Dynamic Defenses for Improved Resilience of Connected Cars

Maxime Ayrault

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAT032

Thèse de doctorat

Dynamic Defenses for Improved Resilience of Connected Cars

These de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom ParisTech

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique

These présentée et soutenue à Palaiseau, le 18 Octobre 2022, par

Ayrault Maxime

Composition du Jury :

M. Joaquin GARCIA ALFARO Professeur, Telecom-SudParis	Examinateur / Président
M. Youssef LAAROUCHI HDR, EDF Research	Rapporteur
Mme Emmanuelle ENCRENAZ HDR, Université Pierre et Marie Curie	Examinatrice
M. Jean-Luc DANGER Professeur, Telecom Paris	Examinateur
M. Yves ROUDIER Professeur, Université Côte d'Azur	Examinateur
M. Etienne BORDE Enseignant Chercheur, Telecom-Paris	Directeur de these
M. Ulrich KÜHNE Enseignant Chercheur, Telecom-Paris	Co-directeur de these
M. Benjamin VENELLE Docteur, Valeo	Invité

Dynamic Defenses for Improved Resilience of Connected Cars

Maxime Ayrault

Résumé

Avec l'adoption des voitures connectées dans la vie courante lors de ces dix dernières années, de nouvelles menaces de sécurité voient le jour impliquant de nouveaux traitements devant être traitées. La gravité de ces menaces va dépendre de deux facteurs principaux, La surface d'attaque et l'impact de l'exploitation des vulnérabilités. Il est possible d'observer une augmentation de la surface d'attaque avec l'utilisation croissante de composants électroniques pilotés par logiciel dans les véhicules (ECUs). L'augmentation de cette surface d'attaque est aussi majoritairement dû à l'ajoût de nouvelles interfaces permettant de relier les véhicules entre eux et au monde extérieur. L'électronique présente dans les véhicules permettent le control de plus en plus de fonctionnalités critique. Cela peut aller de fonctionnalités telles que le braking by wire ou d'assistance avancée à la conduite (Adaptative Cruise Control). L'impact de l'exploitation de vulnérabilités présentes sur ces composant électronique devient de plus en plus préoccupant pour les constructeurs au vu des problèmes de sûreté et de privacy que cela pourrait induire. Au cours des dernières années, le nombre de publications traitant de la découvertes de nouvelles vulnérabilités et attaques utilisant les connexions sans fil afin de prendre le contrôle d'une voiture ne fait qu'augmenter.

L'apparition grandissante de ces nouveaux vecteurs d'attaques combiné à l'explosion de la complexité des systèmes embarqués dans les véhicules amènent la sureté de fonctionnement et la (cyber)sécurité au premier plan des objectifs majeurs lors de la conception de nouveaux systèmes automobiles.

Le concept de résilience a du coup fait son apparition dans les différentes études sur les véhicules ainsi que dans la conception de nouveaux systèmes embarqués automobile. Ce terme de résilience fait référence par conception à l'objectif consistant à la sécurisation de l'architecture globale d'un système contrairement à l'introduction de correctif de sécurité locaux durant la durée de vie du produit. Cela inclut des mécanismes de défense tels que la détection d'intrusions ou bien encore une protection coordonnée contre les menaces existantes connues. Des approches comme les approches bio-inspirées utilisent par exemple la rémanence naturelle d'un organisme biologique comme modèle afin de pouvoir proposer des solutions techniques à ce défi de résilience. Un autre exemple d'approche liée à la résilience correspond au principe de ≪Moving Target Defense≫ consistant à la modification dynamique de la configuration d'un système lors de son exécution permettant de rendre les attaques déterministes moins efficaces contre le système défendu.

Lors du déroulement de cette Thèse, nous nous sommes particulièrement intéressé à

l'utilisation de ces techniques de Moving Target Defense dans les véhicules connectés. Ce type d'approche permet en effet de rendre plusieurs aspects d'un système dynamique. Le problèmes actuel liés à l'utilisation de telles techniques de défense dans un véhicule connecté est qu'il n'existe pas encore de prise en compte des contraintes liées aux systèmes embarqués critiques et que leur utilisation pourraient affecter la sûreté des utilisateurs. Il va donc falloir pouvoir garantir que leur ajout dans ce type de système ne provoquera pas de perturbations dans l'utilisation des applications critiques du véhicule.

L'utilisation de MTD est régît par trois grandes questions *quoi* faire bouger?, *comment* le faire bouger? et *quand* le faire bouger. Le traitement de ces deux premières questions *quoi* et *comment* faire bouger on déjà été adressées lors de différentes études existantes. Nous nous sommes donc particulièrement concentré sur la troisième *quand*, ce qui nous a permis d'arriver à la création d'un modèle permettant de calculer de manière automatique la fréquence d'utilisation optimale pour chaque technique de défense de Moving Target Defense présente sur un véhicule tout en prenant en compte les aspects de contraintes liées à l'utilisation de système embarqué critique.

Dynamic Defenses for Improved Resilience of Connected Cars

Maxime Ayrault

Abstract

With the adoption of connected cars in everyday life over the last ten years, new security threats have arisen involving new treatments needing to be dealt with. The severity of these threats will depend on two main factors: the attack surface and the impact of vulnerability exploitation. We can observe an ever increasing attack surface with the growing use of software-controlled electronic components in vehicles (ECUs). The increase of this attack surface is also mainly due to the addition of new interfaces linking vehicles to each other and to the outside world. The electronics present in vehicles enable more and more critical functions to be controlled. These may include functions such as "braking by wire" or advanced driver assistance (Adaptive Cruise Control). The impact of exploiting vulnerabilities in these electronic components is becoming increasingly worrying for automakers, given the safety and privacy issues this could involve. Over the last few years, the number of publications reporting the discovery of new vulnerabilities and attacks using wireless connections to take control of a car has grown steadily.

The growing emergence of these new attack vectors, combined with the explosion of on-board systems complexity in vehicles, has brought operational safety and (cyber)security to the forefront of major objectives when designing new automotive systems.

The concept of resilience has thus made its appearance in various vehicle studies and in the design of new automotive embedded systems. The term resilience refers by design to the objective of securing a system's overall architecture, as opposed to introducing local security patches during the product's lifetime. This includes defensive mechanisms such as intrusion detection or coordinated protection against known existing threats. Approaches such as bio-inspired approaches, for example, use the natural persistence of a biological organism as a model for proposing technical solutions to this resilience challenge. Another example of a resilience-based approach is the principle of "Moving Target Defense", which involves dynamically modifying a system's configuration during execution to make determinist attacks less effective against the defended system.

During the course of this thesis, we were particularly interested in the use of Moving Target Defense techniques in connected vehicles. This type of approach makes it possible to make several aspects of a system dynamic. The current problem with the use of such defense techniques in a connected vehicle is that the constraints associated with critical embedded systems have not yet been taken into account, and their use could affect user safety. We therefore need to be able to guarantee that their inclusion in this type of system will not disrupt the use of critical vehicle applications.

The use of MTDs is governed by three main questions : *what* to move, *how* to move it and *when* to move it. The treatment of these first two questions *what* and *how* to make it move have already been addressed in various existing studies. We therefore focused particularly on the third question, *when*, which enabled us to create a model for automatically calculating the optimum frequency of use for each Moving Target Defense technique present on a vehicle, while taking into account the constraints associated with the use of critical on-board systems.

Dedicace

Dédicace spéciale à mon Homer.

Remerciements

Je tiens à remercier tout ceux qui ont pu rendre ce projet possible et qui m'ont soutenu lors de ces années de Thèse. Je commence par mes deux directeurs de thèse qui ont eu la patience de me supporter et de m'aider lors de ces presques quatres années. Tous les gens de l'étage de qui j'ai pu recevoir des conseils techniques tels que Thomas, Nicolas (maître du café), Mounira, Florian ou Jean. Ainsi que tous les autres de qui j'ai pu recevoir des conseils autres que techniques comme Florian et nos conversations absudes mais toujours réfléchies, Petr et ces heures de répètes dans le studio de musique, Thomas avec les conversations sur tout et n'importe quoi, Laurent, notre ancien chef et actuel doyen, Ada et le jardinage, Dominique notre Monsieur Patate toujours la pour mettre de la bonne humeur, Chadi, notre grand chef à tous toujours présent et prêt à aider ainsi que Tarik l'homme à tout faire toujours la pour dépanner lui aussi. Mention spéciale à Mounira qui m'a souvent nourri et à James, sa femme et son fils pour toutes ces soirées après le travail pour décompresser. Merci aussi à tous les thésards, Zülal, Lylia, Felipe, Clement.. présent la plupart des jours avec qui j'ai pu passer de bons moment à l'école et en dehors. Au parents qui ont pu m'aider à arriver jusqu'au bout.

Bien sur c'est sans oublier mon ancienne collègue de bureau Docteur Hana avec qui on a partagé un bureau pendant 3 ans et qui a bien voulu supporter mes débuts à la guitare.

Et bien sur à ceux qui étaient la longtemps avant et qui le seront encore plus après, Sarah, Hono, Chuck, Cucus, Amélie, Bounj, Alex et Bibou, pour tous ces bon moments passés et à venir.

Et tout ceux que j'ai pu oublié car je suis pas doué dans cet exercice..

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Il est très important, pour celui qui souhaite découvrir, de ne pas limiter son esprit à un seul chapitre de la science mais plutôt de rester en contact avec plusieurs autres.

Jacques Hadamard.

In the 1960's and 1970's, when the Citroën 2CV was one of the most used cars, it was possible to repair them with a hammer and a wrench. This is not possible anymore on modern cars because of the technological advances that have led to the integration of Electronic Control Units (ECUs) and new functionalities in vehicles since the 1980's. The number of electric cables and computers has been increasing in such a way that on a current Twingo, half of its weight is only due to electronic equipment. At the slightest problem, it is now necessary to ask a car mechanic to make a computer diagnosis of the vehicle in order to locate and repair the problem.

The introduction of all these different calculators has resulted in a greater amount of driving comfort and safety for the driver, allowing him or her to drive while not having to focus on every aspect of the road and letting the vehicle take over parts of the driving tasks. These driving assisting systems can range from Lane Keeping Assists (LKA) to Adaptative Cruise Control (ACC), allowing the vehicle's speed to be automatically managed while adapting to surrounding vehicles. Some services such as the Automated Park Assist (APA), the rear and

front cameras or the Head-Up Display (HUD) have been added to connected vehicles to make car driving easier for users. While services as Electronic Stability Control (ESP), the Anti-lock Braking System (ABS) or the airbags increase the safety of the vehicle passengers by limiting the occurrence or the consequences of an accident.

The introduction of these new ECUs also involves the addition of connectivity in the vehicle, allowing it to communicate with vehicles and the infrastructure around it. This offers access to new services requiring an Internet connection. Vehicles are now becoming capable of communicating with other vehicles in the surrounding area to exchange information on traffic conditions, speed limits, road works, accidents, or detours on the road being used. Internet connection has enabled access to high definition GPS maps, Internet radios, remote system updates, or access streaming services for the passengers of the vehicle.

Unfortunately, there are some downsides in integrating connectivity and various driving assisting systems. Vehicles now have more access points and new vulnerabilities as well as new services, leading to various types of attacks and a bigger attack surface.

As an example of a successful attack, in 2015 two American researchers managed to take control of a Jeep Cherokee [25] remotely and make it stop on the highway after being able to change the information displayed on the dashboard. Or as another example, on Tesla, it has been demonstrated in 2017 that it was possible to send fraudulent messages on the Controller Area Network (CAN) bus through the Wi-Fi access of the car [45], or the creation of a key fob opening any Tesla which has been shown in 2021 [63].

The different car manufacturers are well aware of this new type of problem concerning their vehicles and are looking for ways to efficiently solve it, to make the use of the vehicle as safe as possible for the users.

The Connected Cars & Cyber-Security (C3S) Chair was born in this context, it gathers the following industrials: Nokia, Renault, Thales, Valeo, and Wavestone at Telecom-Paris around five research axes designed to improve the cyber security of connected vehicles. These five research axes cover topics such as risk analysis on vehicles, data protection thanks to cryptography, identity management, resilience by design, and user privacy. The work presented in this thesis is part of the 4^{th} axis of this Chair, “Resilience by design” which aims at making a vehicle capable to defend itself as long as possible against any type of attack, and ensuring that when the defenses are down the vehicle can return to a normal state of execution as quickly as possible with the least disruption.

We focused on the consequences of exploiting connected vehicle vulnerabilities through malicious attacks when such vulnerabilities have been exposed. Some of the ECUs are dedicated to the control of the vehicle's critical services such as acceleration, braking, or steering. This characterizes the connected vehicles as Critical Real-Time Embedded Systems (CRES) and implies that a fault on one of the ECUs managing one of the critical services of the vehicle can lead to serious damage to the driver. This is a consequence that needs to be avoided, so we have tried to identify the various entry points existing in the vehicle as well as the damage that can be done to the vehicle from those entry points. This analysis also serves to characterize the various types of attackers who can be interested in this type of system. Their objective can go from tracking the movement of a vehicle to intrude the privacy of its user, or blocking the wheels on the highway to provoke direct damages on the car and its occupants.

Existing defenses used in a vehicle are mostly static. That is to say that they are set up with a static configuration at the time of the introduction of the vehicle and then remain in place in the same way throughout its life. This leaves opportunities for an attacker to analyze the defenses in place to find potential ways around them.

In addition to static defenses, we decided to rely on dynamic defense techniques to improve vehicle resilience. Moving Target Defense (MTD) is a set of dynamic defenses that can be used to bring dynamics to the vehicle's configuration by dynamically changing certain characteristics of the system.

With MTD, the vehicle's characteristics can be made dynamic, for instance, the IP address of the vehicle, or the memory representation of a service. By adding dynamicity into the vehicle defense, an attacker will be slowed down when trying to launch an attack.

But with the use of MTD on a vehicle there will be new kinds of constraints to take care of that come from the fact that these defense methods have been developed for non critical real-time systems such as smartphones, PCs, or servers. One must be aware that a CRES will have different constraints to respect than other systems, such as limitations on the available computational power or the time constraints used to ensure the proper operation of the critical vehicle functions. The integration of new defense methods in a vehicle will have to go through safety certification.

By Looking at the state of the art, three questions need to be answered to use a new type of MTD in a system.

- The question *"What to move?"*, corresponding to finding the part of the system that we want to make dynamic and protect.

- The question ‘*How to move?*”, corresponding to what type of MTD to use to make dynamic the part of the system that we want to protect, and if this MTD is usable on the targeted system or if it requires an adaptation.
- The question *“When to move?”*, corresponding to the frequency at which the MTD will be used.

The questions *“What?”* and *“When?”* already have some answers, researchers have mainly focused on them. But most of the time, MTDs are used on general purpose systems, such as servers or personal computers, and the constraints based on the quality of service, computational power or time are not critical and will not lead to damage to users if not respected.

To determine the optimal strategy for using an MTD on a vehicle, we have identified game theory as a well adapted technique. Game theory can be used to represent through a mathematical model the different interactions between a vehicle and different types of attackers as well as all the constraints related to CRES. After the model has been designed, solving methods exist to determine the optimal solution for this game. In this thesis, we have been mainly addressing the issue of determining the optimal usage frequency of an MTD so as not to disrupt the nominal operation of the critical vehicle services.

In this thesis, we propose a method based on game theory taking as input the different parameters and characteristics allowing us to represent the gain, costs, and constraints of attackers and car manufacturers. Our method, and the prototype we developed, produce as an output an optimal strategy to use MTDs on a vehicle.

This thesis is divided into four chapters.

Our first chapter 2 presents the context, problem statement, and terminology. It begins with an elaboration of the terminology. Then we define the context in which this research takes place as well as the problem we address. We discuss in details connected vehicles, and Moving Target Defenses, as well as set motivating examples allowing to understand our approach.

In the second chapter 3, we present our first research based on the addition of an MTD in a vehicle that would slow down the discovery and the tracking of a vehicle in the network of the constructor.

In chapter 4, we present our method allowing, thanks to the game theory, to model the interactions between a vehicle and a multitude of types of attacker to determine the optimal defense strategy to apply to a vehicle.

This leads us to the final chapter 5 which concludes this thesis.

In the hope of being able to interest the curious reader...

Chapter 2

Background

Do or Do not, but there is no try.

- Master Yoda

Connected vehicles are becoming more and more present in our daily lives. Since the 1980's, their technological evolution has never stopped, up to the modern vehicle we know today providing a large number of driving assistanting systems. Those driving assisting systems emerged and were democratized through the continuous integration of ECUs into vehicles.

Vehicles are also increasingly being equipped with new electronics, and connected to the outside world via the Internet. This makes them more suitable for new forms of attacks, such as trying to immobilize the vehicle before it starts, causing problems to its integrity in order to injure its passengers, or trying to access information on the private life of users.

To present the problematic of this thesis, we start by reminding the different terms necessary to understand the subject as well as a definition of Resilience before talking in depth about the different elements composing a vehicle and the methods of attacks and defenses.

2.1 Terms Definition

2.1.1 Cyber-Security Terms

Cyber-security is build on three pillars: the CIA triad. The security goal is to provide measures to achieve confidentiality, integrity, and availability (CIA) for

protection of the overall system along with its peripherals. The terms are defined as follows:

- *Confidentiality*: The aim of confidentiality is to protect critical information from unauthorized users. Confidentiality for network security ensures that the critical assets are accessible only to authorized users.

- *Integrity*: This ensures that unauthorized users do not modify or manipulate the data or information when they transit on the network or they being stored.

- *Availability*: Availability aims to ensure that data and network resources can be accessed when requested by authorized users.

Besides the CIA triad, Identification, Authentication, Authorization, Auditing and Accounting (called IAAAA) also play an important role for controlling the access to the system resources. The IAAAA is a term for controlling the access to the system resources, auditing usage, and enforcing policies.

- *Identification*: Identification aims to ensure that a user is really who he claims to be when he attempts access a resource. Providing an identity can involve typing or sending a username or an ID, swiping a smart card, waving a proximity device .... Without an identity the system has no way to correlate an authentication factor to the subject.

- *Authentication*: Authentication is about proving the user identity. It requires the subject to provide additional information that corresponds to the identity that is claimed. The most common form is to provide a password. Authentication verifies the identity of the subject by comparing one or more factors against the database of valid identities.

- *Authorization*: Authorization is defining the permissions of a resource/object access for a specific identity. It ensures that the access to a resource/object is given the right and privileges assigned to the authenticated identity. If the requested action is allowed, the access is authorized, otherwise the access is denied. It is not because a subject is correctly authenticated that he/she has the right to perform any actions on any resource.

- *Auditing*: Auditing is recording a log of events and activities related to the system, subjects and objects. It aims at tracking and recording all requests and actions. Log files provide an audit trail for re-creating the history of

events. It permits to detect malicious actions, system failures but also system performances problems.

- *Accounting*: Accountability aims at reviewing the log files to check for compliance or violation of security policy in order to hold the subject accountable of his actions. It is also a way of evaluating what services have been used and how many resources have been consumed.

The main techniques used to guarantee these five properties are:

- *Encryption*: The transformation of the original information (called clear) using an encryption key, so that the transformed information (called cipher) can only be interpreted by another user with knowledge of the decryption key (which may, in some cases, be identical to the encryption key). To be secure, an encryption algorithm must make it extremely difficult for someone to decipher all or part of the clear information without knowledge of the decryption key or weakness in the encryption algorithm.

- *Access Control*: A set of security rules and policies that limit access to information to subjects (people and/or systems) with a "need to know". This need to know is determined following the correct authentication of the subject, by his identity or role. A set of rules is predefined by the manager of computer security based on the security policy.

- *Signature*: The signature of an information has two objectives; to ensure that the signed information has not been altered since the signature and to authenticate the source of the information (non-repudiation). The signature consists of a cipher depending on a secret key known only by the subject signing the information and the content of the message to be signed. A signature is verifiable, without knowing the secret key, by a third party in case of dispute between parties. If the content of the message or the signature is modified, the correspondence between the initial content of the message and its signature will be invalid, allowing the alteration to be detected and the information to be rejected. Symmetrically, if the content of the message and its signature match then the source of the information will not be able to deny having signed the information because it is the only one to know the secret key.

- *Accountability*: The ability to make the subject responsible for these actions. This is achieved by relying on audit logs. Once the subject is properly authenticated, all these actions are recorded as events in an audit log. In

the event of an investigation or a periodic fault, the IT security manager can perform an audit of the logs to identify the potential source of an attack.

- *Security Awareness*: The main source of risk for an organization does not come from weaknesses in the technology of the equipment but from actions (or inaction) on the part of the users of the system. In order to limit this risk, it is necessary to train the users to the various computer risks and to the good practices to ensure the required level of security.

- *Physical Security*: Implementation of physical barriers to limit access to sensitive resources. These barriers include such things as guarding, video surveillance, locks on cabinets and doors, vaults, the use of soundproofing materials, or even the construction of hardened equipment (tempest) so that electromagnetic signals cannot enter or exit.

- *Fault Tolerance*: A set of techniques that can be used to guarantee the system in operation.

	Confidentiality	Integrity	Availability
Encryption	✓		
Authentication	✓	✓	
Access Control	✓	✓	
Signature		✓	
Accountability	✓	✓	
Security awareness	✓	✓	✓
Physical Security	✓	✓	✓
Fault Tolerance			✓

Table 2.1: Main security techniques

2.1.2 Resilience

Resilience is a term that has been used extensively in the fields of psychology [39, 52], economics [10, 55], environment [46, 28] and in the medical field [30]. In each of these domains a definition has appeared, giving a meaning for this term, yet in the computer field a clear and precise definition has not been established. By merging different definitions we have encountered, we have come up with the following definition:

Resilience : Being able to defend against an attack as long as possible, and once the defenses have fallen, being able to recover to nominal operation as quickly as possible.

In a connected car, the highest resilience is needed. This resilience is here to improve the *Security* of the car, but without any impact on the *Safety* and limited impacts on the overall performance.

Techniques to improve resilience can be divided into six categories:

- *Redundancy*: Using hardware or software replication in a system, allowing to switch from one instance to another in case of failure. There is several types of redundancy; hot redundancy, cold redundancy and a hot redundancy that can be declined in 1 out-of 2 (the simplest one), to m out-of- n (to mix safety and redundancy).
- *Obfuscation*: Used to hide system information from an attacker.
- *Cryptography*: Special use of obfuscation in a mathematical way allowing to encrypt the data in a way that the deciphering is extremely complicated.
- *Monitoring*: Checking the integrity of the system with an external system observing at the system operation. May be done with redundancy.
- *Authentication*: The insertion of data and/or structures to verify the integrity or provenance of code, data or hardware.
- *Isolation*: Divide functionality physically or logically and controlling interfaces to limit a system's attack surface, like TrustZone [50] in ARM Central Processing Unit (CPU).

Three of the five categories: separation/isolation, obfuscation, and authentication are for the most part, used to secure a system, i.e., to maintain consistency between the perceived system functionality and the actual system functionality.

The other two: redundancy and monitoring are employed when the operating parameters exceed the ability of the security techniques to guarantee integrity of the environment.

2.2 Connected Cars

2.2.1 The Software Revolution

In the automotive domain, there is a clear trend toward increased connectivity: Many new services are available today that demand reliable communication between the car and its owner (e.g. via smartphone applications), the manufacturer, or even the infrastructure. Many cars offer a local WiFi network and Bluetooth. Furthermore, semi-autonomous driving – such as parking and lane keeping assistants or automatic emergency braking – are now standard in many vehicle models. These functionalities require a large number of sensors such as cameras, radars, or other distance sensors. While these services are supposed to increase the safety of the passengers (and of pedestrians), they entail new security risks. Indeed, many attacks have been demonstrated in recent years, some of them putting the life of the car's passengers at risk [56, 25, 41].

The security of connected cars is taken very seriously by the manufacturers and it has become a major design goal. For instance, in the system architecture of our case study, the subsystems are divided into different domains, and any message crossing a domain border passes through a secured gateway. This allows to filter unauthorized messages and prevent an attacker who has compromised one subsystem from spreading to other – more critical – subsystems. However, such protections are of a *static* nature. The defenses are programmed and configured once before the car leaves the factory. Because of strict certification requirements, software updates are very costly and usually must be performed by licensed workshops. This makes the relation between potential attackers and the system under attack an *asymmetric* one: The attacker can analyze the car's system to find vulnerabilities and prepare an exploit that can then be applied – potentially to a whole fleet of cars in parallel. On the other hand, car manufacturers cannot easily analyze attackers' behavior.

Embedded software is one of the key innovations in the automotive world. According to an article of Robert Charette ([15]), the first car embedding software was the Oldsmobile Toronado from General Motors in 1977. The Toronado enclosed an Electronic Control Unit (ECU) which managed the spark timing. In 1978, General Motors offered on the Cadillac Seville as an option a trip computer able to display the speed, the fuel level, trip and engine information. This product was based on an embedded version of a Motorola 6802 and had about 50,000 lines of code. Since then, more and more functions are performed by software in a car. To limit the cost of cables in a car, all the sensors and on-board units are now connected to a network backbone using CAN or Ethernet network. As the computing power of the processors grows, new functions appear. Cars can be

Airbag	ABS	Anti-thief system
Air Conditioning	Speed control	motor management
Turn signals	Headlights	Klaxon
Seat management	Navigation system	Audio system
Wheel pressure	management of the doors and windows	...

Table 2.2: Software function in a car

now considered as on-wheel software platforms like airplanes or trains. A modern family car embeds between 30 and 50 ECUs managing the multiple systems (see 2.2). A premium car can have up to 3,000 functions performed by software.

In 2009, Alfred Katzenbach, Director of Information Technology Management at Daimler announced that the radio and navigation management system of an S-class Mercedes-Benz contained over 20 millions line of code and the car embedded nearly as many ECUs as an Airbus A380 (excluding the in-flight entertainment system) ([15]). Software in a car has shown exponential growth in size and complexity. In just 10 years, the software volume in a car has expanded by a factor of 10, to arrive at around 150 million lines of code. A model S from Tesla is equipped with a 17' tactical display based on a Linux kernel which controls almost every driver function. In fact, there are only 2 manual buttons that are not managed by software in the car, the blinker, and the glove compartment.

A side effect of this revolution is the complexity and the richness of the functions proposed to the driver. It is frequent to have the driver's manual of a car with over 500 pages to explain all the functions. Automotive experts estimate that an average driver uses not more than 20% of those functions. This is why some automotive manufacturers are thinking about a new type of internal architecture that can help to reduce the number of ECU in a car.

