Received 25 July 2022, accepted 9 September 2022, date of publication 26 September 2022, date of current version 7 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210117

RESEARCH ARTICLE

HoneyCar: A Framework to Configure Honeypot Vulnerabilities on the Internet of Vehicles

SAKSHYAM PANDA[1], (Member, IEEE), STEFAN RASS[2], (Member, IEEE), SOTIRIS MOSCHOYIANNIS[3], (Member, IEEE), K. LIANG[3], (Member, IEEE), GEORGE LOUKAS[1], (Member, IEEE), AND EMMANOUIL PANAOUSIS[1], (Senior Member, IEEE)

[1]School of Computing and Mathematical Sciences, University of Greenwich, SE10 9LS London, U.K.
[2]LIT Secure and Correct Systems Lab, Johannes Kepler University Linz, 69 4040 Linz, Austria
[3]Department of Computer Science, University of Surrey, GU2 7XH Guildford, U.K.

Corresponding author: Sakshyam Panda (s.panda@greenwich.ac.uk)

ABSTRACT The Internet of Vehicles (IoV), whereby interconnected vehicles that communicate with each other and with road infrastructure on a common network, has promising socio-economic benefits but also poses new cyber-physical threats. To protect these entities and learn about adversaries, data on attackers can be realistically gathered using decoy systems like honeypots. Admittedly, honeypots introduces a trade-off between the level of honeypot-attacker interactions and incurred overheads and costs for implementing and monitoring these systems. Deception through honeypots can be achieved by strategically configuring the honeypots to represent components of the IoV to engage attackers and collect cyber threat intelligence. Here, we present HoneyCar, a novel decision support framework for honeypot deception in IoV. HoneyCar benefits from the repository of known vulnerabilities of the autonomous and connected vehicles found in the Common Vulnerabilities and Exposure (CVE) database to compute optimal honeypot configuration strategies. The adversarial interaction is modelled as a repeated imperfect-information zero-sum game where the IoV network administrator strategically chooses a set of vulnerabilities to offer in a honeypot and a strategic attacker chooses a vulnerability to exploit under uncertainty. Our investigation examines two different versions of the game, with and without the re-configuration cost, to empower the network administrator to determine optimal honeypot investment strategies given a budget. We show the feasibility of this approach in a case study that consists of the vulnerabilities in autonomous and connected vehicles gathered from the CVE database and data extracted from the Common Vulnerability Scoring System (CVSS).

INDEX TERMS Honeypots, cyber deception, internet of vehicles, cybersecurity investment, game theory, optimisation.

I. INTRODUCTION

The Internet of Vehicles (IoV) is a distributed network that utilises data gathered by vehicles to improve safety on roads and allows interaction of vehicles with other heterogeneous networks through smart Vehicle to Everything (V2X) communication [1]. IoV has evolved from Vehicular ad hoc networks (VANETs) and is expected to eventually evolve into the Internet of Autonomous Vehicles [2]. The aim is to achieve an integrated intelligent transportation system with advanced traffic management to ensure road safety, avoid road accidents and improve driving experiences. Although connected vehicles enhance passenger experience and road safety, they have also introduced more opportunities for cyber attackers to endanger passengers and pedestrian lives. Additionally, the growing prevalence of cyber incidents has resulted in a strong demand for security in IoV solutions to ensure cost-effective, scalable and robust services that conform to legal requirements and adequately confront cyber-physical threats [3]. The steady functioning of IoV requires the integration of many different technologies, services and standards as well as increased connectivity to external environments.

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente García-Díaz.

Many external environments being vulnerable and unprotected makes IoV susceptible to cyber attacks. IoV thus needs to address numerous security issues including threats concerning the Internet of Things (IoT) as well as new threats specific to connected vehicular networks.

To attend to the growing concerns on cyber security and privacy preservation within autonomous and connected vehicles, the United Nations Economic Commission for Europe has produced a set of guidelines on best practices to protect these vehicles and reduce cyber-physical risks. In particular, the United Nations Economic Commission for Europe's new regulations[1] highlight the importance of detection and mitigation of cyber incidents. Threat detection, in turn, relies on cyber threat intelligence. Defensive deception is an effective means to collect information on adversaries' intents and actions [4]. Cyber deception, specifically honeypots [5], [6], have been studied within the field of vehicular networks to identify security gaps and collect cyber threat information to address security challenges.

In IoV, the authorisation and the communication modules which handle the most critical security features are often targeted by attackers leading to serious security, privacy, and physical impacts. Honeypots can significantly contribute to protecting these systems by absorbing damage and collecting information about attackers' intents and actions. The knowledge from honeypots can be leveraged to identify and close security gaps, implement measures to avoid attacks on vehicles and infrastructure, support intrusion detection and prevention systems, and realise a wide range of attacks and threats to IoV. The gathered information may also be used to manage cyber risk through protection [7], [8], mitigation [9], [10], [11] as well as support forensic investigations of cyber security breaches [12] and obtain evidence to take legal actions or support cyber insurance [13], [14], [15]. Even so, it is important to know how attackers gain access and proceed in the network and how the vehicles and their connected environment behave in case of such attacks.

As honeypots do not involve regular users and usage patterns, a key challenge in designing honeypots is convincing attackers that these are real systems [16]. This problem can be addressed by strategically configuring honeypots such that they resemble components of the IoV network [17]. Our research is motivated by the fact that an effective cybersecurity strategy must consider the advancing threat landscape, which can only be achieved by understanding attackers' actions. In this paper, we propose HoneyCar, a novel decision support framework that offers cost-effective honeypot configuration capabilities to a network administrator. HoneyCar allows the decision-maker to compute optimal strategies for active re-configuration of honeypots based on a set of available vulnerabilities, available budget and the expected benefit acquired through each available configuration action.

[1]UN Regulation No. 155 - Cyber security and cyber security management system, `https://unece.org/transport/documents/2021/03/standards/unregulation-no-155-cyber-security-and-cyber-security`

Note that, we only consider long-distance communication mode in IoV which includes most of the critical connections concerning security; in particular, V2X which supports all the communication between a vehicle and its environment whether it is another vehicle, roadside infrastructure, or the cloud. The attacker is considered an external entity aiming to exploit a vulnerability in the IoV network.

More specifically, HoneyCar uses a game-theoretic model to capture the interaction between the Defender and the Attacker using a repeated imperfect-information zero-sum game. This work makes the following contributions:

- presents a novel decision support framework called HoneyCar to examine practical (cost-effective) honeypot investment decisions.
- presents a novel game-theoretic model demonstrating the strategic use of honeypot through cost-effective selection and configuration (or re-configuration) of honeypot within a budget.

We evaluate HoneyCar using a case study of connected and autonomous vehicles vulnerabilities that are based on real-world data gathered through:

- the Common Vulnerabilities and Exposure (CVE) repository, which is one of the most popular databases that identifies, defines and catalogues publicly disclosed cybersecurity vulnerabilities.
- the Common Vulnerability Scoring System (CVSS), which provides various characteristics of a vulnerability and a numerical score reflecting its severity. HoneyCar uses several of these characteristics to support the decision making.

The rest of this paper is structured as follows. Section II presents the related work on the use of honeypots in connected vehicular networks. It also presents an overview of the application of game theory to honeypot deception to secure networks and highlights how this work extends the current literature. Section III introduces the honeypot deception model together with the decisions of the Defender and the game-theoretic model for optimal configuration of honeypots. Section IV presents the mathematical analysis for computing optimal honeypot configurations to support the investment decisions. Section V presents a case study using the known vulnerabilities related to autonomous and connected vehicles to evaluate HoneyCar. Finally, Section VI concludes this paper by discussing the limitations of the proposed method and potential future work.

II. RELATED WORK

As IoV inherit many of the characteristics of regular computer networks, most of the countermeasures for computer networks apply to IoV. This leads to many open security problems from identification and communication to standardisation [21], [22] that challenge the privacy and security in IoV [23]. Successful cyber attacks on IoV networks have been documented including those on security keys used in electronic control units, wireless key fobs, tyre-pressure

TABLE 1. Honeypots and vehicular networks.

Ref.	Solution	Strategic	Honeypot re-configuration
Patel and Jhaveri [5]	honeypot detection approach to mitigate selfish vehicles from the network	✓	x
Verendel et al. [6]	use of honeypot to collect attacker information in in-vehicle network	x	x
Babu and Usha [18]	honeypot detection approach to identify and isolate Black Hole attacks in MANETs	x	x
Daubert et al. [19]	use of honeypots to detect intrusion and collect attacker information in unmanned aerial vehicles	x	x
You et al. [20]	honeypot framework to gather large scale attack data in programmable logic controllers	x	x
Our approach	framework to determine cost-effective honeypot configuration to protect and collect attacker information in IoV	✓	✓

monitoring systems, inertial measuring units, braking systems, and more [24], [25], [26]. Some of the proposed countermeasures include threat modelling of security risks in IoV [27], intrusion detection systems for protection against internal and external attacks [28], secure routing protocols [29], [30], privacy mechanisms [31], encryption key management [32] and honeypots to collect adversarial information in IoV [17].

Deception is used for both defensive and offensive interactions. Researchers have developed various defences against offensive deception in connected vehicular networks such as Sybil attacks [33], false positioning attacks [34], illusion attacks [35] and topology poisoning attacks [36]. Honeypots have been widely implemented in network security for defensive deception [37], [38], moving target defence [39] and against advanced persistent threats [40]. Honeypots could be used to detect a broad range of attacks on connected and autonomous vehicles. These attacks could include random attacks on the vehicles and connected infrastructure as well as targeted attacks on IoV. This section presents only those works that are most relevant to our approach and describes how HoneyCar differs or improves upon it.

