligne [121], étiqueté et en format CSV. Nous avons effectué la même opération de prétraitement de nettoyage et de normalisation décrite dans la section précédente. Le tableau 4.7 présente la répartition des échantillons de l'ensemble de données sur la circulation des véhicules.

| Échantillons | | Nb | | | % | | |
|---|---|---|---|---|---|---|---|
| | | Apprentissage | Test | Totale | Apprentissage | Test | Totale |
| Bénigne | | 1239 | 826 | 2065 | 26.93% | 17.95% | 44.88% |
| Malicieux | DoS Attaque | 504 | 336 | 840 | 10.95% | 7.3% | 18.25% |
| | Fuzzy Attaque | 304 | 203 | 507 | 6.61% | 4.40% | 11.01% |
| | Gear Attaque | 264 | 176 | 440 | 5.74% | 3.82% | 9.56% |
| | RPM Attaque | 449 | 300 | 749 | 9.76% | 6.51% | 16.27% |
| Totale | | 2760 | 1841 | 4601 | 60% | 40% | 100% |

TABLE 4.7 – *Statistiques des échantillons normaux et d'attaque de l'ensemble de données du trafic in-véhiculaire*

### 4.4.3 Résultats et discussion

Pour construire les modèles de classification pour les deux analyseurs (réseau et in-véhicule), nous appliquons les algorithmes d'apprentissage automatique supervisé décrits dans la section 4.3.3. Pour chaque analyseur, nous choisissons l'algorithme qui donne les meilleures performances. Nous formons, validons et testons les deux modèles de classification séparément (ana-

lyseur de réseau et analyseur in-véhicule). Pour chaque ensemble de données, nous utilisant 60 % pour l'apprentissage et la validation du modèle de classification par une validation croisée décuplée (10-fold cross-validation), et les 40 % restants pour tester le modèle. Six mesures communes, Accuracy, Precision, Recall, F1_score, False Positive Rate (FPR), et le False Negative Rate (FNR) ont été sélectionnées pour évaluer les performances de la classification, les mesures susmentionnées peuvent être calculées comme suit :

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \tag{4.7}$$

$$Precision = TP/(TP + FP) \tag{4.8}$$

$$Recall = TP/(TP + FN) \tag{4.9}$$

$$F1\_score = (2 * (precision * recall))/(precision + recall) \tag{4.10}$$

$$FPR = FP/(TN + FP) \tag{4.11}$$

$$FNR = FN/(TP + FN) \tag{4.12}$$

où TP, FP, TN et FN désignent respectivement un vrai positif, un faux positif, un vrai négatif et un faux négatif.

#### 4.4.3.1 *Détection de trafic réseau malveillant*

Dans cette expérimentation, nous évaluons les performances de notre modèle AntibotV pour la classification d'une connexion réseau comme légitime ou comme malveillante. Comme nous pouvons le voir dans le tableau 4.8, tous les classifieurs présentent une grande accuracy ($> 85\%$), la plus grande valeur d'accuracy est obtenue par l'algorithme decision tree (99,4%) suivi par le Random forest (99,3%), tandis que le Naive Bayes présente l'accuracy le plus faible. Cependant, l'accuracy ne suffit pas pour évaluer et sélectionner le meilleur modèle de classification. Par conséquent, d'autres paramètres importants tels que le recall (rappel, ou taux de détection), la précision, le taux de faux négatifs (FNR) et le taux de faux positifs (FPR) doivent être pris en considération. Figure 4.6 compare entre le recall, le F1-score, le FNR, et le FPR pour le trafic bénin et malveillant.

| Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|
| KNN | Trafic Bénin | 98,5% | 99,6% | 99,0% | 2,3% | 0,3% | 98,70% |
| | Trafic Malicieux | 74,53% | 72,15% | 73,23% | 0,05% | 27,83% | |
| | Valeur moyenne | 86,5% | 85,9% | 86,1% | 1,2% | 14,1% | |
| Neural Netowrks | Trafic Bénin | 97,20% | 98,60% | 97,90% | 4,30% | 1,30% | 97,30% |
| | Trafic Malicieux | 77,30% | 74,94% | 75,94% | 1,04% | 25,02% | |
| | Valeur Moyenne | 87,25% | 86,77% | 86,92% | 2,67% | 13,16% | |
| Decision Tree | Trafic Bénin | 99,50% | 99,60% | 99,50% | 0,70% | 0,30% | 99,40% |
| | Trafic Malicieux | 82,88% | 79,70% | 80,88% | 0,03% | 20,28% | |
| | Valeur Moyenne | 91,19% | 89,65% | 90,19% | 0,36% | 10,29% | |
| Random Forest | Trafic Bénin | 99,30% | 99,50% | 99,40% | 1,00% | 0,40% | 99,30% |
| | Trafic Malicieux | 74,80% | 74,70% | 74,75% | 0,05% | 25,28% | |
| | Valeur Moyenne | 87,05% | 87,10% | 87,08% | 0,53% | 12,84% | |
| SVM | Trafic Bénin | 96,40% | 99,40% | 97,80% | 5,80% | 0,50% | 97,20% |
| | Trafic Malicieux | 74,28% | 66,75% | 69,68% | 0,13% | 33,23% | |
| | Valeur Moyenne | 85,34% | 83,08% | 83,74% | 2,96% | 16,86% | |
| Naive Bayes | Trafic Bénin | 96,60% | 83,60% | 89,60% | 4,50% | 16,30% | 87,70% |
| | Trafic Malicieux | 65,95% | 86,23% | 71,48% | 2,73% | 13,73% | |
| | Valeur Moyenne | 81,28% | 84,91% | 80,54% | 3,61% | 15,01% | |

TABLE 4.8 – *Classification binaire du trafic réseau en utilisant AntibotV*

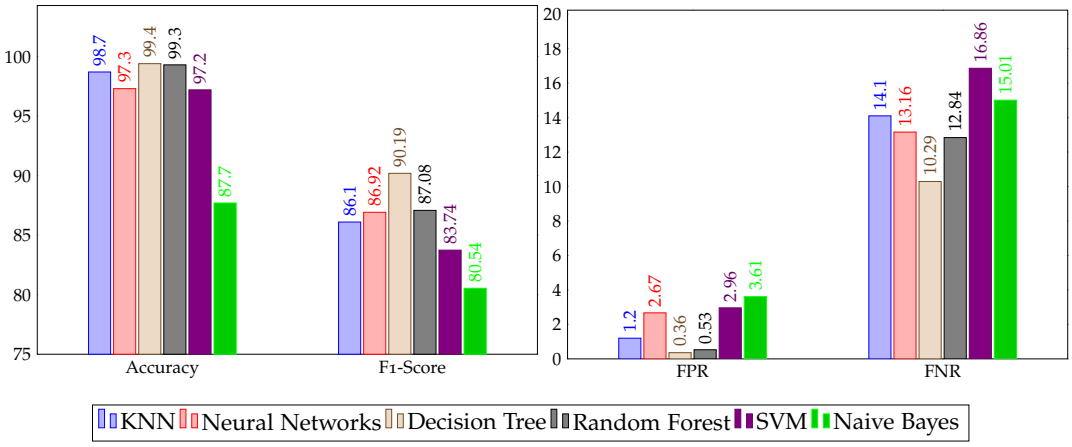


FIGURE 4.6 – *Résultats de la classification binaire du trafic réseau en utilisant AntibotV : Gauche - Précision et F1-score. Droite - FPR et FNR*

Decision tree donne la meilleur valeur de recall, cependant, le taux de détection de malveillance s'avère être de 79,70%, ce qui représente un taux de faux négatifs intolérable (>20 %). Bien que le decision tree soit capable de reconnaître le trafic légitime avec une grande précision, les performances

| Classes | | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|
| Bénigne | Trafic WSMP | 99,06% | 97,70% | 98,38% | 0,16% | 2,29% | 99,40% |
| | Trafic IP | 99,57% | 99,71% | 99,64% | 0,39% | 0,28% | |
| Malicieux | GPS Tracking | 99,61% | 100,00% | 99,80% | 0,08% | 0,00% | |
| | Phishing Attack | 99,47% | 100,00% | 99,73% | 0,07% | 0,00% | |
| | WSMP Flood | 97,43% | 98,70% | 98,06% | 0,14% | 1,29% | |
| | GeoFlood | 100,00% | 91,66% | 95,65% | 0,00% | 8,30% | |
| Valeur Moyenne | | 99,19% | 97,96% | 98,54% | 0,14% | 2,03% | |

TABLE 4.9 – *Résultats de la classification multiclasse du trafic réseau en utilisant AntibotV basé sur l'algorithme Decision Tree*

de détection du trafic malveillant ne sont pas prometteuses. Le faible taux de détection du trafic malveillant est dû à l'hétérogénéité du trafic légitime. Contrairement aux autres types de réseaux, il existe deux types de trafic réseau dans les réseaux véhiculaires : IP et WSMP. Les deux types de trafic réseau présentent des schémas de trafic différents, influencés par plusieurs facteurs tels que : le contexte d'utilisation, la durée, la taille des paquets, etc. Cette différence représente un défi pour la formation du modèle de classification, et par la suite induit en erreur le classificateur sur la différenciation entre le trafic WSMP légitime et le trafic malveillant.

Pour surmonter le problème susmentionné, nous créons deux classes de trafic légitime en séparant le trafic WSMP et le trafic IP. Nous pensons que la formation du modèle de classification avec deux catégories séparées de trafic légitime, réduira le taux de faux négatifs, et améliorera ainsi le taux de détection du trafic malveillant. En outre, pour fournir une mesure de réponse appropriée, le framework de détection doit identifier le type d'attaques. Par conséquent, nous séparons le trafic malveillant en différentes classes : GPS tracking, WSMP flood, phishing, and Geo-WSMP flood. D'après le tableau 4.9, nous pouvons constater une amélioration significative du taux de détection du trafic malveillant, le recall est passé de 79,7% à 97,59%. La figure 4.7 compare le recall, le F1-score, le FNR et le FPR pour chaque classe de trafic séparément. Nous voyons que le decision tree acheive les meilleures performances. Bien qu'il y'a une légère baisse dans les résultats en générale, le recall de la classe

bénigne reste élevé (>98). Le taux de détection de l'attaque Geo-WSMP flood n'est pas aussi bon que les autres attaques. Cela peut s'expliquer par le fait que la majorité des applications de sécurité sur le protocole WSMP utilisent la diffusion, qui est assez similaire à l'attaque Geo-WSMP flood.

D'après les résultats susmentionnés, il est important de noter que l'attaque Geo WSMP flood représente l'attaque la plus difficile à détecter. Le Decision Tree a atteint le taux de détection le plus élevé avec le taux de faux positifs le plus bas. Outre la détection du trafic malveillant, il offre également des performances élevées pour identifier le type de trafic bénin (WSMP, IP) et malveillant (suivi GPS, phishing, inondation WSMP, Geo flood). Par conséquent, le Decision Tree est le meilleur classifieur à sélectionner pour le module d'analyse de réseau.

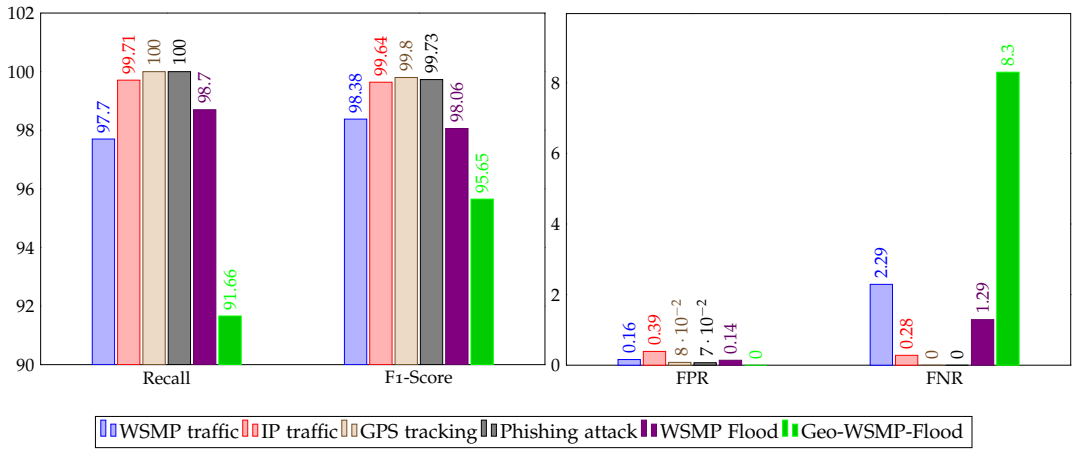


FIGURE 4.7 – *Résultats de la classification multiclasse du trafic réseau. Algorithme Decision Tree : Gauche - Recall, F1-Score. Droite - FPR et FNR*

#### 4.4.3.2 *Détection du trafic in-véhiculaire malveillant*

Nous évaluons les performances du modèle AntibotV en matière de détection et d'identification des attaques in-véhiculaires pouvant être menées par un bot malveillant. Tout d'abord, nous effectuons une classification binaire (bénigne/malveillante) en utilisant les algorithmes d'apprentissage automatique supervisé décrits plus haut dans la section 4.3.3, puis nous évaluons le meilleur algorithme de classification pour identifier le type d'attaque. Les diagrammes d'accuracy dans la figure 4.8 montrent clairement que le framework proposé est très performant pour la détection et l'identification du trafic

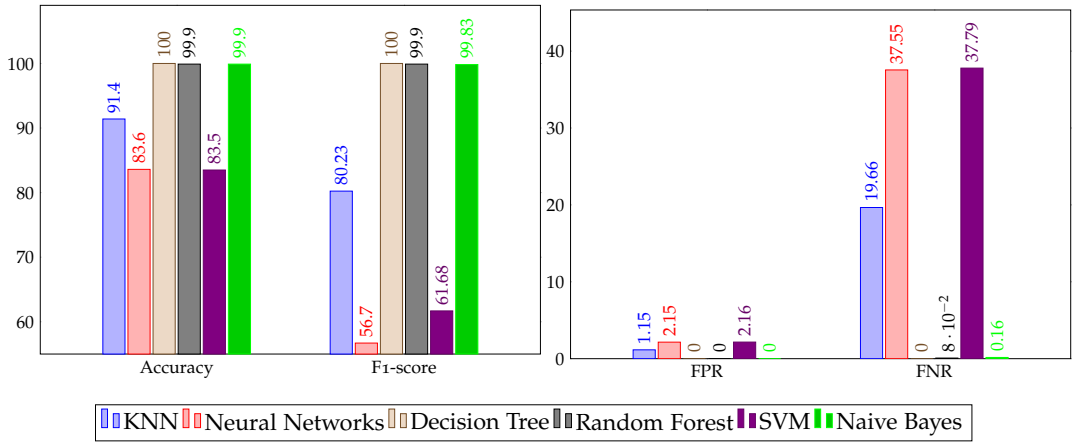


FIGURE 4.8 – *Résultats de la classification binaire du trafic in-véhiculaire : Gauche : Accuracy et F1-score. Droite : FPR et FNR*

in-véhiculaire, avec plus de 90 % pour les algorithmes KNN, decision tree, random forest, et naive bayes. Les résultats de la classification binaire sont présentés dans le tableau 4.10. On constate que le decision tree fournit le taux de détection maximal pour les trames bénignes et malveillantes. Les algorithmes KNN, Neural Networks et SVM n'ont pas réussi à détecter la classe de trafic malveillant, avec un faible taux de détection (entre 25 et 61 %). Dans le tableau 4.11, on peut voir que le decision tree peut parfaitement identifier les différents types d'attaques in-véhiculaires. Ces résultats confirment l'efficacité des algorithmes basés sur les arbres pour la détection des intrusions. Par conséquent, le decision tree est le meilleur classifieur à sélectionner pour le module d'analyse in-véhiculaire.

