information

Article

Finding Optimal Moving Target Defense Strategies: A Resilience Booster for Connected Cars

Maxime Ayrault *, Ulrich Kühne * and Étienne Borde

Télécom Paris, Institut Polytechnique de Paris, Information Processing and Communications Laboratory (LTCI), 91120 Palaiseau, France; etienne.borde@telecom-paris.fr

* Correspondence: maxime.ayrault@telecom-paris.fr (M.A.); ulrich.kuhne@telecom-paris.fr (U.K.)

Abstract: During their life-cycle, modern connected cars will have to face various and changing security threats. As for any critical embedded system, security fixes in the form of software updates need to be thoroughly verified and cannot be deployed on a daily basis. The system needs to commit to a defense strategy, while attackers can examine vulnerabilities and prepare possible exploits before attacking. In order to break this asymmetry, it can be advantageous to use proactive defenses, such as reconfiguring parts of the system configuration. However, resource constraints and losses in quality of service need to be taken into account for such Moving Target Defenses (MTDs). In this article, we present a game-theoretic model that can be used to compute an optimal MTD defense for a critical embedded system that is facing several attackers with different objectives. The game is resolved using off-the-shelf MILP solvers. We validated the method with an automotive use case and conducted extensive experiments to evaluate its scalability and stability.

Keywords: moving target defense; game theory; automotive systems

Citation: Ayrault, M.; Kühne, U.; Borde, É. Finding Optimal Moving Target Defense Strategies: A Resilience Booster for Connected Cars. *Information* **2022**, *13*, 242. https://doi.org/10.3390/info13050242

Academic Editors: Giedre Sabaliauskaite, Jeremy Bryans and Farhan Ahmad

Received: 13 April 2022
Accepted: 2 May 2022
Published: 9 May 2022

Authors' Note: This work is supported by the research chair Connected Cars and Cyber Security (C3S) founded by Nokia, Renault, Thales, Valeo, Wavestone, Fondation Mines-Télécom and Télécom Paris.

1. Introduction

In the automotive domain, there is a clear trend towards increased connectivity: many new services are available today that demand a 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—is now standard in middle to higher class 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 the recent years, some of them putting the lives of the car's passengers at risk [1].

The security of connected cars is taken very seriously by the manufacturers, and it has become a major design goal. For instance, in a typical system architecture of a connected car, the subsystems are divided into different domains, and any message crossing a domain border passes through a secure gateway. This allows filtering unauthorized messages and preventing 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 in order to find vulnerabilities and prepare an exploit that can then be applied—potentially to a whole fleet of cars in parallel.

In order to limit the risk of this type of large-scale attack, proactive defenses are an interesting option. In this paper, we consider a class of bio-inspired techniques, called *Moving Target Defense* (MTD). The basic idea of an MTD is to proactively modify the

configuration of a system, thereby making previously acquired knowledge of the attacker useless and preventing deterministic attacks. There are several variants of MTDs, targeting different aspects of the system, such as the network configuration, data representations, or the execution environment.

Real-time critical embedded systems (*CRESs*)—such as connected cars or other intelligent transportation systems—are an interesting application domain for MTDs. Since a car is typically in use for more than ten years, it will be exposed to newly discovered vulnerabilities (*zero-day attacks*) during its life-cycle with a high probability. Furthermore, since such systems have strict resource constraints—e.g., regarding computational power, storage, and communication bandwidth—they are difficult to defend using traditional methods. MTDs can help slow down the implementation and deployment of new attacks when a new vulnerability is discovered, and thereby limit its scope. This can be seen as a way to increase the *resilience* of the system under attack. The resilience of a system is defined as the ability to provide its services safely, even in the face of unknown attacks. This can be achieved by, among other things, degrading certain services or returning the system to a safe state before critical damage is done.

However, these defense techniques come at a cost: the reconfiguration requires time and computational resources, and during the reconfiguration phase, the service delivered by the component might be interrupted or degraded. For example, changing IP addresses induces communication overhead and temporary connection or packet loss. If we want to combine several MTD techniques in a distributed system such as a connected car, we need a good *strategy*, which determines *where*, *how*, and *when* to defend, i.e., which components should be reconfigured in which way and at what frequency. Clearly, the question of *when* is in fact *how often*, and the easiest answer is *as often as possible*; however, such a strategy is overly pessimistic, and it does not take into account the mentioned costs. In this paper, we want to answer the following question: How can the benefits and costs of using an MTD be modeled to find optimal MTD strategies (i.e., *where* and *when* to move) in the CRES domain? We propose a combination of risk analysis techniques, MTD mechanisms, and a game-theoretic approach to determine the best defense strategy to adopt in terms of the frequency for each MTD method used.

At this time, the majority of work on MTDs focuses on the issue of *where* and *how*, proposing and evaluating new mechanisms of MTDs [2,3]. There are several MTD methods that can be applied to CRESs and for which the question *how* has already been addressed: Burow et al. [4] analyzed the applicability of some of these techniques in the context of real-time systems. This can be seen as a good starting point for the questions we try to answer.

The problem of finding an optimal strategy is best answered by game-theoretic methods. Indeed, there exist several works in the context of MTDs [5–9], but these contributions are mainly focused on web applications or general-purpose computing systems, and therefore, their requirements, MTD mechanisms, and models are not suitable for CRES design. Some studies have been conducted to evaluate the effectiveness of MTDs in the context of risk analysis methods [10]. Regarding the underlying game, we identified Bayesian Stackelberg games as the most suitable model for our purpose [11], since it reflects well the asymmetry and the probabilistic nature of the attacker. Besides the choice of the theoretic framework and the method for its resolution, a common problem of game-based approaches to determine is the *input*. Indeed, many parameters need to be taken into account in order to establish an appropriate model of a real-world application such as a connected car architecture. Therefore, we propose a complete methodology that helps to determine the input parameters for the game.

In summary, the contributions of this paper are:

1. A game theoretic model for the defense of CRESs;
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,

The article is organized as follows: In Section 2, we present the necessary background in order to keep the article self-contained. In Section 3, we introduce a motivating example and we outline the problem we are trying to solve. Section 4 presents our model, as well as a methodology allowing characterizing the different input parameters. In Section 5, we further formalize the model as a game, giving rise to a Mixed-Integer Quadratic Program (*MIQP*). Section 6 shows the transformation of the problem to a Mixed-Integer Linear Program (*MILP*), as well as its resolution and the form of the obtained strategies. In Section 7, we present a case study for the proposed method and further experimental evaluations. The results and limitations are discussed in Section 8. Related work is presented in Section 9, before concluding the article in Section 10.

The work is based on previous work published in [12]. This article differs from the previous one by a new and refined model, extensive experimentation, including a stability analysis, and a methodology to define the input parameters.

2. Background

2.1. Moving Target Defense

Before an attacker can successfully launch an attack, he or she must gather as much information as possible about the targeted system. Depending on the type of target, the information gathered during this *reconnaissance phase* can include, for example, network addresses, open ports, the version of the operating system in use, libraries, and running applications. This will help an attacker develop an appropriate exploit to be used in the *attack phase*.

This gives the attacker a natural advantage: she/he can spend the time and effort necessary to find flaws in the targeted system and attack whenever he or she wants. The defender, on the other hand, must establish a defense strategy beforehand, when designing the system. Thus, he/she can take into account known vulnerabilities, but will have difficulties in protecting the system against unknown attack vectors. This is why security updates are crucial to protect a system throughout its life-cycle. However, unknown exploits (*zero-day*) always remain a threat. In addition, frequent updates are not an option for safety-critical embedded systems, where the software must be thoroughly checked and certified before being released.