A. HONEYPOTS FOR VEHICLES

Honeypots have been used to identify vehicles behaving selfishly to preserve their resources [5], lure and isolate attackers [18], collect adversarial information [6], and support intrusion detection systems for VANETs [17] and unmanned aerial vehicles [19]. Verendel *et al.* [6] proposed in-vehicle honeypot models to gather attack data to support a secure wireless infrastructure for vehicular communication. You *et al.* [20] proposed a framework to deploy a physical high-interaction programmable honeypot. The honeypot has cross-network access, one-to-many mechanisms and high-interaction capabilities replicating a programmable logic controller to gather large-scale attack data. The authors have created an operational scenario to deploy the honeypot and collect attack data at a low cost. Moreover, honeypots are used for intrusion detection in connected vehicular networks. The aforementioned approaches have focused on identifying the number of honeypots to deploy, the location and the type of honeypot to deploy in the network. HoneyCar extends the frontier of the application of honeypots by focusing on the strategic re-orientation of honeypots to deceive attackers and collect threat data in IoV. It introduces honeypot re-configuration strategies to enhance the effectiveness of honeypots as the interaction capabilities and location of honeypots are not the only predictive factors in deceiving attackers. Table 1 provides a list and brief description of existing work on the application of honeypots in vehicular networks in comparison to our work.

B. STRATEGIC DECEPTION USING GAME THEORY

Garg and Grosu [41] investigated the allocation of honeypots in a network as a signalling game where the Defender implements k honeypots out of n possible host systems. Carroll and Grosu [37] studied a similar signalling game in which the Defender can disguise a real system as a honeypot and a honeypot as a real system. Çeker *et al.* [42] build on [37] to design a honeypot-based deception game where the Defender can either choose a system to be a real system or a honeypot and the Attacker can either observe, attack or retreat. The goal was to deceive attackers by placing several honeypots in the network to mask valuable systems in the network. When attackers cannot determine the type of systems due to deployed deception, they either postpone the attack or spend additional resources to determine the identity of a system. Píbil *et al.* [43] and Kiekintveld *et al.* [44] investigated ways of designing honeypot systems to optimise the probability of the Attacker choosing to attack the honeypot rather than the real system. La *et al.* [45] analysed an IoT network along with a honeypot to defend systems from attacks and extended the analysis from a single-shot game to a repetitive game considering the deceptive aspects of the players. While most of the aforementioned works investigated the deceptive use of honeypots to minimise the probabilities of attacks on the real system, the ways of engaging the attackers remained understudied. HoneyCar closes this gap by looking into the use of honeypot to deceive adversaries with the explicit goal of optimal re-configuration of the honeypot to increase Attackers' engagement and thereby enhancing the information gathered on adversarial behaviour.

There are relatively fewer papers that use game theory to optimise the information learned about attackers through multi-round games. Zhuang *et al.* [46] proposed a multi-round signalling game to investigate strategies of deception and resource allocation. In each round of the game, the Defender decides on investing in either short term expenses or long term capital investments in defence. To deceive the Attacker, Defender then chooses how much of the investment information to reveal as a signal. The Attacker on observing the signal decides whether to attack or not. Durkota *et al.* [47] investigated similar multi-round interactions as a Stackelberg game. The Defender plays first by placing honeypots to harden the network. The Attacker with knowledge of the number of honeypots but not their identities follows the Defender. Attack graphs are used to represent Attacker's multi-round strategies and develop network hardening techniques to enhance security. Unlike most game-theoretic models of deception which are either static games or single-shot games, we study the interaction between the Defender and the Attacker as a multi-round game. Also, concerning the structure of the game, the work closest to ours is that of Durkota *et al.* [47] for regular computer networks. HoneyCar improves on this by introducing re-configuration (similar to hardening operation) of honeypots in each round of the game to engage attackers, rather than just placing honeypots to harden the IoV network. In conclusion, although many models and frameworks to strategically support defensive deception have been proposed, none had investigated the importance of honeypot re-configuration as HoneyCar for effective deception. Contrary to the previous works which aimed to support the strategic use of honeypots either by identifying the best number of honeypots to implement or the best location of the honeypot in the network, HoneyCar introduces honeypot re-configuration strategies to enhance attacker engagement and gather better cyber threat intelligence. Finally, HoneyCar aims to assist the technical aspects of honeypot deception and help network administrators to identify the best honeypot investment strategy by using publicly disclosed vulnerabilities from the CVE database and their associated characteristics from the CVSS repository.

III. SYSTEM MODEL

We assume that a network service provider, henceforth called the Defender, decides to invest in honeypots to dissuade adversaries, the Attacker, from critical infrastructure and to capture adversarial activities in the network. Her goal is to design a honeypot deception strategy to deceive the Attacker into considering the honeypots as systems of importance while minimising the cost of investment in honeypots. To achieve this, the Defender must decide on the type of honeypot to implement, the configuration of the honeypot and the available investment budget. Successful deception may lead to the identification of potential attributes of intrusion and techniques of the Attacker.

A. HONEYPOTS AND CHOICES

To make security investment decisions with a limited budget and a strict reaction time frame is always challenging. Key questions the Defender face are: (i) *whether to invest in cyber deception* and (ii) *how this deception must be implemented and delivered*. To support these decisions, we consider the Defender has an investment budget - which may be determined by a cyber investment decision-making methodology such as [10] and [9] - and aims to cost-effectively address the aforementioned questions. The computation of the available investment budget, as well as its size, is out of the scope of this paper.

Let B denote the required budget for implementing and maintaining a honeypot. We assume that L is the expected loss caused by a successful breach on a system in the IoV network. Adding a honeypot in the network would be beneficial when L is greater than B as investing in deception is affordable and less expensive than the expected loss from the cyber incidents. Further, the information gathered from the honeypot could be used to reduce this expected loss. Likewise, the Defender would prefer not to implement a honeypot when $L < B$ as the implementation cost would exceed its reward leading to a negative Return on Security Investment (ROSI).

When investing is a preferable choice, the next step is to identify what type of honeypot to implement i.e., either a *low-interaction* honeypot (LIH) or a *high-interaction* honeypot (HIH), which are the most common categories utilised. Other categories of honeypots such as hybrid and medium-interaction honeypots could also be considered in the modelling choices. Each honeypot type has an implementation cost, expressed as C_l and C_h for LIH and HIH, respectively, and a learning rate (λ). The implementation cost is exogenous and depends on the resources required to implement and maintain it. For example, LIHs are easy to deploy and maintain, usually hosted on a virtual machine, and offer limited services such as Internet protocols and network services without any interactions with the operating system. The limited interaction enables them to minimise the risk of exploitation by containing the activities of the Attacker within the deceptive environment. These low-level interactions only capture limited information on the Attacker's activities, thus leading to a lower learning rate. On the other hand, HIHs provide greater levels of interaction such as interactions with a real/virtual operating system. These additional functionalities introduce complexities in implementing and maintaining them [48], besides the higher risk of breaching the real systems through them. The learning rate for HIH is thus higher as they can capture more activities of the Attacker contributing to better cyber threat intelligence. Table 2 presents the list of symbols used in this paper.

Let $\mathcal{I} \subseteq \mathcal{V}$ denote the set of vulnerabilities to be offered in a honeypot. The complexity to exploit a vulnerability (i.e., low, medium or high) is obtained from Common Vulnerability Scoring System (CVSS) and is used to differentiate vulnerabilities that can be offered through LIH and HIH. We assume vulnerabilities with ''high'' complexity can only be offered through HIH. The Defender has to strategically offer the vulnerabilities to engage the Attacker, which could involve re-configuring the honeypot multiple times. The re-configuration activities can be achieved using honey-patches, also known as ghost-patches [49]. A honey patch can function similar to a regular patch and has two components: (i) a traditional patch for the known software vulnerability, and (ii) additional code to create fake vulnerabilities to mislead the Attacker [50]. The Defender has to bear costs to perform the reconfiguration and maintenance of the honeypot. We refer to these costs as the re-configuration cost $S(\mathcal{I})$. The re-configuration action is analysed using a game-theoretic framework based on a repeated game, as discussed in the following section. Besides, respecting the available budget of B, the Defender's security investment decisions are determined by:

- the expected loss L, without a honeypot, due to a breach;
- the choice of honeypot type to implement $\in \{l, h\}$;
- the set of vulnerabilities to offer in a honeypot, $\mathcal{I} = \{v_1, v_2, \cdots, v_n\} \subseteq \mathcal{V}$, and their associated exploitation times $\{t_1, t_2, \cdots, t_n\}$.
- the cost of re-configuration $S(\mathcal{I}) = \{(s_1^+, s_1^-), (s_2^+, s_2^-), \cdots, (s_n^+, s_n^-)\}$, where s_j^+ and s_j^- are costs (in time) for deliberately *opening* (indicated by ''+'' superscript) or knowingly *patching* ('$-$'' superscript) vulnerability $v_j \in \mathcal{I}$, respectively.

B. GAME MODEL

The choice to implement a honeypot type leads to a distinct strategic game between the Defender and the Attacker. Figure 1 illustrate the choices and the games. We refer to these games as Honeypot Configuration Games (HCG) represented as $\mathcal{G}_l$ and $\mathcal{G}_h$ (related to LIH and HIH, respectively). We utilise game theory [51] which studies optimal decisions involving multiple decision-makers (also referred to as agents or players), including adversarial settings where two or more players have opposing goals, to support the optimal decision.

We model the HCGs as a *repeated imperfect-information zero-sum two-player game*. Security games involve interaction between players with exactly opposite goals making zero-sum games the best fit to model them [52]. Zero-sum games, in particular, capture the worst-case scenarios for the Defender, which is to face the Attacker who is after the most valuable assets. The key motivation behind using zero-sum games is that they can provide desirable solutions against any opponent and not just against rational opponents [43]. On some occasions, this may mean that the Defender spends more resources particular when the Attacker is non-strategic or naive. However, in the absence of complete knowledge about the Attacker's incentives, it is reasonable to model the

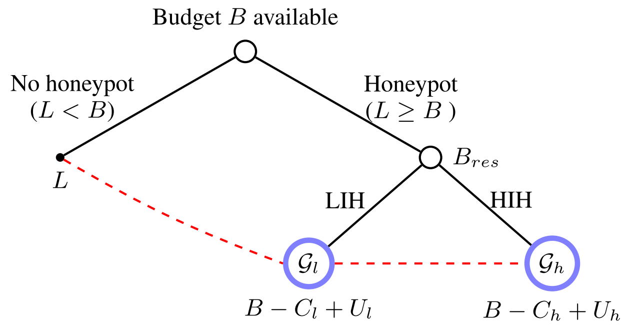


FIGURE 1. Defender's decision tree with sub-games $\mathcal{G}_l$ and $\mathcal{G}_h$.

proposed investment and honeypot configuration decisions as a zero-sum game to achieve robustness against strategic adversaries, which most often are mentioned in the literature as Advanced Persistent Threats (APTs) [53].