### 4.4.4 Discussion et comparaison

A notre connaissance, il n'existe qu'un seul article [1] qui a aborde la détection des botnets véhiculaires, les auteurs ont utilisé une technique de détection d'anomalies pour permettre la détection de nouvelles formes de messages BSM injectés. La solution de détection suppose un protocole de communication spécifique au botnet (GHOST), ce qui la rend incapable de détecter les botnets utilisant un protocole de communication différent. Le framework proposé AntibotV ne suppose pas un protocole de communication de botnet particulier, de sorte que l'approche de détection fonctionne indépendamment du protocole de communication de botnet. En limitant la détection à la seule sur-

| Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|
| KNN | Trames bénignes | 99,90% | 100,00% | 99,90% | 0,10% | 0,00% | 91,40% |
| | Trames malveillantes | 61,30% | 60,58% | 60,55% | 2,20% | 39,33% | |
| | Valeur moyenne | 80,60% | 80,29% | 80,23% | 1,15% | 19,66% | |
| Neural Networks | Trames bénignes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 83,60% |
| | Trames malveillantes | 12,48% | 24,85% | 13,40% | 4,30% | 75,10% | |
| | Valeur moyenne | 56,24% | 62,43% | 56,70% | 2,15% | 37,55% | |
| Decision Tree | Trames bénignes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 100,00% |
| | Trames malveillantes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| | Valeur moyenne | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| Random Forest | Trames bénignes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 99,90% |
| | Trames malveillantes | 99,83% | 99,83% | 99,80% | 0,01% | 0,15% | |
| | Valeur moyenne | 99,91% | 99,91% | 99,90% | 0,00% | 0,08% | |
| SVM | Trames bénignes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 83,50% |
| | Trames malveillantes | 23,33% | 24,33% | 23,35% | 4,33% | 75,58% | |
| | Valeur moyenne | 61,66% | 62,16% | 61,68% | 2,16% | 37,79% | |
| Naive Bayes | Trames bénignes | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 99,90% |
| | Trames malveillantes | 99,68% | 99,65% | 99,65% | 0,00% | 0,33% | |
| | Valeur moyenne | 99,84% | 99,83% | 99,83% | 0,00% | 0,16% | |

TABLE 4.10 – *Classification binaire du trafic in-véhiculaire en utilisant AntibotV*

| Classes | | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|
| Trames bénignes | | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | 100,00% |
| Trames malveillantes | DoS | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| | Fuzzy | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| | Gear | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| | RPM | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |
| Valeur moyenne | | 100,00% | 100,00% | 100,00% | 0,00% | 0,00% | |

TABLE 4.11 – *Classification multiclasse du trafic in-véhiculaire en utilisant AntibotV basé sur l'algorithme Decision Tree*

veillance des communications de réseau, [1] ne peut pas détecter les attaques à l'intérieur des véhicules. Grâce à une surveillance à deux niveaux, le AntibotV peut également détecter les attaques in-véhiculaires. Par rapport à [1], le AntibotV offre un meilleur accuracy et un taux de faux positifs plus faible (voir tableau 4.12). Seo et al. [114] ont considéré les menaces in-véhiculaire, ils ont proposé un framework de détection des anomalies basé sur le convolutional neural network et deep neural network. Bien que le AntibotV offre de meilleures performances, l'approche proposée [114] permet de détecter des

attaques invisibles, mais au détriment des fausses alertes. En outre, les techniques d'apprentissage en profondeur (deep learning) nécessitent une grande quantité de ressources de calcul et de mémoire, ce qui pourrait être contraignant dans le contexte véhiculaire. Le tableau 4.12 fournit une comparaison entre le AntibotV et les solutions existantes [1, 114].

| | AntibotV (notre framework) | [1] | [114] |
|---|---|---|---|
| Niveau de menace | Réseau & In-véhicule | Réseau | In-véhicule |
| Technique de Machine learning | Classification | Anomaly detection | Anomaly detection |
| Algorithme | Decision Tree | 3D DBSCAN | CNN & DNN |
| Exigence de ressources | Faible | Faible | Élevé |
| Accuracy | 99.40% & 100% | 77% | 97.53% |
| Detection rate (Recall) | 97.96% & 100% | NA | 98.65% |
| Precision | 99.19% & 100% | NA | 97.63% |
| FPR | 0.14% & 0% | NA | NA |

TABLE 4.12 – *Comparaison entre les systèmes de détection des botnets véhiculaires*

## 4.5 Conclusion

Dans ce chapitre, nous avons proposé AntibotV, un framework de détection des botnets véhiculaires, basé sur une detection comportementale à plusieurs niveaux. Nous avons envisagé deux nouvelles zero-day attaques de types Déni de Service, ainsi que d'autres in-véhiculaires attaques. Le framework proposé permet de surveiller l'activité du véhicule au niveau du réseau et in-véhiculaire. Pour construire le système de détection, nous avons recueilli des données sur le trafic réseau des applications légitimes et malveillantes. Ensuite, nous avons formé en utilisant le decision tree un nouveau classifieur avec un ensemble de features que nous avons extraites et sélectionnées. De même, nous avons formé le classifieur decision tree avec des données in-véhiculaires. Les résultats expérimentales ont montré que le AntibotV est plus performant que les solutions existantes, il atteint un taux de détection supérieur à 97 % et un taux de faux positifs inférieur à 0,14 %. Cependant, l'idéale était de faire de l'apprentissage de notre IDS (AntibotV) sur un ensemble de données réaliste afin d'ajouter une dimension de réalité sur le modèle formel. C'est pour cela, dans le chapitre suivant, nous allons travaillé sur la génération

d'un prototype d'une nouvelle base de données de traces du trafic réseaux véhiculaire réelle. Une base de donnée qui peut être utiliser pour l'apprentissage de notre IDS, ainsi que pour les autres recherches sur les VANETs.

# Vers le développement d'un ensemble de données DoS réaliste pour les systèmes de transport intelligents

## 5.1 Introduction

Les systèmes de détection d'intrusion a base de machine learning (IDS-ML) représentent l'une des approches les plus efficace pour la protection des réseau contre les cyberattaques [5], et récemment ils ont beaucoup attiré l'attention de la communauté scientifique. Basé sur un modèle de classification, l'IDS analyse un flux de données (ex : ensemble de paquets réseaux) pour vérifier s'il existe une activité suspecte, et éventuellement une cyberattaque. Étant savoir que l'efficacité de l'IDS dépend du modèle de classification, il est primordiale de faire l'apprentissage du classifieur sur un ensemble de données qui contient des traces de trafic réaliste, représentatif et qui couvre une variété d'échantillons normaux et d'attaques.

Cependant, pendant le développement de notre framework (AntibotV), nous n'avons pas trouvé un ensemble de données réaliste dédiés spécialement pour réseaux véhiculaires, et la totalité des ensembles de données existants [2, 3, 4, 5] contiennent des traces de trafic réseau générées et capturées par des simulateurs, ce qui ne prennent pas en valeur l'impact de l'environnement. Par conséquent, les modèles des systèmes de détection d'intrusion basés sur des données générer a partir des simulateurs peuvent ne pas être réaliste, représentatif, et ne fonctionne pas correctement dans un environnement réel.

Dans ce chapitre, nous générons une nouvelle base de données de traces du trafic réseaux véhiculaire, VDoS-LRS. Une base de données qui contient un trafic réseau réel et de divers types de cyberattaques déni de service (DoS). Nous créons un environnement de test réaliste qui tient en compte de différents types d'environnement (urbain, routier et rural). En outre, nous explorons un large éventail de caractéristiques de trafic (en anglais, features) pour détecter et classer le trafic réseau véhiculaire. Nous évaluons la fiabilité de l'ensemble de données VDoS-LRS en utilisant différents algorithmes d'apprentissage automatique à des fins de cyber-criminalistique. Pour être plus précis, les principales contributions de ce chapitre sont les suivantes :

— Un nouvel ensemble de données DoS réalistes pour les réseaux de vé-

hicules est disponible sur demande, avec une description détaillée de la conception et de la configuration du banc d'essai.

— Nous évaluons les performances des méthodes d'analyse de cyber criminalité, basées sur des algorithmes d'apprentissage automatique en utilisant l'ensemble de données VDoS-LRS.

Le reste du chapitre est structuré comme suit : la description et le contexte d'application des attaques DoS et DDoS dans les réseaux véhiculaires est examinée dans la section 5.2. Dans la section 5.4, nous présentons en détail le banc d'essai utilisé pour créer le jeu de données VDoS-LRS. La section 5.5 présente les algorithmes d'apprentissage automatique utilisés pour le processus de classification. Dans la section 5.6, nous présentons et discutons les résultats expérimentaux. Nous terminons par une conclusion.

## 5.2 Les scénarios de l'attaque Déni de service (DoS) déployés

Dans la section 4.2.1 nous avons présenté deux zero-day DoS attaques spécifiques aux réseaux véhiculaires (WSMP-Flood et Geo-WSMP-Flood). Les deux attaques exploitent le Wave Short Message protocole (WSMP), qui est le protocole utilisé par les applications de sécurité et de gestion du trafic. Cependant, le déploiement réel de ces deux attaques n'est pas possible, car jusqu'à aujourd'hui aucun générateur de traffic WSMP n'est disponible. C'est pour cela, nous avons travaillé sur d'autre variante de l'attaque DoS.

En gros, les attaques DoS peuvent être divisées en trois types : les attaques basées sur le volume (qui saturent la bande passante du site attaqué comme UDP-Flood), les attaques basées sur le protocole (qui consomment les ressources réelles du serveur comme SYN-Flood) et les attaques de la couche application (comprenant des requêtes apparemment légitimes et innocentes comme l'attaque Slowloris). Quyoom et al. [122] ont étudié différents scénarios de DoS dans le contexte les réseaux véhiculaires. Les attaques DoS dans les réseaux de véhicules peuvent cibler les ressources des véhicules, les unités

en bord de route (RSU) et les canaux de communication. Par exemple, si un véhicule légitime tente de démarrer une connexion TCP avec un RSU :

1. Le véhicule demande l'établissement de la connexion en envoyant un message SYN (synchronisation).
2. Le RSU répond au véhicule en lui renvoyant un message SYN-ACK.
3. Le véhicule lui renvoie un ACK, et l'opération d'échange peut commencer.

Dans le cas du SYN-Flood (figure 5.1), le véhicule malveillant ne répond pas au RSU par un ACK, ce qui pousse le RSU à attendre l'ACK un certain temps pour éviter l'encombrement du réseau. En fonction du nombre de requêtes envoyées par l'attaquant, le RSU sera temporairement indisponible pour communiquer avec les véhicules légitimes. L'attaquant peut également brouiller le canal, de telle sorte que les véhicules ne pourraient pas accéder au canal et communiquer.

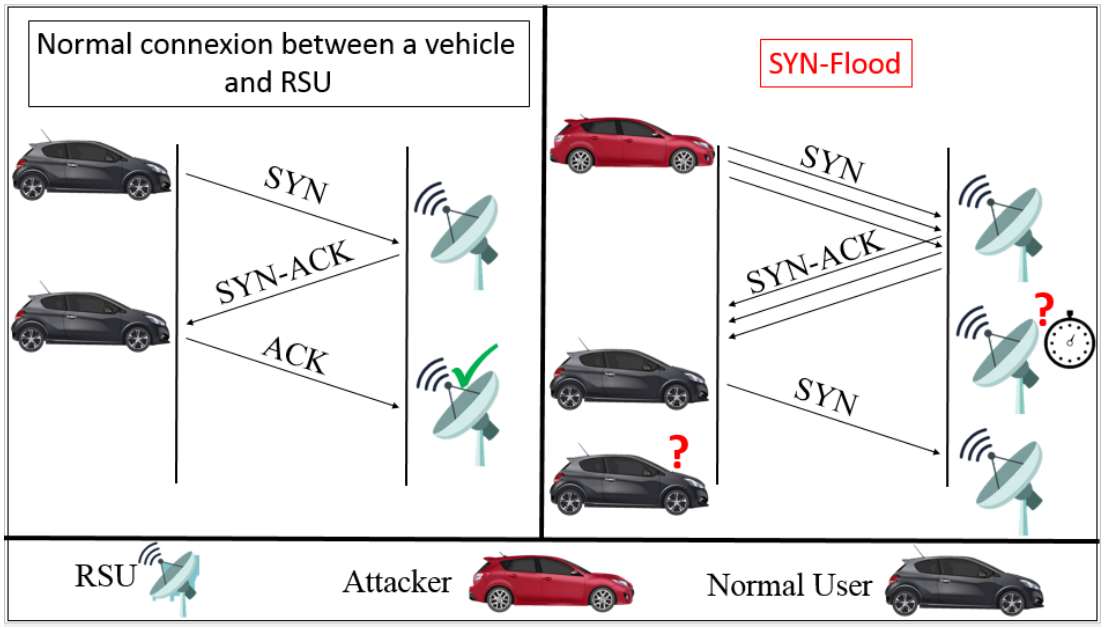


FIGURE 5.1 – *L'attaque SYN-Flood dans le contexte d'un réseau véhiculaire*

## 5.3 Les datasets de trafic réseau véhiculaires existantes

De nombreuses recherches ont été effectuées dans la littérature pour protéger les réseaux véhiculaires contre les cyber-attaques, la plupart d'entre elles

sont basées sur des techniques d'apprentissage automatique [5, 2, 3, 4, 123, 124, 125, 126, 127, 128, 129, 130, 131]. Celles-ci nécessitent des ensembles de données pour analyser les flux de réseau et modéliser le trafic réseau normal et malveillant.

Il y a eu plusieurs ensembles de données contenant le trafic du réseau, par exemple, KDD99 [132], CAIDA datasets [133], NSL-KDD [134], ISCX [135],CICDS2017 [136], UNSW-NB15 [137], qui ont été utilisés par les chercheurs pour la détection des intrusions et l'analyse médico-légale. Tous les ensembles de données susmentionnés ne comprennent pas des données de trafic véhiculaires, ce qui les rend inutilisables dans le contexte des réseaux véhiculaires.

Bien que certaines recherches récentes [2, 4, 126, 5] aient produit des ensembles de données synthétiques sur les réseaux véhiculaires à des fins de détection d'intrusion, le développement d'un ensemble de données réalistes sur le trafic des réseaux véhiculaires qui inclut des scénarios de véhicules est encore un sujet inexploré. Les bancs d'essai et les ensembles de données les plus récents sont brièvement expliqués ci-dessous, avec une comparaison entre eux et notre base VDoS-LRS dans le tableau 5.1.

Singh et al. [2] ont proposé une approche basée sur l'apprentissage automatique pour détecter les attaques Wormhole dans les réseaux véhiculaires. Pour la production de l'ensemble de données, ils ont utilisé le simulateur NS3 [138] qui utilise les traces de mobilité générées par le simulateur de trafic SUMO. Ils ont utilisé quarante véhicules dans le banc d'essai avec une communication multi-hop en utilisant le protocole de routage AODV. Cependant, dans le scénario de simulation, il est supposé que tous les véhicules se sont déplacés à la même vitesse pendant toute la durée de la simulation (15m/s), ce qui est irréaliste dans le contexte des véhicules. De plus, la durée de la simulation est trop courte (400 secondes) pour refléter une activité réelle des véhicules.