The growth of software in a car has also important consequences on the way to maintain and repair a car. An estimation gives that more than 50% of the ECU that are changed by a car mechanic have no software or hardware failure. Mechanics frequently replace pieces without the knowledge of the root cause of the issue. Now, one of the most frequent activities of a car mechanic is to download and upgrade a new version of the software. Some car manufacturers, like Tesla, propose to download software updates, including software corrections (patches) or new functionalities, using the cellular network without any involvement of a car mechanic.

Today, the cost of the software and the supporting electronics in a car is estimated between 35% and 40% of the cost of a car[60]. Investment to develop new software platforms became so expensive that the main European Automotive manufacturers have developed a series of standards to reduce the development

cost:

- a common platform; *Automotive Open System Architecture (AUTOSAR)* [7] allowing suppliers to develop a single platform interoperable between several manufacturers.
- a software development framework for safety; *Road vehicles, Functional Safety package* ISO 26262 [57]
- and recently a standard to address the cybersecurity management system; *Road vehicles - Cybersecurity engineering* ISO 21434 [58].

The introduction of software in the car permits to have safer and less polluting cars but it has opened the door to two new risks the *software safety risk* and the *cyber-security threat.*

2.2.2 Car Architecture Over Time

In this section, we are focusing on the evolution of the network security aspect over time. As for all electronic systems, car systems follow Moore's law going from generic 8 bits processors to car specialized Digital Signal Controllers embedding all functions on one chip. In the course of time and technological evolution, the internal architecture of vehicles has moved from a perimeter defense by routing rules between the different elements of the vehicle to a defense in depth introducing domain controllers, adding management of the quality of service, and introducing connectivity methods to external infrastructures. The following timeline of different architectures has been established during meetings with our industrial partners.

In the 1980's, the electronic system present in most cars is quite basic, it is composed of a dozen ECUs performing simple tasks: Figure 2.1 represents this architecture. The defense of the vehicle is done thanks to gateways dividing the vehicle into several zones, front, middle, and rear. The gateways present on the vehicle are used to route the various messages circulating on the vehicular network. The controls being mainly mechanical, possible attacks on these vehicles are limited, so the defenses were designed accordingly.

In the 1990's, electronics embedded in vehicles becomes a little more developed but does not allow yet to control critical functions of the vehicle. A perimeter defense as presented in figure 2.2 is still enough to protect the vehicle efficiently. The main difference with an architecture of the 1980's is the addition of the quality of service management on the internal network of the vehicle, which was not taken into account until then. The different gateways that were present are now grouped

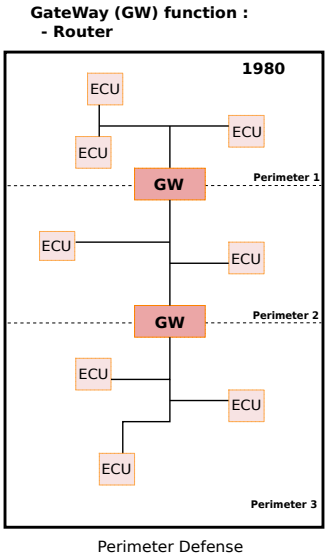

Figure 2.1: Architecture 1980

in a central gateway that connects the ECUs together.

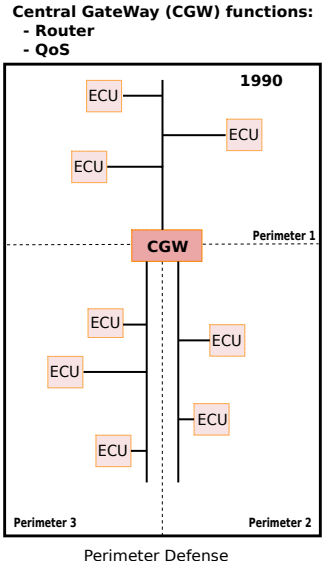

Figure 2.2: Architecture 1990

From the year 2000, with new functions allowing to control critical elements of a vehicle such as brakes, steering, or acceleration, potential attacks on a vehicle become a new source of hazards for the driver and passengers. The car suppliers propose a new architecture designed to address those threats, as presented in Fig-

ure 2.3. The main difference to the previous architecture is the shift to a defense in depth in which each domain is connected to a Domain Controller managing the incoming and outgoing communication of this domain as well as the communication inside the domain. Communications between domains pass through a Secure Gateway, which manages the routing and the quality of service of the network while also acting as a firewall to limit fraudulent or malicious messages circulating on the network.

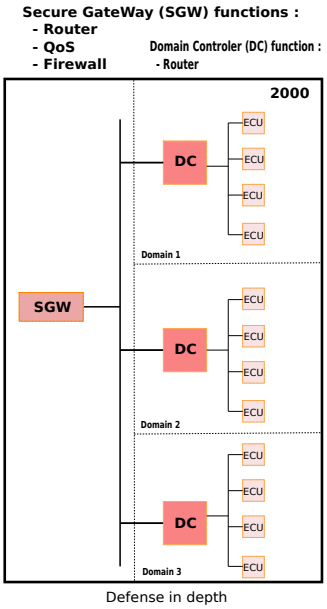

Figure 2.3: Architecture 2000

The 2010's have seen the emergence of wireless communication in vehicles, providing a new entry point for larger attacks that can directly affect a fleet of vehicles. Driving assistants are becoming more and more common, making mandatory the addition of defense methods in vehicles. Figure 2.4 represents this type of architecture. The network configuration remains essentially the same with a defense in depth allowing to isolate each domain and control messages circulating inside each domain based on a secure gateway at the core of the network. As the technologies evolve, secure gateway, as well as domain controllers, need greater processing power.

The architecture used in Tesla vehicles is a little different from the architectures commonly used until now, as shown in Figure 2.5. The Tesla architecture is based on virtualization and interoperability. At the center of the architecture is a large calculator allowing to manage the network and a part of the computation to be

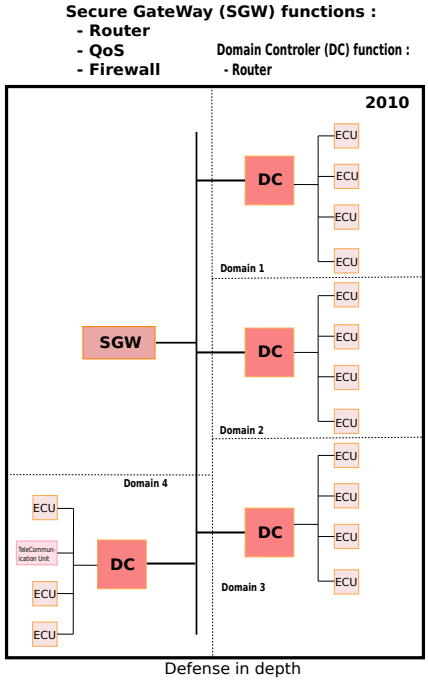

Figure 2.4: Architecture 2010

realized. A grid of four ECUs connected to each other and to the central computer are used to perform the computation of the different applications of the car.

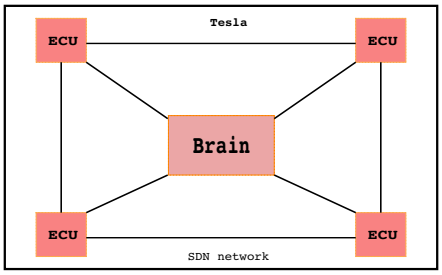

Figure 2.5: Tesla Architecture

2.2.3 Modern Car Software Architecture

AUTOSAR

AUTOSAR was founded in 2003, with the goal to develop an architecture, independent of the underlying ECU hardware that the automotive industry can use to reduce the increasing complexity of software in modern vehicles ([7]). This is

the de facto standard for the automotive software today. AUTOSAR provides an abstract layer of the underlying hardware, so that the applications written on top of AUTOSAR are independent from the actual supplier of the ECU hardware. The AUTOSAR standard documentation guides companies and the automotive industry in designing and implementing software in their vehicles. Besides the software architecture and the standardized Application Programming Interface (API), AUTOSAR provides a software development methodology. By adopting the AUTOSAR standard, companies can develop software solutions that are independent of the hardware they are running on, and this software can run on any ECU in the vehicle.

AUTOSAR is a three-layered architecture ([6]):

- the *application layer* provided by the software company implementing the specific functions of the ECU. This is the highest layer which contains the Software Component (SWC). AUTOSAR application (e.g., ABS, Cruise Control ...) consists of several SWC, which provide the core functions.
- the *Run-time Environment (RTE) layer*. The RTE layer provides the standardized interface between the SWCs and the basic software layer.
- the *Basic Software (BSW) layer* that consists of four sub-layers; the *services layer* providing operating system functions like communication services, memory services, diagnostic services, ... The *ECU abstraction layer* makes higher software layers independent of ECU hardware layout. The *Microcontroller abstraction layer*. It contains drivers for direct access to the underlying microcontroller and internal parameters. It makes higher layers independent of the microcontroller. The *complex drivers layer* provides the ability to integrate special-purpose functions such as drivers for devices that are not specified with the AUTOSAR standard. This layer accesses directly the microcontroller.

The AUTOSAR standard defines security mechanisms that can be used by the software modules implemented in the vehicle system. It further specifies interfaces and procedures to provide Secure On-Board Communication, and the exact implementation is left to the Original Equipment Manufacturer (OEM). OEMs choose the cryptographic algorithms and encryption techniques they want to implement and use in the vehicle system.

AUTOSAR has been mainly built to ensure a high level of safety of the system. Many safety mitigations requested by the standard can be attack vectors. For example (see [3]), to protect the internal and external communication, AUTOSAR defines the End to End (E2E) library which defines the safety mitigations

to prevent safety critical functions from operating on faulty or missing data. The mitigations are:

- Cyclic Redundant Checksum (CRC) check to detect corruption of data
- Sequence counter to detect out of sequence messages and to sort the messages
- Alive counter to prevent operating on old data
- A unique ID for Interaction Layer Protocol Data Unit (I-PDU) group to detect a fault of sending I-PDUon unintended message
- Timeout monitoring to detect communication loss with the sender.

Attack based on those mitigations will lead to a rejection of the messages and a potential Denial of Service (DoS). No mechanism has been provided to detect the difference between a non-malicious errors in the content of a message and an attack.

2.2.4 Case-Study

After several meetings with the industrial partners to determine a vehicle architecture on which to base our research, we finally agreed on the architecture presented in Figure 2.6[1]. It corresponds to an architectures used in modern vehicles, in which each service is grouped with other similar services to obtain an architecture by domains. It uses a defense in depth, supported by the use of a central element called *Bastion*, which has the same role as a Secure Gateway, with the added benefit of managing the defense mechanisms that are present on the vehicle.

This architecture is called *architecture by domain* because the different services of the vehicle are separated into four domains according to their role:

1. **Infotainment**: services related to the user experience such as the radio, the on-board screen, and various applications accessible to the user.
2. **Core Services**: critical services for the operation of the vehicle such as cruise control, brake controls, or lane keeping assistant.
3. **Management**: services dedicated to diagnostics and updates of the vehicle, such as the On-Board Diagnostic-II (OBD-II) plug.
4. **Shared Services**: regroups services allowing to communicate with the outside world (V2X) as well as some services shared among several domains.

[1] In the appendix B, a more complete version is presented in Figure B.1

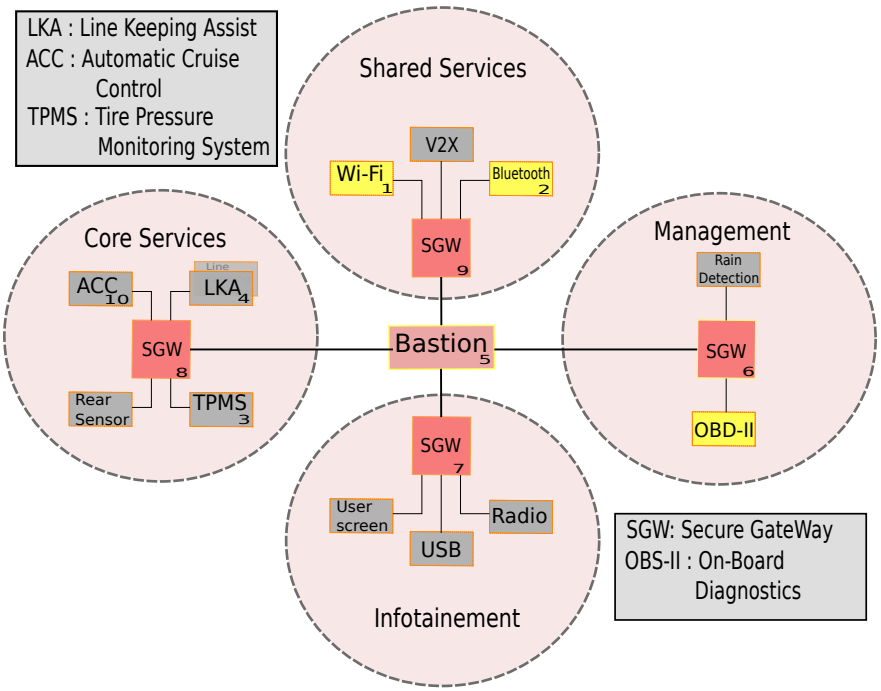

Figure 2.6: Car architecture we defined

Within each of these domains, the communication is managed by a Secure Gateway serving as a router and firewall. For example, in the Infotainment domain, if the USB module wants to send a message to the user screen module, those messages must go through the domain's Secure Gateway.

All the communication between two separate domains must pass through an entity called the *Bastion*, operating as a *"super" Secure Gateway*. If the Bluetooth module located on the Shared services domain wants to communicate with the user screen module located on the Infotainment domain, all the messages will be examined and filtered by the Bastion. The *Bastion* is a router and a firewall for the communication between the domains. Typically, it also embeds an IDS *(Intrusion Detection System)* to detect any attempt of intrusion into the vehicle's system.

2.3 Existing Attacks and Defenses

For several years now, the produced cars have been equipped with more and more driving assisting systems. These systems often have access to the control of critical functions of the car, which if misused or bypassed can affect the safety of the user. These vehicles can also communicate with internet services as well as other vehicles on the road and the various infrastructures they may encounter. This added connectivity introduces new access points [16] allowing new remote attacks that can impact the integrity of the vehicle as well as the privacy of its users. These attacks can be performed at three different levels: with physical access to the vehicle, by being close to the vehicle, or remotely through the internet. This is what we will present in the following subsections.

To this day, defenses are already present on the vehicle to try to prevent an attacker to succeed in his attack. But they are mainly static defenses, having a configuration at the time of the production of the vehicle remaining the same until the software update of this vehicle.

It is easily possible for an attacker to acquire a vehicle and to analyze the defenses in place to develop an attack kit allowing to get a successful attack in an automatic way. The manufacturer can try to implement updates to its system, which will have to pass through several certifications that do not allow for daily updates.

2.3.1 Existing Attacks

There are several known attacks on vehicles, each of which has specific goals and requires a specific context to be carried out.

Physical Attacks

Physical attacks will aim to directly impact the functioning of a vehicle by getting access to the vehicle. These attacks can be performed through the OBD-II socket of the vehicle, or the USB port of the vehicle. Some attacks will aim to cause physical damage to the vehicle. Figure 2.7 shows the attacker entry points on a vehicle architecture for physical attacks.

- **OBD-II** : Attack via the car's OBD-II diagnostic socket [32, 62]. From the OBD-II socket, it is possible to perform those two examples of attacks. One aiming to modify the configuration of the vehicle to unlock new functions. And one aiming at disturbing the vehicle operation.
  - The first objective of going through the diagnostic plug is to be able to modify the car characteristics and/or to add options to the car. The

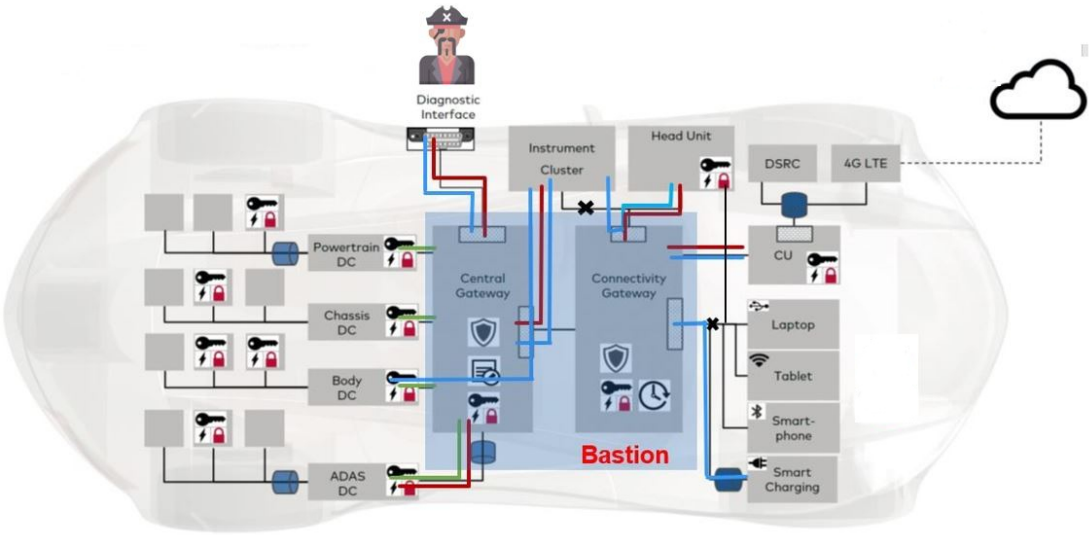

Figure 2.7: Physical attack entry points

software of a car is highly configurable. The same software is installed on several ranges of vehicles of a manufacturer, which only differ regarding the software parameters. If an attacker has the possibility to modify these parameters, he can activate optional functions of the car or modify the engine characteristics to boost the vehicle.

- The second objective is to access the CAN bus of the vehicle via this diagnostic socket. This allows an attacker to send fraudulent messages on this CAN bus to disrupt or crash the vehicle's functionalities.

- **USB port**: Attack via the car's USB port [56]. From the USB port of a vehicle, it is possible to install fraudulent software on the vehicle such as malware to track the user or ransomware blocking the use of a vehicle until payment of a ransom. The use of USB ports allows the creation of an entry point into the vehicle to leave backdoors open to short or long range attacks, as well as the installation of worms, which might spread to surrounding vehicles.

- **Physical Damages** : Direct physical attacks on the vehicle [40]. It is also possible to physically access the vehicle to damage its integrity by cutting wires that can be used to control the brakes or the steering, directly impacting the safety of the user. It is not possible to introduce software defenses in the vehicle to prevent this kind of physical attack.

Short Range Attacks

Short range attacks are designed to communicate with vehicles within a short perimeter. The main objectives are to infect the surrounding vehicles, send false information, block communication to the surrounding vehicles, or track the vehicles present. These attacks are mainly carried out via the vehicle's Bluetooth, Wi-Fi, and Tire Pressure Monitor System (TPMS). Figure 2.9 shows the attacker entry points on a vehicle architecture for short range attacks.

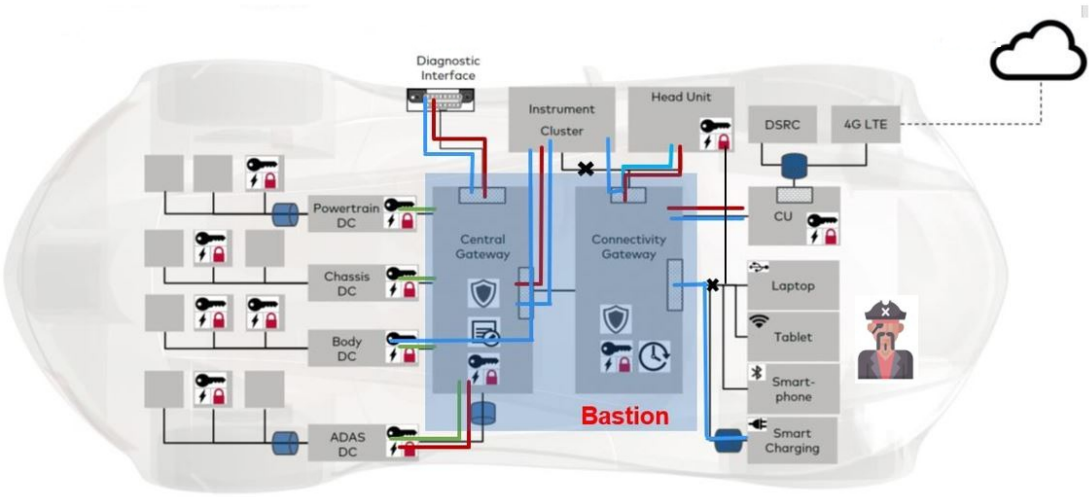

Figure 2.8: Short range attack entry points

- **Bluetooth Attacks**: With the Bluetooth module, it is possible to carry out attacks allowing to have for objective to invade the user's privacy. Indeed, it is possible to recover the MAC address of the Bluetooth module of a vehicle to follow the movements of a user.

  From the Bluetooth module, it is also possible to have a new entry point allowing to attack the integrity of the vehicle, if one manages to recover the password.

- **Wi-Fi Attacks**: Connected vehicles on the road can also communicate with each other. If one of them has been infected, it is possible to use this connection between vehicles to spread worms to surrounding vehicles [56]. These worms can have different purposes, such as collecting privacy data from all infected users, installing malware on the infected systems, or opening and leaving open backdoors in the vehicles.

The information available in clear text in the Wi-Fi frames can identify the version of the system used and the different network elements. This allows an attacker to establish a database of different vehicles and their configurations to find the most suitable method to launch an attack on a specific vehicle.

- **TPMS Attacks**: The Tire Pressure Monitor system is a radio device that allows communication between a tire and a vehicle to keep the vehicle informed of the tire's status. This device can be used against the privacy of the user and the integrity of the vehicle.

  For the privacy side, each TPMS has an identifier accessible in clear text within the frame sent to the vehicle. This allows to keep a database of the comings and goings of a vehicle in a location.

  The TPMS can also be used to interfere with the integrity of the system [56]. It is possible to send false frames to the vehicle continuously to overload the vehicle's receiver and make it crash. This will trigger a light on the dashboard indicating a problem with a tire. This action can be used to force a user to pull over to the side of the road to steal the vehicle. Or to force the user to go to a garage to restart the car and remove the warning light.

- **Interference Attacks**: The use of vehicle related frequencies can also be used to steal and start a vehicle [63]. With the use of radio equipment on the key of the vehicle, allowing to start it remotely, it is possible to duplicate the frequency of a key once at a short range to scan it. The attacker can come back later to open and start the vehicle with a radio equipment allowing to reproduce the scanned frequency.

  In the same way, the use of a frequency scrambler allows preventing the locking of a vehicle once the frequency of the key of the vehicle is identified, allowing to enter the vehicle, potentially to realize one of the physical attacks presented above.

Long Range Attacks

Long range attacks are designed to take over or follow the movements of a vehicle remotely via Internet. Indeed, connected vehicles have 4G and 5G antennas allowing them to access online services, which introduces a new entry point. These attacks will be performed mainly through Wi-Fi, because the Bluetooth or the car's ports do not have a sufficiently long range. Figure 2.9 shows the attacker entry points on a vehicle architecture for long range attacks.

From the **Wi-Fi**, it is possible to take remote control of a vehicle [25]. This has been done and presented in [42], in which the authors have indeed managed to take control of a Jeep Cherokee remotely via Internet and have had control of a

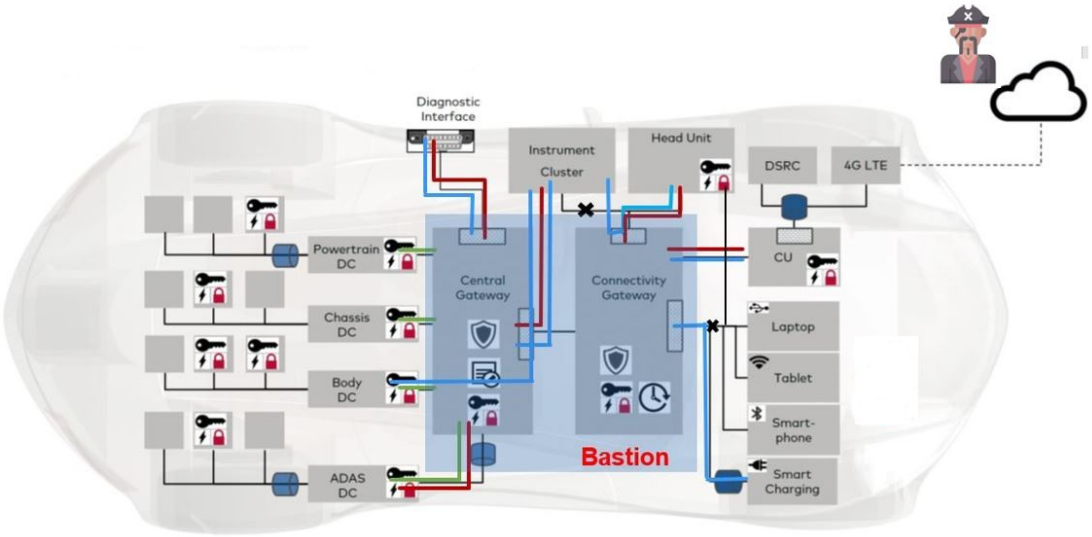

Figure 2.9: Long range attack entry points

car's inside services such as air conditioning, the screen of the vehicle or the radio. Then they managed also to take control of the critical elements of the vehicle to make it stop remotely.

Cars from the same manufacturer are connected to the same manufacturer subnetwork, they are often protected by the same type of defense. If an attacker succeeds in taking control of one car, he might succeed to do so with an entire fleet of vehicles, raising security issues for the users of these vehicles.

It is also possible to attack the privacy of a user remotely. In fact, the data corresponding to its location or the different trips made are recoverable remotely. This allows tracing the movements of the users.

2.3.2 Defenses Deployed In A Connected Vehicle

Connected vehicles are fortunately already protected by defense mechanisms. These defenses allow slowing down an attacker at first, but once a flaw is found, the car is defenseless. As said before, it is complicated to regularly update the system of a vehicle knowing the certifications that it must undergo. The defenses currently in place are therefore said to be static, the main ones corresponding to the attacks mentioned in the previous subsection are the following:

- **Can Bus**: On the CAN bus, every message is broadcasted and can be seen by anybody connected to that bus. In the early days, every one was able to

post a message on the CAN bus, without any restriction, and was able to see every message transiting on the bus.