Moving Target Defenses (MTDs) are a set of defense techniques aimed at breaking the asymmetry between attacker and defender [2,13,14]. The general idea behind MTDs is to reduce the knowledge (utility) that an attacker can acquire by *dynamically reconfiguring* one or more features of the system. The goal is to (i) increase the time the attacker needs to discover vulnerabilities, (ii) reduce the likelihood that the attack will propagate through the system, and (iii) gain valuable time to bring an attacked system to a safe state. MTD techniques can be applied proactively (i.e., periodically) or reactively (i.e., when the presence of an attacker is detected or suspected). MTDs represent a practical and effective approach to improving cyber security and delaying the implementation and propagation of potential attacks. In particular, the use is to some extent indifferent for the type of attack and thus provides at least some protection against unknown exploits.

The MTD techniques have been classified into the following four categories [13,15,16]:

- Runtime environment (e.g., modify the operating system's address space layout);
- Data (e.g., modify data representation in memory);
- Software (e.g., switching between multiple implementations of the same services);
- Network (e.g., switching between multiple active IP addresses and open ports and modifying the network topology).

The use of MTDs in CRESs such as connected vehicles needs to take into account several constraints. Indeed, with limited resources and Quality of Service *(QoS)*, as well as strict timing requirements to respect, not all types of MTDs can be realistically applied in CRESs, and if used, the time and frequency are an important issue.

As an example, changing the runtime environment can be performed on a CRES at startup or while the system is not performing any critical functions by restarting it.

This mainly includes the use of techniques such as *Address Space Layout Randomization* (ASLR) [16] or its variant, KASLR [17]. Those techniques are used to randomly shuffle the basic block addresses of an application or the OS in memory to make direct Return-oriented Programming *(RoP)* attacks [18] unusable.

Another practical MTD is to dynamically change the IP address of a system [19], as well as information about the used network. This will slow down or prevent an attacker from finding and tracking the system. Similar techniques can be also applied to any network asset, such as the Bluetooth MAC address or on the Tire Pressure Monitor System *(TPMS)* ID of a vehicle.

In contrast, the use of data-type or software MTDs in a CRES is difficult, due to limited computational resources and storage capacity. Physical or software redundancy is often not an option regarding budgetary and space constraints.

In conclusion, the use of MTDs can add a real advantage in defense for a CRES, but this requires thorough planning and adjustment before incorporating them into the system.

2.2. Game-Theoretic Concepts

Many decision problems can be modeled using game-theoretic notions, allowing applications in various domains, from program synthesis to resource allocation in smart grids. Game theory provides a natural way to model security problems: it allows formalizing the objectives of potential attackers in terms of objectives and provides a mathematical framework for reasoning about defense strategies. The use of game theory [20] 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 made 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 making a decision.

In this article, we focus on sequential games and in particular on a Stackelberg game [21], where the first player is denoted as the *leader* and the others as the *followers*. This reflects the fact that the defender must commit to a defense that can be taken into account by potential attackers. In this context, the leader would be represented by the vehicle and the followers by the different types of attackers.

Choosing an action to perform results in a *reward*, which 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 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.

Table 1. Attacker/defender normal-form game.

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

The solution we are looking for—once the problem at hand is represented as a game—is an optimal *strategy*, which tells us which actions to take in order to maximize the gain of the defender. Such a strategy can be *pure* or *mixed*. With a pure strategy, a player chooses a single action to perform, while with a mixed strategy, the player randomly chooses a set of

actions according to some probability distribution. In a Stackelberg attack/defense game, it is clear that the defender—who occupies the role of leader—must adopt a mixed strategy in order to maximize his/her gain. On the other hand, it is common to consider that the attacker adopts a pure strategy.

In their standard form, Stackelberg games are *perfect information* games. This means that all players have perfect knowledge of the characteristics of the game, in particular the reward functions. However, this is not a realistic assumption for the applications we are investigating: As an example, a terrorist might want to cause maximal damage by manipulating the brakes, while someone interested in financial gain will prefer to block the engine in the first place, in order to demand a ransom. A third attacker might want to extract personal data or to track a vehicle. The defender does not know in advance which attacker he/she is going to face. This variety makes it impossible to define a reasonable reward function that covers all of the attackers' objectives.

We therefore use *imperfect information* games, Bayesian games [22]. In this variant, there can be several types of adversaries, each associated with a reward function. Furthermore, we define an a priori probability distribution over the different player types, reflecting the probability of encountering one or the other type.

A Bayesian game [23] is represented by a set of players G in which each player g must be of a given type of the set of type θ_g. In this work, the games we consider have two players: the defender d and the attacker a. While there is only one type of defender, θ_a contains as many elements as there are attackers. During the game, the attacker knows her/his own type, but it is unknown to the defender. A probability distribution γ defines the probabilities for each of the players to be of a specific type. Note that the type is chosen once before the game starts.

Each player g has a finite set of available actions A_g. The reward for each player then depends on the actual type of players and the chosen actions: $R_g : \theta_d \times \theta_a \times A_d \times A_a \to \mathbb{R}$ for $g \in \{a, d\}$.

There exists a way to transform a Bayesian game into normal form thanks to the transformation of Harsanyi [24]. This allows us to find a solution to this problem through a linear program to find the best possible existing strategy.

2.3. 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.

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.

$$\begin{aligned} &\max_x c^T \times x \\ &\text{Where: } A \times x \leq b \\ &\text{and: } x \geq 0 \end{aligned} \tag{1}$$

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 (2). Here, y corresponds to the variable we are trying to determine, and c, A, and b are the same elements as in the primal.

$$\begin{aligned} &\min_y b^T \times y \\ &\text{Where: } A \times y \geq c \\ &\text{and: } y \geq 0 \end{aligned} \tag{2}$$

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 $c^T \times x_0 = b^T \times y_0$, then x_0 and y_0 correspond to the optimal solution of the problem.

In our model, complementary slackness is 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.

2.4. Security Risk Analysis

In the context of critical system design, risk analysis is the systematic investigation of potential security risks. Its goal is to qualitatively or quantitatively evaluate the possibility that an adversary can cause damage by attacking the system. It examines the existing entry points of the system, their potential flaws, the difficulty of attacking them, as well as the consequences in case of a successful attack.

Performing a risk analysis is a complex process, and it needs to take into account the stakeholders that are concerned with the examined system and its environment. There exist several standardized methodologies and guidelines explaining how to perform a security risk analysis [25–27]. As an example, in the European context, the EBIOS standard gathers a common base of concepts and analysis tools while respecting the ISO 27001 standard on security risk management [25]. A common tool of such methods is attack trees [26] or attack graphs, which help to model the steps that an attacker needs to take to compromise a system and to estimate the success probability of the considered attacks. The exact methodologies are beyond the scope of this article. For the work at hand, we are mainly interested in the outcome of a risk analysis process, in the form of the quantitative *score* of the identified threats.

One such metric is the *Common Vulnerability Scoring System* (CVSS) [27]. The CVSS score is composed of three metrics, *base*, *temporal*, and *environmental*. The base score reflects the level of danger of a vulnerability according to the system characteristics that do not change over time. The temporal metrics allow adjusting the base score thanks to the elements changing over time such as the degree of evolution of an exploit kit. The more mature or easy to use the exploit kit is, the higher the score will be. The environmental metric allows completing the basic and temporal score by adding information related to a specific environment.

The final CVSS score is a value between 0 and 10, where 10 corresponds to the highest threat level. In this work, the CVSS score is used as an input parameter of our model (see Section 4.3). CVSS has already been used in other work based on game-theoretic models [28].

3. Motivating Example

Connected cars are essentially critical and connected real-time embedded systems, which operate on average for 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 infecting an entire fleet.

Regarding the security of connected cars, software updates are difficult to perform on these widely distributed systems with limited computational 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.

An example of the internal architecture of a connected car is presented in Figure 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 On-Board Diagnostic *(OBD-II)* plug.
4. **Shared Services**: regroups services allowing communication with the outside world (V2X), as well as some services shared among several domains.