Repeated games refer to games that have multiple independent rounds. In each round of HCG, the Defender re-configures the honeypot. We define re-configuring as a hardening operation which involves replacing an observed exploited vulnerability with another from the $\mathcal{I}$. The replacement choice aims to convince the Attacker that the honeypot is a valuable system. The important aspect of this modelling decision is that HoneyCar considers not only the costs for implementing and maintaining the honeypots but also the cost for re-configuring the honeypot. This comes into the model via the cost variables s_i^+ and s_i^- to offer or revoke some vulnerability v_i in a new configuration.

In each HCG, the Defender chooses a set of vulnerabilities $\mathcal{I} \subseteq \mathcal{V}$ to offer. The offered vulnerabilities can be observed by the Attacker during system reconnaissance, who in response chooses a vulnerability from the offered ones to exploit. Having a honeypot in the network makes it difficult for the Attacker to gain the exact configuration of the network as he cannot distinguish between a real system and a honeypot with certainty. The Attacker also has no information regarding the kind of system he is targeting and the value of the asset he will have access to by exploiting the vulnerability. Furthermore, we make the following assumptions about the Attacker: (i) he needs to compromise at least one vulnerability to mount an attack on the IoV network; and (ii) he is aware of the possibility of a honeypot (decoy system) and plays the best strategy, that is, targets a vulnerability that maximises the chances of successfully breaching the targeted system.

LIH is a cheaper option than HIH, but it is more likely to be recognised by the Attacker leaving the Defender with a lower reward. The Defender, therefore, has to undertake some cost-benefit analysis to identify the best type of honeypot for defending the IoV network. The choices of the players determine the game utility which is the outcome of the cost-benefit analysis performed when these strategies are played. As illustrated in Figure 1, the choice of a honeypot type to implement depends on the residual security budget defined as

$$B_{res} = \max \left\{ B - C_l + U_l \, , B - C_h + U_h \right\} \tag{1}$$

where U_l and U_h are the Defender's expected game utilities from $\mathcal{G}_l$ and $\mathcal{G}_h$, respectively. These utilities correspond to the payoffs of the Defender for implementing a honeypot type with a set of vulnerabilities ($\mathcal{I}$) to be exploited by the Attacker. In each of these games, the Defender selects $\mathcal{I} \in \mathcal{V}$, which is different for LIH and HIH. It is evident from Figure 1 that the selection of a honeypot type and the vulnerabilities to offer would alter B_{res}.

C. GAME ITERATIONS AND UTILITIES

The Defender configures the honeypot with a selection of m out of $n = |\mathcal{V}|$ possible vulnerabilities to let the Attacker mount any available exploit on the offered m weaknesses denoted by $\mathcal{I} = \{v_1, \ldots, v_m\} \subseteq \mathcal{V}$. The Attacker would anticipate that the Defender will eventually discover the vulnerability expecting the weakness to be patched as in a real system. But other vulnerabilities will remain, and new vulnerabilities are opened up in an attempt to retain the attractiveness of the honeypot for the Attacker. This is exactly the dynamics that HCG implements. The games proceed with an assumption that in each iteration the Attacker exploits only one vulnerability from the offered ones and the Defender re-configures the honeypot by patching the exploited vulnerability to demonstrate activity as she would on a real system. The Defender further offers a new vulnerability v_t from the set of remaining vulnerabilities as an attempt to keep the Attacker interested in the honeypot.

Each vulnerability has its own cost and benefits based on the time required to exploit it and the observed activities of the Attacker during the interaction with the vulnerability. For reference, Figure 2a and Figure 2b illustrate the players' interaction in HCG. The strategic choices of the players are the following:

- the Defender chooses an m-element subset $\mathcal{I} \subseteq \mathcal{V}$, making the strategy space having cardinality $\binom{n}{m}$. The choice of m is exogenous.
- the Attacker chooses a vulnerability $v_j \in \mathcal{I}$ to exploit. This choice can be motivated based on several criteria such as time, resources and skills.

Remark 1: Investigating adversarial profiles is beyond the scope of this paper and for simplicity, yet realistic, we consider that the Attacker prefers the "easiest" of all $v_t \in \mathcal{I}$ to break into a targeted system. Likewise, we do not further study sequential combinations of several exploits, and focus our analysis on a single exploitation trial, corresponding to a single round of our game. Repeated attempts as occur in practice then manifest as rounds of our repeated game.

The Attacker aims to minimise the time spent to exploit v_t to decrease the engagement time with a targeted system. Furthermore, intelligent attackers always aim to compromise a targeted system while remaining undetected. The execution of such a stealthy attack allows the attacker to compromise the targeted system without raising any alerts. While the Defender has a contrary objective of maximising the Attacker's time spent interacting with the honeypot rather than the real system to gather as much intelligence on the Attacker's activities as possible. The Defender's expected utility in HCG is defined by the amount of *cyber threat intelligence gained* $g(\mathcal{I})$ and the *cost of monitoring* offered vulnerabilities $c(\mathcal{I})$ i.e.,

$$U_D = g(\mathcal{I}) - c(\mathcal{I}) \tag{2}$$

We consider $g(\mathcal{I})$ and $c(\mathcal{I})$ to be logistic functions, detailed in the following section. Note that, for ease of presentation, U_D is used to express the expected game utility of the Defender regardless of the chosen type of honeypot.

Remark 2: The Defender aims at maximising the overall time spent by the Attacker interacting with the honeypot to increase cyber threat intelligence. The Defender's expected game utility is defined as a function of the gained cyber threat intelligence and the cost of monitoring the offered vulnerabilities.

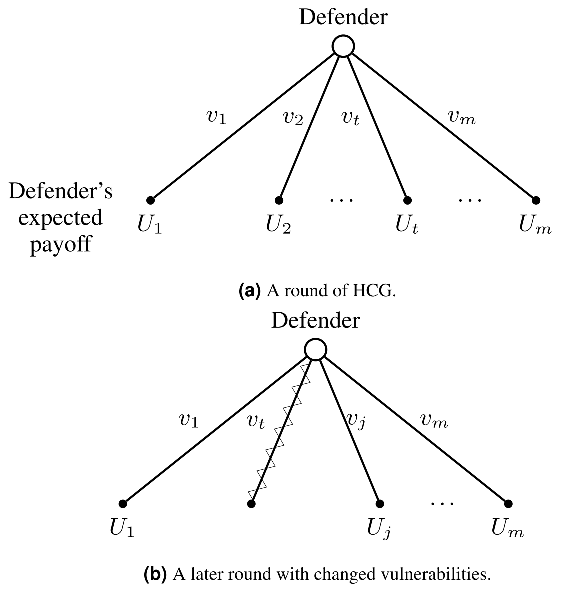


FIGURE 2. Dynamic of HCG: (i) In a round of HCG, the Defender offers a set of vulnerabilities to be exploited by the Attacker; (ii) In the next round of HCG, the Defender patches a vulnerability v_t and offers a new one v_j from $\mathcal{V}$.

IV. OPTIMALITY ANALYSIS

In this section, we present the core methodology of HoneyCar, which allows it to generate the optimal honeypot configuration strategies for the Defender.

A. DECISION COMPLEXITY

In the game, the pure strategy of the Attacker is to choose a vulnerability v_j to exploit the desirable target. Likewise, the Defender must choose a set of vulnerabilities $\mathcal{I}$ to offer out of $\mathcal{V}$. Thus, the Defender select m out of n vulnerabilities from $\mathcal{V}$ to offer through a honeypot with payoffs being defined as if i iterate over all $\binom{n}{m}$ sets of vulnerabilities, and $v_j \in \mathcal{I}$ being the targeted vulnerability by the Attacker. We can represent the HCG in a matrix form with $m \cdot \binom{n}{m} \in O(m \cdot n^m)$ elements

TABLE 2. List of symbols.

Symbol	Interpretation
	Constants and Functions
B	budget to invest in honeypot
C_h	implementation cost of HIH
C_l	implementation cost of LIH
$\mathcal{V}$	set of available vulnerabilities
$\mathcal{I}$	set of offered vulnerabilities ($\mathcal{I} \subseteq \mathcal{V}$)
L	expected loss without honeypot
$\mathcal{S}(\mathcal{I})$	re-configuration cost for $\mathcal{I}$
	Game Variables
B_{res}	residual budget after implementing honeypot
s_j^+	cost of adding vulnerability $v_j \in \mathcal{I}$
s_j^-	cost of patching vulnerability $v_j \in \mathcal{I}$
$c(\mathcal{I})$	honeypot monitoring cost when offering $\mathcal{I}$
$g(\mathcal{I})$	gained cyber threat intelligence when offering $\mathcal{I}$
p_j	probability of offering vulnerability $v_j \in \mathcal{I}$
t_j	time to exploit vulnerability $v_j \in \mathcal{I}$
U_h	expected game utility of the Defender with HIH
U_l	expected game utility of the Defender with LIH
$\nu(\mathcal{I})$	optimum value obtained from optimisation
α	average monitoring cost factor
β	optimisation smoothing factor
λ	honeypot learning rate

which is computationally intractable even for a moderately small n, m [51], as there is no efficient way of determining the best combination. Even for a small number of vulnerabilities such as $n = 13$ there are over a billion combinations to choose from. To avoid the combinatorial explosion, we restrict the Defender's choices to only the set $\mathcal{V}$ of all vulnerabilities from which she can select n vulnerabilities and assign her randomised choices as a probability distribution over $\mathcal{V}$. We represent this in the form of a matrix game with the action space being $\mathcal{V}$ instead of a family of all n-element subsets of $\mathcal{V}$ transforming the action space to a tractable size $n = |\mathcal{V}|$. The equilibrium strategy of the Defender in the mixed extension of this revised game is represented by a vector $\mathbf{x} = [p_1, \ldots, p_n]^T \in \Delta(\mathcal{V}) \subseteq [0, 1]^n$, *constrained* to satisfy

$$\left|\left\{i : p_j \neq 0\right\}\right| \leq m \tag{3}$$

so that HoneyCar offers the freedom to implement m or fewer vulnerabilities in the honeypot. The use of the equilibrium strategy by the Defender allows for the optimisation of the configuration of the honeypot given the use of the equilibrium strategy by the Attacker. On the other hand, the use of the equilibrium strategy allows the Attacker to increase the chances of successfully exploiting a vulnerability while limiting the interaction time. This assumption is considered acknowledging that attackers are opportunists [54].

