UNIVERSITÉ DE LORRAINE

Rapport de PTA

Étude de l'impact de l'attaque « Ghost Vehicle » sur un simulateur de réseaux véhiculaires en Python

Étudiant : Florian RICHARD

Tuteurs : Ye-Qiong SONG, Runbo SU

LORRAINE INP Ensem

15/02/2024

Table des matières

Abstract— In the realm of autonomous vehicles, the future promises a transformative shift towards self-driving cars. As these vehicles become integral to our daily lives, it is essential to comprehend both the opportunities and challenges they present. Autonomous vehicle networks will be created to ensure self-driving, and security will be mandatory in this kind of network to prevent harm to people or any threats. This report presents the creation of a Python simulator to measure the feasibility of "CPM-base Ghost Vehicles" attacks. The "Ghost vehicle" attack consists of sending false information in the shape of the presence of another vehicle in the network by an attacker (malicious vehicle) to disrupt traffic. The work done is a Python vehicle network simulator to measure the opportunity for an attacker to create this attack. As expected, results show that as the traffic get denser, the attacker is rapidly identified by the network, as other nearby vehicles do not detect the presence of the "Ghost vehicle".

Résumé—Dans le domaine des véhicules autonomes, l'avenir promet une transformation majeure vers les voitures sans conducteur. Alors que ces véhicules deviennent indispensables dans notre quotidien, il est primordial de saisir les opportunités ainsi que les défis qu'ils présentent. Des réseaux de véhicules autonomes seront établis pour assurer la conduite automatisée, et la sécurité sera impérative dans ce type de réseau afin d'éviter tout préjudice aux individus ou toute menace potentielle. Ce rapport expose la conception d'un simulateur Python visant à évaluer la faisabilité des attaques de "véhicules fantômes" basées sur le CPM. L'attaque de "véhicule fantôme" consiste à diffuser de fausses informations sous la forme de la présence d'un autre véhicule dans le réseau par un attaquant (véhicule malveillant) afin de perturber la circulation. Le travail accompli consiste en un simulateur de réseau de véhicules Python destiné à évaluer la possibilité pour un attaquant de mener cette attaque. Comme prévu, les résultats démontrent qu'à mesure que le trafic s'intensifie, le réseau identifie rapidement l'attaquant, car les autres véhicules à proximité ne détectent pas la présence du "véhicule fantôme".

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude envers mes encadrants, M. Runbo SU et M. Ye-Qiong SONG, pour leur soutien et leur disponibilité tout au long de ce projet.

Je remercie également M. Bilal Himite, le créateur du simulateur d'origine sur lequel s'appuie ce projet. Son travail préalable a posé les bases indispensables pour le développement de mon simulateur.

1 INTRODUCTION

Les progrès fulgurants dans le domaine des véhicules autonomes ouvrent la voie à une transformation dans notre façon de concevoir la mobilité. L'émergence des voitures sans conducteur promet non seulement d'améliorer l'efficacité et la sécurité du transport, mais aussi de redéfinir les interactions entre les individus et leur environnement routier. Toutefois, avec cette révolution technologique viennent de nouveaux défis, notamment en matière de sécurité des réseaux de communication.

Les réseaux de véhicules autonomes sont au cœur de cette transformation, permettant aux véhicules de communiquer entre eux et avec l'infrastructure routière pour partager des informations cruciales sur l'environnement routier. Ces échanges de données, réalisés via des messages V2X (Vehicle-to-Everything), permettent d'assurer la sécurité et l'efficacité du système de transport.

Cependant, la connectivité accrue des véhicules ouvre la porte à de nouveaux vecteurs d'attaque. Les attaques malveillantes ciblant les réseaux de véhicules peuvent compromettre la sécurité, l'intégrité et la fiabilité des systèmes, mettant en danger tous les acteurs de son système. Parmi ces attaques potentielles, l'attaque Ghost vehicle émerge comme une préoccupation.

L'attaque Ghost vehicle (plus précisement CPM-base Ghost Vehicles), basée sur l'injection de fausses informations dans les messages de perception collective (Collective Perception Messages), vise à perturber le fonctionnement normal des réseaux de véhicules autonomes en induisant en erreur les véhicules et l'infrastructure routière sur la présence de véhicules fictifs.

Dans ce contexte, ce rapport présente le développement et l'analyse d'un simulateur Python depuis un existant dédié à l'étude de l'attaque Ghost vehicle dans les réseaux de véhicules autonomes [1]. Ce simulateur vise à évaluer la faisabilité de l'attaque. Dans les chapitres suivants, nous examinerons l'état de l'art des réseaux de véhicules autonomes, les types d'attaques potentielles et les solutions de sécurité existantes. Nous présenterons ensuite en détail le modèle de l'attaque Ghost vehicle et les mécanismes de détection envisagés. Enfin, nous décrirons la conception et

l'implémentation du simulateur Python, ainsi que les résultats obtenus et difficultés rencontrées pendant le PTA.