Now manufacturers have added light-weight encryption of the messages and an access code depending on the manufacturer. However, the use of encryption is limited by the low computing power of most of the ECU connected to the bus.

- **Bluetooth**:

  On the Bluetooth module, the defenses used are the same as on usual devices, a password to get access. But once the password is discovered, the attacker has the ability to connect to it very easily.

- **Wi-Fi**:

  For the Wi-Fi, the defenses are the same as in a domestic Wi-Fi, a password to connect. But as for the Bluetooth, when the password is retrieved, an attacker can connect to the vehicle.

- **TPMS**:

  For the *tpms*, no defense has been used. The ECU used for the TPMS is not powerful enough to embed cryptography or other types of defense.

- **Internal Network**:

  In the internal network of the car, some defenses are used to prevent attackers that have gained access to an entry point of the car from advancing further into the internal network. Firewalls are used to filter messages sent on the network between applications separated into multiple domains. All of those defenses help to protect a car from a potential attacker.

2.4 Moving Target Defense

The principle of **Moving Target Defense (MTD)** as defined in [47] [48] [34] is to break the asymmetrical relationship between defender and attacker by giving the advantage of time to the defender. This is done by dynamically reconfiguring different properties of the system. It is necessary, however, that the reconfiguration period be smaller than the time which the attacker needs to realize his attack. Otherwise, this technique is ineffective.

2.4.1 MTD Categories

There are five categories of moving target defense :

- *Dynamic Data*. Change the format of data representation.
- *Dynamic Software*. Change the code of the application.
- *Dynamic Runtime Environment*. Change the execution environment.
- *Dynamic Platform*. Change the properties of the platform.
- *Dynamic Network*. Change the configuration and properties of the network.

Dynamic Data

This MTD method consists in changing the data representation format in memory to make the reading and decoding of different data stored in memory more complicated because there is no consistency. Figure 2.10 shows two different representation formats for the same data.

This is a hard to implement method in a real system, and it has been mostly used for research purposes, since using non-uniform data in a file or database make the reading not only hard for an attacker but also for the user.

```
Age :24
Gender : Male
ID : 443
```

(a) Format 1

```
ID : 443
Age :24
Gender : Male
```

(b) Format 2

Figure 2.10: Different data representation

Dynamic Software

The *Dynamic Software* method consists in having multiple assembly versions of the same high level code to make attacks by fault injection or branching predictions more complicated. By adding this form of code redundancy, the attacker is unable to predict which binary code is running on the victim system. As an example, Figure 2.11 shows two equivalent ARM assembly codes.

This type of method is complicated to integrate in practice, because it is necessary to have the source code of the application available, to be able to compile it with different options and/or compilers. This option also raises performance issues, since a software is usually compiled with options to get the best performance.

```
add r5, 0, 0
add r3, 0, 3
lsl r5, r3
```

(a) Code 1

```
xor r5, r5, r5
xor r3, r3, r3
add r3, r3, 8
mul r5, r5, r3
```

(b) Code 2

Figure 2.11: two equivalent assembly code

Dynamic Runtime Environment

The *Dynamic Runtime Environment* is a category of methods that consist in modifying the execution environment of the application in order to prevent the attacker from directly accessing certain memory zones or to perform easy Return-oriented Programmin (RoP) attacks. This category encompasses two main methods : Address Space Layout Randomization (ASLR) and Instruction Set Randomization (ISR). *ASLR* is the most used method in real systems, it is implemented in all the mainstream Operating System (OS) like Windows, Linux, Android, or MacOS.

ASLR is used to randomize the virtual memory layout of a program at the time of its execution. This can change the base address of different segments, stacks, heaps, shared libraries... Usage of *ALSR* will make all those addresses more difficult to determine. A variant called Kernel Address Space Layout Randomization (KASLR) can be used to randomize the addresses of different OS segments. However, this method is not very effective on small memory or address space due to a small entropy, which will lead to a shorter time to break it by brute force attacks.

The difference between ASLR, KASLR and Kernel Address Randomization Link (KARL) have been discussed in [20]. We already spoke about *ASLR*, so we will focus on *KASLR* and *KARL*. In **KASLR**, we randomize the kernel code location when the system boots. It was introduced in the Linux kernel in 2014, and has been enabled by default since 2017. But its effectiveness is questioned because, until the next system reboot, there will be no other new random distribution in the memory.

KARL [1] has been recently released in OpenBSD, and differs from ASLR. With KARL, the kernel is still located in the same addresses as the KVA (Kernel Virtual Address Space), but every time the system is rebooted or updated, the kernel binary files are randomized. So each time the system is booted, we have a unique kernel different from other systems at a binary level. In [2] we can see the difference between the BSD kernel code with and without the Karl system. Using this type of method to randomize an entire BDS kernel takes around 1 second, so for a small embedded OS, it will take even less time.

The *ISR* randomizes the current instruction set of an application. For example, loading a new instruction set when loading the program will make it more difficult for an attacker to exploit memory vulnerabilities. With ISR, code injected by attackers will not be properly encoded but still go through the decoding process. This leads to illegal CPU instructions and exceptions.

There is also the System Call Number Randomization (SCNR) that can be used for Dynamic Runtime Environment MTD. The principle of this technique is related to the ISR, here we randomize the number of the System Call instead of the instruction. So when there is code injection the system call leads to an error or a different result as expected.

Dynamic Platform

This method consists in changing the properties of the platform on which the application is running to prevent attacks relying on a specific architecture. This technique is based on the change, for example, of the OS, the processor architecture, virtual machine instance, file system, communications... It also proposes migrating an application from one platform to another to avoid persistent attacks. MTD like temporal changes (virtual machine rotation) or diversity (multiple variant execution) can be really useful to protect systems like severs. As an example, used in a server context, having multiple redundant servers with different software running on each of them will increase the availability of the server and will make it more difficult to attack.

Dynamic Network

This method consists in changing the configuration and properties associated with the network of the platform. Examples include regularly changing the IP addresses, the open ports of communication as well as the topology of the network.

With IP randomization, the IP address is changed periodically and the communication of the new address is only for an authorized user. This will reduce the attacker's capabilities to scan or exploit the system. Optionally, if the system

receives some request from a non authorized user, it can return fake information, like the OS or server version.

We can combine the IP shuffling technique with *Honey Pots* dynamically changing location, to confuse the attacker. *Honey Pots* generally consists in letting some data, ports, or anything open that appears legitimate to the attacker, but is actually isolated and monitored, and seems to contain valuable information. So the attacker has the impression to be in the system and having access to real data, but in fact wastes time on fake information. While moving dynamically the honey pot, the attacker can never know if the attacked system will be a honey pot or his actual target.

2.4.2 Measuring The Effectiveness of MTD

Attack based Experiments, *Probability Model* and *Simulation-based Evaluation* are the three main groups of methods used to measure the efficiency of an MTD.

- *Attack-based Experiments* are used to evaluate an MTD method and how difficult is it to compromise a program. It can be used directly on a system with some MTD methods running. With this kind of approach, we can measure the time needed to access a system with MTD protection and without. With this measure, it is possible to conclude if used MTD is effective against the tested attacks and if it produces better results than the same system running without MTD.

- *Probability Models* are used to analyze the attack success probability when MTD are used in a system. In this probability model, we abstract the specificities of the system, the attacks and the defenses using probabilities. For example, the length of an ISR randomization key is represented as a variable related to the probability of a successful attack.

- *Simulation-based Evaluation* consists in performing periodically some attacks on a predefined attack graph. The success of an attack is determined by a probability model and a specific MTD implementation. Results of the simulation quantify the relationship between successful attacks and different MTD settings (like frequency of diversification). In their settings, for instance, the ratio of successful attacks is 50% when MTD is *switch off* which goes to 15% if we *switch on* the MTD. With a simulation-based approach, a whole system can be abstracted as numeric parameters, and there is no restriction on the specifics of attacks and defenses.

An attack-based approach can be used to see what happens directly inside a function or application. But it is difficult to compare MTD only on evaluation

through attack-based experiments. We only have results for the program context that cannot be used directly as a comparable metric. The attack-based approach works at the scope of an individual program. When used in a system with multiple interconnected programs, those approaches are limited. Such limitations are caused by the absence of a model for interaction between different programs. Attack effects that propagate through program interactions could not be captured. On the other hand, simulation or probability approaches permit us to see the system as a black box. But low levels contexts are abstracted or neglected, so it is difficult to reflect what happens in a system.

A probabilistic model mixed with a simulation can be used to measure the effectiveness of network address shuffling [13]. The probabilistic model used in this study is the urn model. It is a simple model in which we consider that we have a bag of marbles. There are two kinds of marbles, green ones and blue ones. The green ones represent the IP addresses that are not valid or not used by our system. And the blue ones the IP addresses used and vulnerable by our system. We have n blue marbles and m green marbles. Every time the attacker scans the network for vulnerable IP addresses, we consider that he picks k marbles, if in those marbles he gets at least one blue marble, then he has something to attack. In the configuration without MTD, the number of marbles in the bag decrease at every pick, and after a certain time, the attacker will have all the marbles. But if we use perfect MTD that shuffles IP addresses every time after every probe, then we put back every marble in the bag after every time the attacker scan the system. So the attacker will take more time to discover the vulnerable entry. In [36], Luo and all go with the same urn model approach to evaluate the efficiency of port shuffling, reaching the same conclusion.

2.4.3 Attack Timeline

The attack timeline can be seen as five different phases. Each phase has a purpose during the attack, and each MTD category intends to counter one or more of the attack phases:

- **Reconnaissance** phase in which the attacker tries to find the system on the network, for example by scanning the IP addresses around him to locate a target, or trying to find the open ports of the system. If this attack phase is countered, an attacker's ability to get access to the system will be reduced.
- **Access** phase in which the attacker tries to break into the system by exploiting a vulnerability he has found during the first phase. Most of the time this is possible because he knows the architectural platform and all the possible breaches.

- **Development** phase in which the attacker deploys his malicious code in the application he has access to. We have seen that most of the existing MTD methods can counter this phase.
- **Launch** phase in which the attacker gives the command to the system to launch his attack, and in which the system will probably encounter faults. An attacker can be in this attack phase once he has access to the system and already deployed his malicious code.
- **Persistence** phase in which the attacker tries to leave a backdoor open for the next time he wants to break into the system.

Table 2.3 describes which MTD category is effective against which phase of an attack.

MTD/attack	Reconnaissance	Access	Development	Launch	Persistence
Network	✓			✓	
Platform		✓	✓		✓
Runtime Env			✓	✓	
Software			✓	✓	
Data			✓	✓	

Table 2.3: MTD effectiveness against which attack phase

In order to get the best protection during all the attack phases, we need to combine multiple MTD techniques.

In this section, we discussed about the several terms linked to this thesis subject, connected cars and their architectures, existing attacks that target vehicles, the existing defenses and one type of dynamical defense technique, Moving Target Defenses. In the next chapter we will present a way to protect the IP address with MTD techniques.

Chapter 3

Dynamic IPv6 Addresses Switching

You need to accept the fact that you're not the best and have all the will to strive to be better than anyone you face.

- Roronoa Zoro

As has been shown for cars such as Jeep [25] or Tesla [45], a vehicle's Wi-Fi access is a potentially vulnerable entry point that can be exploited to perform various kinds of attacks. Even if the specific vulnerabilities of the above attacks have now been fixed, other attacks might still be possible. In addition to countermeasures against *known* attacks, it is thus necessary to design and deploy defense mechanisms against *unknown* vulnerabilities, aiming at (i) delaying attacks implementation, (ii) reducing the risk of attacks propagation. Therefore, extra defense methods must be added to the vehicles for protecting their Wi-Fi access. For this purpose, we have been looking for a new MTD method that can be used to hide a vehicle inside the network to which it is connected.

In this chapter, we propose an MTD solution that targets the network configuration of critical connected objects. In particular, we present an application scenario in the context of connected vehicles, where typically all cars of a specific manufacturer share a subnetwork, which makes them particularly vulnerable to remote attacks. Consequently, the goal of the proposed method is to stop potential attackers at the first stage of their attack: by changing the IP address

sufficiently often, we aim to prevent an attacker from establishing a connection and deploying his or her payload on the target system. The main idea is to equip the connected objects with redundant network interfaces, thereby allowing for faster address changes without losing connectivity to network services. Adjusting the period during which an address is valid allows to achieve a trade-off between security and traffic overhead on the network.

3.1 Background

3.1.1 Multipath TCP

MultiPath TCP is a derivative of the TCP protocol, both being used to implement communications between two objects. The purpose of MultiPath TCP (MPTCP) was initially to improve the connectivity between two objects on the network by defining several possible routes between them. Therefore, if one of the routes is not working properly, packets can still be transmitted using one of the other routes.

MPTCP was first specified in 2013 and updated in 2019, and is available on most existing platforms (Linux kernel, FREE BSD, IOS, MAC OS, etc.). Nowadays, MPTCP is mainly used in mobile networks, as it allows a mobile device to use both its Wi-Fi and cellular network interfaces to communicate with services: the interface connected to the least congested route will be used.

In an MPTCP configuration, it is possible to declare the set of interfaces through which an object can communicate, as well as the priority associated with each interface. This allows a system to keep the TCP connection open, even when one of these interfaces no longer works.

In our work, we propose to use MPTCP and its capacity to define multiple routes, as a way to maintain connections among objects while being able to redefine their communication routes. We also propose to use the IPv6 communication protocol for connections with services on the Internet. We explain the reasons of this choice in the next subsection.

3.1.2 IPv6

IPv6 is the successor of IPv4, even though the latter one is still massively used by many existing systems. However, the number of available addresses on IPv4 networks is limited: being encoded with *32 bits*, approximately four billion addresses are available on an IPv4 network. On the other hand, IPv6 addresses are encoded with *128 bits*, leading to approximately $3.4 * 10^{38}$ available addresses.

An IPv6 network is divided into subnetworks identified in IPv6 addresses by a prefix of size varying from 1 to 64 bits. The remaining bits are then used to identify a host on this subnetwork. Assuming a subnetwork is identified with 64

bits – which is the smallest possible subnetwork – these host identifiers are then encoded with 64 bits. The number of available addresses to identify hosts in this subnetwork is approximately $1.84 * 10^{19}$. These numbers are required to compute the time an attacker would need to identify the IP address of an object connected to a given subnetwork.

In the remainder of this section, we describe how IPv6 addresses can be assigned to objects: automatically, manually and managed by a DHCP server.

An automatic address assignment may cause security problems since IPv6 addresses are derived from the interface MAC address, and each MAC address is unique. This allows an attacker to probe the exchanges from a particular host and track its communications. This is problematic in the context of connected objects: using automatic assignment, the IP addresses of connected objects would not change over their lifetime. Therefore, a vulnerable object can be detected very easily by an attacker who already identified it.

A manual assignment of IPv6 addresses consists in letting objects determine their own IP address. As a consequence, it is under the responsibility of the object provider to define mechanisms preventing attackers to track this object on the network. This also poses a network management problem since address assignment is decentralized, which can result in address collisions.

The last possibility is to assign IP addresses to objects using a Dynamic Host Configuration Protocol (DHCP) server. This allows to assign or reassign addresses to objects without collisions by using the routing table of the DHCP server. The address assigned to an object is no longer derived from its interface MAC address, making it more difficult to track. However, this poses another problem: the DHCP server must be well secured in order not to allow access to the different data it contains. This problem is out of the scope of this chapter, and could also take advantage of deploying MTD mechanisms on the DHCP server.

3.1.3 Car Network Topology

The hosts of the network we consider in this chapter are cars seeking to connect to the Internet in order to have access to different services, such as navigation or entertainment services as well as traffic information. Traffic information may be provided by other cars and infrastructure objects located on the road (*e.g.* signs, lights or traffic lights).

In order for these different devices to communicate with each other, they must be assigned IP addresses, which will be done by a DHCP server. For the sake of simplicity, we assume each car manufacturer holds an IPv6 subnetwork to which cars produced by this manufacturer are connected. Of course, other options are possible, for instance, car manufacturers could rent and share a network provided by a third party.

The manufacturer's subnetwork is accessible all over the world, and all the connected cars of its brand are connected to it. The subnetwork's DHCP server is used to assign IP addresses to all the cars, and manage the addresses consistency in order to avoid addresses collisions during addresses (re)assignment. This DHCP server is also used to route the different messages on the subnetwork.

The subnetwork topology described here above is depicted in figure 3.1: the DHCP server links the cars connected to its subnetwork to the Internet. All the cars of a given manufacturer are connected to the same IPv6 subnetwork, managed by this DHCP server. All the messages circulating on the subnetwork are transmitted to the DHCP server which routes them to the recipient car.

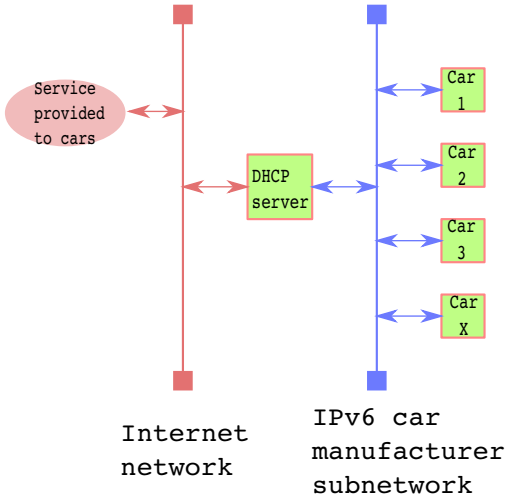

Figure 3.1: Network Topology Representation

Given this topology, a car's connection is performed as follows. Each time a car is started, it tries to connect to the internet by broadcasting a message to the entire subnetwork. This message is a request for an IP address, and should be treated by the DHCP server. This is followed by a series of message exchanges until the DHCP server assigns to this car an IPv6 address. Note that these various messages are sent in plain text and readable by any object connected to the subnetwork and listening to it. Of course, this raises security issues since an attacker can easily know the IP address of hosts: the IP addresses assigned initially have to be modified.

In addition, an host receiving an IP address from a DHCP server receives at the same time a lease time for this address: the host will have to request a new address to the DHCP server before the end of its lease. As these messages are also sent in plain text on the network, it makes it easy for an attacker to track address changes of a car.

3.1.4 Attack Entrypoints

Attackers may attempt to comprise a car from outside the subnetwork the car is connected to (i.e. from a system connected to the Internet from outside the subnetwork), or from the inside of this subnetwork (i.e. from a system already present on the subnetwork). In each case, the attacker's method will be different since he or she does not have access to the same information and resources.

An attacker located outside the subnetwork cannot easily read the various messages circulating on that subnetwork. He or she then must scan the different subnetwork addresses in order to find a system connected to this subnetwork. Once found, he or she will then need to retrieve information about the car system he or she found before launching his or her attack. It must be ensured that defense methods deployed in connected cars allow these cars to hide from an attacker scanning their subnetwork from the outside (*e.g.* from the Internet).

An attacker located inside the subnetwork can see the messages circulating on the network. Even though the payload of these messages can be encrypted, the address of the sender and receiver of each message circulates in plain text. An attacker located inside the subnetwork can thus see the IP address of cars connected to this subnetwork by listing to the circulating messages. Therefore he or she does not need to scan the subnetwork to find the IP addresses of connected cars. We must then ensure that defense methods deployed in connected cars allow these cars to escape from an attacker trying to attack from within the subnetwork.

It is, therefore, necessary that the defense methods we propose in this chapter allows a car connected to a subnetwork to *both hide* from an attacker outside the subnetwork, *as well as to escape* from an attacker connected to this subnetwork.

This problem is very significant in the context of connected cars, because attacks may propagate from inside the subnetwork itself. This is one of the reasons why the attack against the Jeep [41] became so popular: after taking control of the car which the scientists had physical access to, they demonstrated their capacity to take control over other cars remotely.

3.1.5 Time To Find An IP

Consider a car manufacturer that produces on average 6 millions cars per year. If these cars have a 20-years lifespan, and all of them are connected at the same time to the network, the maximum number of hosts is 120 millions. This is way below the number of addresses that can be encoded for hosts using IPv6, even when considering subnetworks identified with 64 bits in the IP address.

In order to consider the worst-case, we assume all the cars of the manufacturer are used at a given time, *i.e.* $120 * 10^6$. We want to estimate how long it would take for an attacker outside the subnetwork to reach a 50% probability to find the

IP address of one car on this subnetwork. We assume a subnetwork identified by a 64 bits prefix, which is also a worst-case hypothesis: it minimizes the possible number of identifiers for hosts. Still, the number of possible identifiers is huge: 2^{64}, *i.e.* approximately $1.85 * 10^{19}$.

For estimating the time an attacker needs to scan the network, we consider a well-known open source tool for scanning networks: *nmap*. In fact, *nmap* can be used to discover hosts and services on a network by sending packets and analyzing the responses. *nmap* can also provide further information on targets (*e.g.* reverse DNS names, device types, *MAC* addresses, etc.). Using *nmap* with highly optimized options, the time needed to scan one IP address is approximately 255 ms [38].

Now that we have all the inputs to compute the time needed for an attacker to reach 50% probability to find the IP address of a car, here is how the computation works. We apply the *urn statistical model* [13], to solve the problem of *Drawing With Replacement*. This means we consider the attacker randomly scans the network until he or she finds a host, and the same address can be scanned more than once.

The mathematical formulation of this problem is given by equation (3.1), where x is the number of attempts, h is the number of hosts on the network, a is the number of available addresses on the subnetwork and $Proba$ is the probability to find an object over the network.

$$Proba = 1 - (((a - h)/a)^x) \tag{3.1}$$

For an attacker outside a subnetwork identified with a prefix of 64 bits in IP addresses (*i.e.* $a = 1.85 * 10^{19}$), the time to reach $Proba = 50\%$ chance of finding the IP address of a vehicle among $h = 120 * 10^6$ million vehicles, considering 255 ms to scan an IP address, is $3 \cdot 10^{13}$ s, or approximately 951 years.

3.2 Problem Statement

In this chapter, we focus on techniques aiming at mitigating the risk of attacks coming from an object's connectivity. In particular, we concentrate on attack entry points induced by their Internet connection. We consider two types of attackers: (i) attackers connected to the Internet and seeking to infect a fleet of connected objects via their own Internet connection; (ii) attackers connected to the same subnetwork as the target objects.

On the one hand, attackers located outside the subnetwork of connected objects are forced to scan the address range of this subnetwork in order to find the IP address of some existing objects. On the other hand, attackers located inside the subnetwork can *see* the IP address of connected objects, since the address is transmitted in plain text in the header of *each* packet. The latter is even more plausible for connected objects on sale for the general public, *e.g.* connected cars, which share a common network. Therefore, an attacker located inside the respective subnetwork does not need to scan the subnetwork and can launch his or her attack faster.

In this chapter, we address the following problem: *how to escape from an attacker inside the subnetwork of an objects' manufacturer while being hidden from attackers targetting these objects from outside their subnetwork (i.e. from the Internet)*?

In this problem, *escaping* refers to the capacity of an object to actively move away from an attacker using MTD, while hiding refers to the passive capacity of an object to be difficult to identify.

Of course, other objectives have to be considered when answering this problem, *e.g.* limiting the impact of the defense mechanism on cost and Quality of Service (QoS) of an object's Internet connection.

Compared to existing approaches pursuing the same objective [5, 31, 17, 13, 24] the originality of our work lies in the following characteristics: firstly, we not only consider attackers proceeding from outside objects network, but also from within this network. Secondly, we propose to limit the impact of escaping techniques on the Internet connection QoS. Lastly, we consider a more recent version of the Internet Protocol (IPv6). In the next section, we present our contribution on this subject in this chapter.

3.3 Pool-based Ip address Switch

An attacker outside a car manufacturer's subnetwork will have a great difficulty learning the IP addresses of potential target vehicles. However, assuming that an attacker owns a connected car, he or she can use it to connect to the same subnetwork as all the other cars by the same manufacturer. It is then possible for the attacker to use the car as a network terminal and listen to messages circulating on the subnetwork. The IP addresses of the sender and receiver of each message are visible in plain text to every host connected to the subnetwork. Therefore, the attacker can effectively bypass a network scan and directly harvest the IP addresses of the cars communicating on this network. In this way, attacks can be launched on a fleet of cars connected to this network. In case of success, the attacker can potentially take control of the entire fleet [61]. In order to prevent this scenario, our objective is to make it more difficult for an attacker to attack a car via its network interfaces, from inside and outside of the manufacturer's subnetwork.

3.3.1 Pool-based IP Address Switches

In this section, we describe an approach allowing a car on a manufacturer's subnetwork to avoid attacks from within that network. Following this approach, each car is assigned a pool of IP addresses, allowing to switch among these addresses. We have designed this approach in such a way as not to lose the advantage of the large IPv6 address range: cars remain hidden from attackers outside of the manufacturer's subnetwork.

In order to find an optimal way to avoid attacks, several parameters need to be considered. In the approach presented in more detail hereafter, a car is given a pool of N IPv6 addresses. The greater the number of addresses available to each car, the faster it can switch addresses before returning to already used addresses and thus hinder an attacker to understand and circumvent the defense. Furthermore, if N is too small, an attacker may easily launch a DoS against most cars of the fleet.

Considering the time that each address is used (the IP address switching period P), it must be sufficiently small to be effective, such that an attacker will not have time to complete the attack before the car switches to a new address accepting incoming messages. On the other hand, the time R before reusing an IP address from the pool should be large, so that the attacker cannot easily predict the next usable address too easily.

In order to optimize these two parameters, we need a large number of IP addresses per vehicle (N) to allow more time to return to an already used address and thus slow down potential denial of service attacks. The number of network interfaces per car is limited by the cost of a network interface. Also, if the number

of interfaces and thus of valid addresses is too large, a scan of the subnetwork from outside this network will become easier (*i.e.* reducing the benefits of IPv6 addresses).

3.3.2 Approach Principle

The main principle of our approach is to equip each car with N network interfaces, each of which is associated to an IPv6 address. Each address of each interface is an active address, *i.e.* it appears in the routing tables and can be accessed from any other host connected to the Internet. However, the different active addresses do not necessarily accept all incoming messages: following the internal firewall rules of the car, some will be accepted while others will be dropped. In our case, there is exactly one network interface accepting incoming messages at any time.

As described in section 3.1.1, MPTCP allows a host to declare several accessible paths. This makes it possible to optimize travel times for packets if some transportation facilities are saturated. In the context of this work, we are relying on MPTCP to avoid loss of connectivity with services used by the car once its address has changed. In order to do this, we declare all interfaces of a vehicle, but only one possible path at any time, corresponding to the accepting IP address.

