Foureye: Defensive Deception based on Hypergame Theory Against Advanced Persistent Threats

Zelin Wan, Jin-Hee Cho, *Senior Member, IEEE*, Mu Zhu, Ahmed H. Anwar, Charles Kamhoua, *Senior Member, IEEE*, and Munindar P. Singh, *IEEE Fellow*

Abstract—Defensive deception techniques have emerged as a promising proactive defense mechanism to mislead an attacker and thereby achieve attack failure. However, most game-theoretic defensive deception approaches have assumed that players maintain consistent views under uncertainty. They do not consider players' possible, subjective beliefs formed due to asymmetric information given to them. In this work, we formulate a hypergame between an attacker and a defender where they can interpret the same game differently and accordingly choose their best strategy based on their respective beliefs. This gives a chance for defensive deception strategies to manipulate an attacker's belief, which is the key to the attacker's decision making. We consider advanced persistent threat (APT) attacks, which perform multiple attacks in the stages of the cyber kill chain where both the attacker and the defender aim to select optimal strategies based on their beliefs. Through extensive simulation experiments, we demonstrated how effectively the defender can leverage defensive deception techniques while dealing with multi-staged APT attacks in a hypergame in which the imperfect information is reflected based on perceived uncertainty, cost, and expected utilities of both attacker and defender, the system lifetime (i.e., mean time to security failure), and improved false positive rates in detecting attackers.

Index Terms—Defensive deception, hypergame theory, uncertainty, attacker, defender, advanced persistent threat

1 Introduction

The key purpose of a defensive deception technique is to mislead an attacker's view and make it choose a suboptimal or poor action for the attack failure [30]. When both the attacker and defender are constrained in their resources, strategic interactions can be the key to beat an opponent. In this sense, non-game-theoretic defense approaches have inherent limitations due to lack of efficient and effective strategic tactics. Forms of deception techniques have been discussed based on certain classifications, such as *hiding the truth* vs. *providing false information* or *passive* vs. *active* for increasing attackers' ambiguity or confusion [3, 9].

Game theory has been substantially used for dynamic decision making under uncertainty, assuming that players have consistent views. However, this assumption fails as players may often subjectively process asymmetric information available to them [19]. Hypergame theory [5] is a variant of game theory that provides a form of analysis considering each player's subjective belief, misbelief, and perceived uncertainty and accordingly their effect on decision making in choosing a best strategy [19].

This paper leverages hypergame theory to resolve conflicts of views of multiple players as a robust decision-making mechanism under uncertainty where the players may have different beliefs towards the same game. Hypergame theory models players, such as attackers and defenders in cybersecurity to deal with advanced persistent threat (APT) attacks. We dub this effort *Foureye* after the *Foureye butterflyfish*, demonstrating deceptive defense in nature [35].

To be specific, we identify the following nontrivial challenges in obtaining a solution. First of all, it is not trivial to derive realistic game scenarios and develop defensive deception techniques to deal with APT attacks beyond the reconnaissance stage. This aspect has not been explored in the state-of-the-art. Second, quantifying the degree of uncertainty in the views of attackers and defenders is challenging, although they are critical because how each player frames a game significantly affects its strategies to take. Third, given a number of possible choices under dynamic situations, dealing with a large number of solution spaces is not trivial whereas the deployment and maintenance of defensive deception techniques is costly in contested environments. We partly addressed these challenges in our prior work in [12]; however, its contribution is very limited in considering a small-scale network and a small set of strategies with a highly simplified probability model developed using Stochastic Petri Network.

To be specific, this paper has the following **new key contributions**:

- We modeled an attack-defense game under uncertainty based on hypergame theory where an attacker and a

- *Zelin Wan and Jin-Hee Cho are with the Department of Computer Science, Virginia Tech, Falls Church, VA 22043, USA. Email: {zelin, jicho}@vt.edu. Mu Zhu and Munindar P. Singh are with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA. Email: {mzhu5, mpsingh}@ncsu.edu. Ahmed H. Anwar and Charles A. Kamhoua are with the US Army Research Laboratory, Adelphi, MD 20783, USA. Email: a.h.anwar@knights.ucf.edu; charles.a.kamhoua.civ@mail.mil.*

defender have different views of the situation and are uncertain about strategies taken by their opponents.
- We reduced a player's action space by using a subgame determined based on a set of strategies available where each subgame is formulated based on each stage of the cyber kill chain (CKC) based on a player's belief under uncertainty.
- We considered multiple defense strategies, including defensive deception techniques whose performance can be significantly affected by an attacker's belief and perceived uncertainty, which impacts its choice of a strategy.
- We modeled an attacker's and a defender's uncertainty towards its opponent (i.e., the defender and the attacker, respectively) based on how long each player has monitored the opponent and its chosen strategy. To the best of our knowledge, prior research on hypergame theory uses a predefined constant probability to represent a player's uncertainty. In this work, we estimated the player's uncertainty based on the dynamic, strategic interactions between an attacker and a defender.
- We conducted comparative performance analysis with or without a defender using defensive deception (DD) strategies and with or without perfect knowledge available towards actions taken by the opponent. We measured the effectiveness and efficiency of DD techniques in terms of a system's security and performance, such as perceived uncertainty, hypergame expected utility, action cost, mean time to security failure (MTTSF or system lifetime), and improved false positive rate (FPR) of an intrusion detection by the DD strategies taken by the defender.

2 Related Work

Garg and Grosu [14] proposed a game-theoretic deception framework in honeynets with imperfect information to find optimal actions of an attacker and a defender and investigated the mixed strategy equilibrium. Carroll and Grosu [10] used deception in attacker-defender interactions in a signaling game based on perfect Bayesian equilibria and hybrid equilibria. They considered defensive deception techniques, such as honeypots, camouflaged systems, or normal systems. Yin et al. [36] considered a Stackelberg attack-defense game where both players make decisions based on their perceived observations and identified an optimal level of deceptive protection using fake resources. Casey et al. [11] examined how to discover Sybil attacks based on an evolutionary signaling game where a defender can use a fake identity to lure the attacker to facilitate cooperation. Schlenker et al. [29] studied a sophisticated and naïve APT attacker in the reconnaissance stage to identify an optimal defensive deception strategy in a zero-sum Stackelberg game by solving a mixed integer linear program.

Unlike the above works cited [10, 11, 14, 29, 36], our work used hypergame theory which offers the powerful capability to model uncertainty, different views, and bounded rationality by different players. This way reflects more realistic scenarios between the attacker and defender.

Hypergame theory has emerged to better reflect real-world scenarios by capturing players' subjective and imperfect belief, aiming to mislead them to adopt uncertain or non-optimized strategies. Although other game theories deal with uncertainty by considering probabilities that a certain event may happen, they assume that all players play the same game [31]. Hypergame theory has been used to solve decision-making problems in military and adversarial environments House and Cybenko [17], Vane [33], Vane and Lehner [34]. Several studies [15, 16] investigated how players' beliefs evolve based on hypergame theory by developing a misbelief function measuring the differences between a player's belief and the ground truth payoff of other players' strategies. Kanazawa et al. [18] studied an individual's belief in an evolutionary hypergame and how this belief can be modelled by interpreter functions. Sasaki [28] discussed the concept of *subjective rationalizability* where an agent believes that its action is a best response to the other agent's choices based on its perceived game.

Putro et al. [27] proposed an adaptive, genetic learning algorithm to derive optimal strategies by players in a hypergame. Ferguson-Walter et al. [13] studied the placement of decoys based on a hypergame. This work developed a game tree and investigated an optimal move for both an attacker and defender in an adaptive game. Aljefri et al. [2] studied a first level hypergame involving misbeliefs to resolve conflicts for two and then more decision makers. Bakker et al. [4] modeled a repeated hypergame in dynamic stochastic setting against APT attacks primarily in cyber-physical systems.