2 ÉTAT DE L'ART

2.1 Réseaux véhiculaires

Les réseaux ad hoc véhiculaires (VANET) sont un sous-ensemble des réseaux ad hoc mobiles (MANET) spécifiquement conçus pour la communication entre les véhicules et les infrastructures routières. Les VANET permettent aux véhicules de communiquer entre eux et avec l'infrastructure routière afin d'améliorer la sécurité routière et l'efficacité du trafic et de fournir divers services d'info-divertissement. La mise en réseau des véhicules implique l'échange d'informations entre les véhicules et l'infrastructure à l'aide de technologies de communication sans fil.

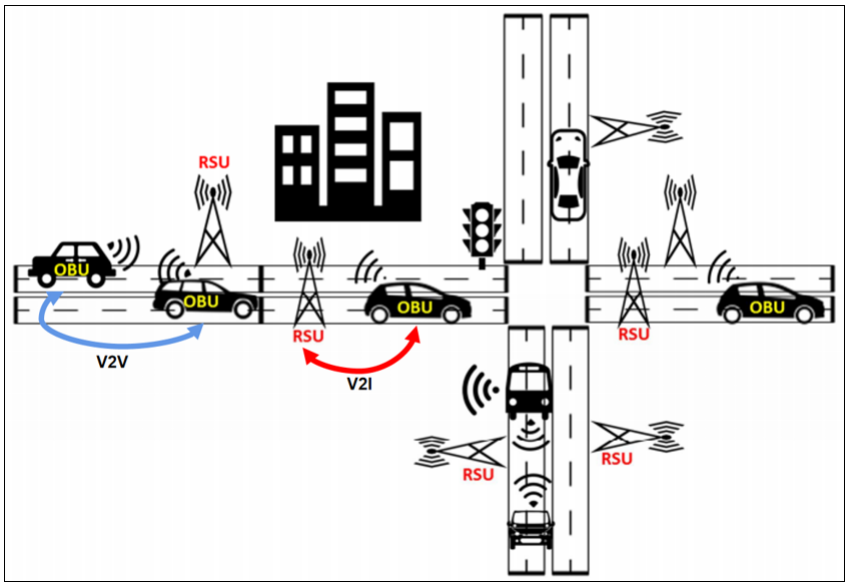

FIGURE 2.1: Composants et modèle de communication de réseaux véhiculaires, source : [2]

Les réseaux de véhicules sont confrontés à des défis tels que les topologies de réseaux dynamiques, la mobilité élevée, la connectivité intermittente et les problèmes de sécurité. Les chercheurs explorent des solutions pour améliorer les performances et la fiabilité des réseaux VANET, telles que des protocoles de routage intelligents, des mécanismes de contrôle de la congestion et

des techniques de cryptage et confiance pour sécuriser les communications.

Comme illustré dans Fig. 2.1, l'OBU (On-Board Unit) est un dispositif installé dans les véhicules pour la communication au sein des VANET. La RSU (Roadside Unit) est une infrastructure déployée le long des routes pour faciliter la communication. V2V (Vehicle-to-Vehicle) désigne la communication entre les véhicules, tandis que V2I (Vehicle-to-Infrastructure) désigne la communication entre les véhicules et l'infrastructure routière.

2.2 Messages V2X

Plusieurs types de message ont été standardisés par l'ETSI (European Telecommunications Standards Institute) afin d'échanger des informations routières de manière coopérative : CAM (Cooperative Awareness Message)[3], DENM (Decentralized Environmental Notification Messages)[4], et CPM (Collective Perception Messages)[5].

— **CAM** : Chaque véhicule diffuse des messages CAM périodiquement contenant sa vitesse, sa position, son type, son accélération, etc., pour faire connaitre de la situation routière. Ce type de message facilite la communication de V2X, permettant aux véhicules proches d'être informés de la présence et des mouvements des autres, ce qui renforce la sécurité et permet des fonctionnalités de conduite coopérative.

— **CPM** : Le CPM comprend des informations sur les objets environnants, tels que les piétons, les cyclistes, les obstacles et l'infrastructure routière. En partageant des données de perception riches, les véhicules peuvent collaborer plus efficacement dans des scénarios de conduite complexes, améliorant ainsi la connaissance de la situation et les capacités de prise de décision.

— **DENM** : En plus des messages DENM peuvent être générés lorsqu'un événement spécifique est détecté, il comprend des alertes sur les dangers de la route, les accidents, les conditions de circulation et d'autres données pertinentes. Les messages DENM sont essentiels pour permettre aux véhicules de réagir rapidement à l'évolution des conditions routières et aux dangers potentiels.