The first time a car connects to the network, it broadcasts messages over the entire subnetwork via its different network interfaces in order to request IP addresses, which is the usual way to obtain a valid address on a network ruled by a DHCP server. The DHCP server responds to the car and assigns an IP address to each of its interfaces. The car will then use MPTCP to declare the different IP addresses belonging to it. In our approach, we propose to enrich the connection protocol in the following way: The DHCP server sends to the car an encrypted message containing the next IP address that will be considered as the active address as well as the period of time during which this address will remain active, call *lease*. The car will then edit the MPTCP rules to declare the new address received as the main route to be used for all communications. The change will remain valid until the end of the period received and the car will then again change its current active address with the new address it will receive from the DHCP server during this period.

The DHCP server manages the active addresses for each car. Every time an active IP address reaches the end of its lease, the server picks a new IP address from the car's address pool. Furthermore, it picks a new duration for the new lease of this address. This duration is drawn randomly from a predefined interval (noted *[a,b]* and centered around P). Randomization is necessary to prevent an attacker who is listening on the network to anticipate and circumvent the defense mechanism. The new active address and its lease are sent to the car in an encrypted message, which can only be decrypted by the destination car. All other network

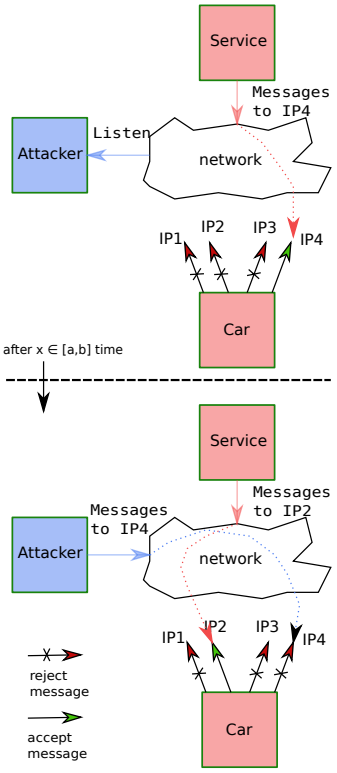

Figure 3.2: Illustration of the proposed pool-based address switching scheme

participants can only see a message from the DHCP server to this particular car. After receiving the encrypted message, the car updates its internal firewall rules in order to accept messages on the new address, and changes the MPTCP rules to declare the new external route corresponding to the new address.

3.3.3 Example

Figure 3.2 illustrates a manufacturer's subnetwork, with a connected car equipped with four network interfaces with each one an IP address assigned, *IP1*, *IP2*, *IP3*, and *IP4*, as well as an external service and an attacker connected to the subnetwork. In the beginning, *IP4* is the active IP address that accepts incoming messages, while the other three reject all incoming messages. The car communicates with the service, which responds by sending messages to *IP4*. Since the IP address of the car is visible in the header of any circulating message in the network, the attacker can discover the IP address of the car and may try to communicate with the car via *IP4*. Once the active time period of *IP4* ends (after $x \in [a, b]$ time units), the DHCP server elects the new IP address accepting messages, and sends – in an encrypted message – the new configuration to the car. As shown in

the lower part of the figure, *IP2* now becomes the active address, while *IP4* will drop any incoming messages.

3.3.4 Protection

If the attacker launches the attack before the currently receiving IP address expires, he or she can start to communicate with the discovered car. If the valid time period per IP address is short enough, the address switch will take place before the attacker can complete a successful attack. Assuming that an entire fleet of vehicles is equipped with the proposed MTD mechanism, a massive parallel attack against the whole fleet will be impractical: if the attacker assembles a hit-list of collected IP addresses, these will likely have changed at the time the attack starts. Since the assignment of the receiving IP addresses and their active time window is sent encrypted, the attacker will not be able to learn which address the car has switched to once the currently active interface stops responding.

In conclusion, changing the IP address slows down an attacker coming from inside of the subnetwork. This attacker will see the IP addresses of the different cars communicating on the subnetwork. If he or she is seeking to retrieve information about one discovered car before launching his or her attack, he or she may even not be able to launch his or her attack at all. Periodically changing the address, makes the time available for an attacker to perform his or her attack shorter than without this mechanism.

3.3.5 Cost

The presented defense mechanism can easily be implemented on top of an existing network. It does not entail major infrastructure changes nor heavy network traffic overhead. To apply this method in an existing system, it will be necessary to setup MPTCP on the side of the vehicles. Furthermore, it requires adding N new physical network interfaces on each car and adapting the car's firewall rules. On the side of the network infrastructure, it will be necessary to slightly modify the DHCP server for each subnetwork in order to implement the assignment of an address pool to each newly connected car and the address renewal mechanism choosing a new address from the pool.

3.3.6 Discussion

It can be argued that the addition of several IP addresses would facilitate the discovery of a fleet of vehicles from outside the network, and this is actually the case. The time to reach a 50% probability of finding one of the IP addresses is $3 \cdot 10^{12}$ s, or 95 years instead of 951. This remains a very acceptable time.

Especially since when an attacker succeeds in finding one of the vehicles, there is only a $1/N$ chance that this IP address is the active IP address that receives the messages.

This technique is able to repel attackers even from inside the network if the period of IP address change is sufficiently small. However, there remains a risk of DoS attacks. This vulnerability is due to the fact that each car is assigned a *static* address pool from the beginning to the end of its connection on the network. This means that the addresses from this pool will be used several times as the accepting address for incoming messages. This periodic re-use of the same addresses allows an attacker to establish a list of known addresses and to flood them with messages continuously, resulting in a denial of service. Therefore, an alternative approach – which will be presented in the following – limits address reuse.

3.4 Randomized Pool-based IP Address Switches

In this section, we describe an alternative MTD approach that additionally aims to protect the connected vehicles against DoS attacks, while trying to limit the impact on the network infrastructure. We keep the same assumptions about the system: all cars of the same manufacturer are connected to the same IPv6 subnetwork controlled by the manufacturer; each car has N network interfaces providing it with N IP addresses on this network. Likewise, we are still using MPTCP to maintain several open connection paths as well as the periodic change of the IP address accepting messages. However, we make the address pool associated with each car dynamic, i.e. addresses will change over time, such that no address will be used twice by the same car, in order to avoid DoS.

3.4.1 Approach principle

As before, at the first connection of a car on a network, the DHCP server assigns an IP address to each of its interfaces and indicates the interface that will accept incoming messages.

However, in addition to regularly switching the active interface of each car, the DHCP server will send an encrypted message, ordering the receiving car to replace the IP address of one of the *inactive* interfaces with a fresh one. The car needs to do the necessary changes in its configuration, and update the corresponding interface to use the new address once it will become the active interface. This implies also adding the new address to the MPTCP rules as a new route to the car. In order to avoid packet loss, and give the car sufficient time for reconfiguration, the address change occurs *in the background*, *i.e.* it never affects the currently active interface and does not interfere with the car's communication.

The period by which new addresses will be assigned can be chosen freely, leading to different trade-offs of costs and security. If it is longer than the validity period, this will lead to address reuse. By always renewing the address of the least recently deactivated interface, the address pool of size N is completely renewed every N address changes. As a consequence, no address will ever be active twice[1], as illustrated by the following example.

3.4.2 Example

Figure 3.3 shows the randomized address switching for the case $N = 2$, i.e. each car has two network interfaces. The car *Car* is connected to the manufacturer's subnetwork and has currently addresses $IP1$ and $IP2$ assigned to its interfaces,

[1] unless the same address is randomly chosen twice by the DHCP server, which has a sufficiently low probability.

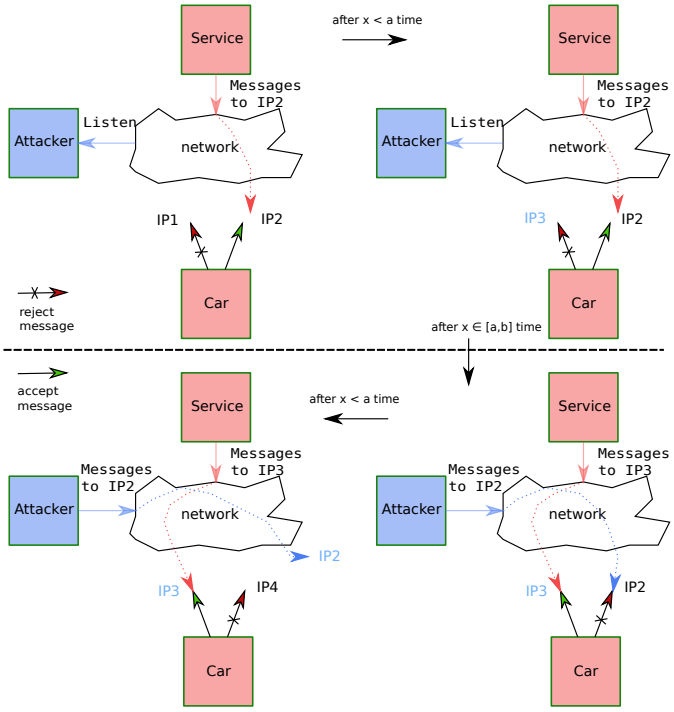

Figure 3.3: Illustration of the randomized pool-based address switching with 2 network interfaces per vehicle

where $IP2$ is the address accepting incoming messages. the *Car* communicates with a service on the network, while an attacker listens to the network traffic. Meanwhile, the *Car* receives instructions from the server to replace the address $IP1$ with the fresh address $IP3$, so it updates its interfaces, MPTCP rules and firewall rules. Once the valid time period of $IP2$ expires, $IP3$ becomes the active address accepting incoming messages, while the attacker still tries to communicate via $IP2$. The *Car* continues its communication with the service while the attacker's messages are rejected on $IP2$. Meanwhile, the car then receives instructions to change $IP2$ to $IP4$, so the attacker now talks to an unknown address. By changing addresses at each time period, the attacker only has this period to launch his or her attack and can no longer anticipate changes in the *Car's* addresses.

3.4.3 Costs

If the address renewal period is equal to the validity period – as in the above example – each address is only used once. Thus, there is no need to choose N greater than two, as this does not increase the entropy. Therefore, the constant costs related to the number of network interfaces are lower than in the static pool-based approach with $N > 2$. However, there is an additional overhead for the network infrastructure: firstly, there are additional address renewal messages sent

out by the DHCP server. Secondly, the DHCP server itself has additional work to do choosing new addresses and changing its internal routing tables.

3.4.4 Randomization on Demand

An alternative application of the randomized address-renewal would be to use it in addition to the first method. In other words, equipping the cars with N network interfaces with addresses assigned per interface that do not automatically change over time. But when a persistent attack on the network is discovered, an address renewal is triggered: for the list of vehicles under attack, the pool of available addresses is updated, thus preventing denial of service.

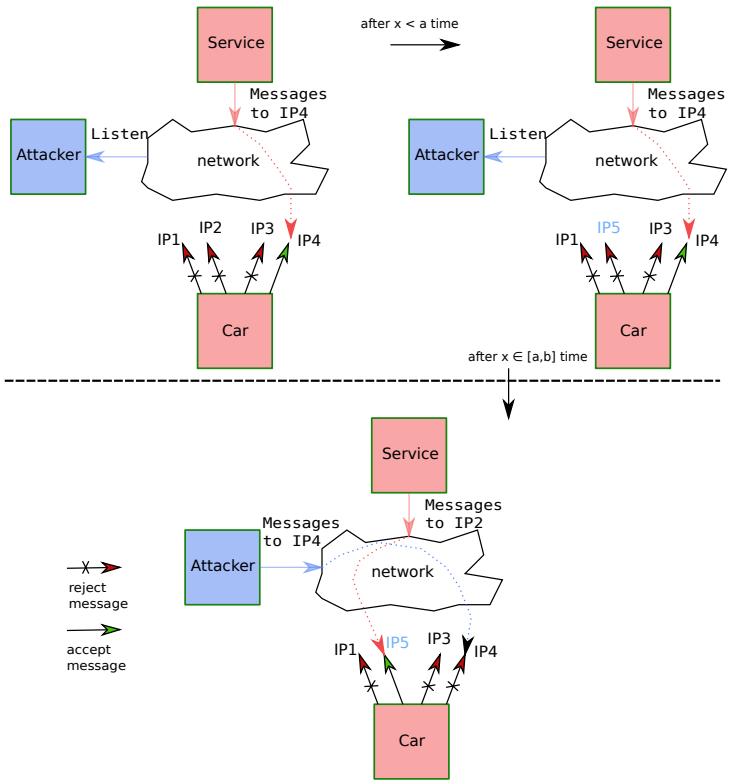

Figure 3.4: illustration of the method with N network interfaces per vehicle

Figure 3.4 illustrates this combined approach. There is a car connected to the subnetwork with 4 network interfaces, corresponding to the IP addresses $IP1$, $IP2$, $IP3$, and $IP4$. In the beginning, $IP4$ is the only address accepting incoming messages. As before, an attacker is listening to messages circulating on the network. Upon detection of the intruder, the car receives instructions from the DHCP server to change $IP2$ address to IP5, and it updates the MPTCP and firewall rules. At the end of the validity period $IP5$ becomes the new accepting

address. The attacker will try to attack $IP4$ which no longer accepts incoming messages. In contrast, in the absence of a detectable threat, the address pool will remain stable, avoiding additional network traffic and server load.

3.5 Related Works

As explained by Lei & al. in [34], MTD can be used in different layers of a computer-based system. Since we concentrate on the network configuration in this chapter, the related works we present hereafter are primarily on MTD at the network level.

Most of these works propose to assign IP addresses dynamically, which allows an object to escape from an attacker who had identified it. This simple protection can be effective in some cases. If a system's address is changed from time to time, it allows the user to be protected against hit-list attacks. A loss of connectivity from time to time is acceptable under conditions where no critical applications are being executed. The authors of [5] illustrate how changing IP address allows users to defend themselves from hit-list attacks. However, opposed to our approach, they do not consider connection losses when changing IP addresses: they assume services can support connection failure without consequences. In practice, this may lead to poor QoS, and in the context of critical systems this might even have unacceptable effects.

In [13], researchers analyzed different IP address shuffle mechanisms without providing solutions for connection losses. Furthermore, they did not consider attack scenarios from within the network, where the attacker can simply intercept broadcast messages from the DHCP server.

Jiao & al. have studied how to counter denial of service attacks with MTD techniques [31]. They explored the added value of proxy shuffles and names shuffles in Domain Name System (DNS) servers for the defense against DoS attacks. Again, this method does not prevent connection loss during shuffles.

In [24], Fraunholz & al. explore a defense method that includes MTD techniques in order to hide a system connected to a network from potential attackers. Their method is based on an IP stack randomization process which allows to modify the information describing the current system in sent TCP messages, with the goal to leave modified traces on the network on regular time-frames. They consider IPv4 sub-networks of classes A, B, and C. Furthermore, they study the probability of collisions when searching for a new address as well as the link with the randomization stack and how to be seen by an attacker as a new system at each shuffle.

This defense method is effective in a network with a multitude of objects from different vendors and manufacturers. In our work, we are considering a network with connected cars. For each given car model of car, the network traces will be identical for all individual vehicles, since they all run on the same system, software version, and hardware. As a consequence, the method presented in [24] is not reusable for this scenario.

In [17], the authors study the problem of address collision when changing IP

addresses. In our approach, we wanted to avoid this issue which is limiting the rate of shuffles. Therefore, we have integrated the address change management into the DHCP server, which is aware of used and available IP addresses.

In [21], authors advocate the composition of MTD techniques in order to improve their efficiency. Even though we did not consider the composition of our method with other MTD techniques, we expect it to be compatible with most of the existing MTD techniques. For instance, following the same idea as Fraunholz & al. in [24], we could have composed our method with a randomization of the IP stack in order to modify the information describing the current system in the TCP messages.

As a conclusion, the novelty of our approach lies in the use of redundant network interfaces in combination with MPTCP to guarantee QoS during address switches. Furthermore, by delegating the address switching to the DHCP server sending encrypted messages, in our setup, we can even escape attacks from within the same network.

3.6 Discussion

By adding randomized renewal, DoS attacks can be thwarted, with higher costs incurred by network traffic overhead and increased DHCP server load. Between the two extremes – no address renewal at all or after each interface swap – the renewal period can be chosen in order to adapt the overhead and the security level to the use case and available infrastructure. As an alternative, addresses can be renewed on demand only, in combination with an appropriate attack detection mechanism.

3.7 Conclusion

In this chapter, we investigate a moving target defense in the context of connected objects on an IPv6 network. In particular, we consider connected cars sharing a common manufacturer's subnetwork, which makes them particularly vulnerable to attacks from within the same network.

As a remedy, we propose a dynamic address switching scheme relying on the network's DHCP server. In our approach, each connected car disposes of several redundant network interfaces, each with a separate IP address. This allows for dynamic address switching without connection losses. The basic protection mechanism is that each car only accepts incoming messages on one interface at any time, leaving attackers talking to inactive addresses after a switch. While the basic approach uses fixed address pools, an additional address renewal by the DHCP server can help to thwart denial of service attacks, since it allows to avoid address reuse. Adjusting both the periods for address switching and address renewal leads to a flexible MTD approach that can be adapted to the security requirements and network infrastructure at hand. To the best of our knowledge, this is the first network MTD approach presented in the context of an IPv6 subnetwork which considers attacks from within the same network.

Chapter 4

Finding Optimal Defense Strategies

If more of us valued food and cheer and song above hoarded gold, it would be a merrier world.

- Thorin Oakenshield

So far, most works on MTD have focused on the question *where* and *how*, thus providing and evaluating new MTD mechanisms [64, 59]. Some studies have also been conducted to assess the efficiency of MTD in the context of risk analysis methods [29]. More recently, a few methods have examined the question of *when* to reconfigure [33, 54, 22, 35, 65], but these contributions focus on the domain of web applications, therefore their requirements, MTD mechanisms, and models are not suitable for the design of CRES.

Nevertheless, there exist several MTD mechanisms applicable to CRES, and for which the question *how* is already answered. Instead, we focus on the questions of *when* and *where* to reconfigure parts of a system. Obviously, the question *when* is actually: *how often*? A question to which one may simply answer *as often as possible*; but implementing an MTD mechanism has a cost, and its execution inevitably impacts the availability of the components that must be reconfigured. For instance, switching IP addresses will induce communication overheads and temporary connection or packets losses. Therefore, MTD comes with an implementation cost and a QoS degradation.

In this chapter, we aim at answering the following question: How to model MTD benefits and cost in order to find optimal MTD strategies (i.e. *where* and *when* to move) in the context of CRES? We propose a combination of risk analysis techniques, MTD mechanisms, and a game-theoretic approach in order to define the best defense strategy to adopt in terms of frequency of each available MTD mechanism. The contributions of this chapter are:

1. A game theoretic model for the defense of CRES
2. A resolution method based on the transformation to an MILP problem
3. A complete methodology to define the input parameters of the presented model
4. An experimental case study for a typical connected car architecture

Before presenting our contributions, the first section of this chapter introduces the necessary game-theoretic concepts.

4.1 Game Theory

4.1.1 Introduction

Game theory is the science of strategic decision making. Game theory has been used in fields as diverse as evolutionary biology, decision management (politics), and economics. Many decision problems can be modeled using game-theoretic concepts, with applications to various domains, reaching from program synthesis to resource allocation in smart grids. It can be defined as the study of mathematical models of conflict and cooperation between intelligent and rational decision makers. Game theory provides general mathematical techniques for analyzing situations in which two or more people make decisions that influence each other.

In the language of game theory, a game refers to any situation involving two or more subjects. The subjects involved in a game may be called the *players*. As indicated in the definition above, there are two basic assumptions that game theorists generally make about players: they are rational and intelligent. Each of these adjectives is used here in a technical sense that requires some explanation. A player is rational if he makes decisions in a manner consistent with his own goals. In game theory, based on the basic results of decision theory, we assume that each player's goal is to maximize the expected value of his own payoff, which is measured in some utility scale. The idea behind a *rational decision maker* is that the actions taken (called *strategies*) will maximize the player's expected utility benefits. This idea goes back at least to Bernoulli (1738), but the modern justification of this idea comes from Von Neumann and Morgenstern (1947) [44].

Games are a natural model for security problems: They allow to formalize the goals of potential attackers in terms of objectives of an adversarial player, and provide a mathematical framework to reason about defense strategies. The use of game theory [53] provides a representation of the problem posed in the form of an optimization problem.

In general, in a mathematical game, each player chooses from a set of available *actions*. The choice of actions of the different players can be done in different ways corresponding to two basic categories of games:

- **Simultaneous games** in which the players choose their respective actions at the same time without knowing in advance the choices of the other players.
- **Sequential games** in which the players are playing in a (fixed) order, such that all other players can observe the first player's action before taking a decision.

Choosing an action to perform results in a *reward* that could be more or less attractive depending on the adversary's choice. The payoffs of choosing an action

for the different players are defined in terms of *reward functions*: The value of the payoff depends on the combination of actions chosen by the different players. The most common way to represent reward functions is by a matrix (see for example Table 4.1), called the *normal form*. In this example, the attacker has the choice to attack or not to attack and the defender to defend or not to defend. For each combination of actions, the table gives a pair of constants, corresponding to the reward for each player: r_a for the attacker and r_d for the defender.

	Defend	Not Defend
Attack	(r_{a1}, r_{d1})	(r_{a2}, r_{d2})
Not Attack	(r_{a3}, r_{d3})	(r_{a4}, r_{d4})

Table 4.1: Attacker/defender normal form game

When creating the rewards associated with each move, we distinguish two types of games, the *zero-sum game* and the *non zero-sum game*. In a *zero-sum game*, the reward of one player equals the loss of the other player. In this type of game, there cannot be several winners. To solve them, finding an equilibrium between the rewards obtained by the players will be the choice most of the time. In a *non-zero-sum game*, the rewards associated with the movement of both players are not related to each other, all players can win something from the same action, as well as lose all from the same action.

4.1.2 General Definition

To explain the concepts of game theory, let's take an example; "congestion avoidance in the Internet" [4]. The Internet is based on the TCP/IP protocol. In TCP/IP, a file is broken down into packets that pass between the different nodes of the network between the sender and the receiver. Each time the receiver receives a packet, it sends a receipt to the sender. In this way, the sender knows that the packet has arrived. The TCP/IP protocol seeks to increase the network's throughput until it reaches saturation. When one of the nodes of the network is saturated, it deletes packets until it becomes depleted. The sender receiving no more acknowledgments for a packet will resend it after a certain delay (managed by the congestion avoidance algorithm). If all the transmitters follow this rule, it allows the node to be de-stressed for the benefit of all the users. However, it is possible for some senders to violate this rule and to re-send the message without latency.

This behavior can be modeled in game theory. Let's imagine two people using Internet. They have two possible choices:

- Use the correct version of the network congestion avoidance algorithm. This strategy is noted C.
- Use an imperfect version of the congestion avoidance algorithm of the network. This strategy is noted D.

The behavior of the network is the following; if the two persons choose the strategy C, each packet will be delayed by 2ms. If the two persons choose strategy D, each packet will be delayed by 4ms. If one person chooses the C action and the other chooses the D action, the first person will have his packets delayed by 6ms and the second person will have his packets delivered without delay. Of course, each person is rational and seeks to increase his throughput, i.e. in our example, to minimize the delay of his packets.

4.1.3 Game Representation

Definition 4.1. *A game can be modeled by a tuple* $< N, S, u >$ *where :*

- N *represents the players.* $N = \{1, 2, \ldots, n\}$ *is a set of size* n *(number of players in the game).*
- S *represents the set of combinations of possible strategies for the set of players* $S : S_1 \times S_2 \times \ldots \times S_n$ *where* S_i *is the set of possible strategies for the player* i*. Not all players necessarily have the same strategies and some combinations of strategies may be impossible.*
- u *represents the function* $u : S \to \mathbb{R}^n$ *that associates each strategy combination with a utility value with each* u_i *:*
  - *a negative value representing a loss for the player* i
  - *a positive value representing a gain for the player* i

It is more convenient to represent the utility function for player i as $u(s_i, s_{-i})$, where $s_{-i} = (s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$ which represents the strategy played by all the other players and to note u_i the i-th component of the utility function.

For our example, we have:

- two players, $N = \{1, 2\}$
- The possible strategy for the two players are identical $S_1 = S_2 = \{C, D\}$. And the set of possible strategy combinations is defined as $S = \{(C, C), (D, C), (C, D), (D, D)\}$
- The utility function can be represented by the following matrix :

$Player_1$ \ $Player_2$	C	D
C	$(-2,-2)$	$(-6,0)$
D	$(0,-6)$	$(-4,-4)$

Table 4.2: Normal form of the game for the example

4.1.4 Strategy

Dominance

Definition 4.2. *A strategy* s_i^* strictly dominates *the strategy* s_i *if and only if :*

$$\forall s_{-i} \in S_{-i}, u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i})$$

Whatever the other player's strategy is, the strategy s_i^* is the best choice for the player i. And since the player i is rational, he will never play a dominated strategy s_i.

Definition 4.3. *A strategy* s_i^* weakly dominates *the stategy* s_i*, if and only if*

$$\forall s_{-i} \in S_{-i}, u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$$

and

$$\exists s_{-i} \in S_{-i}, u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i})$$

Whatever strategy the other players choose, the strategy s_i^* is at least as good as the strategy s_i for the player i.

It is now possible to extend these definitions over all the possible strategies for every player i.

Definition 4.4. *A strategy* s_i^* *is* strictly dominant *(resp.* weakly dominant*) for the player* i *if and only if* s_i^* *strictly dominates (resp. weakly dominate) all other strategies for the player* i*.*

For our example, for player 1, the strategy D strictly dominates the strategy C. Indeed, we have $S_{-1} = \{C, D\}$. for $s_{-1} = C, u_1(D, C) > u_1(C, C)$ and for $s_{-1} = D, u_1(D, D) > u_1(C, D)$. In fact, the strategy D is also the strictly dominant strategy for player 2.

Since player i is rational, and that there exists a strictly dominant strategy, player i will always play this strategy.

Strategy Type

Following a pure strategy, a player always chooses one fixed action to perform, while playing a mixed strategy corresponds to randomly choosing between a set of actions following some probability distribution.

For the type of game used in the following contribution – which corresponds to a Bayesian Stackelberg game – it can easily be shown that the defender – who is the leading player – needs to play a mixed strategy in order to optimize his gain. On the other hand, it is a common hypothesis that the attacker sticks to a pure strategy [18]. The different types of games will be described in the following section.