Unlike the works using hypergame theory above [2, 4, 13, 15, 16, 17, 18, 27, 28, 33, 34], our work considered an APT attacker performing multi-staged attacks where attack-defense interactions are modeled based on repeated hypergames. In addition, we show the effectiveness of defensive deception techniques by increasing the attacker's uncertainty leading to choosing non-optimal actions and increasing the quality of the intrusion detection (i.e., a network-based intrusion detection system, NIDS) through the collection of attack intelligence using defensive deception strategies.

3 System Model

3.1 Network Model

This work concerns a software-defined network (SDN)-based Internet-of-Things (IoT) environment characterized by servers and/or IoT devices, such as an SDN-based smart environment [7]. The key benefit of using the SDN technology is decoupling the network control plane from the data plane (e.g., packet forwarding) for higher flexibility, robust security/performance, and programmability for a networked system in which an SDN controller can efficiently and effectively manage security and performance mechanisms. We use the SDN controller to involve packet forwarding decisions and to deploy defense mechanisms, such as firewalls or NIDSs. SDN-enabled switches handle forwarding packets, where they encapsulate packets without exact matching flow rules in flow tables in which the encapsulated packets, 'OFPT PACKET IN' packets in OpenFlow (OF) protocol (i.e., a standard communication protocol between SDN-enabled switches and the SDN controller), are provided to the SDN controller handling the flow.

The nodes in this environment collect data and perform a periodic delivery of those collected data to the servers via multi-hop communications, in which the servers may need to process further to provide queried services. The nodes

may be highly heterogeneous in their types and functionalities and spread over different Virtual Local Area Networks (VLANs) of the IoT environment. Each VLAN may have one or more servers and is assigned with a set of nodes based on the common characteristics of their functionalities. We leverage the advanced SDN technology [24] for the effective and efficient management of IoT nodes with the help of an SDN controller.

3.2 Node Model

A node, including web servers, databases, honeypots, and IoT devices, is characterized by the following set of features:

- *Criticality:* This metric, c_i, indicates how critical node i is in terms of its given role for security and reachability (i.e., influence) in a network to maintain network connectivity, and given by:

$$\mathrm{c}_i = \mathrm{importance}_i \times \mathrm{reachability}_i, \tag{1}$$

  where $\mathrm{importance}_i$ is given as an integer ranged in $[0, 10]$ during the network deployment phrase. $\mathrm{reachability}_i$ is computed based on the faster betweenness centrality metric [8] by the SDN controller. Note that the algorithmic complexity of the faster betweenness in this work is $O(|V|^2)$ as a given network follows Erdös–Rényi (ER) network model [25]. $\mathrm{reachability}_i$ is estimated in the range of $[0, 1]$ as a real number.
- *Security vulnerability*: A node's vulnerabilities to various types of attacks are considered based on three types of vulnerabilities: (1) vulnerabilities associated with software installed in each node, denoted by sv_i; (2) vulnerabilities associated with encryption keys (e.g., secret or private keys), denoted by ev_i. As a longer-term key exposes higher security vulnerability, the attacker can exploit encryption vulnerability over time with $\hat{\mathrm{ev}}_i = \mathrm{ev}_i \cdot e^{-1/\mathrm{T}_{\mathrm{rekey}}}$ and $\mathrm{T}_{\mathrm{rekey}}$ is the time elapsed since the attacker has investigated a given key; and (3) an unknown vulnerability, denoted by uv_i, representing the average unknown vulnerability. We assume that all the vulnerabilities are computed based on the Common Vulnerability Scoring System (CVSS) [1] with the severity value in $[0, 10]$ as an integer. We measure the average vulnerability associated with node i being vulnerable by:

$$\mathrm{vulnerability}_i = \frac{\sum_{v_j \in V_i} v_j}{|V_i|}, \tag{2}$$

  where V_i is a set of vulnerabilities associated with node i (e.g., $\{\mathrm{sv}_0, \mathrm{sv}_1, \mathrm{sv}_2, \mathrm{ev}_0, \mathrm{ev}_1, \mathrm{ev}_2, \mathrm{uv}_0\}$), v_j refers to one of vulnerabilities, associated with node i where v_j is measured based on $[0, 10]$ following the CVSS. We denote $P_i^v = \mathrm{vulnerability}_i/10$ as a normalized vulnerability probability. P_i^v is used as the probability to exploit (i.e., compromise) node i by an attacker.
- *Mobility*: We model the mobility rate of node i by considering a rewiring probability P_i^r only for IoT devices where node i can be connected with a new IoT node with P_i^r. For rewiring connections, node i will select one of its neighbors with P_i^r to disconnect and then select a new node to be connected to maintain a same number of neighbors (nodes being directly connected).

TABLE 1
EXAMPLE NODE CHARACTERISTICS.

	Importance	Software Vul.	Encryption Vul.
Web servers	$[8, 10]$	$[3, 7]$	$[1, 3]$
Datbases	$[8, 10]$	$[3, 7]$	$[1, 3]$
Honeypots	0	$[7, 10]$	$[9, 10]$
IoT devices	$[1, 5]$	$[1, 5]$	$[5, 10]$

Table 1 shows an example set of node characteristics showing the ranges of each node type's attributes and the shown values used as default settings for our experiments in Section 6. We select each attribute value at random based on uniform distribution in a given range. Notice that we consider zero importance for honeypots, implying no performance degradation and security damage upon its compromise. In addition, we put a fairly high range of the number of vulnerabilities in the honeypots in order to lure attackers with high attack utility. Since a legitimate user can be compromised by the attacker, cp refers to the status of a node's compromise (i.e., $\mathrm{cp}_i = 1$ for compromise; 0 otherwise). We summarize node i's profile as:

$$n_i = [\mathrm{c}_i, \mathrm{cp}_i, \mathrm{ev}_i, \mathbf{V}_i, P_i^v, P_i^r]. \tag{3}$$

Recall that c_i, cp_i, ev_i, $\mathbf{V}_i$, P_i^v, and P_i^r are node i's criticality in $[0, 1]$, the status of being compromised (=1) or not(=0), evicted (=1) or not (=0), vulnerability vector in software, encryption, and unknowns, the probability of the overall vulnerability, and rewiring probability for mobility in $[0, 1]$.

3.3 Assumptions

We assume that the SDN controller and control channel are trusted and considering their security vulnerabilities is beyond the scope of this work. Since each SDN controller should be well informed of basic network information under its control and other SDN controllers' control, each SDN controller periodically updates the network topology and software vulnerabilities of nodes under its control to other SDN controllers. Via this process, each SDN controller can periodically check an overall system security state and take actions accordingly.

We also assume that a network-based IDS (NIDS) is deployed in the SDN controller and is characterized by the probabilities of false positives (P_{fp}) and false negatives (P_{fn}). The NIDS runs throughout the system lifetime. The NIDS's P_{fp} and P_{fn} will be dynamically updated as it receives more attack intelligence from the defensive deception techniques used in this work. We assume that the collected signatures from the deception-based monitoring mechanisms can decrease P_{fn} due to an increased volume of additional signatures. We simply use Beta distribution to derive Beta$(P_{fn}; \alpha, \beta)$ where α refers to false negatives (FN) and β is true positives (TP) with $P_{fn} = FN/(TP+FN)$. Similarly, as more attack intelligence is forwarded to NIDS via defensive deception-based monitoring, β (TP) increments by 1 per monitoring interval. Similarly, false positives will be reduced as defensive deception techniques are used where $P_{fp} = FP/(TN + FP)$ and TN increases by 1.