Dans ce projet, nous nous concentrons sur les CPM, en particulier les messages CPM contenant de fausses informations qui ont été générés par les véhicules malveillants, plus précisément, CPM-base Ghost Vehicles (Ce type d'attaque sera détaillé dans la section suivante).

2.3 Simulateurs

TABLE 2.1: Les simulateurs existants

Simulateur	Open-source	Langage	Mobilité	Communication	Route/Carte
iTETRIS	Oui	C++	SUMO	NS3	*, +
VANET Toolbox	Non	Matlab	Matlab	Matlab	Matlab
Veins	Oui	C++	SUMO	OMNET++	*, +
TrafficSimulator	Oui	Python	*	Non	*

* = user-defined, += map data import

Les simulateurs existants permettant la modélisation des réseaux véhiculaires sont résumés dans le tableau 2.1 : iTETRIS [6], VANET-Toolbox [7], Veins [8], and TrafficSimulator [9]. Comme on peut le voir, les technologies prises en compte dans ces simulateurs sont très diverses. Par exemple, les langages de programmations utilisés sont C++, Python et Matlab. VANET Toolbox n'est pas un simulateur open-source, toutes ses fonctionnalités sont appliquées par Matlab. iTETRIS et Veins sont tous les deux basés sur SUMO [10] (Simulation of Urban MObilit) pour les mobilités des véhicules, et NS3 [11] (Network Simulator 3) et'OMNET++ [12] sont conçus pour démontrer le fonctionnement des réseaux. Différemment, le TrafficSimulator est un simulateur open-source basé sur Python, où la communication V2X est supposée être traitée de manière "invisible", grace à ça, ce simulateur pourra aider à construire un outil de calcul au niveau de measure la confiance dans les messages V2X. Compte tenu des contraintes de temps de ce projet, nous avons choisi trafficSimulator pour la validation du modèle de confiance liées aux messages CPM.

3 ÉTUDE THÉORIQUE

3.1 Les attaques contre les réseaux véhiculaires

Les réseaux de véhicules sont susceptibles d'être la cible d'attaques par des individus malveillants. Dans notre cas, l'attaque Ghost vehicle.

3.1.1 Généralités sur les attaques

Comme toute technologie connectée, les messages V2X sont susceptible d'être sujette à des attaques malveillantes qui peuvent compromettre la sécurité et la confidentialité des données, ainsi que la fiabilité des systèmes. Dans notre cas, nous nous sommes focalisés sur les attaques concernant le Collective Perception Message (CPM). Le Collective Perception Message dans le contexte des V2X est un aspect crucial du partage d'informations entre les véhicules et l'infrastructure. Le CPM permet aux véhicules et aux infrastructures de communiquer des informations sur l'environnement routier afin d'améliorer la sécurité et l'efficacité du système de transport.

Les attaques potentielles sur le CPM pourrait avoir un impact significatif sur la sécurité et le bon fonctionnement des systèmes V2X. Voici quelques exemples d'attaques qui pourraient être dirigées contre le CPM :

1. **Falsification des données CPM** : Les attaquants pourraient intercepter et modifier les messages CPM pour inclure de fausses informations sur les conditions routières, les obstacles ou d'autres éléments de l'environnement, induisant ainsi en erreur les autres véhicules ou l'infrastructure.
2. **Attaques de répudiation** : Les attaquants pourraient nier avoir envoyé des messages CPM spécifiques, ce qui pourrait rendre difficile l'attribution de la responsabilité en cas d'incident ou de problème sur la route.

3. **Attaques de déni de service (DoS)** : Les attaquants pourraient tenter de saturer le réseau en envoyant un grand nombre de messages CPM ou en perturbant les communications entre les véhicules et l'infrastructure, compromettant ainsi la capacité du système à fournir des informations cruciales en temps réel.

4. **Attaques par injection de données malveillantes** : Les attaquants pourraient injecter des données malveillantes dans les messages CPM pour induire en erreur les autres participants au réseau ou pour perturber le fonctionnement normal du système.

Pour contrer ces attaques, des mesures de sécurité telles que le chiffrement des communications, l'authentification des messages, la détection d'anomalies et la résilience du système sont essentielles pour garantir l'intégrité, la confidentialité et la disponibilité des données CPM dans les systèmes V2X.

3.1.2 L'attaque Ghost vehicle

L'attaque Ghost vehicle (CPM-base Ghost Vehicles) suit la logique d'une attaque par injection de données malveillantes. Cette attaque est la suivante : dans le réseau véhiculaire, un attaquant se faisant passer pour un véhicule légitime crée de nouvelles informations, dans notre cas la position d'un nouveau véhicule dans le réseau. Ce nouveau véhicule, n'étant pas dans le réseau, il n'est repérable que depuis une distance restreinte (capteur d'une voiture). Les véhicules connectés au CPM vont alors agir en pensant qu'il existe un véhicule de plus dans l'infrastructure routière, ce qui va perturber le réseau et compromettre la sécurité au sein de celui-ci.