Alheeti et al. [4], ont proposé un ML-IDS pour détecter les attaques de grey hole dans un réseau véhiculaires. Pour la classification, ils ont utilisé le MVC et les réseaux neuronaux à action directe (FFNN). L'ensemble de don-

nées utilisé pour entraîner l'IDS a été extrait d'un fichier de trace généré par simulation. Les auteurs n'ont pas fourni de détails sur la configuration du banc d'essai, ce qui rend leur simulation difficile à reproduire. Dans [3], les auteurs ont proposé une approche basée sur ML pour classifier le comportement du nœud, c'est-à-dire si les nœuds de communication dans le réseau de véhicules sont honnêtes ou malveillants. Ils ont testé différents algorithmes de classification : Naive Bayes, IBK, J-48, Random Forest et Ada Boost. Cependant, l'IDS proposé a été formé sur un ensemble de données de traces de réseau collectées par simulation à l'aide du simulateur NCTUns-5.0 [139]. En outre, les paramètres de simulation ne reflètent pas l'environnement réel, une zone de 6 km et une durée de simulation de 2000 secondes. Dans [126], les auteurs ont proposé une approche basée sur l'exploration de données pour la détection en temps réel des attaques de refus de service par brouillage radioélectrique dans les communications de véhicule à véhicule (V2V) IEEE 802.11p. Cependant, le système proposé a été entraîné statistiquement pour de courtes séquences d'entraînement de 5 et 100 secondes, ce qui semble insuffisant pour entraîner un IDS et couvrir de multiples scénarios d'attaque par brouillage radioélectrique.

Dans [5], les auteurs ont essayé de développer une méthode et un outil pour générer des ensembles de données proches de la réalité pour la détection des intrusions dans les réseaux véhiculaires. L'ensemble de données des traces de réseau a été généré à l'aide du simulateur NS-3 [138], et comprend différents types d'attaques. Il est intéressant de voir qu'aucune des [2] [3] [4] [5] les recherches dans la littérature ont utilisé des traces de réseaux réels pour faire l'apprentissage des ML-IDS. Sachant que la formation d'un IDS avec des données extraites d'un simulateur peut ne pas être fiable, réaliste et représentative des propriétés du réseau de véhicules liées à l'environnement réel.

À la lumière de ce qui précède, cette recherche vise à concevoir un ensemble de données réelles sur le trafic des réseaux véhiculaires afin de donner une dimension de fiabilité à nos recherches pour la détection des attaques par déni de service dans les réseaux véhiculaires.

| Dataset | Configuration réaliste du banc d'essai | Trafic réaliste | l'impact de l'environnement | Données étiquetées | Traces VANET | Divers scénarios DoS | Full packet capture | Nouvelles features générées |
|---|---|---|---|---|---|---|---|---|
| [2] | F | F | T | T | T | F | F | F |
| [3] | F | F | F | T | T | F | F | F |
| [4] | F | F | T | T | T | F | T | F |
| [5] | F | F | F | T | T | T | T | F |
| Notre travail | T | T | T | T | T | T | T | T |

TABLE 5.1 – *Comparaison des ensembles de données de trafic véhiculaires (T=vrai, F=faux)*

## 5.4 Génération de jeux de données

Dans cette section, nous présentons en détail les étapes suivies pour générer notre ensemble de données : configuration du banc d'essai, extraction des caractéristiques, pré-traitement des données et sélection des caractéristiques comme le montre la figure 5.2.

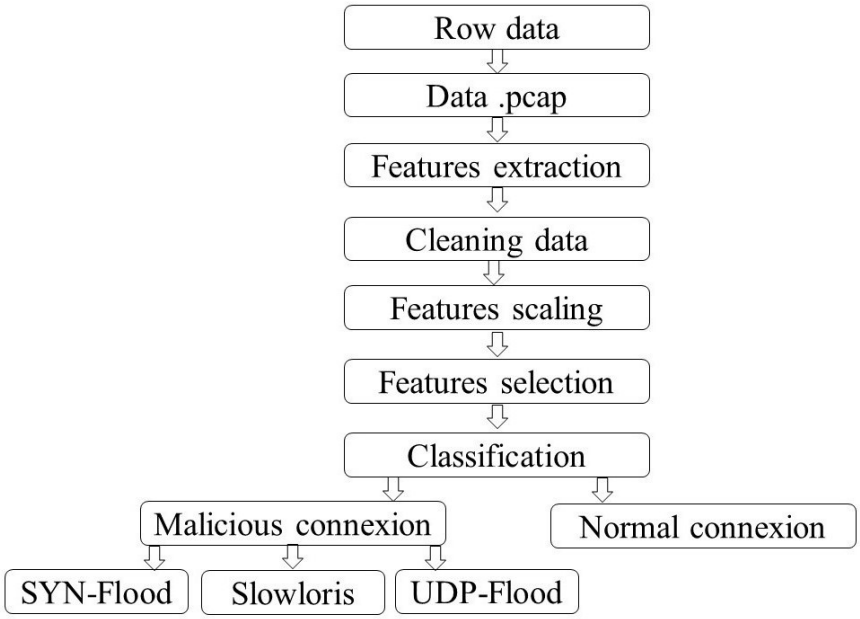


FIGURE 5.2 – *Processus de détection des DoS attaques*

### 5.4.1 Le banc d'essai proposé

Le banc d'essai a été déployé à Annaba, Algérie, nous avons utilisé : deux véhicules, 3 machines physiques, 4 machines virtuelles, 2 points d'accès, un modem 4G et 2 antennes Cisco, comme le montre la figure 5.3. En raison de contraintes de ressources, nous avons utilisé deux véhicules avec plusieurs machines virtuelles. Les deux véhicules (figure 5.4.a) ont été connectés en

mode V2V en utilisant le réseau IEEE 802.11a fourni par deux points d'accès AIR-AP1231G-E-K9 (figure 5.4.b). Les points d'accès ont été connectés avec deux antennes pour l'extension de la couverture (Sharkee GPSB -figure 5.4.d- et Cisco AIR -ANT1949 Antennes -figure 5.4.c-). Cette configuration de réseau assure la connexion entre les deux machines physiques (ordinateur Dell équipé d'un processeur Intel® Pentium® 3558U @ 1,70 GHz dans le premier véhicule et ordinateur Dell équipé d'un processeur Intel® Core i5-4200 U @ 1,60 GHz 2,30 GHz dans le second véhicule).

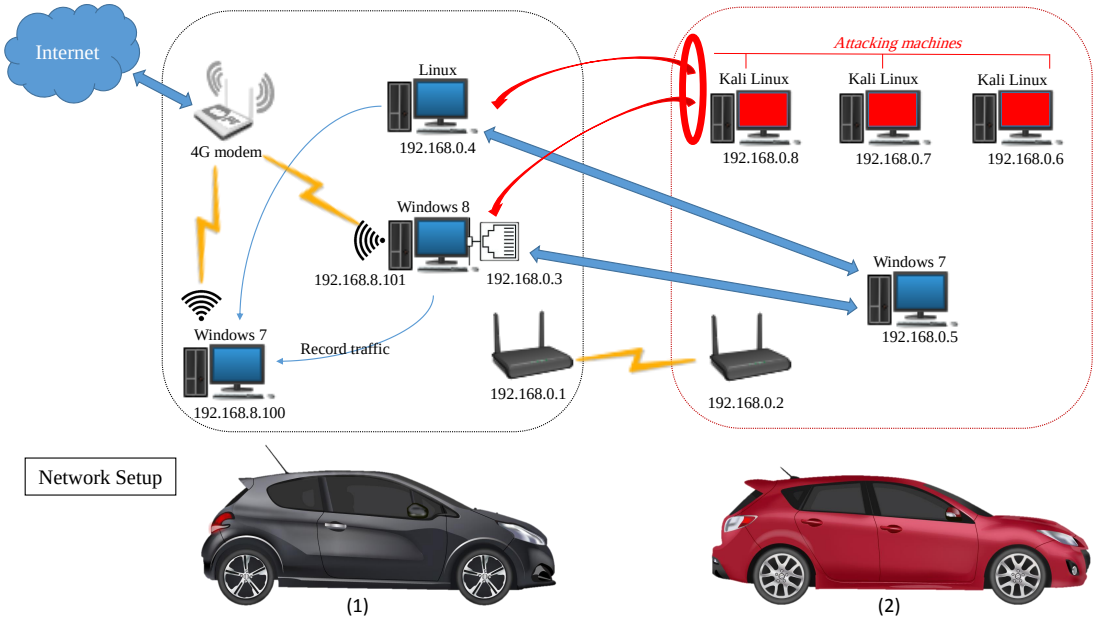


FIGURE 5.3 – *L'environnement du banc d'essai utilisé pour la génération de la nouvelle base de données véhiculaires, VDoS-LRS*

La première machine fonctionne sous le système d'exploitation Windows 8 avec deux interfaces réseau (interface Ethernet avec adresse IP '192.168.0.3' et interface Wi-Fi avec adresse IP '192.168.8.101'). La deuxième machine fonctionne sous le système d'exploitation Windows 7 avec l'adresse IP '192.168.0.5'. En utilisant l'outil Oracle VM virtual Box, la première machine exécute un autre système Linux avec l'adresse IP '192.168.0.4', et la seconde machine physique (dans le second véhicule) exécute 3 autres machines virtuelles Kali-Linux, avec les trois adresses IP suivantes '192.168.0.6', '192.168.0.7', '192.168.0.8'. Les trois machines Kali représentent les attaquants dont chaque machines exécute un type différent d'attaque DoS. Les trois autres machines comprennent deux victimes.

Comme le montre la figure 5.3, pour le trafic malveillant, les trois machines Kali-Linux (dans le second véhicule) lancent une attaque DoS prenant pour cible les deux machines victimes (dans le premier véhicule) avec des paquets UDP Flood, SYN Flood et Slowloris en alternance. Pour le trafic bénin, nous avons essayé de simuler différents services VANETs. Par exemple, pour l'échange d'informations et la collaboration entre les nœuds des VANETs, nous avons utilisé Packet Sender [140], un logiciel générateur de trafic, qui envoie des paquets UDP, TCP et SSL à plusieurs clients simultanément. Pour le partage de fichiers entre des nœuds bénins, nous avons réalisé un partage de fichiers en utilisant le protocole SMB (Server Message Block) qui fonctionne comme un protocole de réseau de couche application ou de couche présentation. Nous avons également pris en compte les activités communes des utilisateurs telles que l'accès à Google Maps, YouTube, les réseaux sociaux et les applications en temps réel comme les appels vidéo et audio. Pour la collecte des données, nous avons utilisé une autre machine physique Windows 7, qui effectue une analyse du trafic réseau en executant l'outil Wireshark [141]. L'expérimentation a été menée dans trois environnements différents : urbain, rural et routier. Le trafic généré a été stocké sous forme de fichiers pcap.

Nous avons nommé l'ensemble de données produit "VDoS-LRS", qui signifie Vehicular Denial of Service - Networks and Systems Laboratory. L'ensemble de données contient du trafic normal et du trafic malveillant de déni de service. Le VDoS-LRS a pris en compte trois types d'attaques par déni de service. SYN Flood exploite les vulnérabilités du protocole TCP ; il consiste en un envoi massif de requêtes SYN au véhicule. L'objectif de cette attaque est de rendre le véhicule indisponible pour les véhicules légitimes en épuisant ses ressources. UDP Flood, qui représente la catégorie basée sur le volume, dans laquelle l'attaquant submerge des ports aléatoires sur l'hôte cible avec des paquets IP contenant des datagrammes UDP. Le destinataire renvoie un paquet "Destination Unreachable". Comme de plus en plus de paquets UDP sont reçus et font l'objet d'une réponse, le système devient submergé et ne répond plus aux autres clients. Slowloris, qui représentent les attaques DoS sur les couches applicatives. Il peut faire tomber une autre machine avec une bande

passante minimale et des effets secondaires sur des services et des ports non liés, en gardant ouvertes de nombreuses connexions vers la cible et en les maintenant ouvertes aussi longtemps que possible (en utilisant une requête HTTP non complétée).

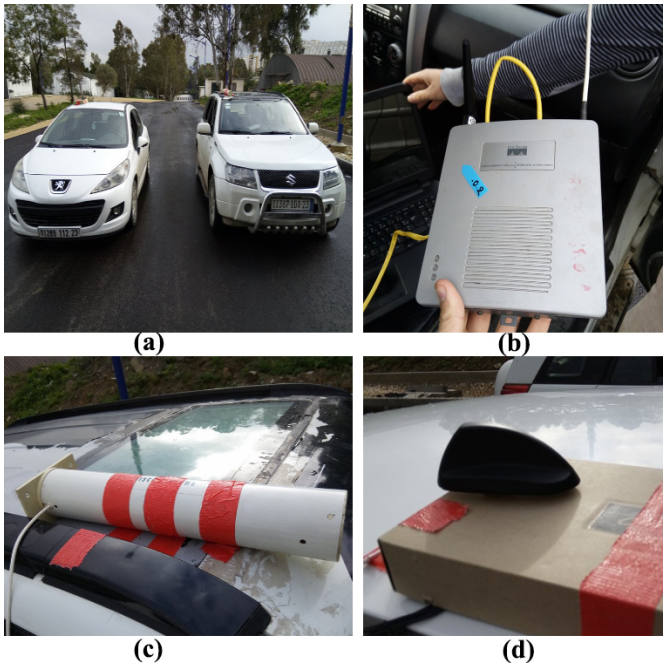


FIGURE 5.4 – *Équipements utilisés dans le banc d'essai*

Dans notre banc d'essai, nous prenons en considération les caractéristiques intrinsèques des VANET. Trois environnements différents ont été pris en considération : urbain, routier et rural. Chaque environnement a ses propres caractéristiques. Par exemple, la disponibilité de la couverture du réseau en milieu rural n'est pas aussi bonne qu'en milieu urbain. En ce qui concerne la vitesse des véhicules, nous considérons les vitesses recommandées dans chaque environnement (vitesse moyenne en milieu urbain 40 Km/h, 90 Km/h sur autoroute et 30 Km/h en milieu rural) (Plus de détails dans l'Annexe A).

### 5.4.2 Features extraction

Après avoir collecté le trafic réseau (sous forme de fichiers PCAP) correspondant aux trois environnements (urbain, rural, routier), nous extrayons un ensemble de 79 caractéristiques du réseau (voir tableau 5.2). Pour extraire les caractéristiques, nous utilisons CICFlowMeter [118], un générateur de flux de trafic réseau distribué par l'Institut canadien de cybersécurité (CIC). Il génère

des flux bidirectionnels, où le premier paquet détermine les directions avant (source à destination) et arrière (destination à source). Notez que les flux TCP sont généralement terminés lors du démontage de la connexion (par le paquet FIN) tandis que les flux UDP sont terminés par un délai d'attente. La valeur du délai d'attente du flux peut être attribuée arbitrairement par le schéma individuel, par exemple 600 secondes pour TCP et UDP.