B. PAYOFFS MAXIMISATION

Next, we formalise the players' objectives by defining the probabilities of certain outcomes and the expected payoff values each player gains from these outcomes. The zero-sum payoff is derived using the equation (2) and depends on the time required to exploit a vulnerability. Time to exploit a vulnerability is a critical element of our analysis. It shapes the players' objectives, strategies and expected payoffs. The Attacker aims to minimise the exploitation time whereas the Defender aims to maximise the exploitation time to gather information about the attacker. As we do not offer a single vulnerability from $\mathcal{I}$ as a pure strategy of the Defender but a set of vulnerabilities $\mathcal{V}$ to choose from, the game does not satisfy the usual form of a matrix game. Considering t_j as the time to exploit vulnerability v_j, the expected payoff of the Defender equals

$$u_j := \begin{cases} p_j \cdot t_j, & \text{if } p_j > 0; \\ \infty, & \text{otherwise.} \end{cases} \tag{4}$$

This expresses that the payoff is either the expected time to exploit a vulnerability times the probability that the vulnerability was offered by the Defender ($p_j > 0$) or infinite if the Attacker invests time in creating an exploit for a vulnerability that was not offered by the Defender. Since the Attacker aims at minimising the exploitation time t_j, any choice leading to ∞ would be a dominated strategy and hence would never be chosen to be exploited by the Attacker.

In standard game theory with real-valued utility functions, the existence of a Nash equilibrium is assured when the strategy spaces are non-empty compact subsets of a metric space and the utility functions are continuous to the metric [51]. It can be observed in (4) that the expected payoff is an unbounded, even discontinuous function and thus needs to be revised to obtain the equilibrium. We address the unbounded-discontinuous nature of the utility function by defining the nature of the Attacker to be a maximiser of the expected residual time obtained for using an exploit. This assumption is in line with the rational characteristics of attackers seen in the wild who aim to maximise their payoff from an attack [37]. We choose any constant $T > \max\left\{t_j : v_j \in \mathcal{V}\right\}$, and let the Attacker maximise the expected residual time $T - t_j$, which is by construction equivalent to minimising t_j. Substituting t_j by $T - t_j$ in (4), we let the Attacker maximise the expected payoff without altering the objective and just changing the optimisation goal. The revised expected payoff function of the Attacker is

$$u_j := p_j \cdot (T - t_j) \tag{5}$$

whenever the v_j-th vulnerability in $\mathcal{I} \subseteq \mathcal{V}$ is chosen. Depending on whether vulnerability v_j was offered by the Defender in the first place, u_j is either

- $u_j > 0$, if $p_j > 0$, when the vulnerability was offered, or
- $u_j = 0$, if $p_j = 0$, which is a strategically dominated choice for a maximising Attacker.

It can be asserted that the revised payoff function (5) is bounded and no longer has the troublesome discontinuities as observed in (4). Hereafter, we consider (5) as the expected payoff of the Attacker and equivalently multiplied by -1 as the expected payoff of the Defender.

Our analysis is supported through two distinct versions of HCG: (i) *without* the cost of re-configuration (HCG-a); and (ii) *with* the cost of re-configuration (HCG-b). HCG-a presents the baseline model of interaction between the Defender and the Attacker with no cost for re-configuring the honeypot in each round of the game. In HCG-b, we increase the complexity of the model to account for the cost of patching and opening a new vulnerability (re-configuring the honeypot in each iteration of the game) which affects the overall budget available to the Defender. The re-configuration cost moderates the number of rounds the game could be played thus affecting the Defender's expected utility. Through these games, we investigate the importance of re-configuration cost in the Defender's investment decisions, which was identified by Rass *et al.* [55] as a key component for an acute game-theoretic model. A comparative analysis of these games based on the developed use case is discussed in Section V.

C. DECISION ANALYSIS FOR HCG-a

HCG-a presents a baseline model for the optimal honeypot configuration when the re-configuration cost is ignored to provide insights on the baseline configuration strategies. These baseline strategies are then used to assess the impact of the re-configuration cost on the Defender's decisions which are studied by the HCG-b version, presented in the following section. We let $\mathbf{x} = [p_1, \ldots, p_n] \in [0,1]^n$ and $\mathbf{y} = [q_1, \ldots, q_n]^T \in [0,1]^n$ both being probability distributions, i.e., constrained to satisfy: (i) $p_k \geq 0, q_k \geq 0, k = 1, 2, \cdots, n$, and (ii) $\sum_k p_k = 1$ and $\sum_k q_k = 1$. The payoff of the game depends on the Attacker's probability of choosing a vulnerability to exploit (represented as q_j) and the Defender's probability of choosing the targeted vulnerability to offer (represented as p_i). Since both make their choices at random, the expected utility is expressed as

$$U(\mathbf{x},\mathbf{y}) = \sum_{i,j} p_i \cdot q_j \cdot \bar{u}_j \overset{(5)}{=} \sum_{i,j} p_i \cdot q_j \cdot (p_j \cdot [T - t_j]) \quad (6)$$

From equation (6), it is observed that the expected utility implicitly depends on i. The variable i iterates over all choices $i \in \mathcal{V} = \{v_1, \ldots, v_n\}$ of the Defender, but the payoff to the Attacker occurs if the vulnerability v_i was actually chosen for exploitation. The expected utility depends on the sum running over i in (6).

Since the expected payoff continuously depends on the mixed strategies $\mathbf{x}, \mathbf{y}$ of both players, using Glicksberg's theorem [51], we have an equilibrium when $\min_\mathbf{x} \max_\mathbf{y} U = \max_\mathbf{y} \min_\mathbf{x} U$. Towards solving this minimax optimisation, let us rewrite U in matrix notation. To this end, observe that

$$\mathbf{x}^T\mathbf{x} = \begin{pmatrix} p_1p_1 & p_1p_2 & \cdots & p_1p_n \\ p_2p_1 & p_2p_2 & \cdots & p_2p_n \\ \vdots & \vdots & \ddots & \vdots \\ p_np_1 & p_np_2 & \cdots & p_np_n \end{pmatrix} \text{ and}$$

$$\times \begin{pmatrix} q_1(T-t_1) & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & 0 \\ \vdots & \vdots & q_j(T-t_j) & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & 0 & q_n(T-t_n) \end{pmatrix}$$

$$= \underbrace{\text{diag}[T-t_1, \ldots, T-t_n]}_{=:\Gamma} \cdot \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} = \Gamma \cdot \mathbf{y} \quad (7)$$

and multiplying $\mathbf{x}^T\mathbf{x}$ by the diagonal matrix $\Gamma \cdot \mathbf{y}$, we technically multiply the j-th row in (7) by the value $q_j(T - t_j)$ for all $j = 1, 2, \ldots, n$. Now, to reproduce (6), we are left with adding up the rows in the resulting $n \times 1$ vector, which is doable by a multiplication with the row-vector $\mathbf{e} = [1, 1, \ldots, 1] \in \mathbb{R}^{1\times n}$. The matrix representation being $U(\mathbf{x},\mathbf{y}) = \underbrace{\mathbf{e} \cdot \mathbf{x} \cdot \mathbf{x}^T \cdot \Gamma}_{=:\mathbf{A}(\mathbf{x})} \cdot \mathbf{y} = \mathbf{A}(\mathbf{x}) \cdot \mathbf{y}$, in which $\mathbf{A}(\mathbf{x})$ is a $(1 \times n)$-matrix for all $\mathbf{x}$.

Let the Defender choose $\mathbf{x}$. The Attacker's problem, following the Defender, is choosing $\mathbf{y}$ towards

$$\max_\mathbf{y} \mathbf{A}(\mathbf{x}) \cdot \mathbf{y} = \max_{1 \leq i \leq n} \mathbf{A}(\mathbf{x})\mathbf{e}_i^T \quad (8)$$

since the inner maximisation is the selection of the largest element from the vector $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^n$, achievable by a discrete optimisation over all unit vectors $\mathbf{e}_i$ with a 1-entry only at the i-th coordinate and being zero elsewhere. Introducing the scalar ν to represent the value of the inner optimisation (8), we arrive at the Defender's problem

$$\min \nu \quad \text{s.t.} \begin{cases} \nu & \geq \mathbf{A}(\mathbf{x})\mathbf{e}_i^T \quad \text{for all } i = 1, 2, \ldots, n, \\ \sum_{i=1}^n p_i & = 1, \\ p_i & \geq 0. \end{cases} \quad (9)$$

This is a nonlinear problem with smooth objectives and constraints. The value obtained from this optimisation is used to compute $g(\mathcal{I})$ and $c(\mathcal{I})$, used in the equation (2). We define $g(\mathcal{I})$ and $c(\mathcal{I})$ to be logistic functions [56], and are represented as

$$g(\mathcal{I}) = e^{\frac{1}{1+\lambda\cdot\nu_\mathcal{I}}} \quad \text{and} \quad c(\mathcal{I}) = e^{\frac{1}{\alpha\cdot|\mathcal{I}|\cdot\nu_\mathcal{I}}} \quad (10)$$

where $\nu_\mathcal{I}$ denotes the optimum ν obtained by solving (9) (or (12) for HCG-b in next section); and $\lambda \in (0,1)$ denotes the learning rate for a honeypot type as detailed in Section III.

The λ value can also be expressed as the degree of effectiveness of a honeypot which can be represented as [57]: (i) the time required by the Attacker to realise that he is attacking a decoy system; (ii) the type of honeypot implemented to deceive the Attacker; or (iii) the amount of collected attack data. As any of the above factors are higher for HIH, we consider λ for HIH is larger than LIH. The $\alpha \in (0,1)$ value denotes the average monitoring cost factor of a honeypot type. HIH must be supported with adequate data collection and control mechanisms to ensure reliable adversarial information gathering, and to prevent the honeypot from being used as a foothold to attack connecting devices and networks. HIH requires significant resources for continuous monitoring and logging of all the interactions to determine the Attacker's motives and methods. Thus, high-interaction honeypots generally exhibit greater monitoring costs than low-interaction honeypots leading to a higher α.