3.2 Modèle de vérification

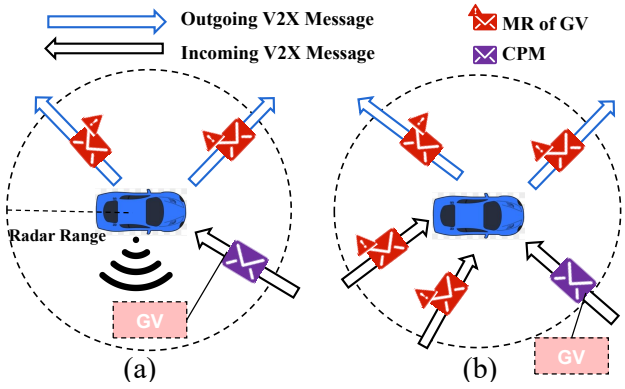

FIGURE 3.1: Modèle de vérification

Pour repérer cette attaque, un système de confiance peut être mis en place dans le but de caté-

goriser l'attaquant et de l'écarter par la suite. Le modèle de vérification est le suivant : les individus du réseau comparent si la position donnée par l'attaquant du ghost vehicle est possible. Ainsi, si le réseau reçoit une majorité d'individus ne trouvant pas le véhicule fantôme alors en portée de détection, alors l'attaquant est repéré. En cas d'attaque seule, il suffit de deux individus du réseau pour disqualifier l'attaquant. Cette disqualification permet de discréditer tous les messages V2X et donc la participation CPM de l'attaquant par la suite.

4 SIMULATION

Pour le travail de ce PTA, j'ai donc dû effectuer un simulateur Python capable de modéliser l'attaque ghost vehicle. Ce travail consistait à une adaptation d'un simulateur déjà existant disponible sur GitHub[9]. Le simulateur déjà existant est adaptable et consiste en une simple simulation de trafic où les voitures évitent les collisions entre elles. Il a donc fallu adapter cet environnement pour pouvoir recréer l'attaque ghost vehicle ainsi que de nouvelles fonctionnalités afin de répondre au sujet du PTA.

4.1 Code existant

Pour bien comprendre les ajouts que j'ai effectués à ce simulateur, je vais dans un premier temps vous expliquer son fonctionnement. Le code compte quatre classes principales :

1. *simulation*
2. *segment*
3. *vehicle*
4. *window*

Simulation est une classe qui est composée de segments (de la classe *segment*) et de véhicules (de la classe *vehicle*) avec les classes du même nom. C'est à partir de cette classe que le cœur de la simulation est faite : les déplacements des véhicules au sein des segments et l'évitement de collisions entre eux sur les segments.

C'est dans la classe *window* qui est la plus importante, c'est elle qui fait le lien entre la simulation et l'interface graphique Python utilisée : elle permet le passage d'un passage de une dimension à deux (axe y) pour l'affichage car celle-ci, par souci d'optimisation, n'a pas été implémentée dans la classe *Simulation*.

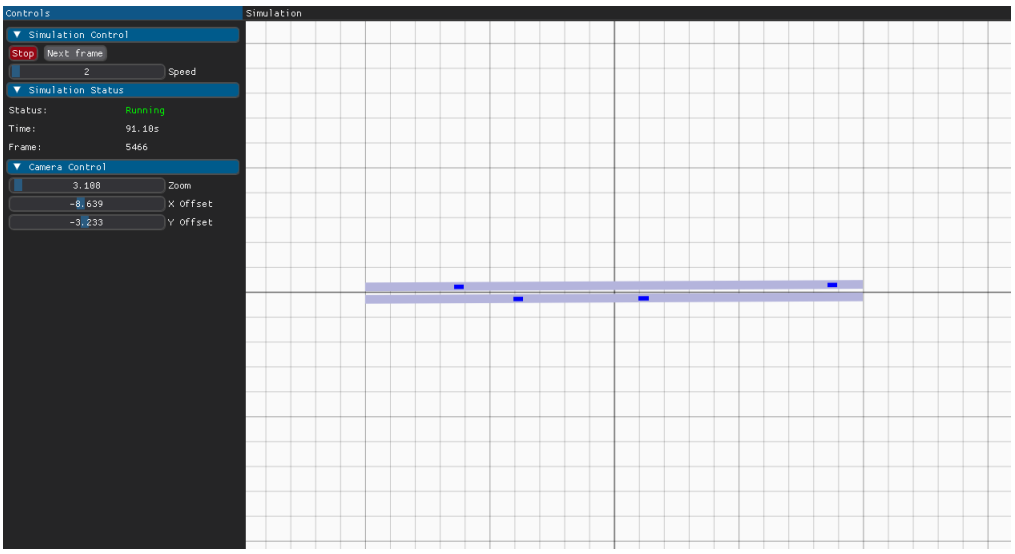

FIGURE 4.1: Simulateur d'origine

La figure 4.1 montre le simulateur d'origine. Celui-ci utilise le module Dearpygui pour la partie graphique.