4.1.5 Designing A Game

There are different types of games, all of which have in common the basic definitions as presented in Section 4.1.3: a set players, a set of possible moves for each player and a reward function associated to each move. However, they differ in the way the players interact with each other.

A game can be played either simultaneously or turn by turn (*Stackelberg games*). Furthermore, the information related to the reward of the players can be known exactly or some information can be missing (*Bayesian game*). These characteristics can be combined to obtain four types of games in order to be able to represent different types of situations and to be as close as possible to the system we want to represent.

- The first type of game corresponds to the easiest and most common one, the **simultaneous games with perfect information** [53] games are the most commonly used to represent a situation in which the players make their decision concerning the strategy to follow at the same time without knowing the choices of the other players at the time of the decision. Depending on the type of game, cooperative or not, zero-sum game or not... each player will have to try to make the best decision according to the objective he has to fulfill.

  Let's go back to the example to the Internet congestion dilemma presented in section4.1.2 in which each player can choose to use a correct or incorrect version of the congestion algorithm. In the case of a simultaneous game, both players know the reward each combination of chosen actions will entail. Both players have to choose an action at the same time without knowing what action the other player will perform before they both have made their choice.

- The second type of game corresponds to a **Stackelberg game with perfect information** [11, 35], where the order in which the players choose their actions is defined. The first player to choose and reveal his action is called the *leader*. The second player, called the *follower*, will choose his action based on the action of the first player. These games are also commonly called *turn-based games*, because each player chooses an action to perform in turn.

  We consider again the Internet congestion dilemma to illustrate this type of game if which the players play turn by turn. This corresponds to a situation where one of the players chooses his action, when the second one stands right behind him observing the chosen action. The second player will then choose to respect the protocol depending on the first player's choice.

- The third type of game corresponds to a **simultaneous Bayesian game** [49, 51]. The particularity of this game comes from the fact that each of the players can be of several types. The other players do not know at the time of their choice the type of the other players. He only knows the probability of appearance of each type of the other player. Each type of the second player has a different objective to achieve, which leads to a new normal form for each type of player, each with different rewards depending on the type of player.

  To illustrate the simultaneous Bayesian game we will once again use the Internet congestion dilemma. In this situation there are still two people that want to send some messages over the network. However, the second player now can have one of two possible objectives: either the same objective as the first player (sending his message as fast as possible), or he might want to disturb the internet traffic. In the latter case, the reward could for example be calculated as the sum of the delays of all players. At the beginning, the first player does not know the goal of the second player, only the probability of encountering the one or the other type of player. Thus, he needs to adapt his strategy according to this probability.

- The last type of game is a **Bayesian Stackelberg game** [49] which corresponds to a Stackelberg game with imperfect information on the type of the other player.

  To illustrate this last type of game, consider again the Internet congestion dilemma. As in the previous example, the first player does not know the

goal of the second player. The second player – who might be interested in causing congestion – has the additional advantage to know the first player's move before choosing his own. The first player is called the leader, while the second player is the follower.

4.1.6 Game Resolution

There are several ways to find a solution for a game. The method to be used will depend on the type of game chosen as well as on the result we want. For the same game and the same reward functions, it is possible to have a different solution depending on the type of resolution or equilibrium we are trying to achieve.

The simplest way to solve a game corresponds to the **Min-Max**, in which one will seek to minimize the loss for the player in the worst case scenario. It is assumed in this type of resolution that the other player will not necessarily do what is expected of him, hence the worst case scenario. This concept can be used in both directions, maximizing the minimum reward, or minimizing the maximum loss.

In the following examples, we will take as an example the case of *maximin* in which a player tries to maximize the minimum he can win. The definition of the maximin is presented in equation 4.1. Here, $\mathcal{R}(\delta_n, \alpha_m)$ corresponds to the reward function of player 1 when player 1 chooses the action δ_n and player 2 chooses the action:

$$\mathcal{R}^* = \max_{\delta_n} \min_{\alpha_m} (\mathcal{R}(\delta_n, \alpha_m)) \tag{4.1}$$

Using the normal form of the example shown in figure 4.2, players 1 and 2 can perform either action C or action D. We will calculate the maximin value for player 1. By choosing action C, player 1 can win either -2 or -6, so the worst reward he can get by playing action C is -6. If this time he chooses to play the D action, he can either win 0 or -4, the worst reward he can get is -4. Since -4 is greater than -6, if the 1 player wants to maximize the worst payoff he can get, he should choose to play the D action.

To compute the *minimax*, we will instead try to minimize the maximum payoff that the other player can get according to the same principle as the *maximin*.

Another way to compute a solution for a non-cooperative game corresponds to the **Nash Equilibrium**. The principle of this solution is that each player is considered to know the equilibrium strategy of the other player and that each player has no interest in deviating from this equilibrium strategy. The equilibrium

consists in the fact that each player will find an action to perform in such a way that according to the action of the other players there is no other action that can bring a greater gain to either player. It has been proved by Nash[43] that every finite game has a Nash equilibrium.

Considering δ the strategy of player 1 and α the strategy of player 2 and $R(\delta, \alpha)$ the reward obtained for the pair of actions (δ, α), if there exists a pair of strategies as presented in equation 4.2, we consider that it is the Nash equilibrium of the system.

$$R(\delta*, \alpha*) \leq R(\delta, \alpha*) \tag{4.2}$$

If there is only one $\delta*$ strategy that achieves the Nash equilibrium, it is called a strict equilibrium. If on the other hand there exists one or more other δ allowing to obtain the same result, we call this equilibrium weak equilibrium.

We look again at the table 4.2 presenting the normal form of a game, in order to calculate its Nash equilibrium. If we take the (C, C) strategy, if one of the two players changes his strategy to the D strategy, he will get a larger reward, so this does not correspond to the Nash equilibrium of this game. If we now take the (D, D) strategy, if one of the two players changes his strategy to the C strategy, he will get a smaller reward than what he already gets. The (D, D) strategy corresponds to the Nash equilibrium of the system, and since there is only one strategy that allows player 1 to obtain this equilibrium, we have a strict Nash equilibrium for the game.

When we have defined a Stackelberg game and we want to maximize our gains as a leader, we use the **Stackelberg Equilibrium**. In a Stackelberg game, the two players do not play at the same time, the follower will choose his strategy according to the observed strategy of the leader. The basic assumption is that the follower is rational and will choose the strategy that gives him the most profit according to the strategy chosen by the leader.

The search for optimization will go through an *backward induction* in which the leader will look for the strategy allowing him to maximize his gains. To do this, the leader begins by calculating the optimal response of the follower for each sub-game. A sub-game corresponds to a game in which the leader's move has been fixed and we consider all possible moves of the follower. Once the list of the follower's responses to each of these moves is established, the leader will choose the strategy that will allow him to get the biggest rewards while knowing what the follower will do next.

Using our example of the table 4.2, in this example we start by looking at the optimal response of the follower for each of the moves of the leader (player 1).

For the action C, the best response of the follower is to choose the action D, this brings a reward of -6 for the leader and 0 for the follower. For the action D, the best answer of the follower is to choose the action D, it brings a reward of -4 for the leader and -4 for the follower. The leader now has the choice between playing the action C which should bring him -6 and the action D which will bring him -4. The leader will choose to play D which will be followed by the follower also playing the action D.

Harsanyi Transformation

The above resolution methods do not allow to directly solve Bayesian games. In order to solve this problem, Harsanyi has developed a transformation method [27] allowing to represent Bayesian games in normal form and then to solve them with the presented methods.

If we take a Bayesian game with 2 players, where one of the two players can be of 2 types (such as type A in 40 % of the time and type B in 60 % of the time), we can represent the interactions between player 1 and each of the two types of player 2, as shown in Tables 4.3.

$Player_1$ \ $Player_{2A}$	C	D
C	$(-2,-2)$	$(-6,0)$
D	$(0,-6)$	$(-4,-4)$

$Player_1$ \ $Player_{2B}$	C	D
C	$(-1,-3)$	$(0,-4)$
D	$(-4,0)$	$(-5,-5)$

Table 4.3: Normal form of the game for the example

The two types appear 40 % and 60 % of the time, respectively. Using these probabilities, we can combine the two normal forms to a new one. Harsanyi's proposal consists in adding the rewards associated with the combination of actions between two players in the corresponding box of the new normal form, weighted by the probability of appearance of the type of player. This gives the new normal form presented in table 4.4. We can now use the classical resolution methods on the resulting normal form.

Complementary Slackness

The search for an optimal strategy gives rise to an optimization problem. In some cases, it can be efficiently solved using for example linear programming.

$Player_1$ \ $Player_{2A/B}$	C	D
C	$(-2*0.4+-1*0.6,$ $-2*0.4+-3*0.6)$	$(-6*0.4+0*0.6,$ $0*0.4+-4*0.6)$
D	$(0*0.4+-4*0.6,$ $-6*0.4+0*0.6)$	$(-4*0.4+-5*0.6,$ $-5*0.4+-4*0.6)$

$Player_1$ \ $Player_{2A/B}$	C	D
C	$(-1.4, -2.6)$	$(-2.4, -2.4)$
D	$(-2.4, -2.4)$	$(-4.6, -4.6)$

Table 4.4: Normal form of the game for the example

In general, a linear program is represented by an objective function to maximize, and a set of constraints, such as presented in the equation below. Here, x corresponds to a vector of variables we are trying to determine, c, b are vectors, and A is a matrix.

$$\begin{aligned} &\max_x c^T * x \\ &\text{Where: } Ax \leq b \\ &\text{and: } x \geq 0 \end{aligned} \tag{4.3}$$

This is also called the *primal* of the problem. It can be rewritten in another way, in order to try to find an equivalent solution, called the *dual* of the problem presented in 4.4. Here, y corresponds to the variable we are trying to determine, and c, A and b are the same elements as in the primal.

$$\begin{aligned} &\min_y b^T * y \\ &\text{Where: } Ay \geq c \\ &\text{and: } y \geq 0 \end{aligned} \tag{4.4}$$

In order to show that we have found an optimal solution to the original problem, we must verify the *complementary slackness* which says that if x_0 is a solution to the primal and y_0 is a solution to the dual, and that $c^T * x_0 = b^T * y_0$ then x_0 and y_0 correspond to the optimal solution of the problem.

In our model, complementary slackness will be added as a constraint in order to force the attacker to choose an optimal solution. This reflects the fact that we assume a rational attacker.

4.2 Risk Analysis

All current systems have both known and unknown vulnerabilities which correspond to bugs or flaws. If those vulnerabilities are exploited, the system's behavior can be manipulated by an attacker. In case of critical systems, this can lead to serious consequences for its users. If the system has Internet access, it might even be possible to remotely exploit these vulnerabilities by bypassing the security of this system. A system designer needs to decide where to deploy additional protections in the system in order to prevent such attacks in the most efficient way. This requires a thorough evaluation of the existing vulnerabilities and the associated risk for the user.

The goal of carrying out a risk analysis is to *quantify* the vulnerability of a system, the existing threats to this system, the attractiveness of this system to an attacker as well as its capacity to attack [26]. The result can be used a guideline for safety and security improvements, and to judge if the risk for the system's users is acceptable or not. Several techniques have been developed to model the different aspects of security and safety in order to highlight the different threats and consequences of an event that may occur.

- **Attack Tree**: Attack trees are used to model in a formal way the security aspects of a system. The attacks are represented in a tree structure, with the goal to achieve as the root node, and actions performed by the attacker as leaf nodes. The internal nodes are logic gates (and/or) combining several actions. With this method, it is possible to model the different steps an attacker will have to perform to reach his goal.

- **Fault Tree**: In a similar manner as attack trees, fault trees are used to represent safety faults that can occur inside a system, and to check the different consequences those errors can lead to.

- **(Dynamic) Reliability Block Diagrams (D)RBD**: Block diagrams are a way to represent a system structurally. Each component of the system is represented as a block connected, either serially or in parallel. Each of these blocks is subdivided into smaller blocks that represent the behavior of the component. Each block is associated with a failure rate. An RBD can be represented as a logical formula in which the blocks in series are connected by *and* and the blocks in parallels by *or*. This can be used to evaluate the consequences of a single component failing.

4.2.1 CVSS

Born from an idea to gather under an international standard a universal way to express the main characteristics related to the vulnerabilities of a system, Common

Vulnerability Scoring System (CVSS) [23] has been created. This standard is used to calculate and express the level of severity of a vulnerability of a system or an asset of this system in a simple way. This vulnerability is expressed through a numerical score between 0.0 and 10.0, for which a score of 10.0 indicates that the studied asset is extremely vulnerable while a score of 0.0 indicates that the asset is not vulnerable at all.

The score is calculated using the CVSS open framework, based on three groups of metrics that represent the basic, temporal and environmental aspects of an asset or a system. In the first group of metrics called "base group", all the necessary and mandatory metrics to calculate the CVSS score are grouped together. This group is split into two subgroups allowing to represent the metrics related to the exploitability of the studied asset and the second grouping the metrics related to the consequences of a successful attack on this asset. The second group, called "Temporal Group", centralizes all the metrics that represent the different characteristics that can evolve over time such as the state of development of an exploit kit on the asset. The third group, called "environmental group" allows to consider the metrics related to the environment in which the asset evolves. This allows to adjust the level of importance given to confidentiality, integrity or availability for the asset taken into account.

The final CVSS score is calculated by combining the metrics related to these three groups. This calculation can be done with the online generator available on the site of the founders of this system: Forum of Incident Response and Security Teams (FIRST), or locally with the description of the computational framework in the specification document. We use the CVSS score as a standardized metric for different input parameters in our model.

4.3 Motivating Example

Connected cars are essentially critical and connected real-time embedded systems, that operate for an average of 15 years. These vehicles can be purchased by virtually anyone with a sufficient budget, including someone looking for a way to attack them. As a consequence, an attacker has time to study a specific vehicle and its defenses in order to discover vulnerabilities and ways to exploit them. An attacker who owns a vehicle can also use this vehicle to gain access to the manufacturer's network in order to mount remote attacks on vehicles on the same network. This could eventually allow to infect an entire fleet.

Regarding the security of connected cars, software updates are difficult to perform on these widely distributed systems with limited computation and communication capacities. This means that car manufacturers must mainly rely on defenses deployed when the vehicle is sold or updated. The asymmetry between attackers and defenders is therefore very significant in the context of connected cars.

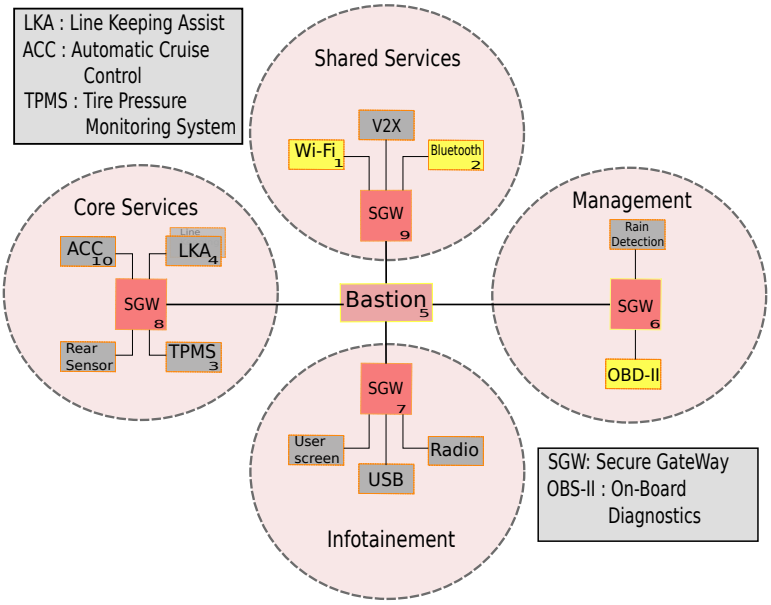

Figure 4.1: Vehicle Architecture scheme

An example of the internal architecture of a connected car is presented in Figure 4.1. This type of architecture is called *architecture by domain* because the different services of the vehicle are separated into four domains according to their role:

1. **Infotainment**: services related to the user experience such as the radio, the on-board screen, and various applications accessible to the user.

2. **Core Services**: critical services for the operation of the vehicle such as cruise control, brake controls, or lane keeping assistant.

3. **Management**: services dedicated to diagnostics and updates of the vehicle, such as the OBD-II *(On-Board Diagnostic)* plug.

4. **Shared Services**: regroups services allowing to communicate with the outside world (V2X) as well as some services shared among several domains.

Within each of these domains, the communication is managed by a Secure Gateway serving as router and firewall. For example, in the Infotainment domain, if the USB module wants to send a message to the user screen module, those messages must go through the domain's Secure Gateway.

All the communication between two separate domains must pass through an entity called the *Bastion*, operating as a *"super" Secure Gateway*. If the Bluetooth module located on the Shared services domain wants to communicate with the user screen module located on the Infotainment domain, all the messages will be examined and filtered by the Bastion. The *Bastion* is basically a router and a firewall for the communication between the domains. Typically, it also embeds an Intrusion Detection System (IDS) in order to detect any attempt of intrusion into the vehicle's system.

This architecture has been designed with safety and security in mind, and the vehicle already possesses some defenses located on the secure gateways. But these defenses are static, i.e. they have a configuration that will remain the same throughout the entire life cycle of the vehicle. If an attacker finds a vulnerability and a way to bypass an existing defense, the attack has a high chance of being reproducible and may be applied to any number of vehicles of the same type.

Consider as an example the access code to the vehicle's Bluetooth module – providing an entry point for attackers that target the vehicles integrity – and the Bluetooth MAC address, which is of interest for attackers that aim to compromise the driver's privacy (e.g. by tracking the vehicle). The Bluetooth access key is generated only once, allowing a trusted device to access the Bluetooth module. It is relatively easy for an attacker to retrieve the access key of this module. The attacker will then be able to use the recovered key to connect to the Bluetooth module and gain access to the CAN bus of the vehicle in order to send forged messages, potentially compromising the integrity of the vehicle. The MAC address of the Bluetooth module is visible in clear to all nearby peripherals. Once the correspondence is made between the MAC address and the associated vehicle, it is possible to follow the comings and goings of this vehicle in the areas monitored by an attacker.

The introduction of MTD on the Bluetooth module can help to limit such attacks, and thus to slow down the progression of an attacker. For example,

we can periodically change the access code to the Bluetooth module. By using a period smaller than the time necessary for an attacker to discover the access code, it becomes harder for an attacker to succeed in connecting to the vehicle's Bluetooth module. Similarly, by periodically changing the MAC address of the vehicle's Bluetooth module, it becomes more onerous for an attacker to maintain a correspondence between a MAC address and a vehicle, making it more difficult to track.

However, there is a drawback of using MTD in a connected vehicle. Indeed, connected vehicles are critical embedded systems with limited computing power and strong time and QoS constraints. The use of MTD on a vehicle will have an impact on these three elements. It is therefore necessary to find a good balance between the protection of the vehicle and the operating constraints related to this type of system. Therefore, we are interested in finding the best strategy for using the different MTD in the vehicle and thus be able to determine the frequency of use of each MTD on each asset, allowing the vehicle to be as well protected as possible against all types of existing and future attacks. As a solution to this problem, we propose a model representing the interactions between a set of attackers and a connected vehicle, as well as all the constraints that the system must respect.

4.4 Related Work

The method of calculating the optimal strategy for MTDs has been the subject of several recent contributions [33, 54, 22, 35, 65]. This topic represents an important challenge for the configuration of MTD as the practical implementation of these techniques requires a trade-off between different constraints such as entropy, frequency, deployment cost, and QoS impact. The studies mentioned above are close to the topic we deal with in this chapter: they all aim at defining an optimal strategy for an MTD, while using a formalization of the problem based on game theory. On the other hand, these works are mainly devoted to the problem of web applications, which generates strong distinctions with respect to our approach. Indeed, the problems of this domain diverge from those of CRES, which leads to different modeling of MTDs. Another example of these domain-specific issues is the research by Burow & al. [12] on the impact of MTD techniques on response time analysis.

Thus, the fundamental differences of this work from the existing state of the art are based on the following criteria: (i) the type of game, (ii) the interpretation of the strategy as a frequency of MTD execution, (iii) the nature of the costs associated with the actions of the defender and the attacker. We clarify these differences throughout this section.

In [54], Sengupta & al. defined a Bayesian Stackelberg game to determine the best mixed strategy to defend a web application by varying its technology stack configuration. This mixed strategy is determined by identifying the most likely attacks and the most exposed configurations. The authors have essentially shown that their approach leads to an improvement of the defense strategy compared to the use of a uniform distribution in order to determine a strategy of configuration, allowing to highlight the legitimacy of the use of an approach based on the game theory. Although the game is comparable to the one we present in this chapter, the definition of the strategy is not clearly associated with the notion of moving frequency. This is due to the fact that their model lacks precision regarding the execution time of an MTD , its impact on the QoS, as well as the probability of success of an attack in the short time. More recently, the mention of the optimal travel frequency was introduced in detail by Li & Zheng in [35]. In this chapter, the game used corresponds to a Bayesian Stackelberg game that is solved from the defender's point of view as a semi-Markovian decision process. Thus, this approach requires defining some of the parameters of the game for each of the configurations of an MTD. For example, the cost of moving from one configuration to another, or the attack time corresponding to a specific configuration. As we have indicated in our introduction, we have mainly oriented ourselves towards

a non-Markovian game. First, significant configuration changes seem to incur a significant cost for reconfiguration, which is too large to be performed during the operation of a CRES. Second, it is still complicated to determine some important parameters for each situation, such as the probability of success of an attack . In a similar vein, Feng & al. [22] defined a defense strategy using a Markov decision process. Therefore, the cost of configuration switching and its associated assumptions are not suitable for application in the CRES framework.

Within the following paper [65], the authors Zhang & al. also study the problem of determining an MTD strategy using a related game theory approach. These authors suggest to rely on learning (Nash-Q) to determine a Nash equilibrium and consequently the defense strategy to adopt. This approach is therefore quite distinct from ours: it aims at finding a Nash equilibrium, which therefore implies a simultaneous game, and at using a learning method, which therefore implies some observability of the reward functions. In such a context, it is rather a question of implementing a reactive defense by combining MTD with an IDS, rather than deploying a proactive defense that changes the configuration periodically.

Connell & al. have also attempted in their paper [19] to approach the problematic related to the frequency of use of an MTD on a system. In their article, their approach is not limited to the very particular contexts of CRES like ours but rather tries to determine this frequency for more "normal" system. For this they based their research on an approach to determine the availability and performance of a resource to which they are using an MTD. This allowed them thanks to a Continous Time Markov Chain (CTMC) to determine the frequency of use of an MTD for a resource based on a trade-off between the probability of success of an attacker and the availability of this resource.

In conclusion, the various existing works in the state of the art have confirmed the validity of game theoretic approaches in the search for an optimal strategy for an MTD. However, these works have been performed in an environment related to web applications, leading to different assumptions, modeling and correction methods. To the best of our knowledge, the work we present is a first attempt to define a game-theoretic approach to determine an optimal strategy for an MTD in the CRES context.

4.5 Model Presentation

In this section, we present our model, starting with the basic structure, and discussing the different input parameters in the following.

4.5.1 Model Structure

The game we use will be represented by the tuple $< N, (M_i)_{i \in N}, P, R_{nmpn'm'}, \widehat{R}_{nmpn'm'} >$ in which:

- $N = \{0, 1, ..., n\}$: the set of nodes of the system under attack.
- $M_i = \{0, 1, .., m_i\}$: the set of MTD defenses present on node i.
- $P = \{0, 1, ..., p\}$: the set of attacker profiles.
- $R_{nmpn'm'}$: reward obtained by the defender when he chooses to use the MTD m on node n while the attacker p targets the MTD m' on node n'.
- $\widehat{R}_{nmpn'm'}$: reward obtained by the attacker p when she chooses to use to target the MTD m' on the node n' while the defender will defend the node n with the MTD m.

The game is composed of a set of nodes N corresponding to the elements present in each domain, such as the Secure GateWay (SGW), the Bluetooth, or the ACC. On each of these nodes, a set of assets is present, corresponding to valuable information or subsystems. Each of these assets is protected by one or several MTDs of the set M_i. The different attackers are represented by the set of attacker profiles P, each of which has the objective to target some assets present on the different nodes, depending on the profile type.

Resolving the game consists in finding the best defense strategy for the defender against the set of attacker profiles taken into account. The decision variables of the problem corresponding to the strategies chosen by the two players and are represented as follows:

- δ_{nm}: The strategy of the defender on the node n and its MTD m.
- $\alpha_{pn'm'}$: The strategy of the attacker of profile p on the MTD m' of node n'.

On each node, the defender has a budget of 1 to spend. The distribution of this budget corresponds to the frequency of use of each MTD over a period of time and will be represented the decision variable δ_{nm}.

Each attacker has a global budget of 1 to spend on the whole game, and will choose only one node to target. As we will consider several types of attackers, each with different objectives to achieve, they will not necessarily all be interested in all the assets present on a node.

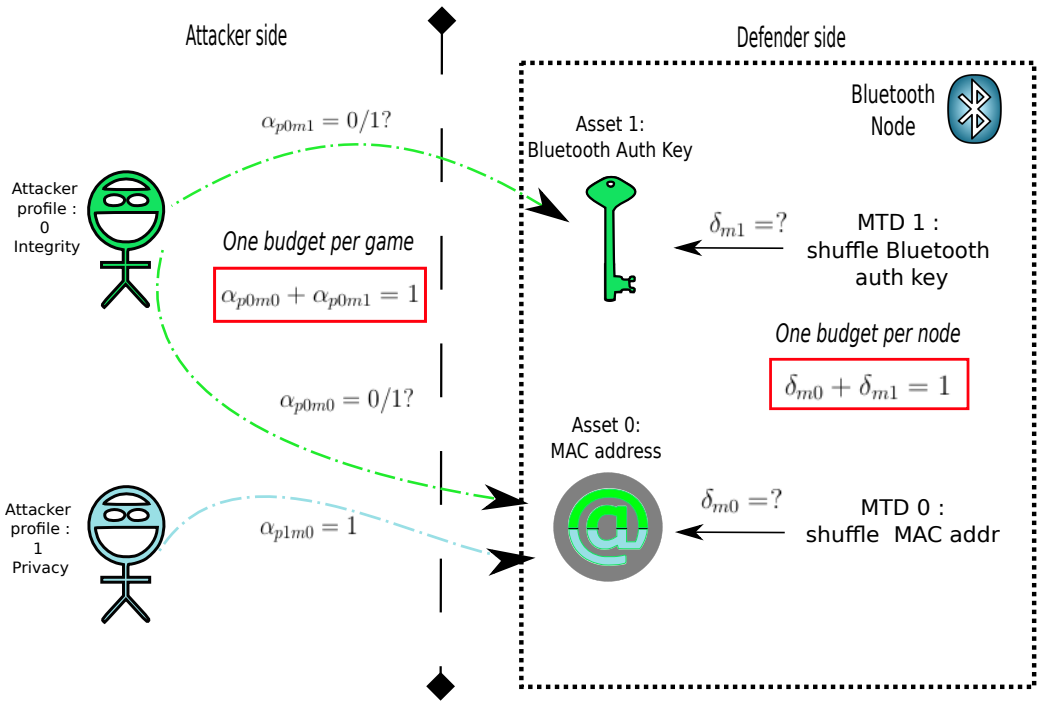

Figure 4.2: Model Representation for one node and 2 attacker profiles

To illustrate our model, Figure 4.2 shows a game composed of one node with two assets to protect and two MTD, as well as two attackers of different types. The attacker of profile 0 is interested in recovering the authentication key of the Bluetooth module as well as the MAC address of the module. The attacker of profile 1 is only interested in the MAC address of the Bluetooth module. Therefore, the attacker of profile 0 will have to choose if she wants to launch an attack on the Bluetooth node via the MAC address or the Bluetooth key. The attacker of profile 1 will always choose to attack the node via the MAC address because it is the only information she is interested in on this node.