1) EXAMPLE WITH HCG-a

Let the Defender be the minimising player with HIH. We consider the exploitation time as categorical values similar to the complexity metric of the vulnerabilities in CVSS scores. We let the exploitation time for *low, medium*, and *high* complexity vulnerabilities be $1, 2$ and 3, respectively. For example, let $m = 3, n = |\mathcal{I}| = 3$, the associated exploitation time $\text{diag}(\Psi) = \{2, 1, 3\}$, and $T = 4$. The payoff matrix Γ is constructed as

$$\Gamma = \begin{pmatrix} T - t_1 & 0 & 0 \\ 0 & T - t_2 & 0 \\ 0 & n0 & T - t_3 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Analysing the game by solving (9), yields $v(\mathcal{I})^* \approx 0.545$ and the result at an equilibrium strategy $x_0^* \approx (0.273, 0.182, 0.545)$. To compute the expected game utility U_D using (2), we let the re-configuration cost $\mathcal{S} = 0$, as we demonstrate this example with HCG-a, $\lambda = 0.6$ and $\alpha = 0.7$. Then, using (10), we compute $g(\mathcal{I}) \approx 2.124$ and $c(\mathcal{I}) \approx 0.418$ leading to $U_D \approx 1.706$.

D. DECISION ANALYSIS FOR HCG-b

In the following, we extend our analysis to include the re-configuration cost of the Defender in HCG and refer to it as HCG-b. For each vulnerability $v_j \in \mathcal{I}$, we denote s_j^+ as the cost (in time) for "offering" vulnerability v_i in the honeypot, and s_j^- as the cost (in time) for patching vulnerability v_i, i.e., removing it from the honeypot. We further consider that if a vulnerability v_j in the current round remains in the honeypot for the next round, no re-configuration cost occurs and c_i^+ is zero. Likewise, if the vulnerability v_j is not offered in this round of the repeated game nor included in the next round, then c_j^- is zero. Assuming stochastic independence of the rounds in the game, we investigate the cases where:

> *Case 1:* vulnerability v_j was offered in the current round with probability p_j and is patched in the next round, with probability $1 - p_j$. Thus, the expected re-configuration cost is $p_j \cdot (1 - p_j) \cdot s_j^-$.
> *Case 2:* vulnerability v_j was not offered in the current round with probability $1 - p_j$ and the honeypot will be configured to have v_j in the next round leading to the expected re-configuration cost of $(1 - p_j) \cdot p_i \cdot s_j^+$.

Since the coefficients of the expected re-configuration cost in both cases above are the same, we end up with the re-configuration penalty term to be the sum of all $p_j \cdot (1 - p_j) \cdot (s_j^+ + s_j^-)$ over $v_j \in \mathcal{I}$. The matrix notation of the re-configuration cost $\mathcal{S}(\mathcal{I})$ can be expressed as

$$\mathbf{x}^T \cdot \Psi \cdot (\mathbf{1} - \mathbf{x}),$$

$$\text{where } \Psi = \begin{pmatrix} s_1^+ + s_1^- & 0 & \dots & 0 \\ 0 & s_2^+ + s_2^- & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_n^+ + s_n^- \end{pmatrix} \tag{11}$$

in which $\mathbf{1}$ is the vector of all ones, and Ψ is a diagonal matrix with the re-configuration cost.

The re-configuration cost matrix Ψ can be included in the optimisation, as detailed in [55], as a penalty term $\mathbf{x}^T \cdot \Psi \cdot (\mathbf{1} - \mathbf{x})$ to the expected payoff function. To express the trade-off between the payoff matrix $\mathbf{A}(\mathbf{x})$ and the re-configuration cost matrix Ψ, we introduce a smoothing factor $0 < \beta < 1$. The so-enhanced optimisation problem can be expressed as:

$$\begin{aligned} &\min v \\ &\text{such that} \\ &\quad v \geq \beta \cdot A(\mathbf{x})\mathbf{e}_i + (1 - \beta) \cdot \mathbf{x}^T \cdot \Psi \cdot (\mathbf{1} - \mathbf{x}) \\ &\quad \forall i = 1, 2, \cdots, n, \\ &\quad \mathbf{1}^T \mathbf{x} = 1, \\ &\quad \mathbf{x} \geq 0. \end{aligned} \tag{12}$$

HoneyCar can also incorporate a more complex HCG where all m-out-of-n subsets are included as separate strategies to re-configure the honeypot. For example, let the set of vulnerabilities in the current round be $\mathcal{I}_i$ and the set of vulnerabilities for the next round be $\mathcal{I}_j$. Then the cost to switch from i to j, i.e., re-configuration cost, is $s_{ij} = \sum_{k \in \mathcal{I}_j \setminus \mathcal{I}_i} s_k^+ + \sum_{k \in \mathcal{I}_i \setminus \mathcal{I}_j} s_k^-$, with i, j ranging over all strategies, forming a matrix $\Psi = (s_{ij})$. According to [55], the penalty term to go into the optimisation is then the plain quadratic form $\mathbf{x}^T \cdot \Psi \cdot (1 - \mathbf{x})$.

1) EXAMPLE WITH HCB-b

We extend the example in Section IV-C1 by considering the re-configuration cost. Let the Defender be the minimising player with HIH and let $m = 3, n = |\mathcal{I}| = 3$, the associated exploitation time $\text{diag}(\Psi) = \{2, 1, 3\}$, and $T = 4$. Then, the payoff and the re-configuration cost matrices are given by

$$\Gamma = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad \Psi = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

Analysing the game in the described way and solving (12), yields $v(\mathcal{I})^* \approx 0.562$ at an equilibrium strategy $\mathbf{x}_0^* \approx (0.332, 0.304, 0.364)$. Naturally, the Defender's expected loss f_v here is higher than the conventional game without re-configuration cost (see IV-C1). To compute the expected game utility U_D using (2), we let $\lambda = 0.6$, $\alpha = 0.7$ and $\beta = 0.5$ leading to $g(\mathcal{I}) \approx 2.112$ and $c(\mathcal{I}) \approx 0.429$. The re-configuration cost for offering these vulnerabilities can be obtained using $(\mathbf{x}_0^*)^T \cdot \Psi \cdot (1 - \mathbf{x}_0^*) \approx 1.35$. Then, taking the re-configuration cost into account gives $U_D \approx 0.33$.

Remark 3: The re-configuration cost is important in determining the number of playable rounds of the HCG and eliminating the unattainable equilibrium (evident from the above examples).

E. SCALABILITY ANALYSIS

To evaluate the scalability of HoneyCar, we ran simulations on a 2.8GHz Intel Core i7 with 16GB RAM using Python 3.7.0 with the *scipy.optimize* package.[2] Our experiments showed that HoneyCar can solve well up to and at least $n = 1000$ vulnerabilities, but has the issue with the optimum assigning of positive probabilities $p_j > 0$ for all 1000 vulnerabilities with the time of exploits (t_j) being represented by a randomly sampled set of values from $\{1, 2, 3\}$.

In a real-world situation, configuring the honeypot with all vulnerabilities in $\mathcal{V}$ will be overly laborious with significant maintenance and monitoring costs. This was evident from the fact that even choosing vulnerabilities with $p_j < 0.001$, we ended up with 319 vulnerabilities in one of the experiments. Offering such a large set of vulnerabilities in a honeypot might be infeasible, let alone economic. Our goal was to keep a small subset of vulnerabilities open, to avoid exposing the honeypot and to unnecessary suspicion from the Attacker. Acknowledging the possibility of the honeypot being identified by the Attacker, we introduced the constraint (3) to enforce an m-out-of-n selection from $\mathcal{V}$. This reduces the problem into one with cardinality constraints, to which specialised approximation methods are applicable [58]. Our experiments with simple methods to replace (3), such as entropy constraints or smooth approximation for the indicator function, has failed when a large vulnerability set (around $n = 100$) is considered unless m is as large as n. However, if the honeypot is used exclusively to monitor unauthorised access, the constraint on the number of configured vulnerabilities could be exempted.

The expected game utility is achieved by the following steps:

1) Solve the optimisation problem as stated above (with or without the cardinality constraint depending on computational feasibility), and call the output value $\mathbf{x} = (p_1, \ldots, p_n)$.
2) Iterate over all $v_j \in \mathcal{I} \subseteq \mathcal{V}$, and with probability p_j, as determined from the optimisation problem, choose to include a vulnerability in the honeypot.
3) Evaluate $\mathbf{A}(\mathbf{x})$ and determine the Attacker's optimal (Defender's worst-case) strategy as the maximum over the elements of $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^n$.
4) Compute v and the re-configuration cost $\mathbf{x}^{*T} \cdot \Psi \cdot (1 - \mathbf{x}^*)$ at the optimum $\mathbf{x}^*$ to calculate U_D in (2).

[2]`https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html`

V. CASE STUDY

This section presents a case study where HoneyCar is assessed using known vulnerabilities related to autonomous and connected vehicles. For this case study, we consider the vulnerabilities from the Common Vulnerabilities and Exposure (CVE) list found within the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD).[3] The CVE data includes a description, Common Vulnerability Scoring System (CVSS) base scores, vulnerable product configuration, and weaknesses' categorisation information on each identified vulnerability. We primarily utilise the CVSS metrics to acquire parametric values required for HoneyCar. CVSS is a publicly available industry-standard that details the characteristics and severity of software vulnerabilities and is built upon three core metric groups: Base, Temporal, and Environment. The *Base metric* represents the intrinsic qualities of a vulnerability that remain unchanged over time and across user environments. The *Temporal metric* reflects the characteristics of a vulnerability that can change over time, while the *Environmental metric* reflects qualities of a vulnerability that are unique to a user's environment. This case study uses the Base metrics to extract the parameters to be used in HoneyCar.

First, we consider a small sample of vulnerabilities to assess HoneyCar. Nevertheless, HoneyCar can be implemented with any number of vulnerabilities to offer decision support to the Defender. Next, the NVD is checked for available patches for a vulnerability. Identifying vulnerabilities with available patches is critical to HoneyCar as without patches it is infeasible to re-configure the honeypot. Once a vulnerability with a patch has been identified, we obtain its CVSS metrics. Finally, HoneyCar is applied to obtain the optimal honeypot configuration for the versions of the Honeypot Configuration Game (HCG) presented in Section IV.