4.2 Ajout de code pour le PTA

Pour le travail que je devais accomplir, j'ai changé en profondeur le code existant. Pour des raisons de compréhension, je vais faire l'énumération des nouvelles fonctionnalités par ordre chronologique.

4.2.1 Création de la carte

Pour comprendre le code ainsi que me familiariser avec celui-ci, j'ai dans un premier temps créé une nouvelle carte qui puisse mieux me permettre de découvrir, pour les voitures, l'attaque ghost vehicle . Ce travail n'est pas vraiment difficile mais très fastidieux car toute la carte doit être créée à la main (aucun outil n'a été créé pour faire la carte plus facilement).

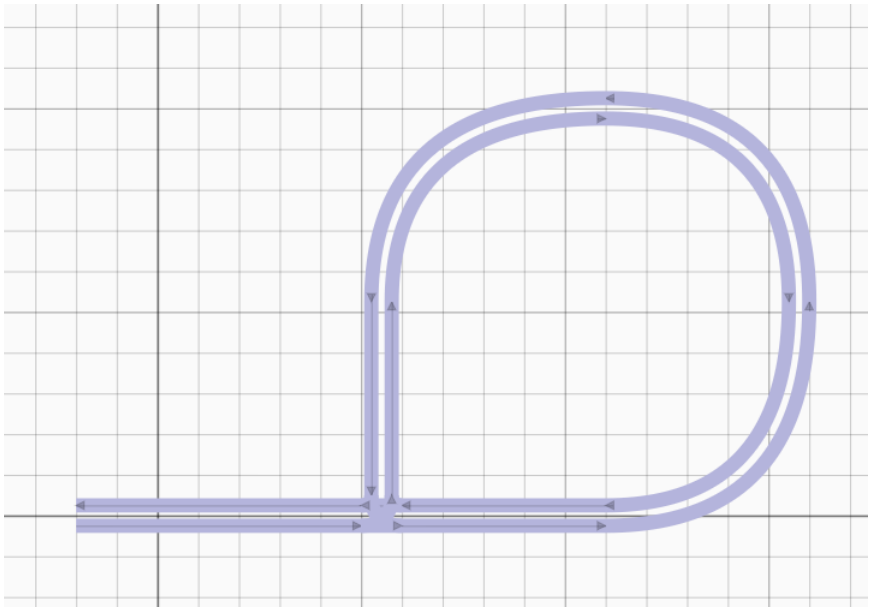

FIGURE 4.2: Carte créée

J'ai donc créé la carte 4.2 qui permet grâce à l'intersection et à la boucle que beaucoup de véhicules se croisent à de multiples reprises afin de détecter efficacement l'attaque en question.

4.2.2 Gestion de la perception de l'environnement

Le deuxième ajout important a été l'implémentation de la perception environnante de chaque véhicule. Cette implémentation a été réalisée au niveau de la classe *window*, car comme expliqué précédemment, c'est à ce niveau que les deux dimensions sont gérées.