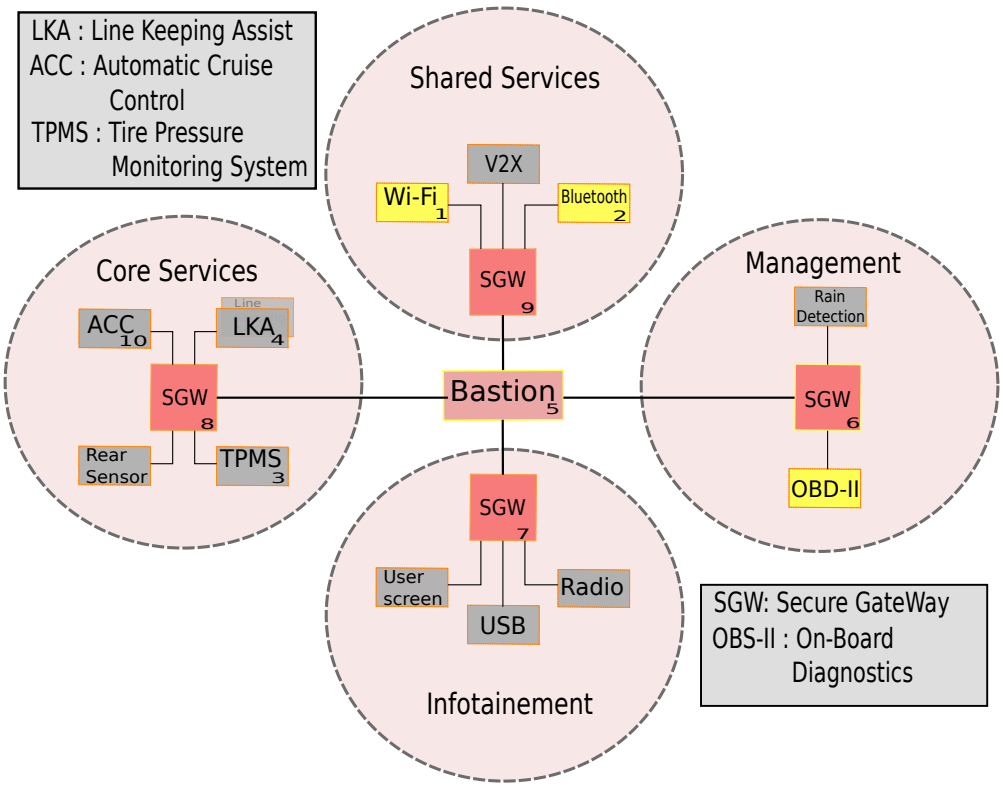

Figure 1. Vehicle architecture scheme.

Within each of these domains, the communication is managed by a secure gateway serving as the 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 in the Shared Services domain wants to communicate with the user screen module located in 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. However, 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 vehicle's 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 MTDs in 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 MTDs in a connected vehicle. Indeed, connected vehicles are critical embedded systems with limited computing power and strong time and QoS constraints. The use of MTDs in 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 MTDs 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. Model

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

4.1. Model Structure

The game we use is 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 MTDs present on node i.
- $P = \{0, 1, ..., p\}$: the set of attacker profiles.
- $R_{nmpn'm'}$: reward obtained by the defender when he/she 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/he chooses 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 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. On each node, it is possible for the defender to choose an action corresponding to doing nothing, denoted $_{idle}$ and included in the set M_i. This action can be chosen by the defender, but cannot be targeted by an attacker.

The different attackers are represented by the set of attacker profiles P, each of which has the objective of targeting some assets present on the different nodes, depending on the profile type.

Resolving the game consists of 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 are represented as follows:

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

To illustrate our model, Figure 2 shows a game composed of one node with two assets to protect and two MTDs, 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/he 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/he is interested in on this node.

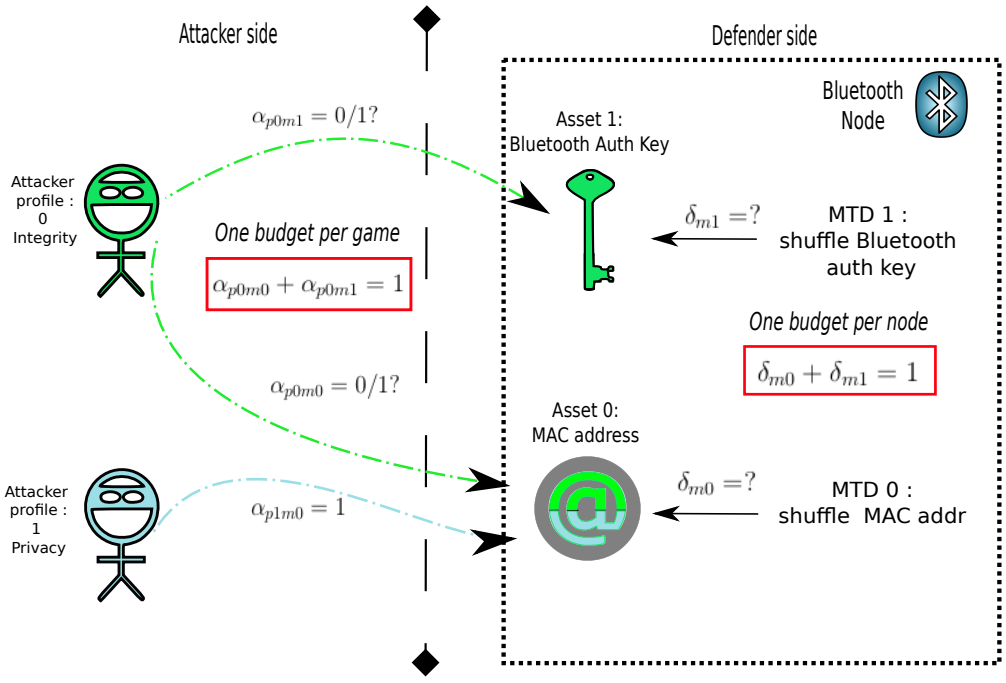

Figure 2. Model representation for 1 node and 2 attacker profiles.

The defender needs to decide how to use the two available MTDs 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 defending the system in the best possible way against the different attackers.

4.2. Input Parameters

Before defining the reward functions associated with each couple of attacker–defender actions, we begin by identifying the different input parameters of the game. These parameters must be 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 interested in node n.
$\widehat{C}_{nmp}$	Cost for the attacker of type p to attack MTD m on node n.
$\widehat{\gamma}_p$	Probability of encountering attacker profile p.

The following parameters describe the defender side:

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

4.3. 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 is known:

- The entropy value associated with 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 into account the difficulty of accessing 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 detail different methods used to determine the input parameters defined in the previous section.

4.3.1. 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* [29], to solve the problem of *drawing with replacement*. In our case, the number of attempts is equal to the node period divided by the time needed to scan one configuration. The formulation of this problem is given by Equation (3), where x is the number of attempts, h is the number of instances of the asset, and a is the number of available values. The probability of finding the information can then be calculated as

$$\widehat{Sp} = 1 - ((a-h)/a)^x \tag{3}$$

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 ad hoc method to estimate the success probability.

Example 1. *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 considered 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 [30].

Considering that the defender can change the IP address of the module sought by the attacker every 10 s, 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 (3), the attacker's probability of success in discovering the IP address of the module is

$$\begin{aligned} \widehat{Sp} &= 1 - ((a-h)/a)^x \\ \widehat{Sp} &= 1 - ((252-1)/252)^{39} \\ \widehat{Sp} &= 1 - 0.856355413 = 0.143644587 \end{aligned} \tag{4}$$

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

The interest of an attacker of profile p in 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.

4.3.3. 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.

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

The definition of the probability of the appearance of an attacker can be given 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 do 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.

4.3.5. Defender Node Gain: W_{np}

The interest in 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 in 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.

4.3.6. 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.

5. Game Formalization

5.1. Game Form

As explained in Section 3, 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. 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 2.2 allow us to represent these different interactions and take into account the different constraints related to the system: the asymmetry between the attacker and the defender gives rise to a Stackelberg game [31], allowing imposing an order in the decision of the actions. Bayesian games [6] 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.

5.2. Reward

For each combination of actions attacker–defender on each node and MTD, a reward function allows computing 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 performed 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 nodes n and n' and MTDs 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 nodes n and n' and MTDs m and m', the associated gain for the attacker p is equal to his/her success probability multiplied by his/her interest in the node n'.