A. DATA AND USE CASE COMPOSITION

Taking advantage of the complexity metric of the CVSS, we derive the exploitation time t_j and the re-configuration cost $\mathcal{S}(v_j)$ for a vulnerability v_j. The complexity metric expresses the *anticipated efforts* needed to exploit a vulnerability. We associate a *low* complexity to "short" time, a *medium* complexity to "medium" time and a *high* complexity to "long" time. With such association, we set $t_j, s_j^+, s_j^- \in \{1, 2, 3\}$ for a vulnerability v_j within the ranks of the categorical values of the complexity metric. We further use the complexity metric to distinguish vulnerabilities that can be supported by LIH and HIH. Vulnerabilities requiring higher effort to exploit require higher access privileges which can only be supported through high-interaction honeypots. We, thus, consider that a low-interaction honeypot can only support vulnerabilities with *low* and *medium* complexities. Table 3 presents the list of security vulnerabilities used from the Common Vulnerabilities and Exposures (CVE) database and relevant CVSS metrics. The CVSS score ranges from 0 to

[3]`https://nvd.nist.gov/vuln/data-feeds`

TABLE 3. Sample set of vehicular security vulnerabilities with CVSS metrics.

#	v_1	v_2	v_3	v_4	v_5	v_6	v_7
CVE ID	CVE-2018-9311	CVE-2018-9318	CVE-2019-9977	CVE-2018-9313	CVE-2012-6510	CVE-2018-6508	CVE-2016-9337
Score	10	10	6.8	5.7	4.3	6	4
Access	Remote	Remote	Remote	Remote	Remote	Remote	Remote
Complexity	Low	Low	Medium	Medium	Medium	Medium	High

TABLE 4. Exploitation time and re-configuration cost for sample vulnerabilities.

v_j	v_1	v_2	v_3	v_4	v_5	v_6	v_7
t_j	1	1	2	2	2	2	3
$T - t_j$	3	3	2	2	2	2	1
$\mathcal{S}(v_j)$	1	1	2	2	2	2	3

10 and specifies the potential impact of a vulnerability. For example, a vulnerability providing access to the braking system of a vehicle will have a higher impact compared to the one that compromises the windscreen wipers.

We assume that the Defender can offer at most $m = |\mathcal{I}| = 6$ vulnerabilities through a honeypot. The Honeypot Configuration Game (HCG) is played for a finite number of rounds equivalent to the maximum number of vulnerabilities to be offered by a honeypot. Naturally, honeypot monitoring cost increases with the offered number of vulnerabilities. Deploying HoneyCar assists the Defender in determining the right type of honeypot to implement and the optimum number of vulnerabilities to offer to optimise the expected utility. To compute the expected game utility (U_D), we first set values for the time constant T, exploitation times t_j and re-configuration cost $\mathcal{S}(v_j)$ for all vulnerabilities $v_j \in \mathcal{V}$.

As detailed earlier, we heuristically consider the "complexity" metric to derive these values. We let $T := 4$ as any choice for $T > \max\{1, 2, 3\}$ is admissible. Table 4 presents the t_j and $\mathcal{S}(v_j)$ for the vulnerabilities listed in Table 3. We further consider that the re-configuration cost of a vulnerability equals its exploitation time. From the vulnerabilities sample in Table 3, vulnerability v_7 being of high complexity cannot be offered through LIH implying that $\mathcal{I} = \mathcal{V} \setminus \{v_7\}$ for LIH. For proportionality, we consider $\mathcal{I} = \mathcal{V} \setminus \{v_1\}$ for HIH. Using these vulnerability sets, we then construct the respective Γ and Ψ required to complete the optimisation as introduced in Section IV.

B. RESULTS

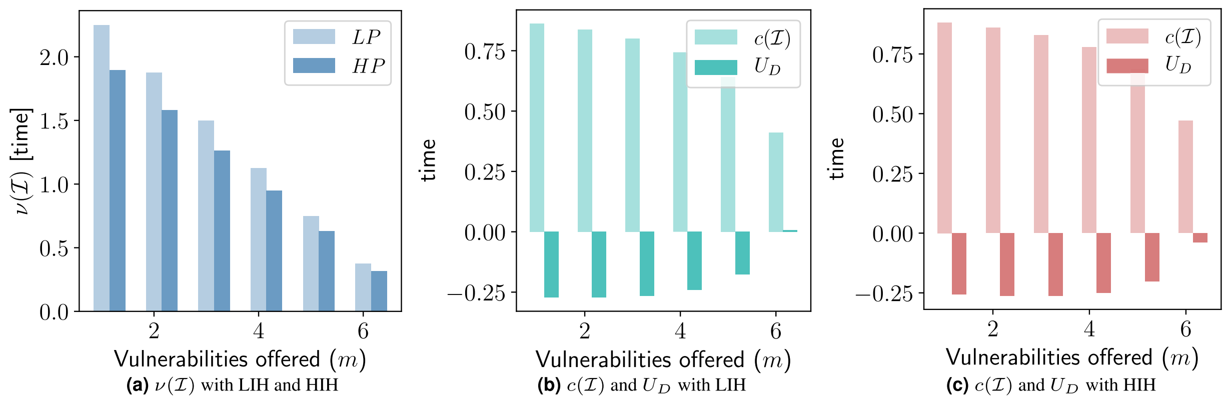
The ν value, obtained from the optimisation, is a key parameter in determining the honeypot configuration strategy. It expresses the maximum of the minimum residual time $(T - t_j)$ that the Defender can achieve regardless of the vulnerability chosen by the Attacker to exploit. The ν value is influenced by (i) the total number of available vulnerabilities; (ii) the number of vulnerabilities selected to offer; and (iii) the type of the offered vulnerabilities. The ν values decrease with the increase in the number of offered vulnerabilities as with an increased number of offered vulnerabilities the Defender has greater chances to deceive the Attacker.

Figure 3a presents the ν value over the range $0 < m \leq n$ with LIH and HIH without the re-configuration cost i.e., for HCG-a. The minimal value for ν is attained at $m = n = 6$. Further, Figure 3b shows the expected game utility U_D and the honeypot monitoring cost $c(\mathcal{I})$ for HCG-a with LIH over the range $0 < m \leq n$. As expected, the cost of monitoring the honeypot and the gather adversarial activities increases with ν. In particular, ν increases with an increase in the number of rounds of the game which enforces the cost of monitoring the honeypot throughout the game. On the contrary to the escalating monitoring cost, a higher ν is preferred as the cyber threat intelligence of the Defender grows with the number of rounds of the game. Thus, an optimal choice would be to select m such that U_D is positive and possibly the largest. The only positive U_D is at $m = 6$ suggesting that the expected game utility is the best when offering all six vulnerabilities at once with LIH. With this strategy, the minimum improvement in U_D is approximately 100% compared to any other selection of m. Similarly, it can be inferred from Figure 3c, which presents U_D and $c(\mathcal{I})$ with HIH, that implementing HIH will be expensive as U_D is always negative regardless of the size of m.

Result 1: In HoneyCar, when trying to maximise the duration of engagement with the attacker, an ideal choice for the Defender would be to offer all available vulnerabilities in one round of the game. This strategy particularly supports the objective of wasting attackers' time before throwing them out of the network and confirming the reasoning behind the use of honeypots to dissuade attackers from critical infrastructure and vehicles in IoV.

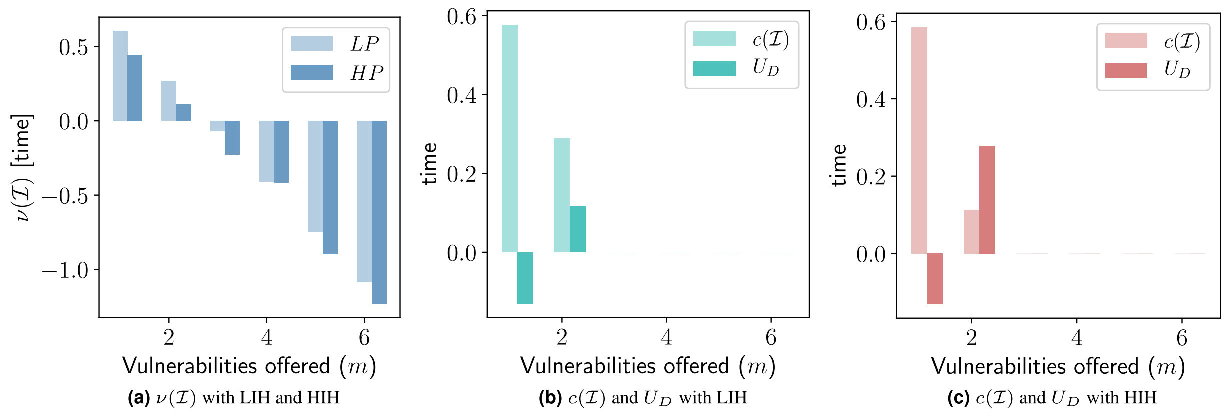
The results of the case study with HCG-b are presented in Figure 4. With regards to re-configuration cost, the Defender has to endure an additional cost for every selection of m vulnerabilities. Figure 4a shows that ν is negative when $m > 2$ implies that the Defender should not offer more than two vulnerabilities in a round. The reason for this stems from the fact that HCG is a zero-sum game and $\nu < 0$ implies a better payoff for the Attacker. Figures 4b and 4c illustrate the U_D and $c(\mathcal{I})$ with LIH and HIH, respectively. According to the results, the Defender gains a better U_D with HIH when $m = 2$. The Defender, while spending approximately 62% less on monitoring costs, can achieve an improved U_D of approximately 134% compared to the best strategy with LIH which also happens to be with $m = 2$.

Result 2: When trying to maximise the cyber threat intelligence, the honeypot configuration should be such that it



(a) $\nu(\mathcal{I})$ with LIH and HIH **(b)** $c(\mathcal{I})$ and U_D with LIH **(c)** $c(\mathcal{I})$ and U_D with HIH

FIGURE 3. Comparison of HCG-a game utility U_D and $c(\mathcal{I})$ for a total of six vulnerabilities with $\beta = 0.5$, λ being 0.4 and 0.6, α being 0.5 and 0.7 for LIH and HIH, respectively.