We assume that legitimate users use a secret key for secure group communications among internal, legitimate users while prohibiting outsiders from accessing secured network resources. If an outsider wants to access a target

network and become an inside attacker with legitimate credentials, it needs to be authenticated and given the secret key to access the target network. In addition, network resources are accessed according to the privilege of each user. Therefore, to compromise a legitimate node, the attacker should obtain appropriate privileges to access them.

3.4 Attack Model

We consider APT attackers performing multi-staged attacks following the cyber kill chain (CKC) for compromising a target node and exfiltrating confidential information to outside [26]. We consider the APT attacks as follows.

APT Attack Procedure to Achieve Data Exfiltration: We define an APT attacker's goal in that the attacker has reached and compromised a target node and successfully exfiltrated its confidential data. We assume that nodes with a higher importance (i.e., having more important, credential information) are more likely to be targeted.

To reach a target node, the attacker needs to compromise other intermediate nodes along the way. We often call the path to the target node 'an attack path.' In reality, the attacker may not have an exact, complete view on network topology. We assume that the attacker only knows its adjacent nodes (i.e., nodes being directly connected) and needs to choose which node to compromise next. The attacker will consider how easily given adjacent node i can be exploited according to an attack cost metric, ac_k, for attack strategy k. Moreover, if the attacker finds already compromised, adjacent nodes, it can leverage it and has no need to put additional effort to compromise it. We call this 'the value of an intermediate node i in an attack path,' denoted by $APV(i,k)$, where k refers to attack strategy ID. Highest $APV(i,k)$ will be added to the attacker's attack path to the target node. $APV(i,k)$ is given by:

$$APV(i,k) = \begin{cases} (1 - \hat{\mathrm{ac}}_k) \cdot P_i^v & \text{if } \mathrm{cp}_i == 0, \\ 1 & \text{otherwise.} \end{cases} \quad (4)$$

Here $\hat{\mathrm{ac}}_k = e^{-(1/\mathrm{ac}_k)} \in [0,1]$ that represents a normalized attack cost where ac_k is a predefined attack cost ranged in $[0,3]$ (see 'Attack Strategy Attributes' later in this section), and $\mathrm{vulnerability}_i$ is the overall vulnerability in Eq. (2). Given a node to be compromised next, its vulnerability degree can be computed as P_i^v $(= \mathrm{vulnerability}_i/10)$. If $\mathrm{cp}_i = 1$ (i.e., node i is compromised), the attacker may add it to the attack path at no cost, which gives $APV(i,k) = 1$. The attacker may need to compromise more than one intermediate nodes before reaching a target node.

Attack Strategy Attributes: An APT attacker can perform multiple attacks through the stages of the CKC. Each attack strategy k can be characterized by: (1) attack cost, ac_k, indicating how much time/effort is needed to launch the attack; and (2) the expected impact (i.e., attack effectiveness) upon attack success, ai_k. ac_k is a predefined constant as an integer in $[0,3]$ reflecting no, low, medium, and high cost, respectively. ai_k is obtained by victim j's criticality, c_j (see Eq. (1)). This implies the attack benefit through compromising a set of exploitable nodes. If there have been multiple nodes being compromised by taking given attack k, ai_k captures the criticalities of the compromised nodes by:

$$\mathrm{ai}_k = \frac{\sum_{j \in C_k} \mathrm{c}_j}{N}, \quad (5)$$

TABLE 2
CHARACTERISTICS OF APT ATTACK STRATEGIES

AS	CKC stage	Attack cost (ac)	Node compromise	Exploited vulnerability
AS_1	R – DE	1	No	UV
AS_2	D – DE	3	Yes (SN)	SV + EV
AS_3	E – DE	3	Yes (MN)	SV
AS_4	E – DE	3	Yes (SN)	SV + UV
AS_5	E – DE	1	Yes (SN)	UV
AS_6	C2 – DE	3	Yes (SN)	EV
AS_7	E – DE	2	Yes (SN)	EV
AS_8	DE	3	Yes (SN)	S + EV

Note: Each CKC stage is indicated by Reconnaissance (R), Delivery (D), Exploitation (E), Command and Control (C2), Lateral Movement (M), and Data Exfiltration (DE). Attack cost is ranged in $[1,3]$ as an integer, representing low, medium, and high, respectively. Node compromise may involve a single node compromise (SN) or multiple nodes compromise (MN). Exploited vulnerability is indicted by Overall (O: Average vulnerability across all three types of vulnerabilities), Software (SV: software vulnerability), Encryption (EV: vulnerability by compromising encryption key(s)); and Unknown (UV: unknown vulnerability).

where C_k is a set of compromised nodes by given attack k and N is the the total number of nodes. If node j is already compromised, then there is no additional attack impact, $\mathrm{ai}_k = 0$, introduced by attack strategy k. Compromising more important nodes with highly confidential information leads to early system failure (see Eq. (8)).

Attack Strategies: Attackers in IoT environments have their own characteristics. We consider several types of attacks at the different stages of the CKC by an APT attacker. The CKC consists of six stages denoted by (R, D, E, C2, M, and DE) (see Table 2). Each attack strategy is characterized by (1) in which CKC stage the attacker is in; (2) whether the attacker will compromise other nodes in an attack path to reach a target; (3) what attack cost ac_k and attack impact ai_k) are associated with each attack strategy k; and (4) what vulnerability an attacker can exploit to perform a given attack strategy (AS_k). For simplicity, when an attacker exploits more than one vulnerability, the average security vulnerability is used to compute the normalized vulnerability, P_i^v. In addition, each attack strategy k's attack impact, ai_k, is obtained based on Eq. (5). Note that an attacker can select a non-compromised adjacent victim with the highest APV value (see Eq. (4)) to maximize the attack success probability while minimizing the attack cost. We describe each attack strategy as follows:

- AS_1 – *Monitoring attack*: This attack is to collect useful system information and identify a vulnerable node to compromise as a target. It can be performed inside or outside the network from R to DE stages. In this attack, no node compromise process is involved and accordingly its attack cost is low, $\mathrm{ac}_1 = 1$.
- AS_2 – *Social engineering:* The typical examples of this attack include email phishing, pretexting, baiting, or tailgating [20]. We assume that an inside attacker can successfully compromise an adjacent node if the attack is successful. If the attacker is an outside attacker, it can identify a node as vulnerable during its reconnaissance stage. This attack can be performed from D to DE stages as an outside or inside attacker. Since it is highly challenging to deceive a human user who can easily detect a social

engineering attack, the associated attack cost for AS_2 is high, $\mathrm{ac}_2 = 3$.