  The cost of performing an action for the attacker corresponds to the parameter $\widehat{C}_{n'm'}$. The gain function of the attacker is then of the following form:

$$\widehat{R}_{nmpn'm'} = \begin{cases} 0, & \text{if } n = n' \text{ and } m = m' \\ \widehat{Sp}_{n'm'p} * \widehat{W}_{n'p}, & \text{if } n \neq n' \text{ or } m \neq m' \end{cases} \tag{5}$$

  The total reward an attacker will obtain for performing an action will be:

$$\widehat{R}_{nmpn'm'} - \widehat{C}_{n'm'p} \tag{6}$$

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 nodes n and n' and MTDs 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 in the node.
- If the attacker p and the defender do not target the same nodes n and n' and MTDs 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_{nm}^{MTD}.

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

$$R_{nmpn'm'} = \begin{cases} \widehat{Sp}_{nmp} * W_{np}, & \text{if } n = n' \text{ and } m = m' \\ 0, & \text{if } n \neq n' \text{ or } m \neq m' \end{cases} \quad (7)$$

The total reward the defender will obtain for performing an action will be:

$$R_{nmpn'm'} - C_{nm}^{MTD} \quad (8)$$

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 of 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.

5.3. Payoff Function

In order to determine the best possible strategy for the defender, a *payoff* function is used to calculate the maximum possible reward. This function basically combines the different reward functions, depending on the decision variables α and δ, representing the strategies for the attacker and the defender, respectively:

$$\sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} \delta_{nm} \times [\sum_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}_{-idle}} (\widehat{\gamma_p} \times \alpha_{n'm'p} \times R_{nmpn'm'}) - C_{nm}^{MTD}] \quad (9)$$

5.4. Optimal Attack Strategy

As discussed in Section 2.3, complementary slackness is used to constrain each attacker to maximize her/his payoff function. The payoff is expressed by the following equations:

$$\begin{aligned} &\forall_{p\in\mathcal{P}} \max_{\alpha_{nmp}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}_{-idle}} [\alpha_{n'm'p}[\sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} (\delta_{nm} \times \widehat{R}_{nmpn'm'}) - \widehat{C}_{n'm'p}]] \\ &\forall_{p\in\mathcal{P}} \sum_{n'\in\mathcal{N}} \sum_{m'\in\mathcal{M}_{-idle}} \alpha_{n'm'p} = 1 \\ &\forall_{p\in\mathcal{P}} \forall_{n'\in\mathcal{N}} \forall_{m'\in\mathcal{M}_{-idle}} \alpha_{n'm'p} \geq 0 \end{aligned} \quad (10)$$

It is then possible to transform the primal problem (10) into its dual (11). 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.

$$\begin{aligned} &\forall_{p\in\mathcal{P}} \min a_p \\ &\forall n' \in \mathcal{N}, \forall m' \in \mathcal{M}, \forall p \in \mathcal{P}\ a_p \geq \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} (\delta_{nm} * \widehat{R}_{nmpn'm'}) - \widehat{C}_{n'm'p} \end{aligned} \quad (11)$$

Using strong duality and complementary slackness, these two problems are transformed into Constraint (12), which must be satisfied when solving the optimization problem for the leader (defender) in order to consider only the best responses by the follower (attackers).

$$\forall_{p\in\mathcal{P}}, \forall_{n'\in\mathcal{N}}, \forall_{m'\in\mathcal{M}}, 0 \leq a_p - \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} (\delta_{nm} \times \widehat{R}_{nmpn'm'}) - \widehat{C}_{n'm'p} \leq (1 - \alpha_{n'm'p})M \quad (12)$$

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

5.5. Mixed-Integer Quadratic Program

With the above elements, we have all the ingredients to define the game as a *Mixed-Integer Quadratic Program* (MIQP). Formulating our problem in this way allows us to find the strategy δ that maximizes the reward obtained for the defender, which corresponds to the best strategy of using the available MTDs.

$$\text{obj}: \max_{\delta_{nm}, \alpha_{n'm'p}, a_p} \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \delta_{nm} \times [\sum_{p \in \mathcal{P}} \sum_{n' \in \mathcal{N}} \sum_{m' \in \mathcal{M}_{-idle}} (\widehat{\gamma_p} \times \alpha_{n'm'p} \times R_{nmpn'm'}) - C^{MTD}_{nm}] \tag{13}$$

$$C1: \forall_{n \in \mathcal{N}}, \sum_{m \in \mathcal{M}} \delta_{nm} = 1 \tag{14}$$

$$C2: \forall_{p \in \mathcal{P}}, \sum_{n' \in \mathcal{N}} \sum_{m' \in \mathcal{M}_{-idle}} \alpha_{n'm'p} = 1 \tag{15}$$

$$C3: \forall_{n' \in \mathcal{N}} \forall_{m' \in \mathcal{M}}, 0 \leq a_p - \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} (\delta_{nm} \times \widehat{R}_{nmpn'm'}) - \widehat{C}_{n'm'p} \leq (1 - \alpha_{n'm'p}) M \tag{16}$$

$$C4: \forall_{n \in \mathcal{N}} \forall_{m \in \mathcal{M}}, \delta_{nm} \in [0,1] \tag{17}$$

$$C5: \forall_{p \in \mathcal{P}} \forall_{n' \in \mathcal{N}} \forall_{m' \in \mathcal{M}}, \alpha_{n'm'p} \in \{0,1\} \tag{18}$$

$$C6: \forall_{p \in \mathcal{P}}, a_p \in \mathbb{R} \tag{19}$$

The complete MIQP is shown in Equations (13)–(19). We quickly review the different parts of the MIQP in the following. The objective function (13) maximizes the defender's payoff function. According to our model, the defender will defend each node independently, spending a budget of 1 (Constraint (14)), using a mixed strategy (Constraint (17)). In contrast, each attacker has a global budget of 1 to spend (Constraint (15)), using a pure strategy (Constraint (18)). Finally, Constraint (16) adds the complementary slackness, forcing the attacker to choose the target that maximizes her/his reward, taking into account the defender's strategy.

Unfortunately, the complexity of solving an MIQP is high, and there are very few tools available. In the next section, we show how to transform the problem into an MILP in order to make its resolution feasible.

6. Game Resolution

6.1. MIQP to MILP Transformation

In this section, we propose a transformation of the above MIQP into an MILP to allow for a more efficient resolution. In general, transforming an MIQP into an MILP consists of eliminating quadratic terms by changing the decision variables. The only place where a nonlinear term occurs is the objective function (13), which contains both δ and α as factors. As a consequence, we factor the two variables to obtain a new variable Z:

$$\forall_{n \in \mathcal{N}} \forall_{m \in \mathcal{M}} \forall_{p \in \mathcal{P}} \forall_{n' \in \mathcal{N}} \forall_{m' \in \mathcal{M}_{-idle}}, Z_{nmpn'm'} = \delta_{nm} \times \alpha_{n'm'p} \tag{20}$$

This new variable will replace α and δ in the MILP formulation. However, this is only an auxiliary construction, and we are not really interested in the value of Z, but in the defender strategy, which is given by δ. Therefore, we need to recover the original decision variables from Z. This can be achieved by the following equations:

$$\forall_{n \in \mathcal{N}} \forall_{m \in \mathcal{M}}, \delta_{nm} = \sum_{p \in \mathcal{P}} \sum_{n' \in \mathcal{N}} \sum_{m' \in \mathcal{M}_{-idle}} Z_{nmpn'm'} \tag{21}$$

$$\forall_{p \in \mathcal{P}} \forall_{n' \in \mathcal{N}} \forall_{m' \in \mathcal{M}_{-idle}}, \alpha_{n'm'p} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} Z_{nmpn'm'} \tag{22}$$

Based on the above correspondence, the resulting MILP is shown in the Equations (25)–(33). We review the individual constraints and their transformation in the following, starting with the objective function. Expanding (9) gives