(a) $\nu(\mathcal{I})$ with LIH and HIH **(b)** $c(\mathcal{I})$ and U_D with LIH **(c)** $c(\mathcal{I})$ and U_D with HIH

FIGURE 4. Comparison of the HCG-b game utility U_D and $c(\mathcal{I})$ for a total of six vulnerabilities with $\beta = 0.5$, λ being 0.4 and 0.6, α being 0.5 and 0.7 for LIH and HIH, respectively.

engages the Attacker longer leading to a positive expected game utility. HoneyCar assists in determining the optimal number of vulnerabilities to be offered to gain the largest positive expected utility. The positive expected utility reflects that the collected amount of cyber threat intelligence is higher than the cost of monitoring the honeypot leading to a positive Return on Security Investment (ROSI).

Result 3: The re-configuration cost can be seen as a cost for switching from one Nash Equilibrium of the game to another. It refines the duration of the game by eliminating unplayable choices regarding the number of vulnerabilities to be offered in a round. HoneyCar, through such strategies, recommends optimal honeypot deception investment plans to engage attackers and achieve cyber threat intelligence.

C. DISCUSSION

In a real-world scenario, Defenders may use HoneyCar to plan a cost-effective and efficient cyber deception strategy. The decision-maker is expected to provide HoneyCar all the required information at the start such as the expected cost of implementing and maintaining a honeypot type, estimates of available budget, as well as the vulnerabilities to configure in the honeypot. HoneyCar uses this data, updates the relevant parameters, calculates the payoffs for the selection of a honeypot type and the configuration of vulnerabilities, determines the number of possible re-configuration steps and finally returns the best choice of a honeypot type and the optimal configuration within the available budget. At the moment, HoneyCar considers the CVSS value of a vulnerability for the cost and benefit parameter but this can be tailored to accept values specific to an organisation. Further, we assume that the data collected from implementing honeypot with the recommended configuration may lead to valuable insights on attackers' behaviour and tactics. Deployment duration and location of the honeypot in the network may also matter.

Furthermore, in the presented case study the model parameters are calculated using the CVSS scores and simulated costs for implementing a honeypot type. However, it is expected that for a real world application several different metrics can be aggregated for the calculation of Ψ and Γ. For instance, the value of the re-configuration cost matrix Ψ could be enhanced if combined with data about exploitation and patching vulnerabilities from available data breach reports and cyber threat intelligence. In many cases, information regarding the exploitation time of a vulnerability and required patching time either does not appear in any reports or is challenging to define in a time frame. Thus, HoneyCar currently uses only the metrics that are available to express the severity of a vulnerability to guide the suggested honeypot configuration.

In our case study, the values of the cost and benefit vectors were calculated using the CVSS Base Score for each selected vulnerability. However, in many cases, these values may vary based on the operations, availability of exploit techniques and patches, and the nature and importance of assets of an organisation. The Defender must be able to reassess the cost and benefit vectors seeing their requirements. This can be achieved by combining the CVSS Temporal Score and Environmental Score with the Base Score. Finally, regarding the honeypot monitoring cost and learning rate we assigned numerical values. However, these values may be learned over time from the deployed honeypots in IoV networks which would result in more generalised values that would make HoneyCar more robust and applicable in a variety of scenarios.

VI. CONCLUSION

The application of honeypots is a promising approach for protecting IoV networks. If an attacker is successfully lured by the honeypot, the adversarial activities captured by the honeypot can be used to learn about the attacker's motives and techniques. Successively, this knowledge, also referred to as cyber threat intelligence, can contribute to protecting existing system components by improving intrusion detection with new attack signatures or anomalous behaviour deviating from norms of protocols and systems' behaviours. Besides the inevitable cost of maintaining IoV honeypots, it is a challenge to design convincing honeypots to successfully deceive attackers.

To this end, this paper proposed a novel framework called HoneyCar which aims to assist IoV network administrators with the optimal configurations and investments in honeypots. HoneyCar is built upon two models: (i) a formal model of assessing the option of the Defender to invest in cyber deception using honeypots; and (ii) a game-theoretic model to strategically determine the configuration and selection of honeypot to be deployed in IoV network. Our framework empowers the network administrator to derive optimal decisions regarding honeypot deception based on an available budget. We take into consideration the number and type of vulnerabilities to be offered by the honeypot, the benefit and cost of implementing a vulnerability, the cost of re-configuring a honeypot and the available budget for investment in deception.

We demonstrate and evaluate HoneyCar using autonomous and connected vehicular security vulnerabilities collected from the Common Vulnerabilities and Exposure (CVE) data found within the National Vulnerability Database (NVD) and the respective Common Vulnerability Scoring System (CVSS) metrics. Our evaluation suggests that HoneyCar is capable of supporting IoV network administrators to determine the optimal configuration of honeypots for (i) wasting attacker's time and (ii) maximising the collection of cyber threat intelligence. HoneyCar also highlighted the importance of re-configuration cost in determining the duration of the game by eliminating unplayable choices leading to a realistic investment plan.

A key question future work could investigate is when to re-configure the honeypot. An option would be to consider finite horizon games where the vulnerability set becomes exhausted at some point leading to reopening a vulnerability that has been closed in the past. However, such an action could raise suspicion and the deception might fail. The Defender's option could be to invest more to introduce an entirely new set of vulnerabilities or refresh the honeypot. Formally, the game can be expressed in extensive form, with a number of stages that correspond to the number of re-configurations allowed given the vulnerability set. This will introduce more solution concepts like Subgame Perfect Nash equilibria or even Stackelberg Nash equilibria. Last, future work can include approaches that combine both game theory and machine learning to develop defensive deception techniques. In such a hybrid approach, reinforcement learning with game theory could be used to formulate players' utility functions, estimate the opponent's beliefs and update the optimal strategy and predict the opponent's actions by analysing data from host vehicles, network and threat actor behaviours.

REFERENCES

[1] O. Kaiwartya, A. H. Abdullah, Y. Cao, A. Altameem, M. Prasad, C.-T. Lin, and X. Liu, "Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[2] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 241–246.

[3] G. Loukas, *Cyber-Physical Attacks: A Growing Invisible Threat*. London, U.K.: Butterworth, 2015.

[4] M. Zhu, A. H. Anwar, Z. Wan, J.-H. Cho, C. A. Kamhoua, and M. P. Singh, "A survey of defensive deception: Approaches using game theory and machine learning," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2460–2493, Aug. 2021.

[5] P. Patel and R. Jhaveri, "A honeypot scheme to detect selfish vehicles in vehicular ad-hoc network," in *Computing and Network Sustainability*. Cham, Switzerland: Springer, 2017, pp. 389–401.

[6] V. Verendel, D. K. Nilsson, U. E. Larson, and E. Jonsson, "An approach to using honeypots in in-vehicle networks," in *Proc. IEEE 68th Veh. Technol. Conf.*, Sep. 2008, pp. 1–5.

[7] M. Ahmed, S. Panda, C. Xenakis, and E. Panaousis, "MITRE ATT&CK-driven cyber risk assessment," in *Proc. 17th Int. Conf. Availability, Rel. Secur.*, Aug. 2022, pp. 1–10.

[8] E. Scott, S. Panda, G. Loukas, and E. Panaousis, "Optimising user security recommendations for AI-powered smart-homes," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, Sep. 2022, pp. 1–8.

[9] S. Panda, E. Panaousis, G. Loukas, and C. Laoudias, "Optimizing investments in cyber hygiene for protecting healthcare users," in *From Lambda Calculus to Cybersecurity Through Program Analysis*. Cham, Switzerland: Springer, 2020.

[10] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Decision support approaches for cyber security investment," *Decision Support Syst.*, vol. 86, pp. 13–23, Jun. 2016.

[11] I. Kalderemidis, A. Farao, P. Bountakas, S. Panda, and C. Xenakis, "GTM: Game theoretic methodology for optimal cybersecurity defending strategies and investments," in *Proc. 17th Int. Conf. Availability, Rel. Secur.*, Aug. 2022, pp. 1–9.

[12] A. Nisioti, G. Loukas, A. Laszka, and E. Panaousis, "Data-driven decision support for optimizing cyber forensic investigations," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2397–2412, 2021.

[13] S. Panda, D. W. Woods, A. Laszka, A. Fielder, and E. Panaousis, "Post-incident audits on cyber insurance discounts," *Comput. Secur.*, vol. 87, Nov. 2019, Art. no. 101593.

[14] A. Farao, S. Panda, S. A. Menesidou, E. Veliou, N. Episkopos, G. Kalatzantonakis, F. Mohammadi, N. Georgopoulos, M. Sirivianos, N. Salamanos, and S. Loizou, "SECONDO: A platform for cybersecurity investments and cyber insurance decisions," in *Proc. Int. Conf. Trust Privacy Digit. Bus.* Cham, Switzerland: Springer, 2020, pp. 65–74.

[15] S. Panda, A. Farao, E. Panaousis, and C. Xenakis, *Cyber-Insurance: Past, Present and Future*. Berlin, Germany: Springer, 2019, pp. 1–4, doi: 10.1007/978-3-642-27739-9_1624-1.

[16] N. C. Rowe, "Measuring the effectiveness of honeypot counter-counterdeception," in *Proc. 39th Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 6, Jan. 2006, p. 129c.

[17] S. Sharma and A. Kaul, "A survey on intrusion detection systems and honeypot based proactive security mechanisms in VANETs and VANET cloud," *Veh. Commun.*, vol. 12, pp. 138–164, Apr. 2018.

[18] M. R. Babu and G. Usha, "A novel honeypot based detection and isolation approach (NHBADI) to detect and isolate black hole attacks in MANET," *Wireless Pers. Commun.*, vol. 90, no. 2, pp. 831–845, Sep. 2016.

[19] J. Daubert, D. Boopalan, M. Mühlhäuser, and E. Vasilomanolakis, "HoneyDrone: A medium-interaction unmanned aerial vehicle honeypot," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2018, pp. 1–6.

[20] J. You, S. Lv, L. Zhao, M. Niu, Z. Shi, and L. Sun, "A scalable high-interaction physical honeypot framework for programmable logic controller," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Nov. 2020, pp. 1–5.

[21] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "VANet security challenges and solutions: A survey," *Veh. Commun.*, vol. 7, pp. 7–20, Jan. 2017.

[22] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, and P. Doody, "Internet of Things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1. Gistrup, Denmark: River Publishers, 2011, pp. 9–52.