- AS_3 – *Botnet-based attack:* A botnet consists of compromised machines (or bots) running malware using C2 of a botmaster. When this attack is chosen, all compromised nodes (including original attackers) will launch epidemic attacks (e.g., spreading malware to compromise) to their adjacent, legitimate nodes [6]. This attack can be used from E to DE stages. This attack incurs high attack cost, $\mathrm{ac}_3 = 3$.
- AS_4 – *Distributed Denial-of-Service (DDoS)*: A set of compromised nodes can form a botnet and perform DDoS by sending multiple requests [6]. When an attacker tries to compromise one of its adjacent nodes as a potential victim node, if all compromised nodes send service requests to the potential victim node, the potential victim node's vulnerability may increase because it could not properly handle all operations due to the large volume of requests received (e.g., not properly executing underlying security operations). This will allow the attacker to easily compromise the potential victim node or exfiltrate confidential data from it. To model this, unknown vulnerability, uv_i, for a given victim node i will increase for the attacker to more easily compromise a node with unknown vulnerability (e.g., increasing $\epsilon_1\%$ for UV). This attack can be performed from E to DE stages, with high attack cost, $\mathrm{ac}_4 = 3$.
- AS_5 – *Zero-day attacks:* This attack can be performed to exploit unknown vulnerabilities of software, which are not patched yet. The attacker can compromise chosen adjacent node i based on normalized uv_j. This attack can be performed from E to DE stages at low cost, $\mathrm{ac}_5 = 1$.
- AS_6 – *Breaking encryption:* Examples include a legitimate node's private or secret key compromise. The attacker with the encryption key is considered an inside attacker with a privilege to exploit system resources. This attack can be launched from C2 to DE stages to collect system configurations or confidential information. Upon the attack success, the attacker can intercept all the information to be sent to a victim node whose private key is compromised. This attack may exploit vulnerabilities $\hat{\mathrm{ev}}_i$ associated with encryption keys and involve high attack cost, $\mathrm{ac}_6 = 3$. We assume that if a legitimate node's private key is compromised, the node is compromised. Hence, the attacker can escalate its attack by reauthenticating itself with a new password and steal confidential information or implant malware into file downloads.
- AS_7 – *Fake identity:* This attack can be performed when packets are transmitted without authentication or internal nodes spoofing the ID of a source node, such as MAC/IP/Virtual LAN tag spoofing in an SDN-based IoT by an SDN switch [23]. This attack involves compromising a node with a fake ID. This attack can be performed from E to DE stages with cost, $\mathrm{ac}_7 = 2$. This attack increases the encryption vulnerabilities of its adjacent nodes (e.g., increasing $\epsilon_1\%$ for EV).
- AS_8 – *Data exfiltration*: This attack will also allow the attacker to compromise one of the adjacent nodes. The attacker will check all data compromised by itself until DE stage. Then, if the accumulated importance of compromised data exceeds a certain threshold (i.e., $\sum_{j \in C_A} \mathrm{c}_j > \mathrm{Th}_c$), the attacker can decide whether to exfiltrate the collected intelligence to the outside. This attack costs high with $\mathrm{ac}_8 = 3$.

We summarize the characteristics of all attack strategies considered in terms of the CKC stages involved, attack cost, node compromise, and exploited vulnerability in Table 2. Except AS_1, the attack success from AS_2 to AS_8 is determined based on whether all nodes on the attack path to reach a target node have been successfully compromised. For AS_1, the attack success is determined based on how long the attacker has monitored a target system. This is computed by the probability $\mathrm{vulnerability}_i \cdot e^{-1/T_A}$ where T_A is the time elapsed the attacker has monitored a given target system. This implies that the attack is likely successful when the attacker has more scanned the targeted system longer and find more vulnerabilities. After the attacker exfiltrates data successfully and leaves the system, a new attacker will arrive. Otherwise, the attacker may be evicted by the NIDS or need to try other attack strategies to escalate its attack to a next level.

An Attacker's Deception Detectability: Depending on an attacker's capability, the attacker may have a different level of intelligence to detect defensive deception techniques. We denote it by ad to represent an attacker's probability (omitted an attacker's ID for simplicity) to detect deception used by the defender. An attacker can use this probability, ad, to detect honeypots or honey information, as described in DS_5 in the next section below.

3.5 Defense Model

Attack Intelligence Collection: Different types of defense strategies can be deployed by the defender to counter APT attackers. At the same time, the NIDS will be run periodically (see Section 3.1). Note that we don't count triggering an NIDS as one of defense strategies in order to meet a high standard of the system integrity. When an attacker arrives at the system as an inside attacker (i.e., after the E stage), it can be detected by the NIDS. However, the system aims to collect more attack intelligence (e.g., attack signatures), which can improve the NIDS as a long-term goal. Thus, depending on the perceived risk level from the attacker, the system will determine whether to keep the detected attacker in the system or evict it. We estimate the perceived system risk level based on the criticality level of the compromised node, c_i, and determine if the system will allow the attacker to reside in the system or be evicted according to predefined risk threshold, $\mathrm{Th}_{\mathrm{risk}} \in [0, 1]$. The decision to evict node i, which is detected as compromised, can be given by:

$$\mathrm{Evict}_i = \begin{cases} 1 \text{ if } \mathrm{c}_i > \mathrm{Th}_{\mathrm{risk}} \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

Here $\mathrm{Evict}_i = 1$ means evicting node i while $\mathrm{Evict}_i = 0$ means allowing node i to reside in the system. Note that this rule is applied when node i is detected as compromised by the NIDS regardless of its correctness. Hence, false positive nodes can be also assessed by this rule while false negative nodes can safely reside in the system without being assessed by $\mathrm{Th}_{\mathrm{risk}}$.

When nodes detected as compromised (i.e., true and false positives) are evicted, all associated edges will be

disconnected, which may generate some non-compromised nodes being isolated from the network. To maintain connectivity of non-compromised but isolated nodes, we connect them to the network based on P_i^r to maintain node i's mean degree based on the ER network model [25]. To deal with the attackers (or compromised nodes) residing in the system, which are either false negatives or attackers kept to collect further attack intelligence, the defender system can take the several defense strategies. Each strategy k will be represented by: (i) defense cost (dc_k) in time/complexity and expense, where $\mathrm{dc}_k \in [0, 3]$ as an integer for no, low, medium, and high cost, respectively; (ii) defense impact (di_k) for its defense effectiveness; (iii) the stage of the CKC (i.e., R, D, E, C2, LM, or DE) for strategy k being used; and (iv) system change on what actual changes are made in the system (e.g., what vulnerabilities are reduced or network topology or cryptographic keys being changed). The defense impact, di_k, is computed by:

$$\mathrm{di}_k = 1 - \mathrm{ai}_k, \tag{7}$$

where ai_k is the attack impact introduced by strategy k in Eq. (5). We measure the effectiveness of a defense strategy as the opposite impact of attack success (i.e., successfully compromising a node). That is, attack failure will increase the impact of the defense strategy.

Defense Strategies: This work considers the following defense strategies:

- DS_1 – *Firewalls*: We assume that firewalls are implemented in the SDN controller to monitor and control the incoming and outgoing packet flows according to predefined rules. We model the effectiveness of firewalls by lowering down unknown vulnerabilities (uv_i) all over the network. Specifically, firewall is assumed to reduce vulnerabilities to outside attackers by a certain percent (i.e., $\epsilon_2\%$).
- DS_2 – *Patch Management*: Known vulnerabilities can be patched by a given defense system [22]. A patch is used to temporarily fix software vulnerabilities or provide updates in a full software package. A patch refers to a software update such as code to be installed in a software program. This will decrease software vulnerabilities (sv_i) of all nodes, such as decreasing a certain percent of the vulnerability (i.e., $\epsilon_2\%$).
- DS_3 – *Rekeying Cryptographic Keys*: Cryptographic keys used for all nodes in the network are rekeyed, which lowers the encryption vulnerability by setting $\mathrm{T}_{\mathrm{rekey}} = 1$ which reduces $\hat{\mathrm{ev}}_i = \mathrm{ev}_i \cdot e^{-1/\mathrm{T}_{\mathrm{rekey}}}$.
- DS_4 – *Eviction*: Recall that an attacker with low risk (see Eq. (6)) is allowed to stay in the system for collecting attack intelligence. However, as the system is at risk due to high security vulnerability in terms of the amount of compromised confidential information (i.e., importance; see Eq. (8)), all inside attackers (or compromised nodes) will be evicted from the system. However, the false negatives will remain in the system while a substantial number of compromised nodes is evicted using DS_4.
- DS_5 – *Low/high-interaction honeypots (LHs/HHs)* [21]: LHs and HHs can be activated as a defense strategy. LHs and HHs differ in their deception detectability and cost. In a given network, we deploy a set of LHs and HHs which are deactivated in the deployment phase. When this strategy is selected, they will be activated, which will change the network topology as LHs and HHs are to be connected with a number of nodes in the network. Hence, DS_5 will change attack paths and lure attackers to the honeypots. To be specific, when DS_5 is selected, LHs and HHs will be activated. This will enable them to be connected to highly vulnerable nodes based on $\mathrm{vulnerability}_i$ where HHs will be connected to nodes of higher vulnerability than nodes connected to LHs. In order for the attacker not to reach legitimate nodes, we will only allow incoming connections (i.e., in-degree) from legitimate nodes to the honeypots. Once the attacker is caught by one of the implemented honeypots, it will be diverted to a fake network for monitoring purposes. Recall that an attacker can detect the deception with ad for a LH and $\mathrm{ad}/2$ for a HH.
- DS_6 – *Honey information*: This defense strategy can lure attackers by disseminating false information, such as honey token, fake patch, honey files, or bait files. This strategy will involve the dissemination of false system vulnerability information, such as providing high (low) vulnerabilities for less (more) vulnerable nodes. The attacker will need to detect whether a known vulnerability of a potential victim node is true or fake according to its deception detectability, ad. If the attacker is successfully deceived, it will make an attack strategy decision based on incorrect vulnerability information.
- DS_7 – *Fake keys* [3]: Fake keys can be planted for potential, inside attackers which may use a fake key obtained by compromising another legitimate, inside node to communicate with other nodes to obtain more confidential information. This will be realized that even if the attacker compromises a cryptographic key (e.g., AS_2, AS_6, AS_7, AS_8), a potential victim targeted by the attacker may not be compromised. We model this using the probability the attacker obtains a fake key implanted in nodes, P_{fake}. When the attacker obtains the fake key of a node, the node will not be compromised.
- DS_8 – *Hiding network topology edges*: This strategy hides $c_{NT}\%$ of network edges in order to hide an actual network topology to an attacker. We use a simple rule for each node to hide the edge with the most critical adjacent node based on its criticality value, c_i.

All defense strategies will have corresponding defense

TABLE 3
CHARACTERISTICS OF DEFENSE STRATEGIES

DS	CKC stage	Defense cost (dc)	System change (dsc)
DS_1	R – D	1	Lowering UV
DS_2	D – DE	2	Lowering SV
DS_3	E – DE	3	Lowering EV
DS_4	E – DE	3	Evict all compromised nodes
DS_5	E – DE	3	Lure attackers to with LHs and HHs
DS_6	C2 – DE	1	Disseminate fake system vulnerability information
DS_7	E – DE	2	Plant a fake key
DS_8	R – DE	2	Hide critical network edges

Note: Each CKC stage is indicated by Reconnaissance (R), Delivery (D), Exploitation (E), Command and Control (C2), Lateral Movement (M), and Data Exfiltration (DE). Defense cost is ranged in $[1, 3]$ as an integer, representing low, medium, and high, respectively. System change may involve lowering unknown vulnerabilities (UV), software vulnerabilities (SV), or encryption vulnerabilities (EV).

costs (dc_k's) and are believed useful when the attacker are in certain CKC stages based on the defender's belief. This is used for the defender to choose each subgame based on hypergame theory. We summarized the characteristics of each defense strategy considered in Table 3.

System Failure Conditions: We define that a system failure (SF) occurs when the following condition is met:

$$SF = \begin{cases} 1 \text{ if } \rho_1 \leq \frac{\sum_{i \in G} \mathrm{cp}_i \cdot \mathrm{Importance}_i}{\sum_{i \in G} \mathrm{Importance}_i} \;||\; \rho_2 \geq \frac{|G_t|}{|G|} \\ 0 \text{ otherwise.} \end{cases} \quad (8)$$

Here G_t refers to a network at time t which does not include nodes being evicted while G is an original network. Hence $|G|$ and $|G_t|$ are the number of the original nodes and the number of the current nodes in the system at time t, respectively. ρ_1 is a threshold as a fraction to determine whether a system fails or not based on the sum of compromised nodes' importance values over the sum of all nodes' importance values. SF mainly captures the system failure caused by the loss of three security goals, such as confidentiality, integrity, and availability. ρ_2 is a threshold to determine whether a system can functionally operate based on a sufficient number of active nodes at time t.

4 Attack-Defense Hypergame

First, the attacker will select strategy AS_1 to monitor a target system in the reconnaissance (R) stage, aiming to penetrate into it as a legitimate user. If the attack is successful based on the success probability $\mathrm{vulnerability}_i \cdot e^{-1/T_A}$, the attacker can proceed to the delivery (D) stage of the CKC. In the D stage, the attacker can choose one of the two strategies AS_1 and AS_2. If the attacker can successfully compromise a targeted victim node, which is one of its adjacent nodes, it can successfully penetrate the system and become an inside attacker with legitimate credentials. Now the attacker is in the Exploitation (E) stage. From E to data exfiltration (DE) stages, any inside attacker detected can be assessed by the defender on whether it can stay in the system based on the risk assessment in Eq. (6). Hence, depending on the criticality of the attacked node, the attacker can be detected by the NIDS or be kept in the system if the defense system intends to collect attack intelligence from it. To assess such risk, the attacker should be detected as an attacker (i.e., true and false positives) by the NIDS. If not (i.e., false negatives), the attacker can safely stay even without being detected. From E to DE, the attack is determined as successful if $\mathrm{ai}_i > 0$ (see Eq. (5)). If the original attacker (i.e., a node the attacker is on) is evicted, then a new attacker will arrive. If an attacker is successful by taking AS_8 (data exfiltrated), it will leave the system and a new attacker will arrive. This process will continue until the system fails based on Eq. (8).

Next we formulate the hypergame between the attacker and defender, and define the game components. We provided the detailed explanation of hypergame theory formulations and its related equations in Appendix A, which are used in the sections below.

4.1 Utilities

An attacker's utility (u^A_{pq}) corresponding to attack strategy p (AS_p) can be expressed as the difference between attack gain and attack loss. The attacker's utility (u^A_{pq}) when the attacker takes AS_p and the defender takes DS_q is calculated by:

$$u^A_{pq} = G^A_{pq} - L^A_{pq}, \;\; G^A_{pq} = \mathrm{ai}_p + \mathrm{dc}_q, \;\; L^A_{pq} = \mathrm{ac}_p + \mathrm{di}_q, \quad (9)$$

where the attack and defense cost (i.e., ac_p and dc_q) and the attack and defense impact (i.e., ai_p and di_q) are discussed in Sections 3.4 and 3.5, respectively.

A defender's utility (u^D_{qp}) by selecting DS_q when the attacker takes AS_p can be computed based on the difference between the gain and loss by:

$$u^D_{qp} = G^D_{qp} - L^D_{qp}, \;\; G^D_{qp} = \mathrm{di}_q + \mathrm{ac}_p, \;\; L^D_{qp} = \mathrm{dc}_q + \mathrm{ai}_p. \quad (10)$$

Similar to u^A_{pq}, the attack and defense cost (i.e., dc_q and ac_p) and the attack and defense impact (i.e., di_q and ai_p) are computed. We consider a zero-sum game between the attacker and defender (i.e., $u^A_{pq} + u^D_{qp} = 0$).

4.2 Estimation of Uncertainty

As in Eq. (7) in Appendix A, an attacker's and defender's hypergame expected utilities (HEUs) are estimated based on the level of uncertainty, g, perceived by each player. In this section, we show how the level of g is estimated by the attacker (i.e., g^A) and the defender (i.e., g^D). Note that we omit an ID of the attacker and defender for simplicity.