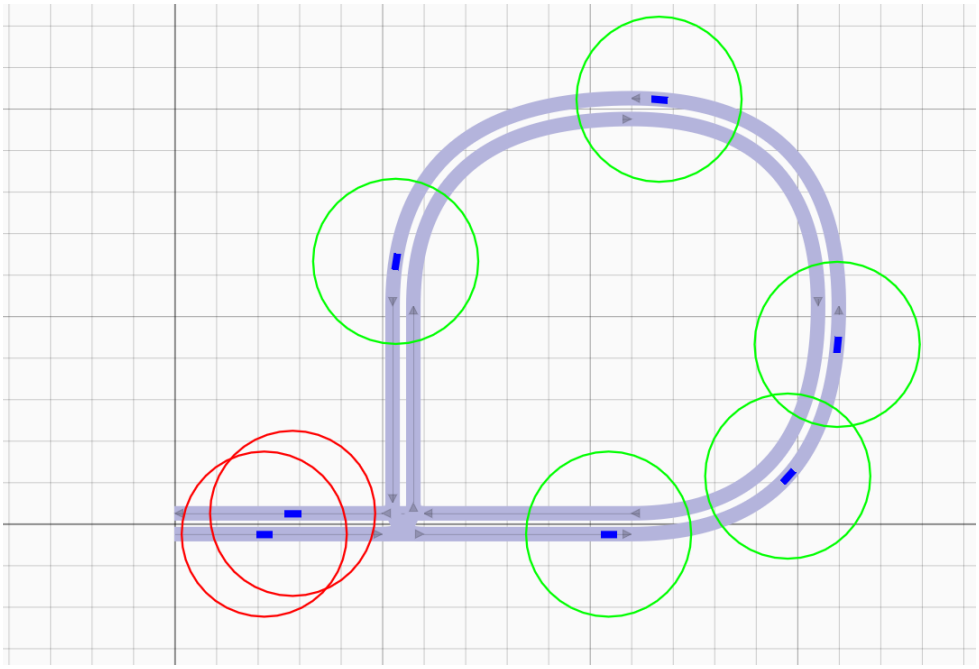

FIGURE 4.3: Perception entre véhicules

La figure 4.3 montre la perception directe entre deux véhicules. Le cercle de détection est rouge lorsqu'il détecte un véhicule et vert s'il n'en détecte aucun.

4.2.3 Définition de l'attaquant et du véhicule de référence

Dans cette partie, j'ai défini un attaquant ainsi qu'un véhicule de référence. Le véhicule de référence est le véhicule dont le point de vue est pris pour détecter l'attaque (dans le programme, il est nommé Trusted vehicle). Sur la carte, l'attaquant est représenté en rouge et le véhicule de référence en vert.

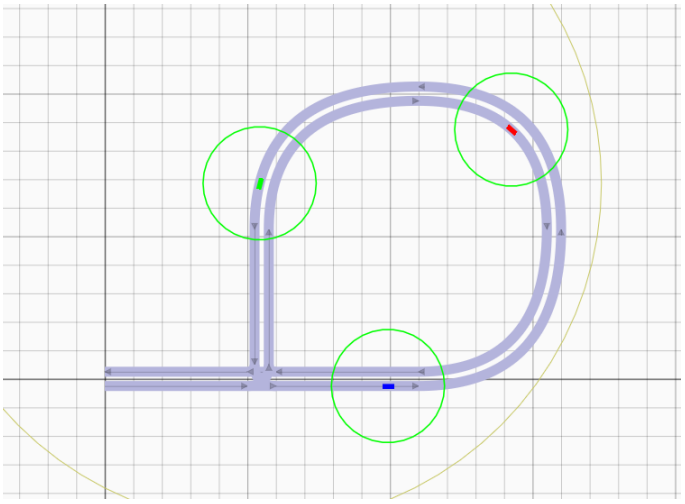

FIGURE 4.4: Définition de l'attaquant et du véhicule de référence

En plus de l'attaquant et du véhicule de référence, on peut voir sur la figure 4.4 en jaune foncé la délimitation de la limite d'envoi utilisée dans le CPM (cette partie est expliquée dans la suite).

Cet ajout vient avec deux nouveaux boutons dans l'interface comme le montre la figure 4.5. Avec le premier bouton, l'ajout de véhicule peut être fait à la main et avec le second, l'attaquant et le véhicule de référence peuvent être définis.

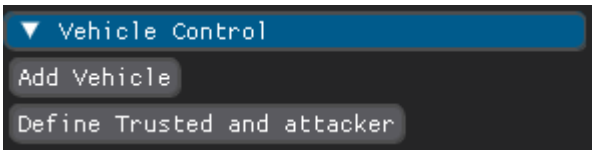

FIGURE 4.5: Nouveaux boutons de l'interface

4.2.4 Ajout du ghost vehicle

L'ajout du véhicule fantôme a été le plus difficile à implémenter dans le simulateur. Le simulateur d'origine n'a pas été conçu pour insérer un véhicule n'importe où ; il a été conçu pour qu'un

véhicule se crée forcément au début d'un segment. Pour réaliser cette implémentation, j'ai dû modifier une grande partie du code d'origine car la gestion du changement de segment se fait grâce à une pile. J'ai donc dû modifier le simulateur pour insérer un élément à une place précise de cette pile. De ce fait, l'avantage de cette pile sur la complexité de calcul du programme a été diminué.