Nous considérons les mêmes cinq catégories de caractéristiques de réseau que Lashkari et al. [142], qui sont basées sur : le comportement, les octets, les paquets, le temps et le flux (Tableau 5.2). Les caractéristiques basées sur le comportement sont utilisées pour évaluer un objet en fonction de ses actions prévues avant qu'il ne puisse exécuter ce comportement. Par exemple, si une connexion dure longtemps, ce comportement peut être celui d'une attaque DoS. Les fonctionnalités basées sur les octets et les paquets sont utilisées pour compter le nombre d'octets/paquets échangés. Les caractéristiques basées sur les octets/paquets permettent de détecter une augmentation importante et anormale du trafic, qui est symptomatique des attaques par déni de service. En outre, le temps écoulé entre la transmission des paquets lors d'une attaque DoS est trop court (en particulier TCP Flood et UDP Flood), c'est pourquoi les fonctionnalités basées sur le temps peuvent être révélatrices des attaques DoS. En outre, les caractéristiques de flux de données sont comme les caractéristiques basées sur les paquets mais avec un espace de stockage réduit. Au lieu de travailler sur les paquets, nous travaillons sur un flux de paquets.

Nous ne prenons pas en compte les adresses IP et les ports source et destination, car les attaquants peuvent les modifier facilement. De plus, cela peut induire en erreur le modèle de classification et l'empêcher d'analyser avec précision le reste des fonctionnalités.

### 5.4.3 Pré-traitement des données et sélection des caractéristiques

Pour l'étape de pré-traitement, nous avons fait le nettoyage et la normalisation des données. Pour vérifier les valeurs manquantes et les traiter, nous avons utilisé une des fonctions du langage de programmation python appelée Dropna(). Dropna supprime une ligne ou une colonne d'une trame de

| Carastérestiques | | Description |
|---|---|---|
| **Nb** | **Nom** | |
| 1 | Flow Duration | Durée du flux en microsecondes |
| 2 | Flow IAT Mean | Temps moyen entre deux paquets envoyés dans le flux |
| 3 | Flow IAT Std | Temps d'écart standard entre deux paquets envoyés dans le flux |
| 4 | Flow IAT Max | Temps maximum entre deux paquets envoyés dans le flux |
| 5 | Flow IAT Min | Délai minimum entre deux paquets envoyés dans le flux |
| 6 | Fwd IAT Tot | Temps total entre deux paquets envoyés dans le sens de l'acheminement |
| 7 | Fwd IAT Mean | Temps moyen entre deux paquets envoyés dans le sens de l'acheminement |
| 8 | Fwd IAT Std | Temps standard entre deux paquets envoyés dans le sens de l'acheminement |
| 9 | Fwd IAT Max | Délai maximal entre deux paquets envoyés dans le sens de l'acheminement |
| 10 | Fwd IAT Min | Délai minimum entre deux paquets envoyés dans le sens de l'acheminement |
| 11 | Down/Up Ratio | Taux de téléchargement et de chargement |
| 12 | Idle Mean | Temps moyen pendant lequel un flux était inactif avant de devenir actif |
| 13 | Idle Std | Temps standard pendant lequel un flux était inactif avant de devenir actif |
| 14 | Idle Min | Durée minimale d'inactivité d'un flux avant de devenir actif |
| 15 | Tot Fwd Pkts | Total des paquets dans le sens de l'acheminement |
| 16 | Flow Pkts/s | Nombre d'octets de flux par seconde |
| 17 | Fwd Pkts/s | Nombre de paquets transmis par seconde |
| 18 | Bwd Pkts/s | Nombre de paquets de retour par seconde |
| 19 | total_Bwd_Packets | Le nombre total de paquets dans le sens inverse d'acheminement. |
| 20 | total/min/max/mean/std_f/b_Pktl | La taille des paquets et la taille standard de l'écart en avant ou en arrière. |
| 21 | Avg_Packet_Size | Taille moyenne des paquets. |
| 22 | f/b_AvgSegmentSize | La taille médiane constatée dans le sense avant ou arrière. |
| 23 | f/b_AvgPacketsPerBulk | Le nombre moyen de paquets en vrac dans le sens de l'avant ou de l'arrière. |
| 24 | f/b_AvgBulkRate | Le nombre moyen de taux de vrac dans le sens de l'avant ou de l'arrière |
| 25 | act_data_pkt_forward | Le nombre de paquets avec au moins 1 octet de données TCP dans le sens de l'acheminement |
| 26 | min_seg_size_forward | La plus petite taille de segment constatée dans le sens de l'avancement. |
| 27 | Total_f/b_headers | Le nombre total d'octets utilisés pour les en-têtes dans le sens avant ou arrière |
| 28 | f/b_AvgBytesPerBulk | Le nombre moyen d'octets en vrac dans le sens avant ou arrière. |
| 29 | Init_Win_bytes_forward/backward | Le nombre total d'octets envoyés dans la fenêtre initiale dans le sens avant ou arrière. |
| 30 | total/min/max/mean/std_Bwd_iat | Le temps total, max, min, moyen et standard entre les paquets pour le sens inverse. |
| 31 | f/b_psh/urg_cnt | Le nombre de flags PSH ou URG a été fixé par paquets dans le sens avant ou arrière. |
| 32 | Min/mean/max/std_active | Le temps min, max, moyen et std pendant lequel un flux était actif avant de devenir inactif. |
| 33 | Flow_Byts_PerSecond | Le nombre d'octets de flux par seconde. |
| 34 | min/max_flowpktl | La durée (min, max) du flux. |
| 35 | Flow_fin/syn/rst/psh/ack/urg/cwr/ece | Le nombre de paquets avec des flags. |
| 36 | Sflow_f/b_Packet | Le nombre moyen de paquets dans un flux secondaire pour la direction avant ou arrière. |
| 37 | Sflow_f/b_Bytes | Le nombre moyen de paquets dans un flux secondaire pour la direction avant ou arrière. |

TABLE 5.2 – *Liste des caractéristiques réseau (features)*

données, qui contient un NaN ou aucune valeur. De plus, pour traiter les énormes différences entre l'amplitude, les unités et la plage dans l'ensemble de données généré, nous avons utilisé la fonction de mise à l'échelle. L'échelle des caractéristiques vise à placer toutes les valeurs de l'ensemble de données entre 0 et 1, afin de rendre les caractéristiques plus cohérentes entre elles et de rendre l'étape d'apprentissage moins sensible à ce problème. Pour appliquer cette technique à notre ensemble de données, nous avons utilisé la fonction StandardScaler de la bibliothèque sklearn du python.

Nous avons utilisé dans notre recherche deux types d'algorithmes de sélection de caractéristiques. Le premier est le Forward selection [119], qui appartient à la classe des wrappers. Le Forward selection est un algorithme itératif qui commence par un ensemble vide de caractéristiques. Dans chaque itération, il ajoute la meilleure caractéristique qui améliore le modèle jusqu'à ce que l'ajout d'une nouvelle caractéristique n'améliore pas les performances du modèle. L'algorithme de sélection directe de caractéristiques a réduit le nombre de caractéristiques de 79 à 10, comme le montre le tableau 5.3.

Le deuxième algorithme de sélection des caractéristiques que nous avons utilisé dans notre étude, est le Linear Support Vector Classifier (LinearSVC) [120], qui fait partie des algorithmes de sélection des caractéristiques intégrées. LinearSVC est un algorithme qui donne à chaque caractéristique un attribut coef_ ou feature_importances_. Toutes les caractéristiques sont considérées comme non importantes au départ et il leur donne des valeurs sous un paramètre de seuil. Ensuite, il utilise des heuristiques intégrées pour trouver le seuil de chaque caractéristique à l'aide d'un argument de type chaîne de caractères. LinearSVC a réduit le nombre de caractéristiques de 79 à 25 comme indiqué dans le tableau 5.3.

Les meilleurs résultats ont été obtenus en utilisant le sous-ensemble de caractéristiques donné par le LinearSVC (comme indiqué dans la section des résultats).

| L'algorithme Forward selection | L'algorithme LinearSVC | Les caractéristiques en commun |
|---|---|---|
| Protocol ; Flow Duration<br>Flow IAT Mean ; IAT Max Flow<br>Flow IAT Min ; Len Bwd Header<br>Len Fwd Header ; Syn Flag Cnt<br>Down / Up Ratio ; Init Bwd Win Byts | Protocol ; Tot Fwd Pkts<br>Tot Bdw Pkts ; TotLen Fwd Pkts<br>TotLen Bwd Pkts ; Fwd IAT Std<br>Bwd IAT tot ; Bwd IAT Max<br>Bwd IAT Min ; Fwd Header Len<br>Bwd Header Len ; Fwd Pkts/s<br>Bwd Pkts/s ; Pkt Len Max<br>Pkt Len Var ; Syn Flag Cnt<br>Ack Flag Cnt ; Down/Up Ratio<br>Pkt Size Avg ; Subflow Fwd Pkts<br>Subflow Fwd Byts ; Subflow Bwd Pkts<br>Subflow Bwd Byts<br>Init Fwd Win Byts ; Init Bwd win Byts | Protocol ; Bwd Header Len<br>Fwd Header Len ; Syn Flag Cnt<br>Down/Up Ratio ; Init Bwd Win Byts |

TABLE 5.3 – *Les caractéristiques sélectionnés*

## 5.5 Les algorithmes de classification

Dans cette section, nous présentons brièvement les algorithmes d'apprentissage automatiques utilisés dans notre expérimentation. Nous avons utilisé les algorithmes suivants pour leurs performances connues en matière d'efficacité et de classification : Naive Bayes, Support Vector Machine, K-Nearest Neighbour (KNN), Random Forest and Decision trees.

### 5.5.1 L'algorithme Naive Bayes

Il s'agit d'un classificateur probabiliste qui effectue des classifications en utilisant la règle de décision maximale a posteriori dans un cadre bayésien. Il fonctionne selon une forte hypothèse d'indépendance, ce qui signifie que la probabilité d'un attribut n'affecte pas la probabilité de l'autre [110].

$$P(c \mid x) = \frac{P(x \mid c) * P(c)}{P(x)} \tag{5.1}$$

$$P(c \mid x) = P(x1 \mid c) * P(x2 \mid c) * ... * P(xn \mid c) * P(c) \tag{5.2}$$

$P(c|x)$ est la probabilité a posteriori de la classe (cible) par rapport à un prédicteur (attribut). $P(c)$ est la probabilité de classe a priori. $P(x|c)$ (en anglais, likelihood) est la probabilité d'un prédicteur donné de classe. $P(x)$ est la probabilité antérieure du prédicteur.

### 5.5.2 Support Vector Machine(SVM)

Chaque élément de données est représenté par un point dans un espace à n dimensions (où n est le nombre d'éléments), la valeur de chaque élément étant la valeur d'une coordonnée particulière. Ensuite, la technique de classification est effectuée pour différencier les classes et définir à laquelle les points de données appartiennent [111].

### 5.5.3 K-Nearest Neighbour

L'algorithme des plus proches voisins (k-NN) est une méthode non-paramétrique utilisée pour la classification et la régression [112]. Il fonctionne sur la base d'une distance minimale entre l'instance d'interrogation et les échantillons d'entraînement pour déterminer les k voisins les plus proches. Après avoir rassemblé les K voisins les plus proches, il faut la majorité simple de ces K voisins les plus proches pour être la prédiction de l'instance de requête. Formellement :

$$score(D, Ci) = \sum Dj \in KNN(d) Sim(D, Dj) \wp(Dj, Ci) \tag{5.3}$$

Au-dessus, *KNN(d)* indique l'ensemble des K voisins les plus proches de l'instance de requête

$$D * \wp(Dj, Ci) \tag{5.4}$$

En ce qui concerne la classe *Ci*, c'est-à-dire :

$$\wp(Dj, Ci) = \begin{cases} 1, Dj \in Ci \\ 0, Dj \notin Ci \end{cases} \tag{5.5}$$

Pour le document de test *d*, il faut lui attribuer la classe qui a la somme pondérée résultante la plus élevée.

### 5.5.4 Decision Trees

Pour construire un arbre de décision, nous avons utilisé dans cet article l'algorithme ID3 (Iterative Dichotomiser 3) qui utilise l'entropie et le gain d'information comme mesures. L'entropie caractérise l'impureté d'une collection arbitraire d'exemples, elle peut être définie formellement comme suit :

$$H(s) = \sum_{c \in C} -P(c) * log_2 P(c) \tag{5.6}$$

où $S$ est l'ensemble de données actuel pour lequel l'entropie est calculée, $C$ est l'ensemble des classes dans $S$, et $P(c)$ représente la proportion d'éléments de la classe c par rapport au nombre d'éléments de l'ensemble S.

### 5.5.5 Random Forest

Comme son nom l'indique, il est constitué d'un grand nombre d'arbres de décision individuels qui fonctionnent comme un ensemble. Chaque arbre individuel de la forêt aléatoire crache une prédiction de classe et la classe ayant le plus de votes devient la prédiction de notre modèle.

Dans cet article, nous avons utilisé les algorithmes mentionnés ci-dessus avec les paramètres par défaut, sauf avec le SVM. Pour ce dernier, nous avons utilisé un noyau non linéaire appelé le Radial Basis Function (rbf), car il est bien appliquée et relativement facile à calibrer, contrairement aux autres noyaux.

## 5.6 Résultats et discussion

Dans cette section, nous présentons et discutons les résultats de la détection des attaques par déni de service (DoS) à travers deux expériences. Dans la première expérience, nous effectuons une classification binaire (attaque/normal). La seconde expérience vise à identifier le type d'attaque par déni de service (SYN-flood, Slowloris, et UDP-flood). Les tableaux 5.4 et 5.5 présentent la distribution des échantillons correspondant respectivement à la classification binaire et multiclasse. Pour ces deux expériences, nous avons

testé plusieurs configurations avec différents ensembles de caractéristiques de réseau : 1) l'ensemble de caractéristiques initial ; 2) l'ensemble de caractéristiques de l'algorithme Forward selection ; 3) l'ensemble de caractéristiques de l'algorithme linearSVC ; et 4) l'ensemble de caractéristiques commun entre le Forward selection et le linearSVC.

Nous avons utilisé 60 % de l'ensemble de données pour l'apprentissage et 40 % pour les tests. Pour évaluer les performances de l'approche de détection de la DoS proposée, nous avons utilisé les mesures de performance suivantes :

— Accuracy : (en francais, justesse) elle désigne la proportion des prédictions correctes effectuées par le modèle. Formellement, elle est définie comme suivant :

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \tag{5.7}$$

TP : true positive (en français, Vrai positif) ; TN : True negative (Vrai négatif) ; FP : false positive (Faux positif) ; FN : false negative (Faux négatif).