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}_{-idle}}\left[\delta_{nm}\times\widehat{\gamma_p}\times\alpha_{n'm'p}\times R_{nmpn'm'}\right]-\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\left[\delta_{nm}\times C^{MTD}_{nm}\right]. \tag{23}$$

Now, using Equations (21) and (22), we obtain the new objective function:

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}_{-idle}}\left[\widehat{\gamma_p}\times Z_{nmpn'm'}\times R_{nmpn'm'}\right]$$
$$-\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\left[\left(\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}Z_{nmpn'm'}\right)\times C^{MTD}_{nm}\right]. \tag{24}$$

$$\text{obj}: \max_{Z_{nmpn'm'},\alpha_{n'm'p},a_p}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}_{-idle}}[\widehat{\gamma_p}\times Z_{nmpn'm'}\times R_{nmpn'm'}]$$
$$-\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}[(\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}}Z_{nmpn'm'})\times C^{MTD}_{nm}] \tag{25}$$

$$D1: \forall_{n\in\mathcal{N}},\ \sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}_{-idle}}Z_{nmpn'm'}=1 \tag{26}$$

$$D2_a: \forall_{n'\in\mathcal{N}}\forall_{m'\in\mathcal{M}}\forall_{p\in\mathcal{P}},\ \alpha_{n'm'p}=\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}Z_{nmpn'm'} \tag{27}$$

$$D2_b: \forall_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}},\ \sum_{m'\in\mathcal{M}_{-idle}}\alpha_{n'm'p}=1 \tag{28}$$

$$D3: \forall_{n'\in\mathcal{N}}\forall_{m'\in\mathcal{M}},\ 0\le(a_p-\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\widehat{R}_{nmpn'm'}\times(\sum_{n''\in\mathcal{N}}\sum_{m''\in\mathcal{M}}Z_{nmpn''m''})-\widehat{C}_{n'm'p})$$
$$\le(1-\alpha_{n'm'p})M \tag{29}$$

$$D4: \forall_{n\in\mathcal{N}}\forall_{m\in\mathcal{M}},\ 0\le\sum_{p\in\mathcal{P}}\sum_{n'\in\mathcal{N}}\sum_{m'\in\mathcal{M}_{-idle}}Z_{nmpn'm'}\le 1 \tag{30}$$

$$D5: \forall_{p\in\mathcal{P}},\forall_{n'\in\mathcal{N}}\forall_{m'\in\mathcal{M}},\ \alpha_{n'm'p}\in\{0,1\} \tag{31}$$

$$D6: \forall_{p\in\mathcal{P}},\ a_p\in\mathbb{R} \tag{32}$$

$$D7: \forall_{p\in\mathcal{P}}\forall_{n\in\mathcal{N}}\forall_{m\in\mathcal{M}}\forall_{n'\in\mathcal{N}}\forall_{m'\in\mathcal{M}},\ Z_{nmpn'm'}\in[0,1] \tag{33}$$

In the same way, the constraint $C1$ (14) is transformed to the constraint $D1$ (26) by substituting δ_{nm} with Equation (21). As in the original MIQP formulation, $D1$ limits the defender's budget to 1.

The attacker's global budget is represented by the two constraints $D2_a$ (27) and $D2_b$ (28), the first of which reintroduces α into the MILP, following (22). The second constraint directly corresponds to $C2$ (15). Note that in addition to the new decision variables Z, we need either α or δ in order to reconstruct the original solution. We chose to use α.

The transformation of the slackness constraint $C3$ (16) to its MILP version $D3$ (29) is again straightforward, substituting δ_{nm} by the corresponding Z variables. Similarly, $D4$ (30) sets the lower and upper bounds of the defender's strategy to 0 and 1, respectively, thus corresponding to $C4$ (17). $D5$ (31) and $D6$ (32), respectively, correspond directly to $C5$ (18) and $C6$ (19).

Finally, $D7$ (33) is introduced as an additional constraint in order to fix the Z variables to 1 or 0. Note that this is due to α being pure strategies (cf. the constraint $D5$).

6.2. Correspondence between the MIQP and the MILP

Following the transformation described above, based on Equations (22) and (21), we ensure that any solution to the resulting Mixed-Integer Linear Problem (*MILP*) is also a solution to the original Mixed-Integer Quadratic Problem (*MIQP*). However, this is not necessarily true in the other direction. Passing from MIQP to MILP by turning two distinct

variables δ and α into a single one introduces additional constraints. 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. As it turns out, such instances do not have a solution under their MILP form.

However, this limitation is not a practical limitation for the considered use cases of our model. In our applications, we generally consider two types of attackers—integrity and privacy—each with a fixed and limited number of levels of expertise. This usually results in ten or less attacker profiles. The number of assets to defend in an automotive system, on the other hand, is usually larger, ranging from a dozen (for a single subsystem) up to a hundred. The number of nodes in the considered models system is therefore greater than the number of attacker profiles.

7. Experimental Results

7.1. Automotive Use Case

In order to evaluate the proposed method, we set up an experimental use case, which is based on the connected car architecture presented in Section 3. Figure 3 shows a selection of ten nodes, for each of which we have selected two or three different MTD defenses. On the attacker side, there are ten attacker profiles, corresponding to two types (integrity and privacy) and five levels of expertise.

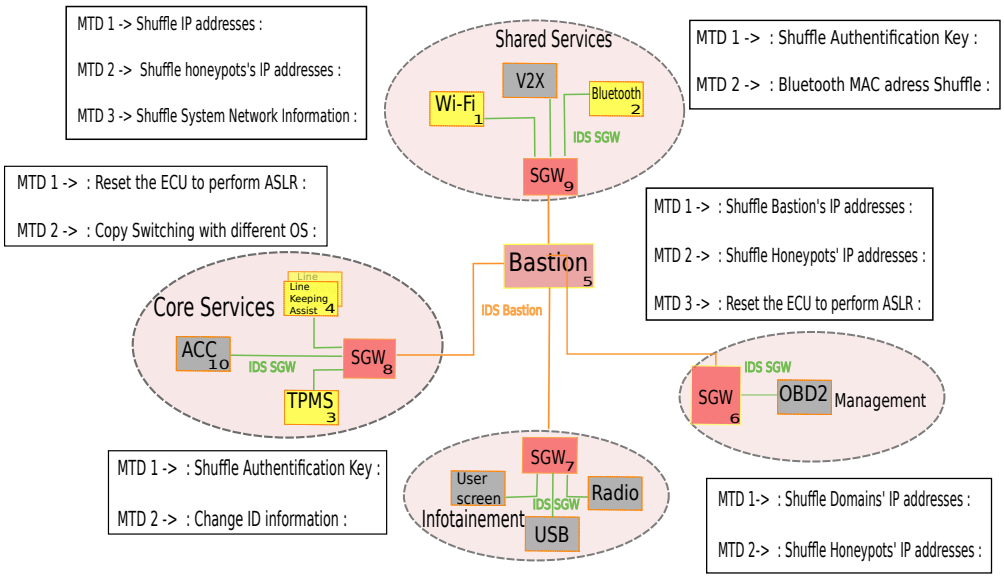

Figure 3. Selected nodes and their defenses.

We implemented a software tool that generates the MILP problem from the given input parameters and solves it using the CPLEX solver [32]. We obtained the optimal defense strategy shown as below equation, in which the defender will defend each node and asset targeted by an attacker by spending his/her full budget per node. As can be seen, for some nodes, one of the defenses dominates (i.e., the corresponding δ has value 1.0), while other nodes are defended using a mixed random strategy of multiple MTDs. The time required to compute this solution with CPLEX was 9.2 s.