We model **an attacker's perceived uncertainty** based whether a defensive deception is used and how long the attacker has monitored a target system. That is, given the time period the attacker has monitored in a target system (T_A) and a defense strategy taken (df), the attacker's uncertainty (g^A) is estimated by:

$$g^A = 1 - \exp(-\lambda \cdot \mathrm{df}/T_A). \quad (11)$$

Here λ is a parameter of representing an amount of initial knowledge towards a given system configuration (higher λ increases uncertainty, and vice versa) and $\mathrm{df} = 1 + (1 - \mathrm{ad}) \cdot \mathrm{dec}$. df returns 1 when no defensive deception is used (i.e., $\mathrm{dec} = 0$); it returns $1 + (1 - \mathrm{ad})$ where ad refers to an attacker's deception detectability in $[0, 1]$ when defensive deception is used (i.e., $\mathrm{dec} = \mathrm{dc}$ where dc is defense cost implying that higher defense cost allows higher quality of deception). The formulation of g^A above implies that the attacker has lower uncertainty as it has monitored the target system longer. On the other hand, the attacker has higher uncertainty when it has lower deception detectability and the defender uses a defensive deception strategy. Hence, we set $\mathrm{dec} = \mathrm{dc}$ (defense cost) when defensive deception strategies, DS_5–DS_8, are taken while setting $\mathrm{dec} = 0$ when non-deception defense strategies, $DS_1 - DS_4$, are taken.

A defender's uncertainty towards an attacker increases as it has monitored the attacker for a longer period where

TABLE 4
Possible Strategies Under Each Stage of the CKC

Subgame	CKC stage	Attack strategies	Defense strategies
0	Full game	$AS_1 - AS_8$	$DS_1 - DS_8$
1	R	AS_1	DS_1, DS_8
2	D	AS_1, AS_2	DS_1, DS_2
3	E	$AS_1 - AS_5, AS_7$	$DS_3 - DS_5, DS_7$
4	C2	AS_1–AS_7	DS_3–DS_8
5	M	AS_1–AS_7	DS_3–DS_8
6	DE	AS_1–AS_8	DS_3–DS_8

the defender's monitoring time towards the attacker is denoted by T_D. In addition, if the attacker has not been deceived by defense strategies, it is assumed to be intelligent not to expose its information to the defender. Considering these two, we model g^D by:

$$g^D = 1 - \exp(-\mu \cdot \mathrm{ad}/T_D), \tag{12}$$

where μ is a parameter of representing an amount of initial knowledge towards an attacker (higher λ increases uncertainty, and vice versa), ad is an attacker's deception detectability, and T_D is a defender's accumulated monitoring time towards the attacker. In g^D, the defender perceives lower uncertainty at longer T_D while perceiving higher uncertainty at higher ad.

4.3 Estimation of HEUs

In order to calculate the HEU for each player (see Eq. (7) in Appendix A), we need to obtain P_κ (i.e., the probability a row player chooses subgame κ), $r_{\kappa p}$ (i.e., the probability that a row player takes strategy k in subgame κ), and $c_{\kappa j}$ (i.e., the probability that a column player takes strategy h in subgame κ based on a row player's belief) because S_q is estimated based on P_κ and $c_{\kappa h}$ while $r_{\kappa p}$ is needed when a row player considers strategy k.

Computation of P_κ: Recall that P_κ refers to the probability that subgame κ is played by a row player. We notate this for an attacker and a defender by P^A_κ and P^D_κ, respectively. We define a subgame based on where an attacker is located in the stages of the CKC which will determine a set of available strategies for both parties. We assume that the attacker clearly knows where it is located in the CKC while the defender is not certain about the stage of the attacker in the CKC. We model the defender's P^D_κ based on its uncertainty g^D. Thus, the defender can know the CKC stage of the attacker with $1 - g^D$ (certainty) and correctly choose a subgame based on the attacker's actual stage in the CKC. With g^D, the defender will choose subgame 0 (i.e., a full game with all available strategies). The set of available strategies may be different depending on what subgame to play, as shown in Table 4.

Computation of $r_{\kappa h}$ **and** $c_{\kappa h}$: $r_{\kappa h}$ is the probability that a row player will play strategy h. We denote this for the attacker and defender by $r^A_{\kappa p}$ and $r^D_{\kappa q}$ for attack strategy p and defense strategy q, respectively. $c_{\kappa h}$ is the probability that a column player will take strategy h based on a row player's belief. We also denote this for the attacker and defender by $c^A_{\kappa p}$ and $c^D_{\kappa q}$ attack strategy p and defense strategy q, respectively. In the very beginning, since no historical information is available, each player will use a uniform probability by choosing one of available strategies in a chosen subgame with an equal probability, meaning choosing a strategy at random. As players participate in repeated games, their recorded history regarding what strategies have been taken is available. Then, we will use Dirichlet distribution [32] to model multinomial probabilities based on the strategies taken for past repeated games. If either an attacker or defender is certain about the opponent's strategy, it will estimate its corresponding $r^A_{\kappa p}$, $r^D_{\kappa q}$, $c^A_{\kappa p}$, and $c^D_{\kappa q}$ as:

$$r^A_{\kappa p} = \frac{\gamma^A_{\kappa p}}{\sum_{p \in \mathbf{AS}_\kappa} \gamma^A_p}, \quad c^A_{\kappa q} = \frac{\gamma^D_{\kappa q}}{\sum_{q \in \mathbf{DS}_\kappa} \gamma^D_q}, \tag{13}$$

$$r^D_{\kappa q} = \frac{\gamma^D_{\kappa q}}{\sum_{q \in \mathbf{DS}_\kappa} \gamma^D_q}, \quad c^D_{\kappa p} = \frac{\gamma^A_{\kappa p}}{\sum_{p \in \mathbf{AS}_\kappa} \gamma^A_p}. \tag{14}$$

Note that $\mathbf{AS}_\kappa$ and $\mathbf{DS}_\kappa$ are a set of attack strategies and defense strategies, respectively. γ^D_q (γ^A_p) is the number of times the defender (attacker) will take strategy q (or p) based on the attacker's (defender's) belief up to time $(t-1)$ where the current state is at time t. Since the probability of a column player playing a particular strategy is estimated by a row player's belief, ground truth $c^A_{\kappa q}$ and $c^D_{\kappa p}$ (as shown in the equations above) will be only detected with the probability $(1 - g^A)$ and $(1 - g^D)$ when the row player is an attacker or a defender, respectively. Otherwise, the row player will select one among the available strategies in a given subgame κ at random due to the uncertainty.

An attacker's HEU (AHEU) is computed with: (1) attack utilities (i.e., u^A_{kh}'s in Eq. (9)); (2) the attacker's belief about defense strategy h (i.e., S^A_q in Eq. (6) in Appendix A); and (3) the attacker's perceived uncertainty (i.e., g^A in Eq. (11)).

Similarly, **a defender's HEU (DHEU)** is estimated using: (1) defense utilities (i.e., u^D_{qp}'s in Eq. (10)); (2) the defender's belief about attack strategy p (i.e., S^D_p in Eq. (6) in Appendix A); and (3) the defender's perceived uncertainty (i.e., g^D in Eq. (12)). Both AHEU and DHEU can be obtained based on Eq. (7) in Appendix A. Since the row player selects each strategy h based on $r_{\kappa h}$ for given subgame κ, we calculate AHEU and DHEU as follows:

$$\begin{aligned} \mathrm{AHEU}(rs^A_p, g^A) &= \mathrm{HEU}(rs^A_p, g^A), \\ \mathrm{DHEU}(rs^A_q, g^D) &= \mathrm{HEU}(rs^D_q, g^D). \end{aligned} \tag{15}$$

A player will play a strategy according to the probability distribution of strategies available in a given subgame κ.