[23] F. Qu, Z. Wu, F.-Y. Wang, and W. Cho, "A security and privacy review of VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2985–2996, Dec. 2015.

[24] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 2898–2915, Nov. 2017.

[25] T. Zhang, H. Antunes, and S. Aggarwal, "Defending connected vehicles against malware: Challenges and a solution framework," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 10–21, Feb. 2014.

[26] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, Aug. 2015.

[27] K.-Y. Lam, S. Mitra, F. Gondesen, and X. Yi, "ANT-centric IoT security reference architecture—Security-by-design for satellite-enabled smart cities," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5895–5908, Apr. 2022.

[28] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101842.

[29] D. A. Rivas, J. M. Barceló-Ordinas, M. G. Zapata, and J. D. Morillo-Pozo, "Security on VANETs: Privacy, misbehaving nodes, false information and secure data aggregation," *J. Netw. Comput. Appl.*, vol. 34, no. 6, pp. 1942–1955, Nov. 2011.

[30] K. Emara, W. Woerndl, and J. Schlichter, "On evaluation of location privacy preserving schemes for VANET safety applications," *Comput. Commun.*, vol. 63, pp. 11–23, Jun. 2015.

[31] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in internet of vehicles: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2339–2352, Oct. 2015.

[32] M. Wazid, P. Bagga, A. K. Das, S. Shetty, J. J. P. C. Rodrigues, and Y. H. Park, "AKM-IoV: Authenticated key management protocol in fog computing-based internet of vehicles deployment," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8804–8817, Oct. 2019.

[33] J. Li, Z. Xue, C. Li, and M. Liu, "RTED-SD: A real-time edge detection scheme for sybil DDoS in the internet of vehicles," *IEEE Access*, vol. 9, pp. 11296–11305, 2021.

[34] P. K. Singh, S. Gupta, R. Vashistha, S. K. Nandi, and S. Nandi, "Machine learning based approach to detect position falsification attack in VANETs," in *Proc. Int. Conf. Secur. Privacy*. Cham, Switzerland: Springer, 2019, pp. 166–178.

[35] N.-W. Lo and H.-C. Tsai, "Illusion attack on VANET applications—A message plausibility problem," in *Proc. IEEE Globecom Workshops*, Nov. 2007, pp. 1–8.

[36] J. Wang, Y. Tan, J. Liu, and Y. Zhang, "Topology poisoning attack in SDN-enabled vehicular edge network," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9563–9574, Oct. 2020.

[37] T. E. Carroll and D. Grosu, "A game theoretic investigation of deception in network security," *Secur. Commun. Netw.*, vol. 4, no. 10, pp. 1162–1172, 2011.

[38] N. Boumkheld, S. Panda, S. Rass, and E. Panaousis, "Honeypot type selection games for smart grid networks," in *Proc. Int. Conf. Decis. Game Theory Secur.* Cham, Switzerland: Springer, 2019, pp. 85–96.

[39] Q. Zhu and T. Başar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *Proc. Int. Conf. Decis. Game Theory Secur.* Cham, Switzerland: Springer, 2013, pp. 246–263.

[40] W. Tian, M. Du, X. Ji, G. Liu, Y. Dai, and Z. Han, "Honeypot detection strategy against advanced persistent threats in industrial Internet of Things: A prospect theoretic game," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17372–17381, Dec. 2021.

[41] N. Garg and D. Grosu, "Deception in honeynets: A game-theoretic analysis," in *Proc. IEEE SMC Inf. Assurance Secur. Workshop*, Jun. 2007, pp. 107–113.

[42] H. Çeker, J. Zhuang, S. Upadhyaya, Q. D. La, and B.-H. Soong, "Deception-based game theoretical approach to mitigate dos attacks," in *Proc. Int. Conf. Decis. Game Theory Secur.* Cham, Switzerland: Springer, 2016, pp. 18–38.

[43] R. Píbil, V. Lisý, C. Kiekintveld, B. Bošanský, and M. Pěchouček, "Game theoretic model of strategic honeypot selection in computer networks," in *Proc. Int. Conf. Decis. Game Theory Secur.* Cham, Switzerland: Springer, 2012, pp. 201–220.

[44] C. Kiekintveld, V. Lisỳ, and R. Píbil, "Game-theoretic foundations for the strategic use of honeypots in network security," in *Cyber Warfare*. Cham, Switzerland: Springer, 2015, pp. 81–101.

[45] Q. D. La, T. Q. S. Quek, J. Lee, S. Jin, and H. Zhu, "Deceptive attack and defense game in honeypot-enabled networks for the Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1025–1035, Dec. 2016.

[46] J. Zhuang, V. M. Bier, and O. Alagoz, "Modeling secrecy and deception in a multiple-period attacker–defender signaling game," *Eur. J. Oper. Res.*, vol. 203, no. 2, pp. 409–418, 2010.

[47] K. Durkota, V. Lisỳ, B. Bošanskỳ, and C. Kiekintveld, "Optimal network security hardening using attack graph games," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 526–532.

[48] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3906–3919, Dec. 2017.

[49] J. Avery and E. H. Spafford, "Ghost patches: Fake patches for fake vulnerabilities," in *Proc. Int. Conf. Syst. Secur. Privacy Protection*. Cham, Switzerland: Springer, 2017, pp. 399–412.

[50] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser, "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 942–953.

[51] D. Fudenberg and J. Tirole, *Game Theory*. London, U.K.: MIT Press, 1991.

[52] C. T. Do, N. H. Tran, C. Hong, C. A. Kamhoua, K. A. Kwiat, E. Blasch, S. Ren, N. Pissinou, and S. S. Iyengar, "Game theory for cyber security and privacy," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–37, 2017.

[53] N. Pitropakis, E. Panaousis, A. Giannakoulias, G. Kalpakis, R. D. Rodriguez, and P. Sarigiannidis, "An enhanced cyber attack attribution framework," in *Proc. Int. Conf. Trust Privacy Digit. Bus.* Cham, Switzerland: Springer, 2018, pp. 213–228.

[54] S. Panda, I. Oliver, and S. Holtmanns, "Behavioural modelling of attackers choices," in *Proc. Asian Control Conf.*, 2018, pp. 119–126.

[55] S. Rass, S. König, and S. Schauer, "On the cost of game playing: How to control the expenses in mixed strategies," in *Proc. Int. Conf. Decis. Game Theory Secur.* Cham, Switzerland: Springer, 2017, pp. 494–505.

[56] K. Hausken, "Returns to information security investment: The effect of alternative information security breach functions on optimal investment and sensitivity to vulnerability," *Inf. Syst. Frontiers*, vol. 8, no. 5, pp. 338–349, Jan. 2007.

[57] M. T. Qassrawi and Z. Hongli, "Deception methodology in virtual honeypots," in *Proc. 2nd Int. Conf. Netw. Secur., Wireless Commun. Trusted Comput.*, 2010, pp. 462–467.
[58] R. Ruiz-Torrubiano, S. García-Moratilla, and A. Suárez, "Optimization problems with cardinality constraints," in *Computational Intelligence in Optimization*. Cham, Switzerland: Springer, 2010, pp. 105–130.

SAKSHYAM PANDA (Member, IEEE) received the Ph.D. degree in computer science from the University of Surrey, U.K., and the dual M.Sc. degrees in human-computer interaction and design from the KTH Royal Institute of Technology, Sweden, and Aalto University, Finland. He is currently a Research Associate in cybersecurity and privacy with the Cyber Risk Laboratory, School of Computing and Mathematical Sciences, University of Greenwich, U.K. His research interest includes application of game theory to support decision-making in cybersecurity.

STEFAN RASS (Member, IEEE) received the double master's degrees in mathematics and computer science and the Ph.D. degree in mathematics. He is currently a Professor of secure systems with the LIT Secure and Correct Systems Laboratory, Johannes Kepler University Linz, Austria, where he is also teaching courses on algorithms and data structures, theoretical computer science, complexity theory, security, and cryptography. He has authored numerous papers related to security and applied statistics and decision theory in security, as well as books on cryptography and game theory (Artech House, Springer). His research interests include decision theory and game-theory with applications in system security, as well as complexity theory, statistics, and information-theoretic security.

SOTIRIS MOSCHOYIANNIS (Member, IEEE) is currently a Reader in complex systems with the University of Surrey and a member of the Surrey Cyber Security Centre, which is the Academic Centre of Excellence in Cyber Security Research (ACE-CSR). He has published over 60 peer-reviewed publications, including four book chapters. His research interests include computational techniques and mathematical methods for the analysis of complex networks, and specifically decision control, network optimization, rule-based, and deep reinforcement learning. He has received six best paper awards at international journals and peer-reviewed conferences. He is on the IEEE Technical Committee on Industrial Informatics and a member of the Executive Committee of the IEEE Technical Committee on Cloud Computing (TCCLD).

K. LIANG (Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong. He is currently an Assistant Professor in secure systems with the University of Surrey, U.K. He has published a series of research works, applying secure tools to tackle real-world problems in many high tier international journals, such as the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and the IEEE NETWORK. His current research interests include data security, user privacy, cybersecurity, blockchain security, and privacy-enhancing technology. He has served as a TPC for International Security/Privacy Conferences, e.g., ESORICS. He is an ISO Member of UK ISO Crypto Sub Committee IST/33/2 and an Associate Editor of *The Computer Journal* and other security journals.

GEORGE LOUKAS (Member, IEEE) received the Ph.D. degree in network security from the Imperial College, in 2006. He is currently a Professor and the Head of the IoT and Security Centre, University of Greenwich. He has been a Project Coordinator or a PI in six national and international research projects related to cyber-physical system security and human-asa-sensor technologies.

EMMANOUIL PANAOUSIS (Senior Member, IEEE) is currently a Professor with the University of Greenwich and the Founder of the Cyber Risk Laboratory. He previously held appointments at the University of Surrey, the University of Brighton, the Imperial College London, and the Queen Mary University of London. His research interests include cybersecurity, the IoT security, and applied maths. His research has been funded by the European Commission, the Engineering and Physical Sciences Research Council (EPSRC), the National Cyber Security Centre (NCSC), the Research Institute in Sociotechnical Cyber Security (RISCS), and the Greenwich Research and Enterprise (GRE).