The defender needs to decide how to use the two available MTD in the most efficient way knowing what the attackers are interested in. The resolution of the game will allow us to determine the optimal strategy of using the MTD, allowing to defend the system in the best possible way against the different attackers.

4.5.2 Input Parameters

Before defining the reward functions associated to each couple of attacker-defender actions, we begin by identifying the different input parameters of the game. These parameters must determined in order to instantiate the model for a specific use case.

The following parameters describe the attacker side:

$\widehat{Sp}_{nmp}$	Success probability of attacker profile p on node n when MTD m is used.
$\widehat{W}_{np}$	Attacker profile p interest in the node n.
$\widehat{C}_{nmp}$	Cost for the attacker of type p to attack MTD m on node n.
$\widehat{\gamma}_p$	Probability to encounter attacker profile p.

The following parameters describe the defender side:

W_{np}	Defender interest on node n when the attacker profile p targets it.
C_{nm}^{MTD}	Defender cost to use the MTD m on node n.

4.5.3 How To Determine The Parameters

To realistically define a model, we need to characterize the parameters of the game in order to define them correctly. We assume that the following information are known:

- The entropy value associated to each element that we want to protect by an MTD. This corresponds e.g. to the number of valid IP addresses that can be used or the number of MAC addresses available for an element.
- The CVSS score associated with each asset of the system we are considering in the game. This will allow us to know the interest that one of the players may have in this node, the more vulnerable it is, the more it may interest an attacker. This score is computed by taking in account the difficulty to access a specific resource and the requirements to launch an attack.
- For each MTD, the time during which the corresponding service is not accessible if the defense is used.
- For each attacker profile and information to protect, the time needed for an attacker to scan an occurrence of the information.
- For each node, the associated reconfiguration period.

In the following, we will detail different methods used to determine the input parameters defined in the previous section.

Success Probability: $\widehat{Sp}_{nmp}$

We start by defining the success probability of an attacker for a given MTD and node.

First of all, when an attacker of profile p is not interested in the asset or information protected by MTD m on node n, we fix the corresponding success probability to 0 in order to incite the attacker not to target this asset. If the attacker is indeed interested in an asset or information; there are several ways to determine the value of the success probability:

1. When the MTD m used is of type shuffle, we apply the *urn statistical model* [14], to solve the problem of *Drawing with Replacement*. In our case, the number of attempt is equal to the node period divided by the time needed to scan one configuration. The formulation of this problem is given by the equation (4.5), where x is the number of attempts, h is the number of instances of the asset, a the number of available values. The probability to find the information can then be calculated as

$$\widehat{Sp} = 1 - ((a-h)/a)^x \tag{4.5}$$

2. When the MTD m is not of shuffle type, and a method to bypass this MTD exists, the attacker's success probability is equal to the duration of the reconfiguration period of the node n divided by the time to bypass the MTD. If the duration of a period is greater, the attacker's success probability is equal to 1.

3. When the MTD m used is neither of type shuffle, nor has a method to bypass, we will need to resort to an adhoc method to estimate the success probability.

Example: Consider an attacker trying to find the IP address of a module on an IPV4 sub-network. On this particular sub-network 252 addresses are valid and usable by a module.

In order to estimate the time an attacker needs to scan the network, we consider a well-known open source tool for scanning networks: nmap. In fact, nmap can be used to discover hosts and services on a network by sending packets and analyzing the responses. It can also provide further information on targets (e.g. reverse DNS names, device types, MAC addresses, etc.). Using nmap with highly optimized options, the time needed to scan one IP address is approximately 255 ms [37].

Considering that the defender can change the IP address of the module sought by the attacker every 10 seconds, the attacker will have $10/0.255 = 39,21$ attempts to find the correct address. Assuming that the module is the only element present on the network, according to formula (4.5), the attacker's probability of success in discovering the IP address of the module is

$$\widehat{Sp} = 1 - ((a-h)/a)^x$$
$$\widehat{Sp} = 1 - ((252-1)/252)^{39}$$
$$\widehat{Sp} = 1 - 0,856355413 = 0,143644587$$

Attacker Gain: $\widehat{W}_{np}$

The interest of an attacker of profile p for a node n will depend on the type of information contained on this node. If there is at least one piece of information on node n that could be of interest to the attacker, the value of the attacker's interest p for this node will be equal to the corresponding CVSS score. The higher the CVSS score for a node's asset, the easier and more interesting it will be for an attacker to launch an attack on it. If on the other hand none of the information of node n is of interest to the attacker p, the interest of the attacker p for this node will be equal to 0.

Attack Cost: $\widehat{C}_{nmp}$

The definition of the cost related to an attack depends on the type of MTD m used on the node n. If this one corresponds to a shuffle MTD, the cost of an attack will be equal to the cost of launching a scan multiplied by the number of scans that can be performed during the reconfiguration period of the node. If the MTD used does not correspond to a shuffle type MTD, the cost will correspond to the cost of using the MTD bypass method.

Attacker Appearance Probability: $\widehat{\gamma}_p$

The definition of the probability of appearance of an attacker can be done in two ways. If we have access to the history of different attacks that have already taken place on the same type of system, it is possible to extract the probability of occurrence of an attacker profile. Obtaining this kind of information is difficult, since car manufacturers typically so not share such information with the public.

Therefore, if this type of history is not available or does not exist, we propose to use an exponential distribution, depending on the level of expertise of the attacker.

This reflects the fact that there are many beginners, some serious attackers and very few experts.

Defender Node Gain: W_{np}

The interest of a node for the defender to defend against an attacker p will depend on the node n as well as the information contained on this node.

If none of the information on node n is of interest to the profile p attacker, the interest of this node for the defender will be equal to 0. If at least one of these pieces of information is of interest to the profile p attacker, the defender's interest in this node will be equal to the corresponding CVSS score. The higher the CVSS score for a node's asset, the more impact the loss of that asset will have for the defender and the greater the need for defense.

MTD Usage Cost: C_{nm}^{MTD}

The cost of using the MTD m on node n is equal to the downtime of the node induced by the use of the MTD divided by the duration of a reconfiguration period of the node.

4.6 Game Formalization

4.6.1 Game Form

As explained in the previous section, we have to model the problem by taking into account the interaction between several attackers and a system as well as the different constraints related to the system used.

To do this, we must first take into account the asymmetry between an attacker and the system, which is translated by the fact that an attacker will choose on which part of the system to launch an attack once she has observed the defenses used by the system. We also need to consider the fact that several types of attackers are interested in the system, each with different objectives and means. The problem is that it is not possible to determine which of these attackers will appear and choose to launch an attack at which time.

The game theoretic concepts presented in section 4.1 allow us to represent these different interactions and to take into account the different constraints related to the system: The asymmetry between the attacker and the defender gives rise to a Stackelberg game [18] allowing to impose an order in the decision of the actions. Bayesian games [54] allow us to represent the fact that we cannot determine the type(s) of attacker(s) to defend against and that we are looking for a strategy that will allow us to defend optimally against all the types of attackers considered.

4.6.2 Reward

For each combination of actions attacker-defender on each node and MTD, a reward function allows to compute the reward corresponding to this specific combination. There is one reward function for each player. The higher the reward obtained for one action, the more the player will be interested in performing this action in the chosen context. The computation of these reward functions is done by calculating the gain of performing the action minus the cost of performing this action.

The value of the gain of a player depends on the action of the other player. In contrast, the cost of using an action does not depend on the action taken by the other player.

For the attacker, the gain of an action is defined as follows.

- If the attacker p and the defender target the same node n and n' and MTD m and m' at the same time, the associated gain for the attacker is 0.
- If the attacker p and the defender do not target the same node n and n' and MTD m and m', the associated gain for the attacker p is equal to his success probability multiplied by his interest for the node n'.

The cost of performing an action for the defender corresponds to the parameter $\widehat{C}_{n'm'}$.

The reward function of the attacker is then of the following form:

$$\widehat{R}_{nmpn'm'} = \begin{cases} -\widehat{C}_{n'm'}, & \text{if } n = n' \text{ and } m = m' \\ (\widehat{Sp}_{n'm'p} * \widehat{W}_{np}) - \widehat{C}_{n'm'}, & \text{if } n \neq n' \text{ or } m \neq m' \end{cases} \tag{4.6}$$

On the defender's side, the gain obtained for performing an action is defined as follows.

- If the attacker p and the defender target the same node n and n' and MTD m and m' at the same time, the associated gain for the defender is equal to the probability of success of the attacker p multiplied by the interest of the defender for the node.
- If the attacker p and the defender do not target the same node n and n' and MTD m and m', the associated gain for the defender is equal to 0.

The cost of performing an action for the defender will correspond to the parameter C^{MTD}_{nm}.

The reward function of the defender is then of the following form:

$$R_{nmpn'm'} = \begin{cases} (\widehat{Sp}_{n'm'p} * W_{np}) - C^{MTD}_{nm}, & \text{if } n = n' \text{ and } m = m' \\ -C^{MTD}_{nm}, & \text{if } n \neq n' \text{ or } m \neq m' \end{cases} \tag{4.7}$$

For each pair of attacker-defender actions, on each node/MTD, the corresponding reward function must be defined. The reward functions are composed of the gain for performing an action minus the cost of performing this action. The values of the rewards will be defined according to the location targeted by the two players. The cost of an action remains the same regardless of the target chosen by the other player.

4.6.3 Payoff Function

The rewards functions are used to indicate for each pair of actions and nodes, the associated rewards. In order to determine the best possible strategy for the defender, he needs a payoff function to calculate the maximum reward he can obtain.

This function will depend on the rewards functions and on the decision variables α and δ representing the strategy for the attacker and the defender.

The payoff function will be the function that once maximized allows to obtain the strategy that gives the biggest possible reward to the defender. The payoff function corresponds to the sum of all the rewards functions according to the value of the associated α and δ strategies and has the following form:

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\delta_{nm} * [\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}(\widehat{\gamma_p} * \alpha_{pn'm'} * R_{nmpn'm'}) - C_{nm}] \tag{4.8}$$

4.6.4 Complementary Slackness

As presented in section 4.1.6, the complementary slackness is used to constrain each attacker to maximize her payoff function. The payoff is expressed by the following equations 4.9.

$$\begin{aligned}
&\forall_{p\in\mathcal{P}} \max_{\alpha_{np}} \sum_{n\in\mathcal{N}} [\alpha_{np}[\sum_{m\in\mathcal{M}} \delta_{nm}(\widehat{R}_{nmp} - \widehat{C}_{nmp}) + \delta_{nd}(\widehat{R}_{ndp} - \widehat{C}_{ndp})]] \\
&\forall_{p\in\mathcal{P}} \sum_{n\in\mathcal{N}} \alpha_{np} = 1 \\
&\forall_{p\in\mathcal{P}}\forall_{n\in\mathcal{N}}\alpha_{np} \geq 0
\end{aligned} \tag{4.9}$$

It is then possible to transform the primal problem 4.9 into its dual 4.10. Here, the function aims at finding for each profile p the smallest value of the variable a_p that will be equal to the maximum reward that the attacker p can obtain given the strategy chosen by the defender.

$$\begin{aligned}
&\forall_{p\in\mathcal{P}} \min a_p \\
&\forall n \in \mathcal{N}, \forall p \in \mathcal{P}\ a_p \geq \sum_{m\in\mathcal{M}} \delta_{nm}(\widehat{R}_{nmp} - \widehat{C}_{nmp}) \\
&+ \delta_{nd}(\widehat{R}_{ndp} - \widehat{C}_{ndp})
\end{aligned} \tag{4.10}$$

Using strong duality and complementary slackness, these two problems are transformed into constraint (4.11), which must be satisfied when solving the optimization problem for the leader (defender) in order to consider only best responses by the follower (attackers).

$$\begin{aligned}
&\forall_{p\in\mathcal{P}}\forall_{n\in\mathcal{N}}, 0 \leq a_p - \sum_{m\in\mathcal{M}} \delta_{nm}(\widehat{R}_{nmp} - \widehat{C}_{nmp}) \\
&+ \delta_{nd}(\widehat{R}_{ndp} - \widehat{C}_{ndp}) \leq (1 - \alpha_{np})M
\end{aligned} \tag{4.11}$$

This constraint is added to the optimization problem for the defender, where M is a large integer and a_p is a free variable.

4.6.5 MIQP

Now that we have defined the payoff function of the defender, the different parameters, the constraints that we want the model to respect, and the payoff function, we can combine all these elements in an MIQP. This program will allow us to find the strategy δ which maximizes the reward obtained for the defender (eq.4.12) and so to obtain the best strategy of use of the MTD present on the vehicle.

$$obj : \max_{\delta_{nm},\alpha_{pn'm'},a_p} \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \delta_{nm} * [\sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} (\widehat{\gamma_p} * \alpha_{pn'm'} * R_{nmpn'm'}) - C_{nm}] \tag{4.12}$$

$$C1 : \forall_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \delta_{nm} = 1 \tag{4.13}$$

$$C2 : \forall_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} \alpha_{pn'm'} = 1 \tag{4.14}$$

$$C3 : \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} 0 \leq (a_p - \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \widehat{R}_{nmpn'm'} * \delta_{nm})) \leq (1 - \alpha_{pn'm'})M \tag{4.15}$$

$$\forall_{n\in\mathcal{N}} \forall_{m\in\mathcal{M}} \delta_{nm} \in [0,1] \tag{4.16}$$

$$\forall_{p\in\mathcal{P}}, \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} \alpha_{pn'm'} \in \{0,1\} \tag{4.17}$$

$$\forall_{p\in\mathcal{P}} a_p \in \mathbb{R} \tag{4.18}$$

The complete MIQP is shown in the equations (4.12) to (4.18). It is written in such a way that the defender will defend each node independently (4.13) with a budget of 1 per node (4.16). It also integrates the fact that the attacker will have a budget of 1 to spend on the whole game (4.14) and that he can choose only one target on the game (4.17)). The constraint (4.15) corresponds to the expression of the complementary slackness allowing to force each attacker to choose the target bringing him the biggest reward by taking into account the strategy of the defender.

Unfortunately, a MIQP is complicated to solve. There are very few solvers available and the underlying algorithms do not scale well. In the next section we will show how we transform it into a MILP to make its resolution feasible.

4.7 Game Resolution

4.7.1 MIQP To MILP Transformation

We are going to transform the MIQP presented in the previous section into a MILP. To begin with, the process of transforming a MIQP into a MILP consists in going from a quadratic program with several decision variables multiplied together to a linear program in which there are no decision variables multiplied together. To do this, we factorise the two variables α and δ to obtain a new one named Z through the following transformation:

$$Z_{nmpn'm'} = \delta_{nm} * \alpha_{pn'm'} \tag{4.19}$$

We need to make sure that we can recover the values of α and δ once the MILP is solved, which is done through the following constraints:

$$\delta_{nm} = \sum_{p \in \mathcal{P}} \sum_{n' \in \mathcal{N}} \sum_{m' \in \mathcal{M}} Z_{nmpn'm'} \tag{4.20}$$

$$\alpha_{pn'm'} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} Z_{nmpn'm'} \tag{4.21}$$

We begin by transforming the objective function of the problem, by a new version containing the Z. We start by taking the initial function ((4.22)) that we have expanded into (4.23), in order to be able to replace the values of α and δ present thanks to the equations ((4.19), (4.20)) in order to obtain the new objective function of the problem (4.24).

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\delta_{nm} * [\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}(\widehat{\gamma_p} * \alpha_{pn'm'} * R_{nmpn'm'}) - C_{nm}] \tag{4.22}$$

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}[\delta_{nm} * \widehat{\gamma_p} * \alpha_{pn'm'} * R_{nmpn'm'}] - \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}[\delta_{nm} * C_{nm}] \tag{4.23}$$

$$\begin{aligned}&\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}[\widehat{\gamma_p} * Z_{nmpn'm'} * R_{nmpn'm'}]\\ &-\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}[(\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}Z_{nmpn'm'}) * C_{nm}]\end{aligned} \tag{4.24}$$

4.7.2 MILP

The new MILP must respect the same constraints as those of the MIQP presented in section 4.6.5, adapted with the new Z variables. The solution of this program must allow to obtain the same optimal strategy as the one that would have been given by the MIQP. This gives the following MILP:

$$obj : \max_{Znmpn'm', \alpha_{pn'm'}, a_p} \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} [\widehat{\gamma_p} * Z_{nmpn'm'} * R_{nmpn'm'}]$$
$$- \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} [(\sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} Z_{nmpn'm'}) * C_{nm}] \quad (4.25)$$

$$C1 : \forall_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}_{-idle}} Znmpn'm' = 1 \quad (4.26)$$

$$C2 : \forall_{n\in\mathcal{N}} \forall_{m\in\mathcal{M}}; 0 \leq \sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}_{-idle}} Znmpn'm' \leq 1 \quad (4.27)$$

$$C3 : \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} \forall_{p\in\mathcal{P}}; \alpha_{pn'm'} = \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} Z_{nmpn'm'} \quad (4.28)$$

$$C4 : \forall_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} \alpha_{pn'm'} = 1 \quad (4.29)$$

$$C5 : \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} 0 \leq (a_p - \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \widehat{R}_{nmpn'm'} * (\sum_{p\in\mathcal{P}} \sum_{n"\in\mathcal{N}} \sum_{m"\in\mathcal{M}} Znmpn"m"))$$
$$\leq (1 - \alpha_{pn'm'})M \quad (4.30)$$

$$\forall_{p\in\mathcal{P}}, \forall_{n\in\mathcal{N}} \forall_{m\in\mathcal{M}} \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} Znmpn'm' \in [0, 1] \quad (4.31)$$
$$\forall_{p\in\mathcal{P}}, \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}} \alpha_{pn'm'} \in \{0, 1\} \quad (4.32)$$
$$\forall_{p\in\mathcal{P}} a_p \in \mathbb{R} \quad (4.33)$$

The first constraint (4.26) consists in limiting the defender's budget per node to 1. We want to make sure with the second constraint (4.27) that the value that a δ can take will never exceed 1. Constraints 3 (4.28) and 4 (4.29) of the MILP combined together ensure that an attacker will have a budget of 1 to spend on the whole game and that the values of the different α can be found in the values of Z. The last constraint (4.30) is the complementary slackness which ensures that each attacker profile will choose the node to target with the highest reward according to the strategy chosen by the defender.

4.7.3 Correspondence between the MIQP and the MILP

In order to show that the MILP is indeed equivalent to the presented MIQP representing the game and its constraints, we will consider all the constraints of the MIQP and show that we have their equivalent in the MILP.

We start with the **objective function** (4.12), whose transformation from MIQP to MILP has been presented with equation (4.22),(4.23) and (4.24). This

shows the equivalence between the two objective functions, showing that we are trying to solve the same problem.

Consider the **MIQP constraint C1** (4.13): $\forall_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \delta_{nm} = 1$
Starting from equation (4.20), we can replace the delta of the MIQP constraint C1 (4.13) by the corresponding Z, this allows us to obtain the new constraint C1 of the MILP (4.26) limiting the budget of the defender to 1 per node.

We move on to the constraint **MIQP constraint C2** (4.14):
$\forall_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}} \alpha_{pn'm'} = 1$
There is the MILP constraint C4 (4.29) which is exactly the same as the MIQP constraint, but it does not constrain in any way the values that Z can take. Using equation (4.21), this allows us to link the values of the α with the one of the corresponding Z, which gives the constraint C3 of the MILP (4.28). By combining these two constraints, we constrain the values of Z in such a way that the attacker will have a budget of 1 to spend on the whole game.

For the constraint **MIQP constraint C3** (4.15): $\forall_{n'\in\mathcal{N}}\forall_{m'\in\mathcal{M}} 0 \leq (a_p - \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \widehat{R}_{nmpn'm'} * \delta_{nm})) \leq (1 - \alpha_{pn'm'})M$
We start from the original constraint in which we use equation (4.19) to replace directly the $\alpha * \delta$ in Z in the constraint and equation (4.20) to replace the δ alone to arrive at the new constraint C5 of the MILP (4.30) representing well the same complementary slackness as that of the MIQP.

We finally want to make sure that the value of a δ included in the Z cannot exceed 1. This is done thanks to the constraint C2 of the MILP (4.27). This constraint indicates that the value of a δ included in a Z will not exceed 1.

We have just shown that a solution for the MILP will correspond to a solution for the MIQP, and that we can use it to find an optimal strategy for the defender. However, this will not necessarily be true in the other direction. The passage from MIQP to MILP introduces a loss of expressivity because we go from 2 distinct variables δ and α, to 1 used to represent them both Z. This restricts our model in the case where the number of nodes that we take into account in the MILP is smaller than the number of attacker profiles considered.

However, this limitation does not have an impact on the use cases of our model. The number of attackers taken into account when creating a model is fixed. In our type of applications, we generally consider 2 types of attackers, integrity and privacy, each having 5 levels of expertise. This results in 10 attacker profiles. The number of assets to defend in an automotive system is around a hundred. The

number of nodes in the system will therefore always be greater than the number of attacker profiles. The limitation induced by the transition from MIQP to MILP will therefore not be blocking in our model.

4.8 Experimentation

4.8.1 Experimental Case

In order to present a use case of our model, we start with the architecture presented in Figure 4.3 as a model for the game.

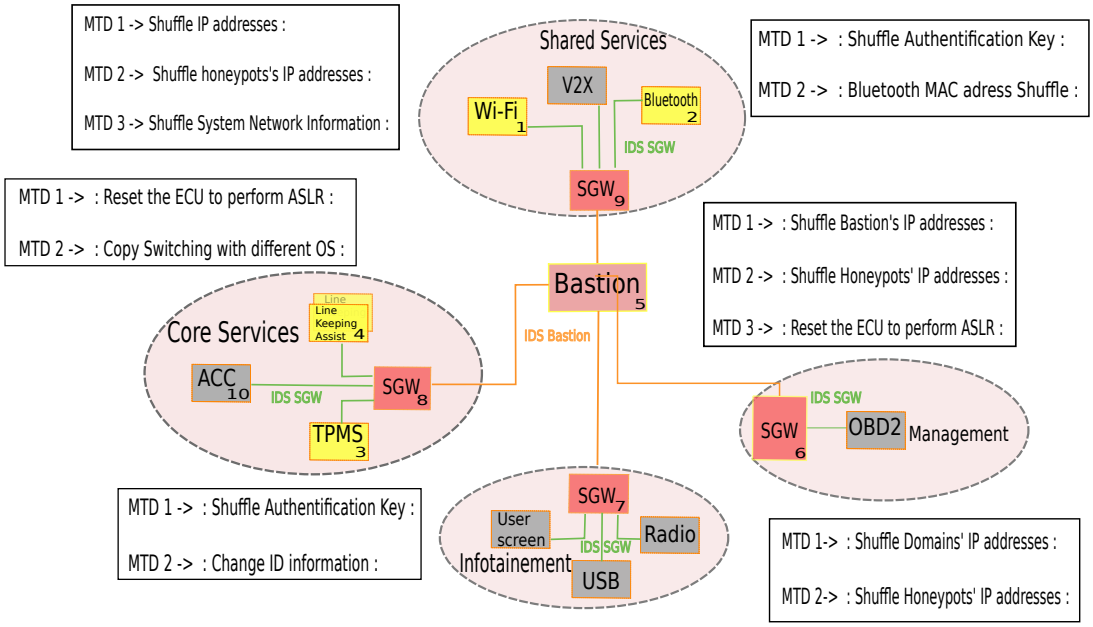

Figure 4.3: Full game representation

This architecture corresponds to a game containing 10 nodes, each having between two and three MTD for defense, with ten attacker profiles taken into account when developing the strategy. We generated the MILP corresponding to this architecture that we resolved using the CPLEX tool. We obtain the following defense strategy presented in 4.34 in which the defender will defend each node and MTD targeted by an attacker by spending its budget of 1 per node. And the attacker strategy presented in 4.35. The time required to compute a solution to the problem with CPLEX is 9.2 seconds.

$$
\begin{array}{llll}
\delta_{n0_{m}0} = 0.351 & \delta_{n0_{m}1} = 0.0 & \delta_{n0_{m}2} = 0.648 & \delta_{n0idl} = 0.0 \\
\delta_{n1_{m}0} = 0.95 & \delta_{n1_{m}1} = 0.05 & \delta_{n1_{m}2} = 0.0 & \delta_{n1idl} = 0.0 \\
\delta_{n2_{m}0} = 0.0 & \delta_{n2_{m}1} = 1.0 & \delta_{n2_{m}2} = 0.0 & \delta_{n2idl} = 0.0 \\
\delta_{n3_{m}0} = 1.0 & \delta_{n3_{m}1} = 0.0 & \delta_{n3_{m}2} = 0.0 & \delta_{n3idl} = 0.0 \\
\delta_{n4_{m}0} = 0.403 & \delta_{n4_{m}1} = 0.310 & \delta_{n4_{m}2} = 0.285 & \delta_{n4idl} = 0 \\
\delta_{n5_{m}0} = 0.0 & \delta_{n5_{m}1} = 1.0 & \delta_{n5_{m}2} = 0.0 & \delta_{n5idl} = 0.0 \\
\delta_{n6_{m}0} = 0.0 & \delta_{n6_{m}1} = 1.0 & \delta_{n6_{m}2} = 0.0 & \delta_{n6idl} = 0 \\
\delta_{n7_{m}0} = 0.0 & \delta_{n7_{m}1} = 0.921 & \delta_{n7_{m}2} = 0.078 & \delta_{n7idl} = 0 \\
\delta_{n8_{m}0} = 0.0 & \delta_{n8_{m}1} = 1.0 & \delta_{n8_{m}2} = 0.0 & \delta_{n8idl} = 0 \\
\delta_{n9_{m}0} = 0.519 & \delta_{n9_{m}1} = 0.480 & \delta_{n9_{m}2} = 0.0 & \delta_{n9idl} = 0
\end{array} \tag{4.34}
$$

$$
\begin{array}{lllll}
\alpha_{p0_{n}0_{m}0} = 1.0 & \alpha_{p1_{n}9_{m}1} = 1.0 & \alpha_{p2_{n}4_{m}1} = 1.0 & \alpha_{p3_{n}1_{m}0} = 1.0 & \alpha_{p4_{n}7_{m}2} = 1.0 \\
\alpha_{p5_{n}0_{m}2} = 1.0 & \alpha_{p6_{n}9_{m}0} = 1.0 & \alpha_{p7_{n}4_{m}0} = 1.0 & \alpha_{p8_{n}7_{m}1} = 1.0 & \alpha_{p9_{n}1_{m}1} = 1.0
\end{array} \tag{4.35}
$$

4.8.2 Scaling Tests

Random Generator

In order to investigate if the proposed solution scales well, we have generated random scenarios of different size [1].