5 Experimental Setting

In this work, we use the following **metrics**:

- *Perceived Uncertainty Level* (g^A or g^D): An attacker's or a defender's mean uncertainty level which is measured as shown in Eqs. (11) and (12), respectively.
- *Hypergame Expected Utility* (HEU): This metric measures the HEU of played strategies profile, according to Eq. (7) in Appendix A in the supplement document.
- *Cost for Taking a Chosen Strategy* (C_A or C_D): This metric measures the average attack (or defense) cost paid by an attacker (or a defender) to play a specific strategy. Attack cost (C_A) and defense cost (C_D) of all available strategies are summarized in Tables 2 and 3, respectively. For a given scenario consisting of a series of games until the system fails based on Eq. (8), the average attack or defense cost per game is demonstrated.
- *Mean Time to Security Failure* (MTTSF): This metric measures a system lifetime based on the system states that do not fall in the system failure states based on Eq. (8).
- *TPR of an NIDS*: This metric measures the true positive rate of the NIDS in order to observe how much defensive deception can improve the quality of the NIDS based on the attack intelligence collected during the time of using defensive deception.

Our work compares the performance following schemes:

- *Game with defensive deception and perfect information (DD-PI)*: This scheme plays a game where each player has perfect information regarding which strategy is played by its opponent, which means there is no uncertainty, $g = 0$ (i.e., $g^A = g^D = 0$), when a defender uses all defensive deception (DD) strategies.
- *Game without defensive deception and perfect information (No-DD-PI)*: This scheme plays a game where each player has perfect information regarding what strategy its opponent plays (i.e., $g = 0$) when the defender does not use DD strategies.
- *Hypergame with defensive deception and imperfect information (DD-IPI)*: This scheme plays a game where each player does not have perfect information regarding the strategy of its opponent (i.e., $g > 0$ with $g_A > 0, g^D > 0$) when the defender uses DD strategies. This is our proposed scheme that considers uncertainty g (i.e., imperfect information, IPI) and DD.
- *Hypergame without defensive deception and imperfect information (No-DD-IPI)*: This scheme plays a game where each player does not have perfect information towards what strategy its opponent takes (i.e., $g > 0$) when the defender does not use DD strategies.

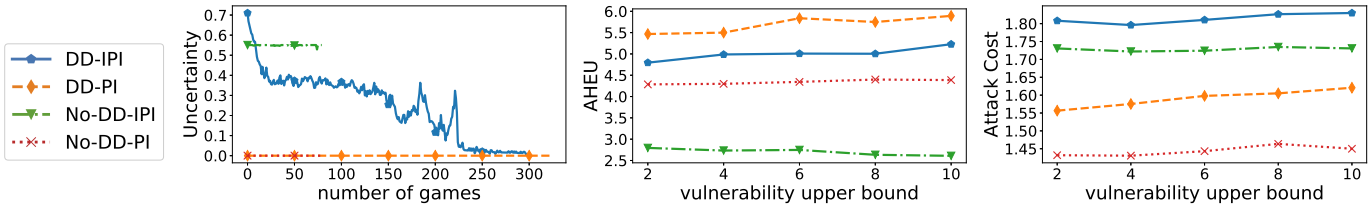
We consider 500 nodes in a given network where a network topology is generated by the ER random graph model with $G(N, P^r)$ where N is the total number of nodes and $P^r (= P_i^r)$ is the connection probability between any pair of nodes [25]. To consider honeypots with low or high interactions, we also assign 75 nodes as honeypots with 50 LHs and 25 HHs. For honeypots, we maintain a directed network where the outgoing edges (i.e., out-degree) are from each honeypot to all other honeypots to ensure an attacker not to be connected with other legitimate nodes. When a honeypot is activated (i.e., DS_5), highly vulnerable nodes are connected to the honeypot as an incoming edge (i.e., in-degree). However, outgoing edges from the honeypot are always forwarded to other honeypots, not real legitimate nodes, which are protected from the attacker. In our experiment, when the honeypots are activated, the top 225 vulnerable nodes are connected to honeypots where top 75 vulnerable nodes are connected to 25 HHs and next top 150 vulnerable nodes are connected to 50 LHs. We assume that the defender has inherently higher uncertainty regarding an attacker while the attacker has a certain level of knowledge regarding a system due to its reconnaissance effort before becoming an inside attacker. This was reflected by setting $\lambda = 0.8$ and $\mu = 8$ in Eqs. (11) and (12), respectively. We summarized the notations of key design parameters, their meaning, and default values used in Table 1 in Appendix B of the submitted supplement document.

6 Results & Analyses

In Fig. 1a, the attacker's perceived uncertainty is plotted for the four defense schemes. When imperfect information (IPI) is considered, the attacker has fairly high uncertainty at the beginning regardless of whether defensive deception (DD) is used or not (i.e., DD-IPI starting from over 0.7 and No-DD-IPI starting from over 0.55). The reason is that using DD strategies can provide a high chance to increase the attacker's uncertainty by misleading the attacker. On the other hand, under perfect information (PI), the attacker's uncertainty is zero. However, without DD (i.e., No-DD-PI and No-DD-IPI), the system lifetime (i.e., MTTSF) is short, so the curve with respect to the number of games stops at around 80 rounds. Under DD-IPI, the system lifetime much more prolongs compared to No-DD schemes. However, the attacker's uncertainty under DD-IPI decreases with more rounds of games played, because playing more games will result in more compromised nodes. A new attacker can leverage this situation to get into the system quicker. This makes the inside attacker stay longer in the system, resulting in lowering uncertainty as more rounds of games are played. Some more fluctuations in later games are due to the small number of runs in simulation show long system lifetimes. Notice that using DD makes the system prolong even if it allows some compromised nodes to reside in the system. This is because the system does not evict detected intrusions immediately after detecting them but does reassess them to reduce false positives while collecting more attack intelligence. This process can give a chance for the NIDS to improve its detection rate. In addition, the attacker perceives lower uncertainty as more games are played as it can perceive less uncertainty about the system since it has been in the system for a while. This was intentionally allowed by the defender to collect attack intelligence.

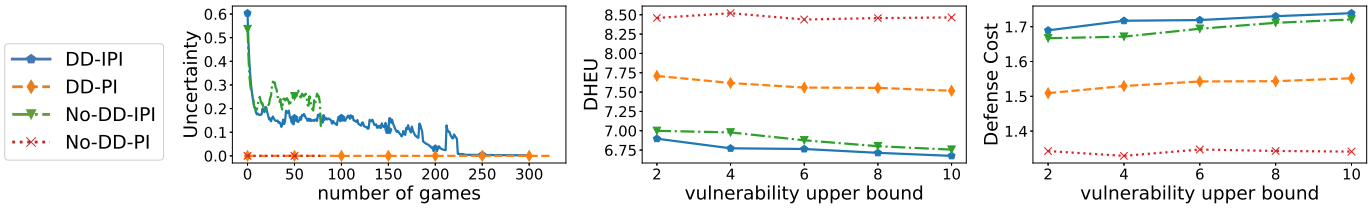
In Figs. 1b and 1c, under varying the vulnerability of nodes in the network, we plotted AHEU and attack cost for each defense scheme. We didn't observe any noticeable sensitivity with respect to varying the extent of node vulnerability. This is because all three metrics, uncertainty, AHEU, and attack cost do not depend on network conditions but rather depend on the choices of strategies by the attacker and corresponding impact and cost in HEU. In terms of AHEU in Fig. 1b, overall, the attacker performs better under DD-based schemes than No-DD-based schemes. The reason is that under DD-based schemes, attackers can use more strategies by being an insider of the system while performing only monitoring attacks as an outside attacker under No-DD-based schemes. In addition, this leads the attacker naturally to perform better under PI than IPI. This explains why the attacker obtains the highest AHEU under DD-PI while having the lowest AHEU under No-DD-IPI. In terms of attack cost, the attacker used more cost under IPI while using less cost under PI as shown in Fig. 1c. Under uncertainty, the attacker cannot choose its optimal, cost-effective strategy. Moreover, the attacker paid higher cost under DD while incurring a lower cost under No-DD. This implies that DD strategies are effective to mislead the attacker to choose less cost-effective strategies by increasing its uncertainty. We also discussed how the attack cost and AHEU with respect to the number of games in Fig. 1 (a)-(b) in Appendix B of the supplement document with the detailed explanations of the observed trends.