— Precision : (en français, Précision) représente la proportion des points pertinents parmi l'ensemble des points proposés :

$$Precision = TP/(TP + FP) \tag{5.8}$$

— Recall : (en français, Rappel) est la fraction des cas pertinents qui ont été retrouvés par rapport au montant total des cas pertinents :

$$Recall = TP/(TP + FN) \tag{5.9}$$

— F1_score : est défini comme la moyenne harmonique entre la précision et le rappel. Elle est utilisée comme mesure statistique pour évaluer la performance du modele :

$$F1\_score = (2 * (precision * recall))/(precision + recall) \tag{5.10}$$

— False positive rate (FPR) : le taux de faux positifs, est la proportion des négatifs qui donnent des résultats positifs :

$$FPR = FP/(TN + FP) \tag{5.11}$$

— False negative rate (FNR) : le taux de faux négatifs, est la proportion des positifs qui donnent des résultats négatifs :

$$FNR = FN/(TP + FN) \tag{5.12}$$

| Échantillons | Autoroute | | Rural | | Urbain | |
|---|---|---|---|---|---|---|
| | Nb | % | Nb | % | Nb | % |
| Bénin | 26480 | 6,51% | 9807 | 3,66% | 9679 | 6,30% |
| Malveillant | 380002 | 93,49% | 257943 | 96,34% | 143895 | 93,70% |
| Totale | 406482 | 100% | 267750 | 100% | 153574 | 100% |

TABLE 5.4 – *Distribution d'échantillons de la base de données pour la classification binaire*

| Échantillons | Autoroute | | Rural | | Urbain | |
|---|---|---|---|---|---|---|
| | Nb | % | Nb | % | Nb | % |
| Bénin | 26480 | 6,51% | 9807 | 3,66% | 9679 | 6,30% |
| SYN-Flood | 157802 | 38,82% | 114943 | 42,92% | 81188 | 52,86% |
| Slowloris | 1057 | 0,26% | 2187 | 0,81% | 274 | 0,17% |
| UDP-Flood | 221143 | 54,40% | 140813 | 52,59% | 62433 | 40,65% |
| Totale | 406482 | 100% | 267750 | 100% | 153574 | 100% |

TABLE 5.5 – *Distribution d'échantillons de la base de données pour la classification multiclasse*

### 5.6.1 Expérimentation 1 : classification binaire

Dans cette expérimentation, nous évaluons la performance du classificateur pour classer une connexion réseau comme légitime ou comme une attaque DoS. Les performances de classification correspondant aux trois types d'environnement (routier, rural et urbain) sont présentées respectivement dans les tableaux 5.6, 5.7, 5.8. Pour chaque environnement, nous présentons les performances de classification correspondant à l'ensemble des caractéristiques initiales, et au meilleur ensemble de caractéristiques sélectionnées.

D'une manière générale, les résultats expérimentaux montrent que les algorithmes basés sur les arbres donnent les meilleures performances de classification dans les trois types d'environnement. L'arbre de décision et le random

| Caractéristiques | Algorithme | Classe | Précision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble initial (79) | SVM | Normal | 0.99996 | 0.95825 | 0.97866 | 0.0208 | 0.0208 | 0.99728 |
| | | Malveillant | 0.99710 | 0.99999 | 0.99854 | | | |
| | | Valeur Moy. | 0.99853 | 0.97912 | 0.98860 | | | |
| | Random Forest | Normal | 0.99984 | 0.99977 | 0.99981 | 0.0001 | 0.0001 | 0.99997 |
| | | Malveillant | 0.99998 | 0.99998 | 0.99998 | | | |
| | | Valeur Moy. | 0.99991 | 0.99988 | 0.99989 | | | |
| | Naïve Bayes | Normal | 0.90766 | 0.06540 | 0.12202 | 0.4675 | 0.4675 | 0.93875 |
| | | Malveillant | 0.93890 | 0.99953 | 0.96827 | | | |
| | | Valeur Moy. | 0.92328 | 0.53247 | 0.54514 | | | |
| | KNN | Normal | 0.99791 | 0.99353 | 0.99571 | 0.0033 | 0.0033 | 0.99944 |
| | | Malveillant | 0.99955 | 0.99985 | 0.99970 | | | |
| | | Valeur Moy. | 0.99873 | 0.99669 | 0.99771 | | | |
| | Decision Tree | Normal | 0.99988 | 0.99992 | 0.99990 | 0.00004 | 0.00004 | 0.99998 |
| | | Malveillant | 0.99999 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99994 | 0.99995 | 0.99994 | | | |
| Meilleur ensemble (25) | SVM | Normal | 1.00 | 0.95747 | 0.97827 | 0.02126 | 0.02126 | 0.99722 |
| | | Malveillant | 0.99704 | 1.00 | 0.99852 | | | |
| | | Valeur Moy. | 0.99852 | 0.97873 | 0.98839 | | | |
| | Random Forest | Normal | 0.99988 | 0.99981 | 0.99984 | 0.00009 | 0.00009 | 0.99998 |
| | | Malveillant | 0.99998 | 0.99999 | 0.99998 | | | |
| | | Valeur Moy. | 0.99993 | 0.99990 | 0.99991 | | | |
| | Naïve Bayes | Normal | 0.99491 | 0.94520 | 0.96942 | 0.02756 | 0.02756 | 0.99611 |
| | | Malveillant | 0.99619 | 0.99966 | 0.99792 | | | |
| | | Valeur Moy. | 0.99555 | 0.97243 | 0.98367 | | | |
| | KNN | Normal | 0.99856 | 0.99524 | 0.99689 | 0.00242 | 0.00242 | 0.99959 |
| | | Malveillant | 0.99966 | 0.9999 | 0.99978 | | | |
| | | Valeur Moy. | 0.99911 | 0.99757 | 0.99834 | | | |
| | Decision Tree | Normal | 0.99996 | 0.99977 | 0.99986 | 0.00011 | 0.00011 | 0.99998 |
| | | Malveillant | 0.99998 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99997 | 0.99988 | 0.99992 | | | |

TABLE 5.6 – *Résultats de la classification binaire dans un environnement routier*

forest sont connus pour leur grande précision et leur stabilité. En revanche, l'algorithme de Bayes naïf est moins performant que les autres algorithmes considérés, donnant les taux de faux positifs les plus élevés. Nous pouvons voir que la sélection des caractéristiques a légèrement affecté les performances de l'arbre de décision et du SVM, en augmentant leurs taux de faux positifs et négatifs dans le cas de l'environnement routier. Ce qui n'est pas le cas dans

| Caractéristiques | Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble initial (79) | SVM | Normal | 0.99407 | 0.96002 | 0.97674 | 0.02009 | 0.02009 | 0.99833 |
| | | Malveillant | 0.99848 | 0.99978 | 0.99913 | | | |
| | | Valeur Moy. | 0.99627 | 0.97990 | 0.98794 | | | |
| | Random Forest | Normal | 0.99906 | 0.98711 | 0.99305 | 0.00645 | 0.00645 | 0.99949 |
| | | Malveillant | 0.99951 | 0.99996 | 0.99973 | | | |
| | | Valeur Moy. | 0.99929 | 0.99354 | 0.99639 | | | |
| | Naïve Bayes | Normal | 0.96151 | 0.16349 | 0.27947 | 0.41837 | 0.41837 | 0.96920 |
| | | Malveillant | 0.96925 | 0.99975 | 0.98426 | | | |
| | | Valeur Moy. | 0.96538 | 0.58162 | 0.63186 | | | |
| | KNN | Normal | 0.99705 | 0.96952 | 0.98310 | 0.01528 | 0.01528 | 0.99878 |
| | | Malveillant | 0.99884 | 0.99989 | 0.99936 | | | |
| | | Valeur Moy. | 0.99795 | 0.98471 | 0.99123 | | | |
| | Decision Tree | Normal | 0.99989 | 0.99969 | 0.99979 | 0.00015 | 0.00015 | 0.99998 |
| | | Malveillant | 0.99998 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99994 | 0.99984 | 0.99989 | | | |
| Meilleur ensemble (25) | SVM | Normal | 0.99618 | 0.95758 | 0.97649 | 0.02127 | 0.02127 | 0.99831 |
| | | Malveillant | 0.99838 | 0.99986 | 0.99912 | | | |
| | | Valeur Moy. | 0.99728 | 0.97872 | 0.98781 | | | |
| | Random Forest | Normal | 1.00 | 0.99989 | 0.99994 | 0.00005 | 0.00005 | 0.99999 |
| | | Malveillant | 0.99999 | 1.00 | 0.99999 | | | |
| | | Valeur Moy. | 0.99999 | 0.99994 | 0.99997 | | | |
| | Naïve Bayes | Normal | 0.77878 | 0.16692 | 0.27491 | 0.41744 | 0.41744 | 0.96774 |
| | | Malveillant | 0.96924 | 0.99819 | 0.98350 | | | |
| | | Valeur Moy. | 0.87401 | 0.58255 | 0.62921 | | | |
| | KNN | Normal | 0.99801 | 0.97277 | 0.98523 | 0.01364 | 0.01364 | 0.99893 |
| | | Malveillant | 0.99896 | 0.99992 | 0.99944 | | | |
| | | Valeur Moy. | 0.99848 | 0.98635 | 0.99233 | | | |
| | Decision Tree | Normal | 0.99979 | 0.99979 | 0.99979 | 0.00010 | 0.00010 | 0.99998 |
| | | Malveillant | 0.99999 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99989 | 0.99989 | 0.99989 | | | |

TABLE 5.7 – *Résultats de la classification binaire dans un milieu rural*

l'environnement rural et urbain, et pas non plus le cas des autres algorithmes de classification.

Le classificateur d'arbre de décision donne le meilleure accuracy score avec les taux de faux positifs et négatifs les plus bas dans le cas de l'environnement routier. En revanche, l'algorithme Random Forest est plus performant que tous les autres classificateurs en milieu rural et urbain. L'algorithme Ran-

| Caractéristiques | Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble initial (79) | SVM | Normal | 0.99977 | 0.93274 | 0.96509 | 0.03363 | 0.03363 | 0.99574 |
| | | Malveillant | 0.99549 | 0.99998 | 0.99773 | | | |
| | | Valeur Moy. | 0.99763 | 0.96636 | 0.98141 | | | |
| | Random Forest | Normal | 0.99958 | 0.99979 | 0.99969 | 0.00011 | 0.00011 | 0.99996 |
| | | Malveillant | 0.99998 | 0.99997 | 0.99997 | | | |
| | | Valeur Moy. | 0.99978 | 0.99988 | 0.99983 | | | |
| | Naïve Bayes | Normal | 0.97132 | 0.34642 | 0.51069 | 0.32713 | 0.32713 | 0.95816 |
| | | Malveillant | 0.95786 | 0.99931 | 0.97814 | | | |
| | | Valeur Moy. | 0.96459 | 0.67286 | 0.74442 | | | |
| | KNN | Normal | 0.98916 | 0.98057 | 0.98485 | 0.01007 | 0.01007 | 0.99809 |
| | | Malveillant | 0.99869 | 0.99927 | 0.99898 | | | |
| | | Valeur Moy. | 0.99392 | 0.98992 | 0.99191 | | | |
| | Decision Tree | Normal | 1.00 | 0.99989 | 0.99994 | 0.00005 | 0.00005 | 0.99999 |
| | | Malveillant | 0.99999 | 1.00 | 0.99999 | | | |
| | | Valeur Moy. | 0.99999 | 0.99994 | 0.99997 | | | |
| Meilleur ensemble (25) | SVM | Normal | 1.00 | 0.93377 | 0.96575 | 0.03311 | 0.03311 | 0.99582 |
| | | Malveillant | 0.99556 | 1.00 | 0.99777 | | | |
| | | Valeur Moy. | 0.99778 | 0.96688 | 0.98176 | | | |
| | Random Forest | Normal | 0.99969 | 1.00 | 0.99984 | 0.00001 | 0.00001 | 0.99998 |
| | | Malveillant | 1.00 | 0.99997 | 0.99998 | | | |
| | | Valeur Moy. | 0.99984 | 0.99998 | 0.99991 | | | |
| | Naïve Bayes | Normal | 0.98636 | 0.41853 | 0.58769 | 0.29092 | 0.29092 | 0.96298 |
| | | Malveillant | 0.96234 | 0.99961 | 0.98062 | | | |
| | | Valeur Moy. | 0.97435 | 0.70907 | 0.78416 | | | |
| | KNN | Normal | 0.98920 | 0.98460 | 0.98690 | 0.00805 | 0.00805 | 0.99835 |
| | | Malveillant | 0.99896 | 0.99927 | 0.99912 | | | |
| | | Valeur Moy. | 0.99408 | 0.99194 | 0.99301 | | | |
| | Decision Tree | Normal | 0.99989 | 0.99989 | 0.99989 | 0.00005 | 0.00005 | 0.99998 |
| | | Malveillant | 0.99999 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99994 | 0.99994 | 0.99994 | | | |

TABLE 5.8 – *Résultats de la classification binaire dans un milieu urbain*

dom Forest avec le meilleur ensemble de caractéristiques est plus performant que l'ensemble de caractéristiques initiales. Cela est principalement dû au fait que dans certains cas, certaines caractéristiques peuvent être bruyantes, redondantes et non pertinentes, ce qui peut induire l'algorithme en erreur.

Les résultats obtenus montrent que la précision de détection du trafic malveillant est légèrement meilleure que celle du trafic normal. Cela peut s'expli-

quer par le volume énorme de requêtes superflues envoyées par l'attaquant, la taille et le temps entre les paquets sont complètement différents d'une connexion normale. La précision de la classification a légèrement baissé dans l'environnement routier, cela peut être dû à la vitesse élevée des véhicules.

De ces expérimentations, nous remarquons que l'environnement a un léger impact sur la faisabilité de l'attaque par déni de service dans le contexte des véhicules et sur ses performances de détection.

### 5.6.2 Expérimentation 2 : classification multiclasse

Comme dans la classification binaire, les algorithmes basés sur les arbres montrent les meilleures performances de classification, avec des scores d'accuracy similaires. Cependant, l'arbre de décision surpasse le Random Forest en termes de taux de faux positifs et négatifs dans les environnements autoroutier et urbain. Le Random Forest a obtenu ses meilleures performances avec l'ensemble des caractéristiques sélectionnées par LinearSVC. Le Bayes naïf donne les pires résultats avec le score d'accuracy le plus bas et les taux de faux positifs et négatifs les plus élevés dans les trois environnements et avec les deux ensembles de caractéristiques. Comme on peut le constater, l'environnement a un léger impact sur la faisabilité de l'attaque par déni de service et sur ses performances de détection.