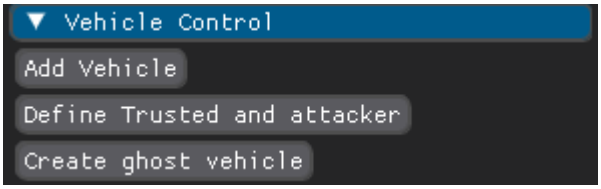

FIGURE 4.6: Bouton pour ajouter le ghost vehicle

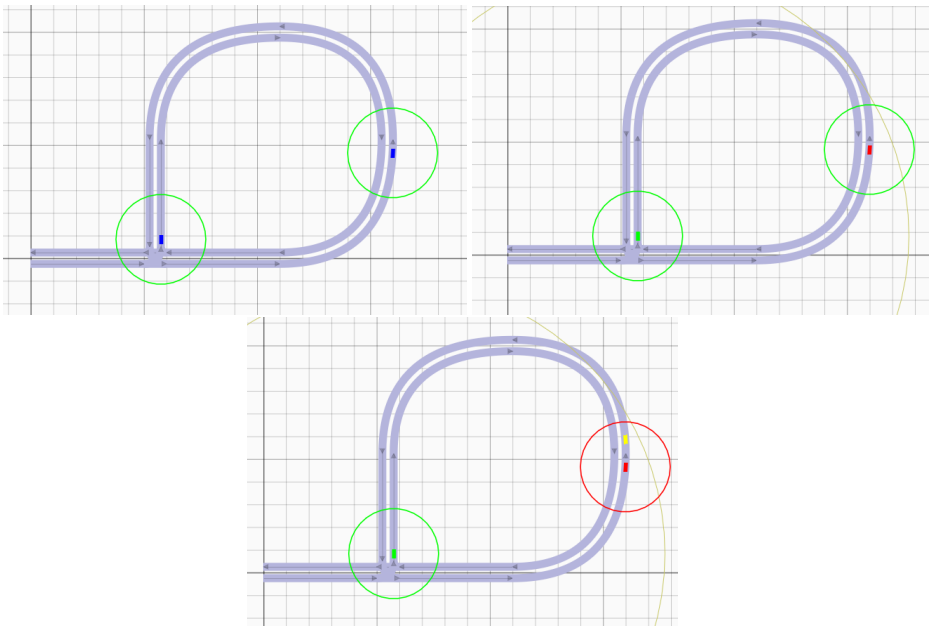

FIGURE 4.7: Différente étapes pour l'ajout du véhicule fantôme

4.2.5 Ajout de légende et de l'indicateur de repérage de l'attaque

Pour une meilleure compréhension, j'ai ajouté une légende expliquant les différents types de véhicules.

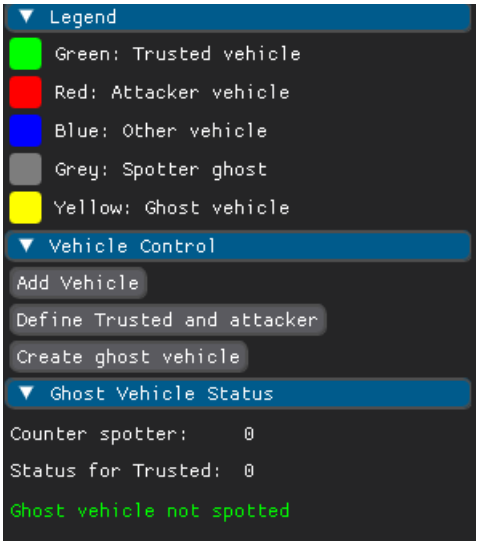

FIGURE 4.8: Interface du simulateur

La figure 4.8 présente des informations supplémentaires par rapport à celles déjà abordées : la gestion du statut du véhicule fantôme. Comme expliqué dans l'étude théorique, la détection de l'attaque se fait soit lorsque le véhicule de référence ne détecte pas le véhicule fantôme alors qu'il est à portée normale de détection, soit lorsque deux autres véhicules le sont ou l'ont été et que leur message est reçu par le véhicule de référence. Dans la figure 4.9, la simulation présente trois véhicules qui ont repéré l'attaque. Pour le véhicule de référence, au moins deux messages sont arrivés, donc l'attaque est également repérée par le véhicule de référence. C'est ici qu'est utilisée la portée d'envoi des messages.

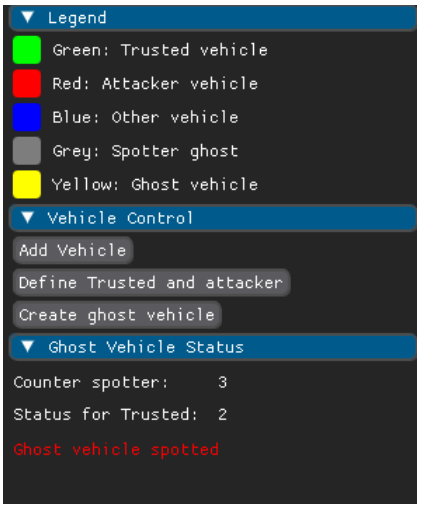

FIGURE 4.9: Interface du simulateur après détection

Ainsi, le simulateur peut désormais notifier sur l'interface graphique si l'attaque a été repérée par le véhicule de référence.