$\delta_{n0_{m}0} = 0.351$	$\delta_{n0_{m}1} = 0.0$	$\delta_{n0_{m}2} = 0.649$	$\delta_{n0idl} = 0.0$
$\delta_{n1_{m}0} = 0.95$	$\delta_{n1_{m}1} = 0.05$		$\delta_{n1idl} = 0.0$
$\delta_{n2_{m}0} = 0.0$	$\delta_{n2_{m}1} = 1.0$		$\delta_{n2idl} = 0.0$
$\delta_{n3_{m}0} = 1.0$	$\delta_{n3_{m}1} = 0.0$		$\delta_{n3idl} = 0.0$
$\delta_{n4_{m}0} = 0.404$	$\delta_{n4_{m}1} = 0.311$	$\delta_{n4_{m}2} = 0.285$	$\delta_{n4idl} = 0.0$
$\delta_{n5_{m}0} = 0.0$	$\delta_{n5_{m}1} = 1.0$		$\delta_{n5idl} = 0.0$
$\delta_{n6_{m}0} = 0.0$	$\delta_{n6_{m}1} = 1.0$		$\delta_{n6idl} = 0.0$
$\delta_{n7_{m}0} = 0.078$	$\delta_{n7_{m}1} = 0.922$		$\delta_{n7idl} = 0.0$
$\delta_{n8_{m}0} = 0.0$	$\delta_{n8_{m}1} = 1.0$		$\delta_{n8idl} = 0.0$
$\delta_{n9_{m}0} = 0.519$	$\delta_{n9_{m}1} = 0.481$		$\delta_{n9idl} = 0.0$

7.2. Scalability

7.2.1. Random Scenarios

In order to investigate if the proposed solution scales well, we generated random scenarios of different sizes (the generating tool we made for the experimentation is available here: https://gitlab.telecom-paris.fr/TheseMA/tool_for_journal.git, accessed on 12 April 2022). During a first series of experiments, we realized that generating the parameters in a totally random way resulted in a worst-case scenario for the defender in which all the attacker profiles are interested in all the nodes and assets of the model. Since this is not realistic—and presents a real challenge to the solver—we limited the interest of an attacker to two thirds of the assets in a random fashion.

The results obtained are summarized in Figure 4, in which we display the computation time taken by CPLEX to solve the generated MILP problems. The scenarios were generated in such a way as to have as many attacker profiles as nodes, while the remaining input parameters were generated randomly. In this way, we obtained an execution time for a scenario with ten nodes and ten attacker profiles in the same order of magnitude as in the case study presented in Section 7.1, which indicates that the random tests are comparable to real-world scenarios.

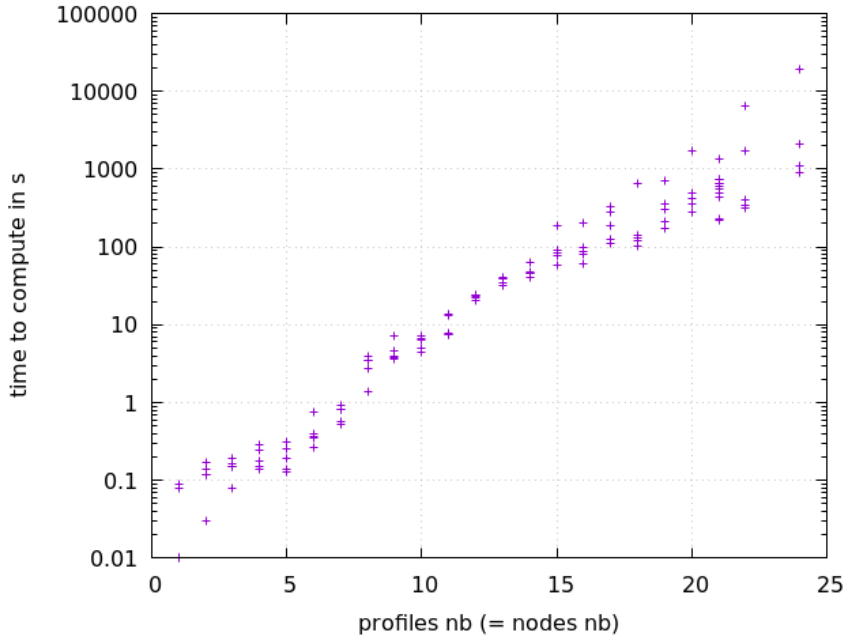

Figure 4. Scaling representation with attacker profiles number = node number.

We managed to solve games composed of up to 25 nodes and 25 attacker profiles with a reasonable runtime. Considering automotive applications, 25 nodes is about the size of a single domain. Thus, one possibility to apply our method to a complete system is decomposing it into its domains and finding an optimal strategy for each of them. This can be considered a conservative approach, since this basically corresponds to a more powerful attacker with one budget to spend for each domain.

7.2.2. Fixed Attacker Profiles

The above experiment served merely to explore the limits when scaling both attackers and nodes. In a realistic scenario, it is more common to choose a fixed number of attacker profiles. In our experiments so far, we considered five levels of expertise (expert, high, medium, low, beginner). If we only consider one type of attacker (integrity or privacy), we will have to face five attackers in total. A typical case was already presented in the case study in Section 7.1. There, we considered privacy and integrity attackers with five levels of expertise each, amounting to ten attacker profiles to take into account in the model. If we want to take things even further, and we are aware of, e.g., ten additional specific attacks, adding those to the model adds up to 20 attacker profiles.

Following the above considerations, we conducted random experiments with a variable number of nodes (Note that in all generated scenarios, the number of nodes is greater than or equal to the number of attacker profiles. As explained in Section 6.2, 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. As a consequence, instances with less nodes than attackers are infeasible in their MILP form because of Constraints (26), (29), and (30)) and a fixed number of attacker types (5, 10, and 20). The results are shown in Figure 5. As can be seen in the figure, with five attackers, our solution scales well up to 50 nodes and beyond. When we add another five attacker profiles, the run-time increases by a factor of around five. Models with a bit more than 40 nodes can be solved in around 1000 s. Finally, the case with 20 attacker profiles takes the MILP solver to its limits. As can be seen in the figure, the variance of the resulting runtimes is much larger. We can still solve models of up to 30 nodes, whereas in some cases, the solver took more than an hour.

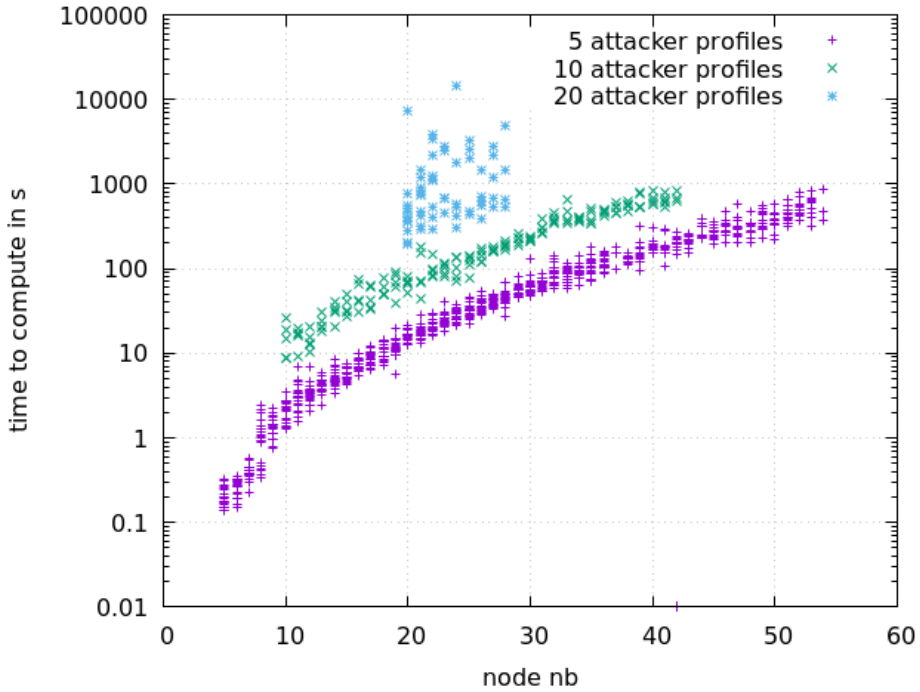

Figure 5. Randomly generated models with a fixed number of attacker profiles and a variable number of nodes.