En ce qui concerne la précision de détection de chaque type d'attaque, nous pouvons remarquer que les attaques SYN-Flood et UDP-Flood sont relativement plus faciles à détecter que l'attaque Slowloris par tous les classificateurs. Cela peut s'expliquer par le fait que le modus operandi de Slowloris est très différent des autres attaques DoS. Slowloris nécessite une bande passante minimale et n'inonde pas le serveur. Périodiquement, l'attaquant envoie des en-têtes ultérieurs pour chaque requête, mais ne la complète jamais. Contrairement aux attaques SYN-Flood et UDP-Flood, qui nécessitent une bande passante maximale. Ces différences rendent le DoS Slowloris plus difficile à détecter par rapport aux attaques SYN-Flood et UDP-Flood.

| Caractéristiques | Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble Initial (79) | SVM | Normal | 0.99996 | 0.95793 | 0.97849 | 0.00210 | 0.24419 | 0.99482 |
| | | SYN Flood | 0.98709 | 1.00 | 0.99350 | | | |
| | | Slowloris | 0.73404 | 0.06527 | 0.11989 | | | |
| | | UDP Flood | 0.99993 | 0.99999 | 0.99996 | | | |
| | | Valeur Moy. | 0.93025 | 0.75580 | 0.77296 | | | |
| | Random Forest | Normal | 0.99932 | 0.99932 | 0.99932 | 0.00003 | 0.00608 | 0.99989 |
| | | SYN Flood | 0.99991 | 0.99999 | 0.99995 | | | |
| | | Slowloris | 0.98755 | 0.97634 | 0.98192 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99669 | 0.99391 | 0.99529 | | | |
| | Naïve Bayes | Normal | 0.97651 | 0.06593 | 0.12353 | 0.05263 | 0.32196 | 0.80324 |
| | | SYN Flood | 0.92336 | 0.65006 | 0.76298 | | | |
| | | Slowloris | 0.01453 | 0.99621 | 0.02864 | | | |
| | | UDP Flood | 0.99996 | 0.99991 | 0.99994 | | | |
| | | Valeur Moy. | 0.72859 | 0.67803 | 0.47877 | | | |
| | KNN | Normal | 0.99844 | 0.99444 | 0.99644 | 0.00019 | 0.01345 | 0.99950 |
| | | SYN Flood | 0.99924 | 1.00 | 0.99961 | | | |
| | | Slowloris | 0.98724 | 0.95175 | 0.96917 | | | |
| | | UDP Flood | 0.99988 | 0.99999 | 0.99993 | | | |
| | | Valeur Moy. | 0.99620 | 0.98654 | 0.99129 | | | |
| | Decision Tree | Normal | 0.99988 | 0.99992 | 0.99990 | 0.0000003 | 0.00025 | 0.99998 |
| | | SYN Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Slowloris | 0.99905 | 0.99905 | 0.99905 | | | |
| | | UDP Flood | 0.99999 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99973 | 0.99974 | 0.99973 | | | |
| Meilleur ensemble (25) | SVM | Normal | 1.00 | 0.95751 | 0.97829 | 0.00212 | 0.24430 | 0.99483 |
| | | SYN Flood | 0.98704 | 1.00 | 0.99348 | | | |
| | | Slowloris | 0.73404 | 0.06527 | 0.11989 | | | |
| | | UDP Flood | 0.99992 | 1.00 | 0.99996 | | | |
| | | Valeur Moy. | 0.93025 | 0.75569 | 0.77290 | | | |
| | Random Forest | Normal | 0.99988 | 0.99981 | 0.99984 | 0.000008 | 0.00075 | 0.99998 |
| | | SYN Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Slowloris | 1.00 | 0.99716 | 0.99857 | | | |
| | | UDP Flood | 0.99997 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.99996 | 0.99924 | 0.99960 | | | |
| | Naïve Bayes | Normal | 0.99736 | 0.07156 | 0.13354 | 0.05230 | 0.32034 | 0.80440 |
| | | SYN Flood | 0.92475 | 0.65177 | 0.76463 | | | |
| | | Slowloris | 0.01456 | 0.99526 | 0.02871 | | | |
| | | UDP Flood | 0.99996 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.73416 | 0.67965 | 0.48171 | | | |
| | KNN | Normal | 0.99852 | 0.99539 | 0.99695 | 0.00015 | 0.01321 | 0.99956 |
| | | SYN Flood | 0.99927 | 1.00 | 0.99963 | | | |
| | | Slowloris | 0.98724 | 0.95175 | 0.96917 | | | |
| | | UDP Flood | 0.99996 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.99625 | 0.98678 | 0.99143 | | | |
| | Decision Tree | Normal | 0.99988 | 0.99966 | 0.99977 | 0.00001 | 0.00032 | 0.99997 |
| | | SYN Flood | 0.99997 | 1.00 | 0.99998 | | | |
| | | Slowloris | 1.00 | 0.99905 | 0.99952 | | | |
| | | UDP Flood | 0.99997 | 0.99999 | 0.99998 | | | |
| | | Valeur Moy. | 0.99995 | 0.99967 | 0.99981 | | | |

TABLE 5.9 – *Résultats de la classification multiclasse dans un environnement routier*

| Caractéristiques | Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble initial (79) | SVM | Normal | 0.99534 | 0.96023 | 0.97747 | 0.00346 | 0.20851 | 0.99202 |
| | | SYN Flood | 0.98219 | 0.99993 | 0.99098 | | | |
| | | Slowloris | 0.99337 | 0.20576 | 0.34090 | | | |
| | | UDP Flood | 0.99997 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.99272 | 0.79148 | 0.82733 | | | |
| | Random Forest | Normal | 0.99969 | 0.99959 | 0.99964 | 0.000008 | 0.00033 | 0.99997 |
| | | SYN Flood | 0.99998 | 0.99999 | 0.99998 | | | |
| | | Slowloris | 0.99908 | 0.99908 | 0.99908 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99969 | 0.99966 | 0.99967 | | | |
| | Naïve Bayes | Normal | 0.98500 | 0.26797 | 0.42132 | 0.04413 | 0.26530 | 0.83281 |
| | | SYN Flood | 0.96416 | 0.67328 | 0.79288 | | | |
| | | Slowloris | 0.04955 | 0.99771 | 0.09442 | | | |
| | | UDP Flood | 1.00 | 0.99981 | 0.99990 | | | |
| | | Valeur Moy. | 0.74968 | 0.73469 | 0.57713 | | | |
| | KNN | Normal | 0.99727 | 0.97144 | 0.98419 | 0.00049 | 0.01608 | 0.99860 |
| | | SYN Flood | 0.99841 | 0.99988 | 0.99914 | | | |
| | | Slowloris | 0.93401 | 0.96433 | 0.94893 | | | |
| | | UDP Flood | 0.99988 | 0.99997 | 0.99993 | | | |
| | | Valeur Moy. | 0.98239 | 0.98391 | 0.98305 | | | |
| | Decision Tree | Normal | 0.99959 | 0.99989 | 0.99974 | 0.000004 | 0.00037 | 0.99998 |
| | | SYN Flood | 1.00 | 0.99999 | 0.99999 | | | |
| | | Slowloris | 0.99954 | 0.99862 | 0.99908 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99978 | 0.99962 | 0.99970 | | | |
| Best set (25) | SVM | Normal | 0.99618 | 0.95758 | 0.97649 | 0.00349 | 0.20916 | 0.99195 |
| | | SYN Flood | 0.98196 | 1.00 | 0.99090 | | | |
| | | Slowloris | 0.99337 | 0.20576 | 0.34090 | | | |
| | | UDP Flood | 0.99997 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.99287 | 0.79083 | 0.82707 | | | |
| | Random Forest | Normal | 0.99959 | 0.99989 | 0.99974 | 0.000005 | 0.00025 | 0.99998 |
| | | SYN Flood | 1.00 | 0.99998 | 0.99999 | | | |
| | | Slowloris | 1.00 | 0.99908 | 0.99954 | | | |
| | | UDP Flood | 0.99999 | 1.00 | 0.99999 | | | |
| | | Valeur Moy. | 0.99989 | 0.99974 | 0.99981 | | | |
| | Naïve Bayes | Normal | 0.99687 | 0.19486 | 0.32599 | 0.04164 | 0.28079 | 0.83629 |
| | | SYN Flood | 0.99310 | 0.68745 | 0.81248 | | | |
| | | Slowloris | 0.04785 | 0.99451 | 0.09131 | | | |
| | | UDP Flood | 0.99999 | 1.00 | 0.99999 | | | |
| | | Valeur Moy. | 0.75945 | 0.71920 | 0.55744 | | | |
| | KNN | Normal | 0.99790 | 0.97297 | 0.98528 | 0.00045 | 0.01557 | 0.99868 |
| | | SYN Flood | 0.99846 | 0.99992 | 0.99919 | | | |
| | | Slowloris | 0.93404 | 0.96479 | 0.94916 | | | |
| | | UDP Flood | 0.99996 | 1.00 | 0.99998 | | | |
| | | Valeur Moy. | 0.98259 | 0.98442 | 0.98340 | | | |
| | Decision Tree | Normal | 0.99959 | 0.99949 | 0.99954 | 0.00001 | 0.00036 | 0.99996 |
| | | SYN Flood | 0.99998 | 0.99999 | 0.99998 | | | |
| | | Slowloris | 0.99908 | 0.99908 | 0.99908 | | | |
| | | UDP Flood | 0.99999 | 0.99999 | 0.99999 | | | |
| | | Valeur Moy. | 0.99966 | 0.99963 | 0.99965 | | | |

TABLE 5.10 – *Résultats de la classification multiclasse dans le milieu rural*

| Caractéristiques | Algorithme | Classe | Precision | Recall | F1-score | FPR | FNR | Accuracy |
|---|---|---|---|---|---|---|---|---|
| Ensemble initial (79) | SVM | Normal | 0.99988 | 0.93294 | 0.96525 | 0.00299 | 0.21567 | 0.99434 |
| | | SYN Flood | 0.98950 | 0.99998 | 0.99471 | | | |
| | | Slowloris | 1.00 | 0.20437 | 0.33939 | | | |
| | | UDP Flood | 0.99990 | 1.00 | 0.99995 | | | |
| | | Valeur Moy. | 0.99732 | 0.78432 | 0.82483 | | | |
| | Random Forest | Normal | 0.99865 | 0.99989 | 0.99927 | 0.00002 | 0.01188 | 0.99990 |
| | | SYN Flood | 0.99998 | 1.00 | 0.99999 | | | |
| | | Slowloris | 1.00 | 0.95255 | 0.97570 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99966 | 0.98811 | 0.99374 | | | |
| | Naïve Bayes | Normal | 0.99076 | 0.38805 | 0.55768 | 0.06680 | 0.25981 | 0.73746 |
| | | SYN Flood | 0.98788 | 0.57641 | 0.72803 | | | |
| | | Slowloris | 0.00682 | 0.99635 | 0.01356 | | | |
| | | UDP Flood | 0.99996 | 0.99993 | 0.99995 | | | |
| | | Valeur Moy. | 0.74636 | 0.74018 | 0.57480 | | | |
| | KNN | Normal | 0.98907 | 0.98233 | 0.98569 | 0.00077 | 0.10387 | 0.99816 |
| | | TCP Food | 0.99809 | 0.99998 | 0.99904 | | | |
| | | Slowloris | 0.97633 | 0.60218 | 0.74492 | | | |
| | | UDP Flood | 0.99971 | 0.99998 | 0.99984 | | | |
| | | Valeur Moy. | 0.99080 | 0.89612 | 0.93237 | | | |
| | Decision Tree | Normal | 0.99989 | 0.99969 | 0.99979 | 0.000008 | 0.00008 | 0.99997 |
| | | SYN Flood | 0.99998 | 0.99998 | 0.99998 | | | |
| | | Slowloris | 0.99275 | 1.00 | 0.99636 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99815 | 0.99991 | 0.99903 | | | |
| Meilleur ensemble (25) | SVM | Normal | 1.00 | 0.93367 | 0.96569 | 0.00296 | 0.21548 | 0.99440 |
| | | SYN Flood | 0.98963 | 1.00 | 0.99479 | | | |
| | | Slowloris | 1.00 | 0.20437 | 0.33939 | | | |
| | | UDP Flood | 0.99983 | 1.00 | 0.99991 | | | |
| | | Valeur Moy. | 0.99736 | 0.78451 | 0.82495 | | | |
| | Random Forest | Normal | 0.99969 | 1.00 | 0.99984 | 0.000005 | 0.00273 | 0.99998 |
| | | SYN Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Slowloris | 1.00 | 0.98905 | 0.99449 | | | |
| | | UDP Flood | 1.00 | 1.00 | 1.00 | | | |
| | | Valeur Moy. | 0.99992 | 0.99726 | 0.99858 | | | |
| | Naïve Bayes | Normal | 1.00 | 0.43010 | 0.60150 | 0.06530 | 0.25009 | 0.74034 |
| | | SYN Flood | 0.99676 | 0.57680 | 0.73074 | | | |
| | | Slowloris | 0.00680 | 0.99270 | 0.01350 | | | |
| | | UDP Flood | 0.99998 | 1.00 | 0.99999 | | | |
| | | Valeur Moy. | 0.75088 | 0.74990 | 0.58643 | | | |
| | KNN | Normal | 0.98911 | 0.98605 | 0.98758 | 0.00065 | 0.10293 | 0.99841 |
| | | SYN Flood | 0.99838 | 1.00 | 0.99919 | | | |
| | | Slowloris | 0.97633 | 0.60218 | 0.74492 | | | |
| | | UDP Flood | 0.99993 | 1.00 | 0.99996 | | | |
| | | Valeur Moy. | 0.99094 | 0.89706 | 0.93291 | | | |
| | Decision Tree | Normal | 0.99958 | 0.99989 | 0.99974 | 0.00001 | 0.00004 | 0.99996 |
| | | SYN Flood | 0.99998 | 1.00 | 0.99999 | | | |
| | | Slowloris | 1.00 | 1.00 | 1.00 | | | |
| | | UDP Flood | 1.00 | 0.99993 | 0.99996 | | | |
| | | Valeur Moy. | 0.99989 | 0.99995 | 0.99992 | | | |

TABLE 5.11 – *Résultats de la classification multiclasse dans le milieu urbain*

## 5.7 Conclusion

Dans ce chapitre, nous avons proposé un nouvel ensemble de données, VDoS-LRS, qui comprend le trafic normal des réseaux ad-hoc véhiculaires, et divers types de trafic d'attaques par déni de service. Cet ensemble de données a été généré et étiqueté sur la base d'un banc d'essai réaliste, qui prend en considération trois types d'environnements (urbain, rural et autoroutier). Pour caractériser les flux réseau, nous avons extrait du trafic réseau brut un ensemble de quatre-vingt caractéristiques liées au comportement des sessions, aux octets/paquets échangés et aux intervalles de temps. Nous avons ensuite procédé à une sélection des caractéristiques pour obtenir le meilleur ensemble de caractéristiques, avant la phase de formation et de test. Ensuite, nous évaluons les performances des ensembles de caractéristiques initiales et sélectionnées à l'aide de cinq algorithmes d'apprentissage machine communs. Les résultats expérimentaux ont montré que le classificateur d'arbre de décision offre la plus grande précision avec les taux de faux positifs et négatifs les plus bas dans les trois environnements. Le système de détection basé sur ce classificateur serait capable de détecter les attaques par déni de service dans VANET ainsi que ses types avec une précision de 99 %. Nous avons observé que l'environnement affecte légèrement la faisabilité et les performances de détection des attaques par déni de service. Dans l'environnement autoroutier, le classificateur a besoin de toutes les caractéristiques pour donner les mêmes performances que dans les autres environnements. Dans les travaux futurs, nous examinerons davantage de scénarios de déni de service (tel que le WSMP-Flood) ainsi que d'autres scénarios bénignes (de sécurité et de gestion de trafic) afin d'utiliser VDoS-LRS pour faire l'apprentissage de notre framework AntibotV. Nous prévoyons également d'étendre notre banc d'essai à un plus grand nombre de véhicules.