During our experiments, we realized that generating the parameters in a totally random way corresponds to the worst case scenario for the defender in which all the attacker profiles are interested by all the nodes and MTD of the game. This slows down the solution search and limits the size of a game to 15 attacker profiles for 15 nodes.

To resolve this issue and to have a more realistic scenario generator, we limit the interest of an attacker to two thirds of the MTD present on the nodes in a random way.

The results obtained are summarized in Figure 4.4, in which we display the computation time taken by CPLEX to solve the problems. The scenarios are generated in such a way to have as many attacker profiles as nodes. With the parameters of the game generated randomly. This way we get an execution time for a scenario with 10 nodes and 10 attacker profiles of the same range as the one presented in 4.8.1.

[1] The generating tool we made for the experimentation is available for clone here : https://gitlab.telecom-paris.fr/TheseMA/tool_for_journal.git

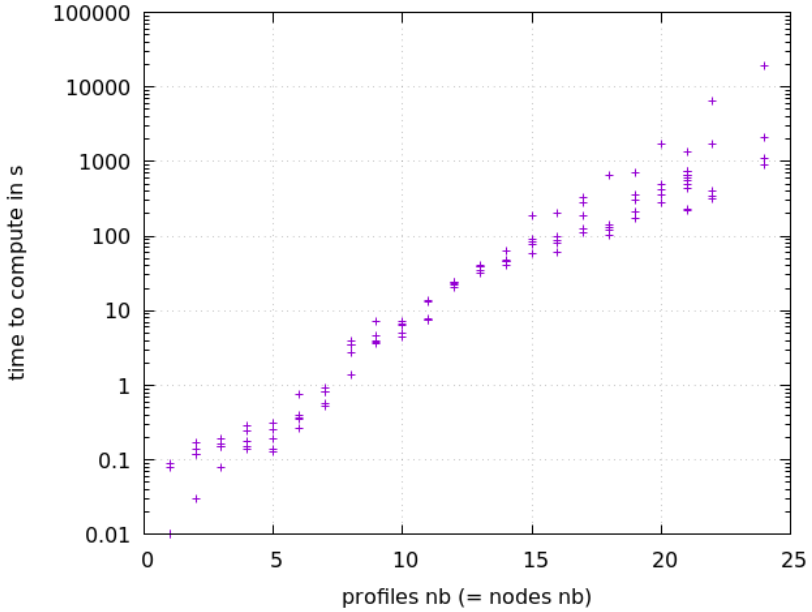

Figure 4.4: Scaling representation with attacker profiles number = node number

We manage to reach reasonable times with games composed of 25 nodes and 25 attacker profiles. An automotive system is made up of an average of a hundred assets that we will try to protect, this amounts to about 25 nodes per domain. If we consider that the game corresponds to the strategy to be defined on a domain, this is not disadvantageous on the attacker's side, who finds himself attacking each domain once. On the defender's side it does not change anything in the principle of finding the strategy.

Fixed Attacker Profiles Generator

In a realistic scenario, it is more common to choose a fixed number of attacker profiles defined in advance according to the type of identified attack scenario being considered. We can look for the best possible strategy by considering that there will only be privacy attackers interested by the vehicle, and that among these types of attackers, there will be 5 levels of expertise taken into account expert/high/medium/low/beginner. If we now consider that we have privacy and integrity attackers targeting the system, with 5 levels of expertise each, we arrive at 10 attacker profiles to take into account in the game.

In the case we want to be even more precise and we know 10 specific attackers trying to attack us in addition to the 10 profiles previously taken into account, we arrive at 20 attacker profiles.

We wanted to check up to how many nodes we could consider in order to find a solution, this is represented on the figure 4.5. As can be noticed, there are no scenarios where the number of nodes is smaller than the number of attacker profiles considered. This is due to the form of our model in which the attacker has a global budget for the game while the defender has one budget per node. This leads to an infeasibility of the problem because of constraints (4.26), (4.28) and (4.29) of the MILP.

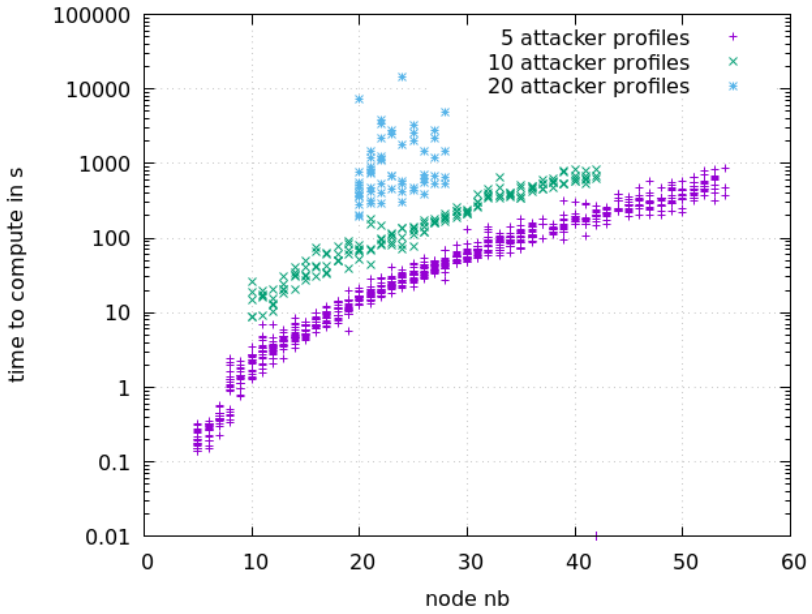

Figure 4.5: Scaling representation with fixed attacker number profiles and increasing node number

By looking at the results of the scaling test with different types of attacker profiles considered, we notice that by taking into account 1 type of attacker (5 profiles), it is possible to scale up to 55 nodes with 3 different MTD each. On systems where only one type of attacker is considered, this type of model allows to calculate a strategy for the defender in a reasonable time.

For systems in which 2 types of attackers are taken into account (10 profiles), it is possible to find a solution in a reasonable time for 40 nodes each having 3 MTD. This corresponds to finding a strategy for a domain of a car-like system.

On the other hand, for systems taking into account a larger number of attacker profiles (20 profiles), we are limited to 30 nodes to find a strategy in a reasonable time. This shows the computational limits of the proposed solution.

4.8.3 Stability Analysis

We realized a stability analysis on different parameters of our game. As it is difficult to characterize some parameters, we wanted to increase the confidence we have in our model in case of an approximation error in the definition of some parameters such as the rate of appearance of an attacker profile, or the cost of an attack for an attacker.

We therefore started with the parameter γ representing the rate of appearance of an attacker profile. We then varied the ratio between the two types of attackers (privacy and integrity) in order to see the impact that this would have on the defender's strategy. We started with a configuration of the game similar to the one presented in the example case, 10 nodes each having between 2 and 3 MTD being targeted by 10 attacker profiles. The results of this experiment for 4 of the nodes in our game are presented in figure 4.6.

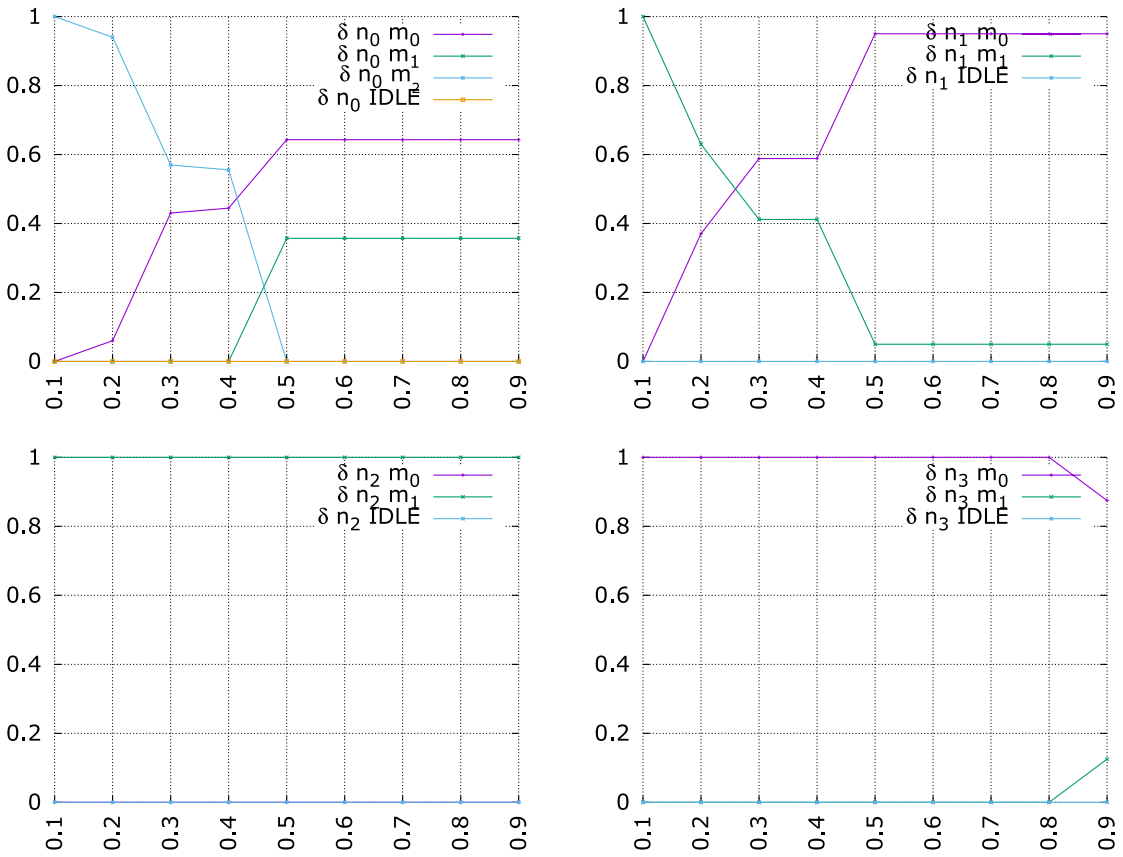

Figure 4.6: How moving the ratio between attacker profiles (gamma) affect the defender strategy

We notice that the variation is linear, and that the defender's strategy reacts in a normal way to the change of ratio between the attackers. On a node like number 0 and 1 on which more attacker profiles target them, depending on the ratio of attacker profile, the strategy evolves in order to defend more efficiently against the type of attacker becoming more present.

On nodes such as node number 2, targeted by no attacker type, this does not change the strategy of the defender, and on the fourth node, when the ratio of the second attacker type becomes more present, the strategy of the defender starts to adapt.

In another experiment, we consider the stability of the given strategy according to a set of input parameters. As a starting point, we choose a fixed scenario using the configuration of the example case. Then we randomly modify the value of the costs and gain for the defender and the attacker as well as the attacker's success probability by a certain percentage. The results are shown in Figure 4.7 as box plots representing the variation of the defender's strategy over 100 different scenarios.

We notice that the strategy variation remains quite stable up to 5%. At this moment, on the node 0 where the balance between the two attackers is borderline, we notice a change of strategy. But we notice that the medians of the strategies on this node are still located in the same order of magnitude, indicating that the strategy remains mostly located in the same area. From 10% we start to notice a more important variance in the strategy of the defender, with an average strategy always located in the same region. We can say that our model reacts in a sane and expected way to the variations in the parameters and is reasonably stable. This allows us to have confidence in the final strategy taking into account a small possible margin of error in the definition of the parameters.

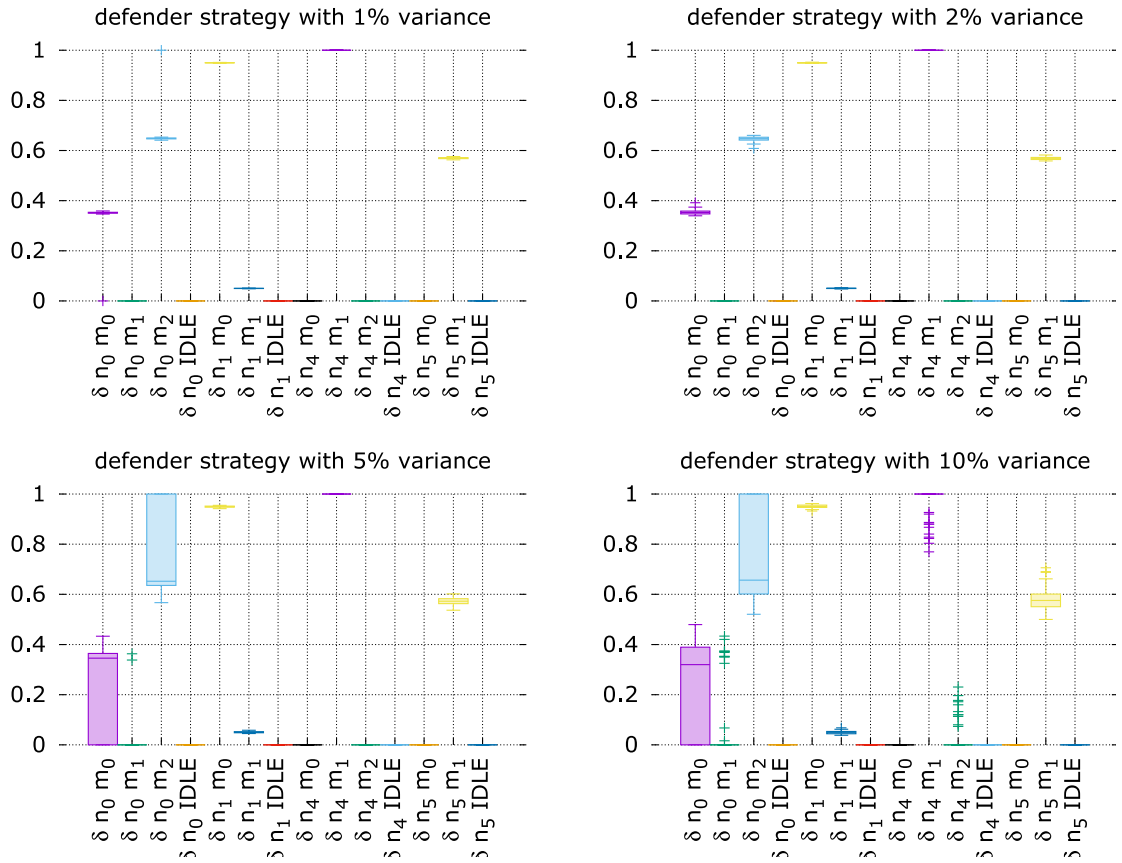

Figure 4.7: How stable the strategies are considering variance in the parameters

4.9 Discussion And Limitations

We have presented a method to find an optimal strategy for using MTD-based defenses in a critical embedded system. The proposed model takes into account the cost of performing an action for the defender in order to use only defenses that protect the system as efficiently as possible and thereby limit quality of service degradation.

Compared to a previous version [9], the presented model has a fine-grained notion of assets and their respective defenses. This allows for a more realistic modeling of the system under attack, as demonstrated with an automotive architecture. During our research, we have identified the input parameters as a critical issue in the modeling process. While some of these parameters can be determined with relatively high confidence – following the explanations in Section 4.5.3 – some inputs remain difficult to estimate. Indeed car manufacturers have little interest in making public their statistic on the frequency of attacks. For this reason, we need to resort to ad hoc estimations.

Another consequence of the refined model is the increased computational complexity of the game resolution. With state of the art solver techniques, the limit of our approach is reached for ten attacker profiles at about 55 nodes. If we consider the number of computational units in a modern car architecture, this is a reasonable size for either a simplified model – taking into account only a subset of nodes of interest – or a complete model of a single subdomain. The method proposed in this article is applied *offline*. Therefore, run-times of one hour or more are acceptable, since the optimal solution needs to be calculated only once. It is left to future work to consider *online* techniques that would recalculate and adapt the defense strategy when new knowledge on attacks or vulnerabilities becomes available.

Regarding the stability of the computed strategy, the results are satisfactory. The observed changes in the output strategies are mostly expected and explicable, but the experiments have also shown that in rare cases, strategies can switch between extremes. In order to detect such local instabilities, conducting such random experiments is recommended as an additional step in order to increase the confidence in the obtained results.

4.10 Conclusion

In this chapter, we have developed a model to represent the interactions between a system and a set of different attacker using a Bayesian Stackelberg game. Our method gives the optimal defense strategy for the defender against the considered attackers. This allows the optimal use of MTD on critical embedded systems, taking into account their limited resources. The game can be resolved using off-the-shelf solvers thanks to an MILP formulation. Our experiments show that the method works for realistic use cases from the automotive domain, and the results exhibit a good stability.

Chapter 5

Conclusion

Je vois refléter dans mon miroir tout mon passé et tout mon avenir.

J. Cortázar.

During this thesis, we studied connected vehicles and the improvement of their resilience to make them more resistant against cyber attacks. We noticed that the current vehicle defenses are mostly static, which leads to an asymmetry between the vehicles and the different types of attackers. So the goal was to study dynamic defenses in order to break this asymmetry and thus improve the resilience of the vehicles. In particular, we considered different types of MTD.

As a first contribution, we have presented a new MTD, to protect the Wi-Fi access of a vehicle. This technique consists in slowing down an attacker during his recognition attack phase and thus making the tracking of the vehicle more complicated. This technique randomly changes the active IP address of a vehicle, forcing an attacker to restart his recognition phase in order to rediscover the new IP address of the vehicle each time he wants to exploit it. We maintain the availability and continuity of services on the asset using this method through the use of multipath TCP. These results have been presented at the MTD workshop 3.

Connected vehicles are critical embedded systems, with limited computational power and time constraints to ensure the proper functioning of critical services. When incorporating new defense methods such as MTDs into a vehicle, one must be sure that it will not affect the user experience by slowing down the entertainment features. And more importantly, that it will not affect the critical functions of the vehicle and cause it to malfunction during use. In order to determine the optimal use of MTD, we have defined a mathematical model to represent the interactions between the vehicle and the different types of attackers. We used this

model to find the best possible defense strategy to respect the various constraints of the vehicle while defending it in an optimal way against any type of attack and attackers. This work has been published in a conference paper [9] and in a journal article [8].

Before being used in practice, the type of MTD based solutions that we developed during this thesis would need to pass through safety certifications to ensure that the critical functions of the vehicles will still operate correctly. Each MTD used would also have to go through a phase of adaptation to the constraints related to the limitation of computing power of vehicles to make them usable without disturbing the operation of the system on which they will be used. The impact of these new defenses on vehicles could lead to the use of larger ECUs in vehicles to manage the newly added defenses more efficiently. This could have an impact on the final selling price of the vehicles. Security always comes at a cost.

In this thesis, we have decided to focus only on rational attackers acting in their own best interest and trying to get the biggest possible "reward" depending on the situation. We did not study the effects that this would have on the efficiency of our strategy if the attacker did not behave in a rational way. Would our strategy remain optimal if an attacker chose to attack an asset that is less defended, not necessarily interesting or easier to corrupt?

This leads us to possible research on the usage of mechanisms such as IDS or machine learning to dynamically change the strategy used on a vehicle according to the attackers actually encountered. Thus allowing to adapt to the real environment in which the vehicle evolves and thus obtain the best possible defense against any attacker behavior, rational or not.

In the light of the existing attacks and what can be done remotely on a vehicle, the question of the development of self-driving vehicles may become a legitimate one: Do we really want to leave the full power on the driving of a vehicle filled with humans to a machine that can be easily corrupted?

Bibliography

[1] *'KARL - kernel address randomized link' - MARC*. URL: https://marc.info/?l=openbsd-tech&m=149732026405941 (visited on 10/31/2018).

[2] *'Re: KARL - kernel address randomized link' - MARC*. URL: https://marc.info/?l=openbsd-tech&m=149732265506347&w=2 (visited on 10/31/2018).

[3] Di Ma Ahmad MK Nasser and Priya Muralidharan. "An Approach for Building Security Resilience in AUTOSAR Based Safety Critical Systems". In: *Journal of Cyber Security* Vol. 6 3 (2017), pp. 271–304.

[4] Aditya Akella et al. "Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP". In: (2002), p. 14.

[5] S. Antonatos et al. "Defending against hitlist worms using network address space randomization". en. In: *Computer Networks* 51.12 (Aug. 2007), pp. 3471–3490. ISSN: 13891286. DOI: 10.1016/j.comnet.2007.02.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S1389128607000710 (visited on 04/15/2019).

[6] AUTOSAR. *Layered Software Architecture*. 2020. URL: https://www.autosar.org/fileadmin/user_upload/standards/classic/20-11/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf.

[7] AUTOSAR. *Platform solutions*. 2003. URL: http://www.autosar.org/standards.

[8] Maxime Ayrault, Ulrich Kühne, and Étienne Borde. "Finding Optimal Moving Target Defense Strategies: A Resilience Booster for Connected Cars". In: *Information* 13.5 (May 9, 2022), p. 242. ISSN: 2078-2489. DOI: 10.3390/info13050242. URL: https://www.mdpi.com/2078-2489/13/5/242 (visited on 07/21/2022).

[9] Maxime Ayrault et al. "Moving Target Defense Strategy in Critical Embedded Systems: A Game-theoretic Approach". In: *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*. 2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC). ISSN: 2473-3105. Dec. 2021, pp. 27–36. DOI: 10.1109/PRDC53464.2021.00014.

[10] Lino Briguglio et al. "Economic Vulnerability and Resilience: Concepts and Measurements". In: *Oxford Development Studies* 37.3 (2009), pp. 229–247. DOI: 10.1080/13600810903089893. eprint: https://doi.org/10.1080/13600810903089893. URL: https://doi.org/10.1080/13600810903089893.

[11] Gerald Brown et al. "Defending Critical Infrastructure". In: *INFORMS Journal on Applied Analytics* 36.6 (Dec. 2006). Publisher: INFORMS, pp. 530–544. ISSN: 2644-0865. DOI: 10.1287/inte.1060.0252. URL: https://pubsonline.informs.org/doi/abs/10.1287/inte.1060.0252.

[12] Nathan Burow et al. "Moving Target Defense Considerations in Real-Time Safety- and Mission-Critical Systems". In: *Proceedings of the 7th ACM Workshop on Moving Target Defense*. CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security. Virtual Event USA: ACM, Nov. 9, 2020, pp. 81–89. URL: https://dl.acm.org/doi/10.1145/3411496.3421224.

[13] Thomas E. Carroll et al. "Analysis of network address shuffling as a moving target defense". en. In: *2014 IEEE International Conference on Communications (ICC)*. Sydney, NSW: IEEE, June 2014, pp. 701–706. ISBN: 978-1-4799-2003-7. DOI: 10.1109/ICC.2014.6883401. URL: http://ieeexplore.ieee.org/document/6883401/ (visited on 10/29/2018).

[14] Thomas E. Carroll et al. "Analysis of network address shuffling as a moving target defense". In: *2014 IEEE International Conference on Communications (ICC)*. ICC 2014 - 2014 IEEE International Conference on Communications. Sydney, NSW: IEEE, June 2014, pp. 701–706. URL: http://ieeexplore.ieee.org/document/6883401/.

[15] Robert N. Charette. "This car runs on code". In: *IEEE Specturm* 7649 (2009).

[16] Stephen Checkoway et al. "Comprehensive Experimental Analyses of Automotive Attack Surfaces". In: (2011), p. 16.

[17] A. Clark, K. Sun, and R. Poovendran. "Effectiveness of IP address randomization in decoy-based moving target defense". In: *52nd IEEE Conference on Decision and Control*. Dec. 2013, pp. 678–685. DOI: 10.1109/CDC.2013.6759960.

[18] Vincent Conitzer and Tuomas Sandholm. "Computing the optimal strategy to commit to". In: *Proceedings of the 7th ACM conference on Electronic commerce*. EC '06. New York, NY, USA: Association for Computing Machinery, June 2006, pp. 82–90. URL: https://doi.org/10.1145/1134707.1134717.

[19] Warren Connell, Daniel A. Menascé, and Massimiliano Albanese. "Performance Modeling of Moving Target Defenses". In: *Proceedings of the 2017 Workshop on Moving Target Defense - MTD '17*. the 2017 Workshop. Dallas, Texas, USA: ACM Press, 2017, pp. 53–63. ISBN: 978-1-4503-5176-8. DOI: 10.1145/3140549.3140550. URL: http://dl.acm.org/citation.cfm?doid=3140549.3140550 (visited on 03/29/2019).

[20] Daniel. *Differences between ASLR, KASLR and KARL*. en-US. July 2017. URL: http://www.daniloaz.com/en/differences-between-aslr-kaslr-and-karl/ (visited on 10/30/2018).

[21] David Evans, Anh Nguyen-Tuong, and John Knight. "Effectiveness of Moving Target Defenses". In: *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Ed. by Sushil Jajodia et al. New York, NY: Springer New York, 2011, pp. 29–48. ISBN: 978-1-4614-0977-9. DOI: 10.1007/978-1-4614-0977-9_2. URL: https://doi.org/10.1007/978-1-4614-0977-9_2.