7.3. Stability Analysis

As in any model-based approach, the quality of the solution heavily depends on the quality of the input model. Therefore, a methodology—as presented in Section 4.3—to determine the model's parameters is crucial for real-world applications. However, some parameters are more easy to determine than others. For some, there will remain some uncertainty with respect to their value. In the following, we describe two further experiments, which aimed to test the *stability* of the obtained results. We can be confident about the result if slight changes in the input parameters lead to slight changes in the output strategies.

The first experiment focused on the parameter γ, representing the rate of appearance of the different attacker profiles. We started from a model similar to the case study: 10 nodes with two or three MTD defenses each and 10 attacker profiles (five integrity attackers and five privacy attackers). In this case, γ corresponds to the probability of encountering privacy attackers, while integrity profiles occur with probability $1 - \gamma$. In the experiment, γ was varied between 0.1 and 0.9 in steps of 0.1, and we observed the evolution of the best strategy for each case.

Figure 6 shows the strategies for four selected nodes. As can be seen, the strategies change in a monotonous fashion, which is expected. As an example, for Node 1 on the upper right of the figure, the output is a mixed strategy of the two MTDs, and the dominant defense is the one that is more effective against the attacker who is more likely to appear.

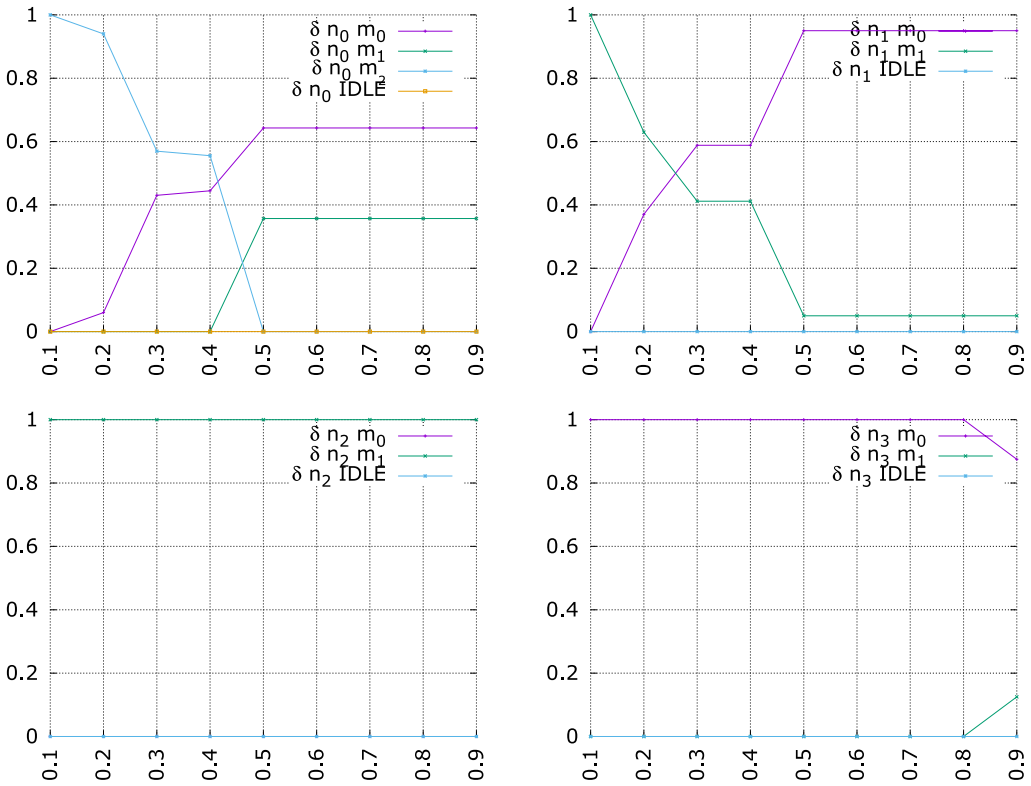

Figure 6. How moving the ratio between attacker profiles (gamma) affects the defender strategy.

In some cases—such as Node 2 on the lower left of the figure—the strategy does not change at all, indicating that either the node is not interesting for the attackers or that one of the defenses is the best one against all attackers. Finally, on Node 3 on the lower right, the strategy of the defender only starts to adapt slightly when the second attacker becomes predominant.

In a second experiment, we took a look at the stability of the given strategy according to a set of input parameters. We started again from the same scenario, where we randomly shuffled the values of several input parameters: the costs and gain for the defender and the

attacker, as well as the attackers' success probability. For each parameter, we chose a value following a Gaussian distribution around the original value and the variance chosen as a given percentage of the original parameters value. For each value of the variance (1%, 2%, 5%, and 10%), we conducted 100 runs.

The result of this experiment is shown in Figure 7. It shows the δ values of a selected subset of MTDs in the form of a box plot. The box contains the two central quartiles of the distributions with the median marked as a horizontal line. Outliers are marked as individual points with cross marks.

Figure 7. How stable the strategies are considering variance in the parameters.

It can be noticed that the variation of the output strategies remains quite low up to 5%, and for many nodes even up to 10%. There is a few exceptions. In particular, the strategy of Node 0 (cf. the three left-most values) shows some volatility. While the original solution is a mixed strategy of three defenses, for a non-negligible number of models, our method

tends to choose a strategy without the first one, setting δ_{n0m0} to zero. The opposite behavior can be seen for Node 4: while the original strategy purely relied on one defense, there are some outliers, where a mixed strategy was chosen.

8. Discussion and Limitations

We presented a method to find an optimal strategy for using MTDs 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 previous work [12], the 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 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 4.3—some inputs remain difficult to estimate. Indeed, car manufacturers have little interest in making public their statistics 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 was 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, runtimes 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 of attacks or vulnerabilities becomes available.

Regarding the stability of the computed strategy, the results are satisfactory. The observed changes in the output strategies were mostly expected and explicable, but the experiments also showed 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.

9. Related Work

The method of calculating the optimal strategy for MTDs has been the subject of several recent contributions [5–9]. This topic represents an important challenge for the configuration of MTDs 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 dealt with in this paper: 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 CRESs, which leads to different modeling of MTDs. Another example of these domain-specific issues is the research by Burow et al. [4] 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, and (iii) the nature of the costs associated with the actions of the defender and the attacker. We clarify these differences throughout this section.

In [6], Sengupta et 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 essentially showed 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 highlighting the legitimacy of the use of an approach based on game theory. Although the game is comparable to the one we present in this paper, 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 the success of an attack in a short time. More recently, the optimal travel frequency was introduced in detail by Li and Zheng in [8]. In this paper, 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 indicated in our Introduction, we 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 the success of an attack. In a similar vein, Feng et al. [7] 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 [9], the authors Zhang et al. also studied the problem of determining an MTD strategy using a related game theory approach. These authors suggested relying 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 the MTD with an IDS, rather than deploying a proactive defense that changes the configuration periodically.

Connell et al. also attempted in their paper [33] to approach the problems related to the frequency of use of an MTD in a system. In their article, their approach was not limited to the very particular contexts of CRESs like ours, but rather tried to determine this frequency for a more "normal" system. For this, they based their research on an approach to determine the availability and performance of a resource for which they were using an MTD. This allowed them, thanks to a Continuous Time Markov Chain [34] *(CTMC)*, to determine the frequency of use of an MTD for a resource based on a trade-off between the probability of the success of an attacker and the availability of this resource.

In conclusion, the various existing works in the state-of-the-art have shown the correctness of the game-theoretic approach 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 presented is a first attempt to define a game-theoretic approach to determine an optimal strategy for an MTD in the CRES context.

10. Conclusions and Future Works

In this paper, we developed a model to represent the interactions between a system and a set of different attackers 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 MTDs 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 showed that the method works for realistic use cases from the automotive domain, and the results exhibited a good stability.

A possible extension of this work would be the integration of the method with an automatic adaptation over time in order to update the strategy according to the observations made during the use of the system.