# Conclusion générale

Au cours de ce mémoire, nous avons proposé AntibotV, un framework de détection des botnets véhiculaires, basé sur la détection comportementale sur plusieurs niveaux. Nous avons envisagé de nouvelles DoS attaques de type "zero day", ainsi qu'un large éventail d'attaques de déni de service et d'attaques à bord de véhicules. Le framework proposé permet de surveiller l'activité du véhicule au niveau du réseau et in-véhiculaire. Pour construire le système de détection, nous avons recueilli des données sur le trafic réseau des applications légitimes et malveillantes. Ensuite, nous avons formé à l'aide des algorithmes des arbres de décision un nouveau classificateur avec un ensemble de caractéristiques que nous avons extraites et sélectionnées. De même, nous avons formé notre framework avec des données embarquées. Les résultats expérimentaux ont montré que AntibotV est plus performant que les solutions existantes, il atteint un taux de détection supérieur à 97% et un taux de faux positifs inférieur à 0,14%. Cependant, étant savoir que l'efficacité de l'IDS dépend du modèle de classification, il est primordiale de faire l'apprentissage du classifieur sur un ensemble de données qui contient des traces de trafic réaliste. Dans notre deuxième contribution, nous avons proposé un nouvel ensemble de données, VDoS-LRS, qui comprend le trafic normal des réseaux ad-hoc véhiculaires, et divers types de trafic d'attaques de déni de service. Cet ensemble de données a été généré et étiqueté sur la base d'un banc d'essai réaliste, qui prend en considération trois types d'environnements (urbain, rural et autoroutier). Les résultats expérimentaux ont montré que le classificateur d'arbre de décision offre la plus grande précision avec les taux de faux positifs et négatifs les plus bas dans les trois environnements. Le système de détection basé sur ce classificateur serait capable de détecter les attaques de déni de service dans VANET ainsi que ses types avec une précision de 99 %. Nous avons observé que l'environnement affecte légè-

rement la faisabilité et les performances de détection des attaques par déni de service. Dans l'environnement autoroutier, le classificateur a besoin de toutes les caractéristiques pour donner les mêmes performances que dans les autres environnements.

## Perspectives

Dans la continuité directe de notre travail de thèse, nous pouvons travailler sur notre ensemble de donnée (VDoS-LRS), en examinons la possibilité d'implémenter d'autres types de scénarios des botnets véhiculaires (vol d'information, WSMP-Flood, et Geo-WSMP-Flood), ainsi que d'autres types de menaces que nous n'avons pas discuté, afin d'utiliser VDoS-LRS pour faire l'apprentissage de notre framework AntibotV. Nous prévoyons également d'étendre notre banc d'essai à un plus grand nombre de véhicules, afin de former la première dataset réelle et complète pour les réseaux des transports intelligent.

# Annexes

A

# Description et configuration réseau de VDoS-LRS

## Jour, Date, Environnement, Vitesse, Météo, Description, Taille (MB)

— Vendredi, 11/01/2019, Autoroute et urbain, 40-100 Km/h, Temps pluvieux-12°C, attaques et activité normale, 1237 MB.
— Samedi, 12/01/2019, Autoroute et montagne, 60-80 Km/h et 20-40 Km/h, grêle-11°C, attaques et activité normale, 1346 MB.
— Lundi, 14/01/2019, Montagne, 20-40 Km/h, temps venteux et brumeux-14°C, Activité normale, 137 MB.

## Informations sur les nœuds victimes et les attaquants

*Les scénarios d'attaques*

— Attaquant (Windows 8 et Ubuntu) : 192.168.0.4
— Non-root Bridge : 192.168.0.1
— Victime (Windows 7) : 192.168.0.3
— Root-bridge : 192.168.0.2

*Bénigne (activités humaines normales comme l'accès aux sites web, réseaux sociaux, et les applications en temps réel)*

— Premier jour (Windows 7) : 192.168.8.101
— Deuxième jour (Windows 8) : 192.168.8.102
— Troisième jour (Windows 7) : 192.168.8.100

*Bénigne (Activités humaines normales : partage de fichiers)*

— Nœud 1 (Windows 7) : 192.168.0.3
— Nœud 2 (Windows 8) : 192.168.0.4

## Vendredi 11 janvier 2019 (Urbain)

*Des attaques DoS*

— Ping flood (16 :07 – 16 :17 p.m.)
— Tcp flood (en utilisant 'Low Orbit Ion Cannon') (16 :18 – 16 :24 p.m.)
— UDP flood (en utilisant 'Low Orbit Ion Cannon') (16 :25 – 16 :29 p.m.)
— Http flood (en utilisant 'Low Orbit Ion Cannon') (16 :29 – 16 :33 p.m.)
— DoS flood (en utilisant 'High Orbit Ion Cannon') (16 :34 – 16 :41 p.m.)
— DDoS (en utilisant 'Low Orbit Ion Cannon' et 'High Orbit Ion Cannon' ensemble) (16 :42 – 16 :49 p.m.)

*Bénigne (Activités humaines normales)*

— Partage de fichier (16 :49 – 17 :10 p.m.)
— L'accès aux sites web, réseaux sociaux, et les applications en temps réel (17 :20 – 17 :56 p.m.)

## Samedi 12 janvier 2019 (Autoroute)

*Des attaques DoS*

— SynFlood (en utilisant l'outil 'Hping3') (15 :50 – 17 :57 p.m.)
— Low and Slow DoS (en utilisant 'Slowloris') (15 :58 – 16 :12 p.m.)
— DDoS (en utilisant Hping3 et Slowloris ensemble) (16 :13 – 16 :20 p.m.)

## Samedi 12 janvier 2019 (Montagne)

*Des attaques DoS*

— SynFlood (en utilisant 'Hping3') (16 :24 – 16 :39 p.m.)
— Low and Slow DoS (en utilisant Slowloris) (16 :40 – 16 :46 p.m.)
— DDoS (en utilisant 'Hping3' et 'Slowloris' ensemble) (16 :47 – 17 :00 p.m.)
— Ping flood (17 :01 – 17 :06 p.m.)
— Tcp flood (en utilisant 'Low Orbit Ion Cannon') (17 :06 – 17 :11 p.m.)

— UDP flood (en utilisant 'Low Orbit Ion Cannon') (17 :12 – 17 :17 p.m.)
— Http flood (en utilisant 'Low Orbit Ion Cannon') (17 :17 – 17 :21 p.m.)
— DoS flood (en utilisant 'high Orbit Ion Cannon') (17 :22 – 17 :44 p.m.)
— DDoS (en utilisant 'Low Orbit Ion Cannon' et 'High Orbit Ion Cannon' ensemble) (17 :45 – 17 :48 p.m.)

*Bénigne (Activités humaines normales)*

— Partage de fichier (17 :48 – 17 :59 p.m.)

## Lundi 14 janvier 2019 (Montagne)

*Bénigne (Activités humaines normales)*

— L'accès aux sites web, réseaux sociaux, et les applications en temps réel (18 :16 – 19 :10 p.m.)

# Mes contributions scienfifiques

1. Rahal, R., Korba, A. A., Ghoualmi-Zine, N., Challal, Y., & Ghamri-Doudane, M. Y. (2020). AntibotV : A Multilevel Behaviour-based Framework for Botnets Detection in Vehicular Networks. Manuscript submitted for publication.
2. Rahal, R., Korba, A. A., & Ghoualmi-Zine, N. (2020). Towards the Development of Realistic DoS Dataset for Intelligent Transportation Systems. Wireless Personal Communications, 115(2), 1415-1444.
3. Rahal, R., Kahya, N., & Ghoualmi-Zine, N. (2019). Resistance against DoS attacks in VANETs using the IDS Snort. Revue de l'Information Scientifique et Technique, 24(1), 20-30.
4. Rahal, R., Kahya, N., & Ghoualmi-Zine, N. (November 6-7, 2018). Resistance against DoS attacks in VANETs using the IDS Snort, presented at The Third International Symposium on Informatics and its Applications, M'sila, Algeria.
5. Rahal, R., Kahya, N., & Ghoualmi-Zine, N. (December 3-4, 2017). Impact of Symmetric Encryption Algorithms in VANET, presented at ISPA 2017 : Image and Signal Processing and their Applications, Mostaganem, Algeria.
6. Rahal, R., Kahya, N., & Ghoualmi-Zine, N. (December 3-4, 2017). The extra-costs of security mechanisms in VANET networks, presented at ISPA 2017 : Image and Signal Processing and their Applications, Mostaganem, Algeria.

# Bibliographie


[1] Mevlut Turker Garip, Peter Reiher, and Mario Gerla. Ghost : Concealing vehicular botnet communication in the vanet control channel. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1–6. IEEE, 2016.

[2] Pranav Kumar Singh, Rahul Raj Gupta, Sunit Kumar Nandi, and Sukumar Nandi. Machine learning based approach to detect wormhole attack in vanets. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 651–661. Springer, 2019.

[3] Jyoti Grover, Nitesh Kumar Prajapati, Vijay Laxmi, and Manoj Singh Gaur. Machine learning approach for multiple misbehavior detection in vanet. In *International Conference on Advances in Computing and Communications*, pages 644–653. Springer, 2011.

[4] Khattab M Ali Alheeti, Anna Gruebler, and Klaus D McDonald-Maier. On the detection of grey hole and rushing attacks in self-driving vehicular networks. In *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*, pages 231–236. IEEE, 2015.

[5] Viacheslav Belenko, Vasiliy Krundyshev, and Maxim Kalinin. Synthetic datasets generation for intrusion detection in vanet. In *Proceedings of the 11th International Conference on Security of Information and Networks*, page 9. ACM, 2018.

[6] Oms | 10 faits sur la sécurité routière dans le monde. https://www.who.int/features/factfiles/roadsafety/fr/. Accessed : 2021-01-09.

[7] Haruki Fujii. The cacs project : How far away are we from the dynamic route guidance system ? In *Transportation for the Future*, pages 145–159. Springer Berlin Heidelberg, 1989.

[8] D.A. Rosen, F.J. Mammano, and R. Favout. An electronic route-guidance system for highway vehicles. *IEEE Transactions on Vehicular Technology*, 19(1) :143–152, February 1970.

[9] Emmanuel Hache Économiste et prospectiviste and Cyprien Ternel Ingénieur économiste « transport et mobilité ». Comment nos transports sont devenus intelligents, Jun 2020.

[10] Hakim Badis and Abderrezak Rachedi. Modeling tools to evaluate the performance of wireless multi-hop networks. In *Modeling and Simulation of Computer Networks and Systems*, pages 653–682. Elsevier, 2015.

[11] Salim Bitam and Abdelhamid Mellouk. *Bio-inspired routing protocols for vehicular ad-hoc networks*. John Wiley & Sons, 2014.

[12] GRICH Sofiane. Contribution à la qualité de service dans les réseaux vanet. *Mémoire, Université d'Oran, département d'informatique*, 4, 2015.

[13] Benchabana Ayoub and Bensaci Ramla. Analyse des protocoles de routage dans les réseaux vanets. *Mémoire, Université Kasdi Merbah-Ouargla Algérie, département d'informatique*, 2014.

[14] Ahmed A Ahizoune. Un protocole de diffusion des messages dans les réseaux véhiculaires. 2011.

[15] Mohammed Ali Hezam Al Junaid, AA Syed, Mohd Nazri Mohd Warip, Ku Nurul Fazira Ku Azir, and Nurul Hidayah Romli. Classification of security attacks in vanet : A review of requirements and perspectives. In *MATEC Web of Conferences*, volume 150, page 06038. EDP Sciences, 2018.

[16] IEEE 802 Standard Working Group et al. Wireless lan medium access control (mac) and physical layer (phy) specifications : high-speed physical layer in the 5ghz band, 1999.

[17] IEEE Computer Society LAN/MAN Standards Committee et al. Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11 : Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11^*, 2007.

[18] Jonathan Petit. *Surcoût de l'authentification et du consensus dans la sécurité des réseaux sans fil véhiculaires*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2011.

[19] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7) :1162–1182, 2011.

[20] Shereen AM Ahmed, Sharifah HS Ariffin, and Norsheila Fisal. Overview of wireless access in vehicular environment (wave) protocols and standards. *environment*, 7 :8, 2013.

[21] 1609.4-2016 - ieee standard for wireless access in vehicular environments (wave) – multi-channel operation.

[22] Caixia Song. Performance analysis of the ieee 802.11 p multichannel mac protocol in vehicular ad hoc networks. *Sensors*, 17(12) :2890, 2017.

[23] 1609.2-2016 - ieee standard for wireless access in vehicular environments–security services for applications and management messages.

[24] John B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7) :1162–1182, July 2011.

[25] Ali Rostami, Hariharan Krishnan, and Marco Gruteser. V2v safety communication scalability based on the sae j2945/1 standard. In *Proc. Workshop ITS Amer.*, pages 1–10, 2018.

[26] Hariharan Krishnan, Fan Bai, and Gavin Holland. Commercial and public use applications. *Vehicular Networking*, pages 1–28, 2010.

[27] CAMP Vehicle Safety Communications Consortium et al. Vehicle safety communications project : Task 3 final report : identify intelligent vehicle safety applications enabled by dsrc. *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC*, 2005.

[28] Patrice Seuwou, Dilip Patel, and George Ubakanma. Vehicular ad hoc network applications and security : a study into the economic and the legal implications. *International Journal of Electronic Security and Digital Forensics*, 6(2) :115, 2014.

[29] José María De Fuentes, Ana Isabel González-Tablas, and Arturo Ribagorda. Overview of security issues in vehicular ad-hoc networks. In *Handbook of Research on Mobility and Computing*, pages 894–911. IGI Global.

[30] Télématique : un système de communication dédié à l'automobile. https://www.journaldunet.fr/web-tech/dictionnaire-de-l-iot/1440704-telematique-un-systeme-de-communication-dedie-a-l-automobile/. Accessed : 2021-01-09.

[31] Soumya Kanti Datta. Connected car as an IoT service. In *BMW Summer School 2016, July 18-23, 2016, Lake Tegernsee, Bavaria, Germany*, Lake Tegernsee, GERMANY, 07 2016.

[32] Abs, esp, asr : des dispositifs utiles ? https://www.lesfurets.com/assurance-auto/guide/abs-esp-asr-dispositifs-vraiment-utiles. Accessed : 2021-01-09.

[33] David Crolla and Behrooz Mashadi. *Vehicle powertrain systems*. John Wiley & Sons, 2011.

[34] Body control module (bcm) in automotive : Intellias blog. https://www.intellias.com/body-control-module-bcm-in-automotive/#text=A%20comprehensive%20body%20control%20module,activation%20of%20auto%20electronics%20units/. Accessed : 2021-01-09.

[35] Jiajia Liu, Shubin Zhang, Wen Sun, and Yongpeng Shi. In-vehicle network attacks and countermeasures : Challenges and future directions. *IEEE Network*, 31(5) :50–58, 2017.

[36] Controller area network. http://www.esd-electronics-usa.com/Controller-Area-Network-CAN-Introduction.html. Accessed : 2021-01-09.

[37] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. VANet security challenges and solutions : A survey. *Vehicular Communications*, 7 :7–20, January 2017.

[38] Kahina Moghraoui. *Gestion de l'anonymat des communications dans les réseaux véhiculaires Ad hoc sans fil (VANETs)*. PhD thesis, Université du Québec à Trois-Rivières, 2015.

[39] Adetundji Adigun. *Gestion de l'anonymat et de la traçabilité dans les réseaux véhiculaires sans fil*. PhD thesis, Université du Québec à Trois-Rivières, 2014.

[40] Christian Tchepnda. *Authentification dans les réseaux véhiculaires opérés*. PhD thesis, 2008.

[41] Ghassan Samara, Wafaa AH Al-Salihy, and R Sures. Security analysis of vehicular ad hoc nerworks (vanet). In *2010 Second International Conference on Network Applications, Protocols and Services*, pages 55–60. IEEE, 2010.

[42] Mohamed Nidhal Mejri. *Securing Vehicular Networks Against Denial of Service Attacks*. PhD thesis, 2016.

[43] ABBAS Assia et al. *DETECTION D'INTRUSION DANS LES RESEAUX VANETS*. PhD thesis, 2016.

[44] Nicole El Zoghby. *Fusion distribuée de données échangées dans un réseau de véhicules*. PhD thesis, Université de Technologie de Compiègne, 2014.