[22] Xiaotao Feng et al. "A Stackelberg Game and Markov Modeling of Moving Target Defense". In: *Decision and Game Theory for Security*. Ed. by Stefan Rass et al. Cham: Springer International Publishing, 2017, pp. 315–335. ISBN: 978-3-319-68711-7.

[23] Inc FIRST.Org. "Common Vulnerability Scoring System v3.0: Specification Documen". In: URL: https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf.

[24] Daniel Fraunholz et al. "Catch Me If You Can: Dynamic Concealment of Network Entities". en. In: *Proceedings of the 5th ACM Workshop on Moving Target Defense - MTD '18*. Toronto, Canada: ACM Press, 2018, pp. 31–39. ISBN: 978-1-4503-6003-6. DOI: 10.1145/3268966.3268970. URL: http://dl.acm.org/citation.cfm?doid=3268966.3268970 (visited on 03/26/2019).

[25] Andy Greenberg. "Hackers remotely kill a jeep on the highway—with me in it". In: *Wired* 7.2 (2015), pp. 21–22.

[26] Yacov Y. Haimes. "On the Definition of Resilience in Systems". In: *Risk Analysis* 29.4 (2009), pp. 498–501. ISSN: 1539-6924. DOI: 10.1111/j.1539-6924.2009.01216.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1539-6924.2009.01216.x (visited on 10/03/2019).

[27] John C. Harsanyi and Reinhard Selten. "A Generalized Nash Solution for Two-Person Bargaining Games with Incomplete Information". In: *Management Science* 18.5 (Jan. 1972), pp. 80–106. ISSN: 0025-1909, 1526-5501. DOI: 10.1287/mnsc.18.5.80. URL: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.18.5.80 (visited on 06/10/2021).

[28] Uta Hassler and Niklaus Kohler. "Resilience in the built environment". In: *Building Research & Information* 42.2 (Mar. 4, 2014), pp. 119–129. ISSN: 0961-3218, 1466-4321. DOI: 10.1080/09613218.2014.873593. URL: http://www.tandfonline.com/doi/abs/10.1080/09613218.2014.873593 (visited on 05/23/2022).

[29] Jin B. Hong and Dong Seong Kim. "Assessing the Effectiveness of Moving Target Defenses Using Security Models". In: *IEEE Transactions on Dependable and Secure Computing* 13.2 (Mar. 1, 2016), pp. 163–177.

[30] Amanda Howe, Anna Smajdor, and Andrea Stöckl. "Towards an understanding of resilience and its relevance to medical training". In: *Medical Education* 46.4 (2012). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2923.2011.04188.x, pp. 349–356. ISSN: 1365-2923. DOI: 10.1111/j.1365-2923.2011.04188.x. URL:

https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2923.2011.04188.x (visited on 05/23/2022).

[31] Q. Jia, K. Sun, and A. Stavrou. "MOTAG: Moving Target Defense against Internet Denial of Service Attacks". In: *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. July 2013, pp. 1–9. DOI: 10.1109/ICCCN.2013.6614155.

[32] Tae Un Kang et al. "Automated Reverse Engineering and Attack for CAN Using OBD-II". In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). ISSN: 2577-2465. Aug. 2018, pp. 1–7. DOI: 10.1109/VTCFall.2018.8690781.

[33] C. Lei, D. Ma, and H. Zhang. "Optimal Strategy Selection for Moving Target Defense Based on Markov Game". In: *IEEE Access* 5 (2017), pp. 156–169. ISSN: 2169-3536.

[34] Cheng Lei et al. "Moving Target Defense Techniques: A Survey". en. In: *Security and Communication Networks* 2018 (July 2018), pp. 1–25. ISSN: 1939-0114, 1939-0122. DOI: 10.1155/2018/3759626. URL: https://www.hindawi.com/journals/scn/2018/3759626/ (visited on 10/23/2018).

[35] Henger Li and Zizhan Zheng. "Optimal Timing of Moving Target Defense: A Stackelberg Game Model". In: *arXiv:1905.13293 [cs]* (May 30, 2019). arXiv: 1905.13293. URL: http://arxiv.org/abs/1905.13293.

[36] Yue-Bin Luo, Bao-Sheng Wang, and Gui-Lin Cai. "Effectiveness of Port Hopping as a Moving Target Defense". en. In: *2014 7th International Conference on Security Technology*. Hainan Island, China: IEEE, Dec. 2014, pp. 7–10. ISBN: 978-1-4799-7776-5 978-1-4799-7775-8. DOI: 10.1109/SecTech.2014.9. URL: http://ieeexplore.ieee.org/document/7023273/ (visited on 10/29/2018).

[37] G. F. Lyon. *Nmap network scanning: The official nmap project guide to network discovery and security scanning*. 2009. ISBN: 978-0-9799587-1-7.

[38] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Nmap Project, 2009. ISBN: 978-0-9799-5871-7.

[39] Laurie McCubbin. "Challenges to the Definition of Resilience". In: (2001), p. 22.

[40] Charlie Miller and Chris Valasek. "A Survey of Remote Automotive Attack Surfaces". In: *black hat - USA* (2014), p. 94.

[41] Charlie Miller and Chris Valasek. *Remote Exploitation of an Unaltered Passenger Vehicle*. en. Aug. 2015. URL: http://illmatics.com/Remote%20Car%20Hacking.pdf (visited on 07/10/2019).

[42] Charlie Miller and Chris Valasek. *Remote exploitation of an unaltered passenger vehicle*. 2015. URL: http://www-cs-faculty.stanford.edu/~uno/abcde.html.

[43] John F. Nash. "Equilibrium points in ¡i¿n¡/i¿-person games". In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.36.1.48. URL: https://www.pnas.org/doi/abs/10.1073/pnas.36.1.48.

[44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 2007. ISBN: 9781400829460. DOI: doi:10.1515/9781400829460. URL: https://doi.org/10.1515/9781400829460.

[45] Sen Nie, Ling Liu, and Yuefeng Du. "free-fall-hacking-tesla-from-wireless-to-can-bus". In: (2017), p. 16.

[46] Magnus Nyström, Carl Folke, and Fredrik Moberg. "Coral reef disturbance and resilience in a human-dominated environment". In: *Trends in Ecology & Evolution* 15.10 (2000), pp. 413–417. ISSN: 0169-5347. DOI: https://doi.org/10.1016/S0169-5347(00)01948-0. URL: https://www.sciencedirect.com/science/article/pii/S0169534700019480.

[47] H. Okhravi et al. "Finding Focus in the Blur of Moving-Target Techniques". In: *IEEE Security Privacy* (Mar. 2014), pp. 16–26. ISSN: 1540-7993. DOI: 10.1109/MSP.2013.137.

[48] H. Okhravi et al. *Survey of Cyber Moving Target Techniques:* en. Tech. rep. Fort Belvoir, VA: Defense Technical Information Center, Sept. 2013. DOI: 10.21236/ADA591804. URL: http://www.dtic.mil/docs/citations/ADA591804 (visited on 10/23/2018).

[49] Praveen Paruchuri et al. "Playing Games for Security: An Efficient Exact Algorithm for Solving Bayesian Stackelberg Games". In: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '08. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 895–902. ISBN: 9780981738116.

[50] Sandro Pinto and Nuno Santos. "Demystifying Arm TrustZone: A Comprehensive Survey". In: *ACM Comput. Surv.* 51.6 (Jan. 2019). ISSN: 0360-0300. DOI: 10.1145/3291047. URL: https://doi.org/10.1145/3291047.

[51] James Pita et al. "Using Game Theory for Los Angeles Airport Security". In: *International Conference on Autonomous Agents and Multiagent Systems, AAMAS-2007*. Vol. 30. 1. Jan. 18, 2009, p. 43. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2173.

[52] Julie Ann Pooley and Lynne Cohen. "Resilience: A Definition in Context". In: 22.1 (2010), p. 8.

[53] Erich Prisner. *Game theory through examples*. OCLC: 986787860. Washington, DC: MAA, The Mathematical Association of America, 2014. URL: http://sfx.urv.cat/urv?url_ver=Z39.88-2004&url_ctx_fmt=info:ofi/fmt:kev:mtx:ctx&ctx_enc=info:ofi/enc:UTF-8&ctx_ver=Z39.88-2004&rfr_id=info:sid/sfxit.com:azbook&sfx.ignore_date_threshold=1&rft.object_id=3710000000141037&rft.object_portfolio_id=&svc.fulltext=yes (visited on 11/18/2020).

[54] Sailik Sengupta et al. "A Game Theoretic Approach to Strategy Generation for Moving Target Defense in Web Applications". In: *AI Magazine*. AAMAS '17 (2017), p. 9.

[55] J. Simmie and R. Martin. "The economic resilience of regions: towards an evolutionary approach". In: *Cambridge Journal of Regions, Economy and Society* 3.1 (Mar. 1, 2010), pp. 27–43. ISSN: 1752-1378, 1752-1386. DOI: 10.1093/cjres/rsp029. URL: https://academic.oup.com/cjres/article-lookup/doi/10.1093/cjres/rsp029 (visited on 05/23/2022).

[56] Craig Smith. *The car hacker's handbook: a guide for the penetration tester*. San Francisco: No Starch Press, 2016. 278 pp. ISBN: 978-1-59327-703-1.

[57] International Organization for Standardization. In: *Road Vehicles Functional Safety Package* (2018).

[58] International Organization for Standardization. In: *Road Vehicles Cybersecurity engineering* (2021).

[59] Joshua Taylor et al. "Automated Effectiveness Evaluation of Moving Target Defenses: Metrics for Missions and Attacks". In: *Proceedings of the 2016 ACM Workshop on Moving Target Defense - MTD'16*. the 2016 ACM Workshop. Vienna, Austria: ACM Press, 2016.

[60] Adam Torok et al. "Automatization in road transport: a review". In: *Production Engineering Archives* 20.20 (Sept. 1, 2018), pp. 3–7. ISSN: 2353-7779. DOI: 10.30657/pea.2018.20.01. URL: https://www.sciendo.com/article/10.30657/pea.2018.20.01 (visited on 07/20/2022).

[61] Chris Valasek and Charlie Miller. *Adventures in Automotive Networks and Control Units*. en. 2014. URL: http://illmatics.com/car_hacking.pdf (visited on 07/10/2019).

[62] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. "Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A New Over-the-Air Attack Surface in Automotive IoT". In: (2020), p. 18.

[63] Lennert Wouters, Benedikt Gierlichs, and Bart Preneel. "My other car is your car: compromising the Tesla Model X keyless entry system". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Aug. 11, 2021), pp. 149–172. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i4.149-172. URL: https://tches.iacr.org/index.php/TCHES/article/view/9063.

[64] Jun Xu et al. "Comparing Different Moving Target Defense Techniques". en. In: *Proceedings of the ACM Conference on Computer and Communications Security* 2014 (Nov. 2014), pp. 97–107. DOI: 10.1145/2663474.2663486. URL: http://dl.acm.org/citation.cfm?doid=2663474.2663486.

[65] Huan Zhang et al. "Strategy Selection for Moving Target Defense in Incomplete Information Game". In: *Computers, Materials & Continua* 61 (Jan. 2019), pp. 763–786.

Glossary

ABS Anti-lock Braking System. 14

ACC Adaptative Cruise Control. 13, 82

APA Automated Park Assist. 13

API Application Programming Interface. 29

ASLR Address Space Layout Randomization. 39

AUTOSAR Automotive Open System Architecture. 25, 28, 29

BSW Basic Software. 29

C3S Connected Cars & Cyber-Security. 14

CAN Controller Area Network. 14, 23, 33, 36

CPU Central Processing Unit. 21, 40

CRC Cyclic Redundant Checksum. 30

CRES Critical Real-Time Embedded Systems. 15, 63, 80

CTMC Continous Time Markov Chain. 81

CVSS Common Vulnerability Scoring System. 75, 84, 86

DHCP Dynamic Host Configuration Protocol. 46, 47, 52, 54, 60, 61

DNS Domain Name System. 60

DoS Denial of Service. 30, 55, 56, 60

E2E End to End. 29

ECU Electronic Control Unit. 23, 24, 28, 29, 37

ECUs Electronic Control Units. 13, 25, 26, 28

ESP Electronic Stability Control. 14

FIRST Forum of Incident Response and Security Teams. 76

HUD Head-Up Display. 14

I-PDU Interaction Layer Protocol Data Unit. 30

IDS Intrusion Detection System. 78

ISR Instruction Set Randomization. 39, 41

KARL Kernel Address Randomization Link. 39

KASLR Kernel Address Space Layout Randomization. 39

LKA Lane Keeping Assists. 13

MILP Mixed Integer Linear Program. 9, 64, 91–93

MIQP Mixed Integer Quadratic Program. 9, 91, 92

MPTCP MultiPath TCP. 45, 52, 54, 56, 61

MTD Moving Target Defense. 15, 38, 41, 60, 61, 63, 80, 84, 105

OBD-II On-Board Diagnostic-II. 30, 32

OEM Original Equipment Manufacturer. 29

OS Operating System. 39, 41, 119

QoS Quality of Service. 50, 60, 63, 80

RoP Return-oriented Programmin. 39

RTE Run-time Environment. 29

SCNR System Call Number Randomization. 40

SGW Secure GateWay. 82

SWC Software Component. 29

TPMS Tire Pressure Monitor System. 34, 35

Appendix A

General terms

- **Security** : The defensive security of an device. This consists in ensuring the security of an application against hacking and taking control of that application. A connected car that involves human lives and the privacy of these users. It is necessary that the connected car on which we are going to work has as little *flaw* as possible, and therefore the smallest possible *attack surface*.
- **Safety** : The operational safety of a device. This consists in ensuring that an object can function properly in a guaranteed way. That everything goes well when it has to go well. The connected cars involve human life and are running *safety critical application*, so we want to be able to guarantee that the car will operate optimally at all times and that the defense methods used do not break this assurance.
- **Flaw** : Something an attacker can exploit to try and launch his attack. Can be found by analyzing the system operation or analysis the system code to find any mistakes that could lead to an attack. We want to avoid as many flaws as possible and/or eliminate existing ones. With the added defensive methods, make sure that it does not add new vulnerabilities.
- **Attack Surface** : What is visible from a system by an attacker, and which could be exploited by this attacker in order to find a flaw. The larger the attack surface of a system, the more likely an attacker may have a way to try to attack that system. This may help to verify that the attack surface once the defensive methods have been deployed has not increased the initial attack surface.
- **Safety Critical Application** : Services and applications that are considered critical. It is therefore necessary to be able to guarantee certain properties on execution/response time, as well as time guarantees before failure.

Connected cars have several critical applications, for which it must be ensured that they remain safe and protected. Once the new defense methods have been applied to the car, these defenses must not disrupt the operation of these critical applications.

- **Confidentiality** : The protection of access to all data relating to the user's private life, habits, travel etc... Because of the network connections provided to the various connected objects, we do not want information relating to the privacy of their users to be disclosed and allow safe use of these objects.
- **Integrity** : Ensure that all applications in a system operate normally. This integrity is due to the defense mechanisms on this system preventing malfunctions of this one. If the integrity of the car is compromised, an attacker can apply the brakes of this car on the highway, installing malware etc.
- **Adaptative Reconfiguration** : The way to defend a system constantly over time by making the defense dynamic. Dynamic defenses are used to increase the resilience of a system. They are used most of the time on systems that are not embedded. Their usage on embedded systems such as connected cars will require some adaptation.
- **Reliability** : The trust that you have in a system or software characterized by the probability of failure per application. The higher the reliability we have in a vehicle asset, the more trust there is in its nominal operation.
- **Vulnerabilities** : Correspond to a weakness or an error left in a system or device's code that, when exploited, can compromise the confidentiality, availability, and integrity of data stored in them through unauthorized access, elevation of privileges, or denial of service. A code or tool used to take advantage of a vulnerability is called an exploit.
- **Exploit** : A code segment or a program that is used to maliciously takes advantage of vulnerabilities or security flaws in software or hardware to infiltrate and initiate a denial-of-service (DoS) attack or install malware, such as spyware, ransomware, Trojan horses, worms, or viruses. So the exploit doesn't correspond to the malware itself but is used to deliver the malware.

Appendix B

Car Architecture

In this second appendix, we present a more detailed internal architecture for the car in figure B.1 than the one present in section 2.2.4. This figure represent in detail how each assets in connected to each other, the kind needed OS to run it, it as well as their computational power. It's also posible to see the dependency between the different asset, and on which aspect of the car each asset get the control of.

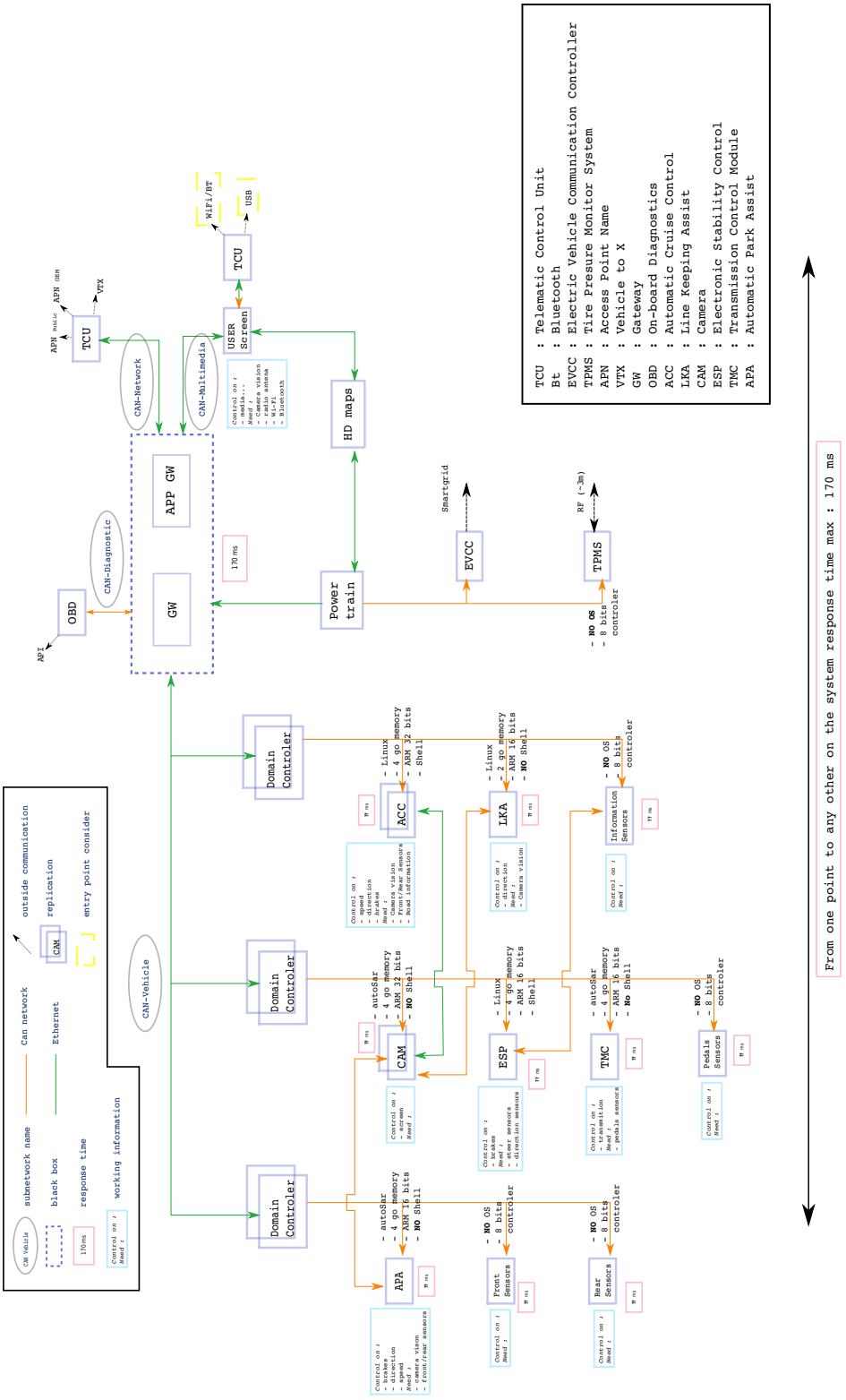

Figure B.1: The detail car architecture

ECOLE DOCTORALE

Titre : Défenses Dynamique pour l'Amélioration de la Résilience des Voitures Connectées

Mots Clefs : Voiture Connectée, Cyber Sécurité, théorie des jeux

Résumé :

Avec l'adoption des voitures connectées dans la vie courante lors de ces dix dernières années, de nouvelles menaces de sécurité voient le jour impliquant de nouveaux traitements devant être traitées. La gravité de ces menaces va dépendre de deux facteurs principaux, La surface d'attaque et l'impact de l'exploitation des vulnérabilités. Il est possible d'observer une augmentation de la surface d'attaque avec l'utilisation croissante de composants électroniques pilotés par logiciel dans les véhicules (ECUs). L'augmentation de cette surface d'attaque est aussi majoritairement dû à l'ajoût de nouvelles interfaces permettant de relier les véhicules entre eux et au monde extérieur. L'électronique présente dans les véhicules permettent le control de plus en plus de fonctionnalités critique. Cela peut aller de fonctionnalités telles que le braking by wire ou d'assistance avancée à la conduite (Adaptative Cruise Control). L'impact de l'exploitation de vulnérabilités présentes sur ces composant électronique devient de plus en plus préoccupant pour les constructeurs au vu des problèmes de sûreté et de privacy que cela pourrait induire. Au cours des dernières années, le nombre de publications traitant de la découvertes de de nouvelles vulnérabilités et attaques utilisant les connexions sans fil afin de prendre le contrôle d'une voiture ne fait qu'augmenter

L'apparition grandissante de ces nouveaux vecteurs d'attaques combiné à l'explosion de la complexité des systèmes embarqués dans les véhicules amènent la sureté de fonctionnement et la (cyber)sécurité au premier plan des objectifs majeurs lors de la conception de nouveaux systèmes automobiles.

Le concept de résilience `a du coup fait son apparition dans les différentes études sur les véhicules ainsi que dans la conception de nouveaux systèmes embarqués automobile. Ce terme de résilience fait référence par conception `a l'objectif consistant a la sécurisation de l'architecture globale d'un système contrairement `a l'introduction de correctif de sécurité locaux durant la durée de vie du produit. Cela inclut des mécanismes de défense tels que la détection d'intrusions ou bien encore une protection coordonnée contre les menaces existantes connues. Des approches comme les approches bio-inspirées utilisent par exemple la rémanence naturelle d'un organisme biologique comme modèle afin de pouvoir proposer des solutions techniques à ce défi de résilience.

Un autre exemple d'approche liée à la résilience correspond au principe de Moving Target Defense consistant `a la modification dynamique de la configuration d'un système lors de son exécution permettant de rendre les attaques déterministes moins efficaces contre le système défendu.

Lors du déroulement de cette Thèse, nous nous sommes particulièrement intéressé a l'utilisation de ces techniques de Moving Target Defense dans les véhicules connectés.

Ce type d'approche permet en effet de rendre plusieurs aspects d'un système dynamique. Le problèmes actuel liés à l'utilisation de telles techniques de défense dans un véhicule connecté est qu'il n'existe pas encore de prise en compte des contraintes liées aux systèmes embarqués critiques et que leur utilisation pourraient affecter la sûreté des utilisateurs. Il va donc falloir pouvoir garantir que leur ajout dans ce type de système ne provoquera pas de perturbations dans l'utilisation des applications critiques du véhicule.

L'utilisation de MTD est régît par trois grandes questions quoi faire bouger?, comment le faire bouger? et quand le faire bouger. Le traitement de ces deux premières questions quoi et comment faire bouger on déjà été adressées lors de différentes études existantes. Nous nous sommes donc particulièrement concentré sur la troisième quand, ce qui nous a permis d'arriver à la création d'un modèle permettant de calculer de manière automatique la fréquence d'utilisation optimale pour chaque technique de défense de Moving Target Defense présente sur un véhicule tout en prenant en compte les aspects de contraintes liées à l'utilisation de système embarqué critique.

Title : Dynamic Defenses for Improved Resilience of Connected Cars

Key Words : Connected Cars, Cyber Security, Game Theory

Abstract :

With the adoption of connected cars in everyday life over the last ten years, new security threats have arisen involving new treatments needing to be dealt with. The severity of these threats will depend on two main factors: the attack surface and the impact of vulnerability exploitation. We can observe an ever increasing attack surface with the growing use of software-controlled electronic components in vehicles (ECUs). The increase of this attack surface is also mainly due to the addition of new interfaces linking vehicles to each other and to the outside world. The electronics present in vehicles enable more and more critical functions to be controlled. These may include functions such as "braking by wire" or advanced driver assistance (Adaptive Cruise Control). The impact of exploiting vulnerabilities in these electronic components is becoming increasingly worrying for automakers, given the safety and privacy issues this could involve. Over the last few years, the number of publications reporting the discovery of new vulnerabilities and attacks using wireless connections to take control of a car has grown steadily.

The growing emergence of these new attack vectors, combined with the explosion of on-board systems complexity in vehicles, has brought operational safety and (cyber)security to the forefront of major objectives when designing new automotive systems.

The concept of resilience has thus made its appearance in various vehicle studies and in the design of new automotive embedded systems. The term resilience refers by design to the objective of securing a system's overall architecture, as opposed to introducing local security patches during the product's lifetime. This includes defensive mechanisms such as intrusion detection or coordinated protection against known existing threats. Approaches such as bio-inspired approaches, for example, use the natural persistence of a biological organism as a model for proposing technical solutions to this resilience challenge. Another example of a resilience-based approach is the principle of "Moving Target Defense", which involves dynamically modifying a system's configuration during execution to make determinist attacks less effective against the defended system.

During the course of this thesis, we were particularly interested in the use of Moving Target Defense techniques in connected vehicles. This type of approach makes it possible to make several aspects of a system dynamic. The current problem with the use of such defense techniques in a connected vehicle is that the constraints associated with critical embedded systems have not yet been taken into account, and their use could affect user safety. We therefore need to be able to guarantee that their inclusion in this type of system will not disrupt the use of critical vehicle applications.

The use of MTDs is governed by three main questions : what to move, how to move it and when to move it. The treatment of these first two questions what and how to make it move have already been addressed in various existing studies. We therefore focused particularly on the third question, when, which enabled us to create a model for automatically calculating the optimum frequency of use for each Moving Target Defense technique present on a vehicle, while taking into account the constraints associated with the use of critical on-board systems.

Institut Polytechnique de Paris

91120 Palaiseau