4.3 Résultats Obtenus et difficultés rencontrées

Les modifications apportées au simulateur fonctionnent après de longues heures de débogage. Elles sont néanmoins très coûteuses en termes de calcul pour la machine, ce qui se ressent sur l'interface graphique qui diminue le nombre d'images par seconde qu'elle est capable de générer. Pour résoudre ce problème, j'ai effectué des optimisations sur certaines boucles de code qui étaient en double au sein du programme.

En termes de détection, ce que l'on peut dire, c'est que, comme on pouvait le prévoir, avec un trafic plus dense, l'attaque est tout de suite repérée. Cela dépend aussi beaucoup des cartes sur lesquelles les véhicules se déplacent; plus les routes sont proches, plus l'attaque est visible. De plus, la distance de repérage personnelle et la distance radio des voitures peuvent être modifiées.

Voici les paramètres pris en compte pour le projet. Ceux-ci peuvent être modifiés dans le code git, en particulier dans le fichier *Firstmap.py* ainsi que dans la classe *window* présente dans le module trafficsimulator disponible sur mon git, dont le lien est présent en annexe 6.1 :

Vitesse véhicule	Portée détection directe	Portée détection CPM	Carte
16.6 m/s	20 m	120 m	Firstmap.py

5 CONCLUSION

En conclusion, les modifications apportées au simulateur Python ont permis de créer un environnement propice à l'étude de l'attaque Ghost vehicle dans les réseaux véhiculaires. Les fonctionnalités ajoutées, comme la gestion de la perception de l'environnement, la définition de l'attaquant et du véhicule de référence, l'ajout du véhicule fantôme et l'indicateur de repérage de l'attaque, offrent une meilleure compréhension du scénario simulé.

Cependant, ces ajustements ont augmenté considérablement la charge de calcul, impactant la fluidité de l'interface graphique. Des efforts d'optimisation ont été réalisés pour atténuer cet effet, mais des améliorations restent nécessaires.

Les résultats obtenus montrent que la détection de l'attaque Ghost vehicle dépend du trafic, de la densité des routes, des paramètres de détection des véhicules et de la distance de repérage. La simulation permet d'observer la réaction des véhicules face à cette attaque et d'en comprendre les conditions de détection.

En résumé, ce simulateur constitue un outil précieux pour l'analyse de l'attaque Ghost vehicle dans les réseaux véhiculaires, fournissant des informations sur les scénarios de détection et les paramètres influençant l'efficacité des contre-mesures. Des perspectives d'amélioration sont envisageables, notamment l'optimisation de la charge de calcul et l'extension du simulateur pour inclure d'autres types d'attaques ou de contre-mesures. L'intégration de messages V2X supplémentaires tels que CAM ou DENM pourrait également enrichir l'analyse.

Bibliographie

[1] Runbo Su, Yujun Jin, and Ye-Qiong Song. A cooperative trust model addressing CAM-and CPM-based Ghost Vehicles in IoV. working paper or preprint, February 2024. URL : `https : //hal.science/hal-04453209`.

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

[3] ETSI. 302 637-2 v1.4.1 -intelligent transport systems ; vehicular communications ; basic set of applications ; part 2 : Specification of cooperative awareness basic service. ETSI, Apr, 2019.

[4] ITS ETSI. Intelligent transport system (its) ; vehicular communications ; basic set of applications ; analysis of the collective-perception service (cps). Intelligent Transport Systems (ITS), 2019.

[5] TS ETSI. 102 637-2, intelligent transport systems (its) ; vehicular communications ; basic set of applications ; part 2 : Specification of co-operative awareness basic service. ETSI,, pages 14–48, 2010.

[6] Michele Rondinone, Julen Maneros, Daniel Krajzewicz, Ramon Bauza, Pasquale Cataldi, Fatma Hrizi, Javier Gozalvez, Vineet Kumar, Matthias Röckl, Lan Lin, et al. itetris : A modular simulation platform for the large scale evaluation of cooperative its applications. Simulation Modelling Practice and Theory, 34 :99–125, 2013.

[7] Le Wang, Renato Iida, and Alexander M Wyglinski. Vehicular network simulation environment via discrete event system modeling. IEEE Access, 7 :87246–87264, 2019.

[8] Christoph Sommer, David Eckhoff, Alexander Brummer, Dominik S Buse, Florian Hagenauer, Stefan Joerer, and Michele Segata. Veins : The open source vehicular network simulation framework. Recent Advances in Network Simulation : The OMNeT++ Environment and its Ecosystem, pages 215–252, 2019.

[9] Trafficsimualtor. https://github.com/BilHim/trafficSimulator.

[10] Sumo. https://github.com/eclipse-sumo/sumo.

[11] Ns-3. https://gitlab.com/nsnam/ns-3-dev.

[12] Omnet++. https://github.com/omnetpp/omnetpp.

6 Annexe

6.1 Git

Tous le code de ce projet est disponible sur ce lien git :

`https://github.com/Azurlors/GhostVehicle`