[45] Elyes Ben Hamida, Hassan Noura, and Wassim Znaidi. Security of cooperative intelligent transport systems : Standards, threats analysis and cryptographic countermeasures. *Electronics*, 4(3) :380–423, 2015.

[46] Sudeep Tanwar, Jayneel Vora, Sudhanshu Tyagi, Neeraj Kumar, and Mohammad S Obaidat. A systematic review on security issues in vehicular ad hoc network. *Security and Privacy*, 1(5) :e39, 2018.

[47] Ruben Morales-Ferre, Philipp Richter, Emanuela Falletti, Alberto de la Fuente, and Elena Simona Lohan. A survey on coping with intentional interference in satellite navigation for manned and unmanned aircraft. *IEEE Communications Surveys & Tutorials*, 22(1) :249–291, 2019.

[48] Irshad Ahmed Sumra, Jamalul-Lail Ab Manan, and Halabi Hasbullah. Timing attack in vehicular network. In *Proceedings of the 15th WSEAS In-*

*ternational Conference on Computers, World Scientific and Engineering Academy and Society (WSEAS), Corfu Island, Greece*, pages 151–155, 2011.

[49] Jaroslaw Magiera and Ryszard Katulski. Detection and mitigation of gps spoofing based on antenna array processing. *Journal of applied research and technology*, 13(1) :45–57, 2015.

[50] Irshad Ahmed Sumra, Halabi Bin Hasbullah, and Jamalul-lail Bin Ab-Manan. Attacks on security goals (confidentiality, integrity, availability) in vanet : a survey. In *Vehicular Ad-Hoc Networks for Smart Cities*, pages 51–61. Springer, 2015.

[51] David Martins and Herve Guyennet. Etat de l'art-sécurité dans les réseaux de capteurs sans fil. 2008.

[52] Ajay N Upadhyaya and JS Shah. Attacks on vanet security. *Int J Comp Eng Tech*, 9(1) :8–19, 2018.

[53] Jin Cui, Lin Shen Liew, Giedre Sabaliauskaite, and Fengjun Zhou. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90 :101823, 2019.

[54] Fernand Lone Sang. *Protection des systèmes informatiques contre les attaques par entrées-sorties*. PhD thesis, 2012.

[55] Kuldeep Kumar and Sandeep Kumar Arora. Review of vehicular ad hoc network security. *International Journal of Grid and Distributed Computing*, 9(11) :17–34, 2016.

[56] Botnet mirai. https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/. Accessed : 2021-01-08.

[57] 9 of history's notable botnet attacks. https://www.whiteops.com/blog/9-of-the-most-notable-botnets. Accessed : 2021-01-09.

[58] Hachem Guerid. *Systèmes coopératifs décentralisés de détection et de contre-mesures des incidents et attaques sur les réseaux IP*. PhD thesis, Paris, ENST, 2014.

[59] Botnet eggheads. http://eggheads.org/. Accessed : 2021-01-08.

[60] Internet relay chat protocol. https://tools.ietf.org/html/rfc1459. Accessed : 2021-01-08.

[61] Unrealakama/agobot. https://github.com/UnrealAkama/AgoBot. Accessed : 2021-01-08.

[62] Sdbot. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/sdbot. Accessed : 2021-01-08.

[63] Armooo/gtbot. https://github.com/arm000/gtbot. Accessed : 2021-01-08.

[64] Zeus virus. https://www.kaspersky.com/resource-center/threats/zeus-virus. Accessed : 2021-01-08.

[65] Citadel botnet. https://www.rogers.com/customer/support/article/citadel-botnet. Accessed : 2021-01-09.

[66] Gameover zeus (goz) botnet. https://www.knowbe4.com/gameover-zeus. Accessed : 2021-01-09.

[67] Carberp botnet. https://www.welivesecurity.com/2012/07/02/all-carberp-botnet-organizers-arrested/. Accessed : 2021-01-09.

[68] Spyeye botnet. https://www.fortiguard.com/encyclopedia/botnet/12. Accessed : 2021-01-09.

[69] Mariposa botnet. https://krebsonsecurity.com/tag/mariposa-botnet/. Accessed : 2021-01-09.

[70] Wei Yan, Zheng Zhang, and Nirwan Ansari. Revealing packed malware. *ieee seCurity & PrivaCy*, 6(5) :65–69, 2008.

[71] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The nepenthes platform : An efficient approach to collect malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

[72] Lance Spitzner. The honeynet project : Trapping the hackers. *IEEE Security & Privacy*, 1(2) :15–23, 2003.

[73] Sergio Vidal-González, Isaías García-Rodríguez, Héctor Aláiz-Moretón, Carmen Benavides-Cuéllar, José Alberto Benítez-Andrades, María Teresa García-Ordás, and Paulo Novais. Analyzing iot-based botnet mal-

ware activity with distributed low interaction honeypots. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, Sandra Costanzo, Irena Orovic, and Fernando Moreira, editors, *Trends and Innovations in Information Systems and Technologies*, pages 329–338, Cham, 2020. Springer International Publishing.

[74] Meng Wang, Javier Santillan, and Fernando Kuipers. Thingpot : an interactive internet-of-things honeypot. *arXiv preprint arXiv :1807.04114*, 2018.

[75] Niels Provos and Thorsten Holz. *Virtual Honeypots : From Botnet Tracking to Intrusion Detection*. 2007.

[76] Juan Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. SIPHON : towards scalable high-interaction physical honeypots. *CoRR*, abs/1701.02446, 2017.

[77] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot : A novel honeypot for revealing current iot threats. *Journal of Information Processing*, 24(3) :522–533, 2016.

[78] Usha Devi Gandhi, Priyan Malarvizhi Kumar, R Varatharajan, Gunasekaran Manogaran, Revathi Sundarasekar, and Shreyas Kadu. Hiotpot : surveillance on iot devices against recent threats. *Wireless personal communications*, 103(2) :1179–1194, 2018.

[79] Pierre-Antoine Vervier and Yun Shen. Before toasters rise up : A view into the emerging iot threat landscape. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 556–576. Springer, 2018.

[80] Kapil Sinha, Arun Viswanathan, and Julian Bunn. Tracking temporal evolution of network activity for botnet detection. *arXiv preprint arXiv :1908.03443*, 2019.

[81] David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali Ghorbani, and Dan Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39 :2–16, 2013.

[82] W Timothy Strayer, David Lapsely, Robert Walsh, and Carl Livadas. Botnet detection based on network behavior. In *Botnet detection*, pages 1–24. Springer, 2008.

[83] Supranamaya Ranjan. Machine learning based botnet detection using real-time extracted traffic features, March 25 2014. US Patent 8,682,812.

[84] Supranamaya Ranjan and Feilong Chen. Machine learning based botnet detection with dynamic adaptation, March 19 2013. US Patent 8,402,543.

[85] Rabah Rahal, Noudjoud Kahya, and Nacira Ghoualmi-Zine. Resistance against dos attacks in vanets using the ids snort. *Revue de l'Information Scientifique et Technique*, 24(1) :20–30, 2019.

[86] Anirudh Ramachandran, David Dagon, and Nick Feamster. Can dns-based blacklists keep up with bots ? In *CEAS*. Citeseer, 2006.

[87] Juan M Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E Diaz-Verdejo. Anomaly detection methods in wired networks : a survey and taxonomy. *Computer Communications*, 27(16) :1569–1584, 2004.

[88] Nicoló Andronio, Stefano Zanero, and Federico Maggi. Heldroid : Dissecting and detecting mobile ransomware. In *International Symposium on Recent Advances in Intrusion Detection*, pages 382–404. Springer, 2015.

[89] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid : an information-flow tracking system for real-time privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2) :1–29, 2014.

[90] Sven Nomm and Hayretdin Bahsi. Unsupervised anomaly based botnet detection in IoT networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, December 2018.

[91] Raihana Syahirah Binti Abdullah, MA Faizal, and Zul Azri Muhamad Noh. Multivariate statistical analysis on anomaly p2p botnets detection. *EVOLUTION*, 8(12), 2017.

[92] Mevlut Turker Garip, Mehmet Emre Gursoy, Peter Reiher, and Mario Gerla. Congestion attacks to autonomous cars using vehicular botnets. In *NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA*, 2015.

[93] Mevlut Turker Garip, Peter Reiher, and Mario Gerla. Botveillance : A vehicular botnet surveillance attack against pseudonymous systems in vanets. In *2018 11th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–8. IEEE, 2018.

[94] Mevlut Turker Garip, Peter Reiher, and Mario Gerla. Riot : A rapid exploit delivery mechanism against iot devices using vehicular botnets. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–6. IEEE, 2019.

[95] Mevlut Turker Garip, Paul Hyungmin Kim, Peter Reiher, and Mario Gerla. Interloc : An interference-aware rssi-based localization and sybil attack detection mechanism for vehicular ad hoc networks. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2017.

[96] Mingyan Li, Krishna Sampigethaya, Leping Huang, and Radha Poovendran. Swing & swap : user-centric approaches towards maximizing location privacy. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 19–28, 2006.

[97] Mevlut Turker Garip, Jonathan Lin, Peter Reiher, and Mario Gerla. Shieldnet : An adaptive detection mechanism against vehicular botnets in vanets. In *2019 IEEE Vehicular Networking Conference (VNC)*, pages 1–7. IEEE, 2019.

[98] Alan C Bovik and Scott T Acton. Basic linear filtering with application to image enhancement. In *The Essential Guide to Image Processing*, pages 225–239. Elsevier, 2009.

[99] Gonzalo De La Torre, Paul Rad, and Kim-Kwang Raymond Choo. Driverless vehicle security : Challenges and future research opportunities. *Future Generation Computer Systems*, 108 :1092–1111, 2020.

[100] Hariharan Krishnan, Fan Bai, and Gavin Holland. Commercial and public use applications. In *Vehicular Networking*, pages 1–28. John Wiley & Sons, Ltd, April 2010.

[101] Martina Stoyanova Todorova and Stamelina Tomova Todorova. Ddos attack detection in sdn-based vanet architectures. *no. June*, page 175, 2016.

[102] Parul Tyagi and Deepak Dembla. Investigating the security threats in vehicular ad hoc networks (vanets) : towards security engineering for safer on-road transportation. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2084–2090. IEEE, 2014.

[103] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*, volume 4, pages 447–462. San Francisco, 2011.

[104] Ben Lovejoy, Ben Lovejoy, Ben Lovejoy, and Eu. Malicious siri commands can be hidden in music and innocuous-sounding speech recordings, May 2018.

[105] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands : Laser-based audio injection attacks on voice-controllable systems. 2019.

[106] Margi Murphy. How google is secretly recording you through your mobile, monitoring millions of conversations every day and storing the creepy audio files, Aug 2017.

[107] Mark Wood and Michael Erlinger. Intrusion detection message exchange requirements. *IETF, draft-ietf-idwg-requirements-10*, 2002.

[108] J Rajahalme, A Conta, B Carpenter, and S Deering. Rfc3697 : Ipv6 flow label specification, 2004.

[109] Controller area network (can) link layer. https://erg.abdn.ac.uk/users/gorry/eg3576/CAN-link.html. Accessed : 2021-01-08.

[110] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[111] Corinna Cortes and Vladimir Vapnik. *Machine Learning*, 20(3) :273–297, 1995.

[112] Songbo Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4) :667–671, 2005.

[113] Machine learning, neural networks and algorithms. https://chatbotsmagazine.com/machine-learning-neural-networks-and-algorithms-5c0711eb8f9a. Accessed : 2021-01-09.

[114] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. Gids : Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018.

[115] Network simulator 3. https://www.nsnam.org/. Accessed : 2021-01-09.

[116] Simulation of urban mobility (sumo). http://sumo.sourceforge.net/. Accessed : 2021-01-09.

[117] Openstreetmap. https://www.openstreetmap.org/. Accessed : 2021-01-08.

[118] Cicflowmeter. http://netflowmeter.ca/. Accessed : 2019-07-08.

[119] Forward selection algorithm. http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/. Accessed : 2019-07-08.

[120] Linear-svc. https://scikit-learn.org/stable/modules/feature_selection.html#l1-based-feature-selection. Accessed : 2019-07-08.

[121] Car-hacking dataset. http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset. Accessed : 2021-01-09.

[122] Abdul Quyoom, Raja Ali, Devki Nandan Gouttam, and Harish Sharma. A novel mechanism of detection of denial of service attack (dos) in va-net using malicious and irrelevant packet detection algorithm (mipda).

In *International Conference on Computing, Communication & Automation*, pages 414–419. IEEE, 2015.

[123] David A Schmidt, Mohamad S Khan, and Brian T Bennett. Spline based intrusion detection in vehicular ad hoc networks (vanet). *arXiv preprint arXiv :1903.08018*, 2019.

[124] Pranav Kumar Singh, Shivam Gupta, Ritveeka Vashistha, Sunit Kumar Nandi, and Sukumar Nandi. Machine learning based approach to detect position falsification attack in vanets. In *International Conference on Security & Privacy*, pages 166–178. Springer, 2019.

[125] Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, and Yaser Jararweh. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90 :101842, 2019.

[126] Nikita Lyamin, Denis Kleyko, Quentin Delooz, and Alexey Vinel. Real-time jamming dos detection in safety-critical v2v c-its using data mining. *IEEE Communications Letters*, 23(3) :442–445, 2019.

[127] Andreas Tomandl, Karl-Peter Fuchs, and Hannes Federrath. Rest-net : A dynamic rule-based ids for vanets. In *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–8. IEEE, 2014.

[128] Vasiliy Krundyshev, Maxim Kalinin, and Peter Zegzhda. Artificial swarm algorithm for vanet protection against routing attacks. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 795–800. IEEE, 2018.

[129] Sushil Kumar and Kulwinder Singh Mann. Detection of multiple malicious nodes using entropy for mitigating the effect of denial of service attack in vanets. In *2018 4th International Conference on Computing Sciences (ICCS)*, pages 72–79. IEEE, 2018.

[130] Sohan Gyawali and Yi Qian. Misbehavior detection using machine learning in vehicular communication networks. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[131] Ayesha Anzer and Mourad Elhadef. Deep learning-based intrusion detection systems for intelligent vehicular ad hoc networks. In *Lecture*

*Notes in Electrical Engineering*, pages 109–116. Springer Singapore, November 2018.

[132] Kdd99. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed : 2019-07-08.

[133] caida. https://www.caida.org/data/. Accessed : 2019-07-08.

[134] Nsl-kdd. https://www.unb.ca/cic/datasets/nsl.html. Accessed : 2019-07-08.

[135] Iscx. http://www.iscx.ca/datasets/. Accessed : 2019-07-08.

[136] Cicids2017. https://www.unb.ca/cic/datasets/ids-2017.html. Accessed : 2019-07-08.

[137] Unsw-nb15. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/. Accessed : 2019-07-08.

[138] Ns3. https://www.nsnam.org/. Accessed : 2019-07-08.

[139] Nctuns. http://nsl.cs.nctu.edu.tw/NSL/nctuns.html. Accessed : 2019-07-08.

[140] Packet sender. https://packetsender.com/. Accessed : 2019-07-08.

[141] Wireshark. https://www.wireshark.org/. Accessed : 2019-07-08.

[142] Arash Habibi Lashkari, Andi Fitriah A Kadir, Hugo Gonzalez, Kenneth Fon Mbah, and Ali A Ghorbani. Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 233–23309. IEEE, 2017.