Author Contributions: Writing—original draft, M.A., U.K. and É.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Smith, C. *The Car Hacker's Handbook: A Guide for the Penetration Tester*; No Starch Press: San Francisco, CA, USA, 2016; 278p, ISBN 978-1-59327-703-1.
2. Xu, J.; Guo, P.; Zhao, M.; Erbacher, R.F.; Zhu, M.; Liu, P. Comparing Different Moving Target Defense Techniques. *Proc. ACM Conf. Comput. Commun. Secur.* **2014**, *2014*, 97–107. [CrossRef]
3. Taylor, J.; Zaffarano, K.; Koller, B.; Bancroft, C.; Syversen, J. 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, Vienna, Austria, 24 October 2016.
4. Burow, N.; Burrow, R.; Khazan, R.; Shrobe, H.; Ward, B.C. Moving Target Defense Considerations in Real-Time Safety- and Mission-Critical Systems. In Proceedings of the 7th ACM Workshop on Moving Target Defense, Online, 9–13 November 2020; pp. 81–89.
5. Lei, C.; Ma, D.; Zhang, H. Optimal Strategy Selection for Moving Target Defense Based on Markov Game. *IEEE Access* **2017**, *5*, 156–169. [CrossRef]
6. Sengupta, S.; Vadlamudi, S.G.; Kambhampati, S.; Doupé, A.; Zhao, Z.; Taguinod, M.; Ahn, G.J. A Game Theoretic Approach to Strategy Generation for Moving Target Defense in Web Applications. In Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems, Rhodes, Greece, 20–22 June 2017; p. 9.
7. Feng, X.; Zheng, Z.; Mohapatra, P.; Cansever, D. A Stackelberg Game and Markov Modeling of Moving Target Defense. In *Decision and Game Theory for Security*; Rass, S., An, B., Kiekintveld, C., Fang, F., Schauer, S., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 315–335.
8. Li, H.; Zheng, Z. Optimal Timing of Moving Target Defense: A Stackelberg Game Model. *arXiv* **2019**, arXiv:1905.13293.
9. Zhang, H.; Zheng, K.; Wang, X.; Luo, S.; Wu, B. Strategy Selection for Moving Target Defense in Incomplete Information Game. *Comput. Mater. Contin.* **2019**, *61*, 763–786. [CrossRef]
10. Hong, J.B.; Kim, D.S. Assessing the Effectiveness of Moving Target Defenses Using Security Models. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 163–177. [CrossRef]
11. Paruchuri, P.; Pearce, J.P.; Marecki, J.; Tambe, M.; Ordonez, F.; Kraus, S. 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 (AAMAS '08), Estoril, Portugal, 14–16 May 2008; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; Volume 2, pp. 895–902.
12. Ayrault, M.; Borde, E.; Kühne, U.; Leneutre, J. Moving Target Defense Strategy in Critical Embedded Systems: A Game-theoretic Approach. In Proceedings of the 2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC), Perth, Australia, 1–4 December 2021; pp. 27–36. [CrossRef]
13. Lei, C.; Zhang, H.Q.; Tan, J.L.; Zhang, Y.C.; Liu, X.H. Moving Target Defense Techniques: A Survey. *Secur. Commun. Netw.* **2018**, *2018*, 1–25. [CrossRef]
14. Cai, G.L.; Wang, B.S.; Hu, W.; Wang, T.Z. Moving target defense: State of the art and characteristics. *Front. Inf. Technol. Electron. Eng.* **2016**, *17*, 1122–1153. [CrossRef]
15. Okhravi, H.; Rabe, M.A.; Mayberry, T.J.; Leonard, W.G.; Hobson, T.R.; Bigelow, D.; Streilein, W.W. *Survey of Cyber Moving Target Techniques*; Technical Report; Defense Technical Information Center: Fort Belvoir, VA, USA, 2013.
16. Okhravi, H.; Hobson, T.; Bigelow, D.; Streilein, W. Finding Focus in the Blur of Moving-Target Techniques. *IEEE Secur. Priv.* **2014**, *12*, 16–26. [CrossRef]
17. 'KARL—Kernel Address Randomized Link'—MARC. Available online: https://undeadly.org/cgi?action=article;sid=20170613041706 (accessed on 12 April 2022).
18. Hund, R.; Holz, T.; Freiling, F.C. Return-Oriented Rootkits: Bypassing Kernel Code Integrity Protection Mechanisms. In Proceedings of the USENIX Security Symposium, Montreal, QC, Canada, 12–14 August 2009; p. 16.
19. Ayrault, M.; Borde, E.; Kühne, U. Run or Hide? Both! A Method Based on IPv6 Address Switching to Escape While Being Hidden. In Proceedings of the 6th ACM Workshop on Moving Target Defense. Association for Computing Machinery, MTD'19, London, UK, 11 November 2019; pp. 47–56. [CrossRef]
20. Prisner, E. *Game Theory through Examples*; MAA, The Mathematical Association of America: Washington, DC, USA, 2014; OCLC: 986787860.

21. Brown, G.; Carlyle, M.; Salmerón, J.; Wood, K. Defending Critical Infrastructure. *INFORMS J. Appl. Anal.* **2006**, *36*, 530–544. [CrossRef]
22. Harsanyi, J.C. Games with Incomplete Information Played by "Bayesian" Players, I–III Part I. The Basic Model. *Manag. Sci.* **1967**, *14*, 159–182. [CrossRef]
23. Paruchuri, P.; Pearce, J.P.; Tambe, M.; Ordonez, F.; Kraus, S. An Efficient Heuristic Approach for Security Against Multiple Adversaries. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, Honolulu, HI, USA, 14–18 May 2007; p. 8. [CrossRef]
24. Harsanyi, J.C.; Selten, R. A Generalized Nash Solution for Two-Person Bargaining Games with Incomplete Information. *Manag. Sci.* **1972**, *18*, 80–106. [CrossRef]
25. Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI). *EBIOS Risk Manager*; Technical Report; ANSSI: Paris, France, 2019.
26. Gadyatskaya, O.; Hansen, R.R.; Larsen, K.G.; Legay, A.; Olesen, M.C.; Poulsen, D.B. Modelling Attack-defense Trees Using Timed Automata. In *Formal Modeling and Analysis of Timed Systems*; Lecture Notes in Computer Science Series; Fränzle, M., Markey, N., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; Volume 9884, pp. 35–50. [CrossRef]
27. Common Vulnerability Scoring System version 3.1. 2021. Available online: https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf (accessed on 12 April 2022).
28. Maghrabi, L.; Pfluegel, E.; Al-Fagih, L.; Graf, R.; Settanni, G.; Skopik, F. Improved software vulnerability patching techniques using CVSS and game theory. In Proceedings of the 2017 International Conference on Cyber Security And Protection of Digital Services (Cyber Security), London, UK, 19–20 June 2017; pp. 1–6. [CrossRef]
29. Carroll, T.E.; Crouse, M.; Fulp, E.W.; Berenhaut, K.S. Analysis of network address shuffling as a moving target defense. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, Australia, 10–14 June 2014; pp. 701–706.
30. Lyon, G.F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*; Insecure.com LLC: Seattle, WA, USA, 2009.
31. Conitzer, V.; Sandholm, T. Computing the optimal strategy to commit to. In Proceedings of the 7th ACM Conference on Electronic Commerce (EC '06), Ann Arbor, MI, USA, 11–15 June 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 82–90.
32. IBM. Overview of Mathematical Programming—IBM® Decision Optimization CPLEX® Modeling for Python (DOcplex) V2.23 Documentation. Available online: https://ibmdecisionoptimization.github.io/docplex-doc/mp.html (accessed on 12 April 2022).
33. Connell, W.; Menasce, D.A.; Albanese, M. Performance Modeling of Moving Target Defenses with Reconfiguration Limits. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 205–219. [CrossRef]
34. Thomas, M.U. Queueing Systems. Volume 1: Theory (Leonard Kleinrock). *SIAM Rev.* **1976**, *18*, 512–514. [CrossRef]