In Fig. 2a, the defender's uncertainty is shown with respect to the number of attack-defense hypergames. Overall under IPI, the defender's uncertainty is much lower than the attacker's uncertainty. This is because the defender can collect more attack intelligence while the attacker can be interrupted by DD strategies which increase the attacker's uncertainty. In addition, there are more fluctuations under



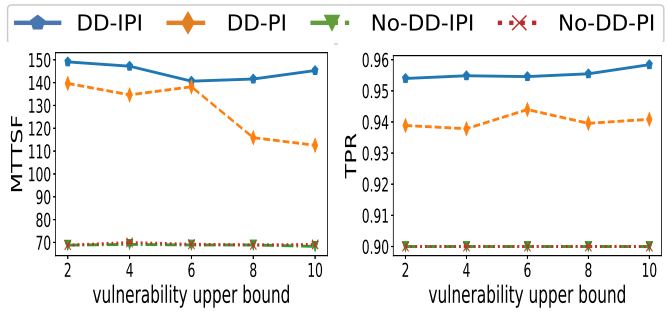
(a) Attacker's uncertainty (b) Attacker's HEU (c) Attack cost

Fig. 1. An attacker's uncertainty, hyergame expected utility (AHEU), and attack cost. The 'vulnerability upper bound' (U_v) refers to the CVSS-based software vulnerability score of IoT devices, Web servers and Databases, which is scaled in $[1, U_v]$.



(a) Defender's uncertainty (b) Defender's HEU (c) Defense cost

Fig. 2. A defender's uncertainty, hyergame expected utility (DHEU), and defense cost. The 'vulnerability upper bound' (U_v) refers to the CVSS-based software vulnerability score of IoT devices, Web servers and Databases, which is scaled in $[1, U_v]$.



(b) MTTSF (c) TPR of an NIDS

Fig. 3. System lifetime (i.e., MTTSF) and true positive rate (TPR) of an NIDS under varying the level of system vulnerability.

No-DD-IPI is because attackers are not allowed to stay in a system if they are detected as compromised. This keeps resetting the time the attacker has stayed in the system, which makes the defender observe the same attacker for a longer time.

In Figs. 2b and 2c, we further show DHEU and defense cost for varying the vulnerability upper bound of nodes (U_v) in the network. In Fig. 2b, compared to AHEU (i.e., 2.5 to 6), we can observe much higher DHEU (i.e., 6 to 8.5). Since HEU is estimated based on the impact and cost of taking a chosen strategy, using DD costs more, leading to lowering DHEU. Besides, under IPI, the defender may not choose its optimal strategy all the time, lowering down DHEU due to less benefit of taking a chosen strategy. Hence, it is reasonable to observe that the highest DHEU is obtained with No-DD-PI while the lowest DHEU is observed with DD-IPI. In Fig. 2c, as expected, the highest defense cost incurs under DD-IPI while the lowest defense cost is observed under No-DD-PI. This also reflects the role of the defense cost in DHEU. DHEU and defense cost with respect to the number of games are also discussed in Fig. 1 (c)-(d) in Appendix B of the submitted supplement document.

In Fig. 3b, we showed how the four different schemes perform under varying the extent of node vulnerability in terms of MTTSF. Regardless of whether PI or IPI is considered, DD-based schemes outperformed non-DD-based schemes. Again this is because DD-based schemes allow for the reassessment of detected intrusions, leading to reduction in false positives while improving TPR of the NIDS. However, DD-IPI outperformed all other schemes in terms of MTTSF. This is because IPI can allow the defender to effectively leverage the nature of DD strategies for misleading the attacker effectively and making it choose non-optimal strategies. Moreover, we notice that the behavior of DD schemes is sensitive to node vulnerability, showing the reduced MTTSF under high vulnerability because the attacker can better exploit vulnerable nodes and more efficiently compromise them. Except insensitivity under high vulnerability nodes, the performance trends in TPR of the NIDS are well aligned with those in MTTSF under the four schemes, as shown in Fig 3c. TPR can be improved under DD-IPI due to the high effectiveness of DD under IPI.

We also discussed the probability of each strategy taken by an attacker and a defender in Figs. 2 and 3 and TPR of the NIDS in Fig. 4 of Appendix B with respect to the number of attack-defense games played under each scheme in the submitted supplement document.

7 Conclusion & Future Work

From this study, we obtained the following **key findings**:

- An attacker's and defender's perceived uncertainty can be reduced when defensive deception (DD) is used. This is because the attacker perceives more knowledge about the system as it performs attacks as an inside attacker. On the other hand, the defender's uncertainty can be reduced by collecting more attack intelligence by using DD while allowing the attacker to be in the system.
- Attack cost and defense cost are two critical factors in determining HEUs (hypergame expected utilities). Therefore, high DHEU (defender's HEU) is not necessarily related to high system performance in MTTSF (mean time

to security failure) or TPR (true positive rate) which can also be a key indicator of system security. Therefore, using DD under imperfect information (IPI) yields the best performance in MTTSF (i.e., the longest system lifetime) while it gives the minimum DHEU among all schemes.
- DD can effectively increase TPR of the NIDS in the system based on the attack intelligence collected through the DD strategies.

This work bring up some important directions for future research by: (1) considering multiple attackers arriving in a system simultaneously in order to consider more realistic scenarios; (2) estimating each player's belief based on machine learning in order to more correctly predict a next move of its opponent; (3) dynamically adjusting a risk threshold, i.e., Eq. (6), depending on a system's security state; (4) introducing a recovery mechanism to restore a compromised node to a healthy node allowing the recovery delay; (5) developing an intrusion response system that can reassess a detected intrusion in order to minimize false positives while identifying an optimal response strategy to deal with intrusions with high urgency; and (6) considering another intrusion prevention mechanism, such as moving target defense, as one of the defense strategies.

ACKNOWLEDGEMENT

This research was partly sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-19-2-0150. In addition, this research is also partly supported by the Army Research Office under Grant Contract Number W91NF-20-2-0140. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] "Common vulnerability scoring system (CVSS)." [Online]. Available: https://www.first.org/cvss/

[2] Y. M. Aljefri, M. A. Bashar, L. Fang, and k. W. Hipel, "First-level hypergame for investigating misperception in conflicts," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 48, no. 12, pp. 2158–2175, 2017.

[3] H. Almeshekah and H. Spafford, "Cyber security deception," in *Cyber Deception*. Springer, 2016, pp. 25–52.

[4] C. Bakker, A. Bhattacharya, S. Chatterjee, and D. L. Vrabie, "Learning and information manipulation: Repeated hypergames for cyber-physical security," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 295–300, 2019.

[5] P. G. Bennett, "Toward a theory of hypergames," *Omega*, vol. 5, no. 6, pp. 749–751, 1977.

[6] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017.

[7] M. Boussard, D. T. Bui, L. Ciavaglia, R. Douville, M. L. Pallec, N. L. Sauze, L. Noirie, S. Papillon, P. Peloso, and F. Santoro, "Software-defined LANs for interconnected smart environment," in *2015 27th Int'l Teletraffic Congress*, Sep. 2015, pp. 219–227.

[8] U. Brandes, "A faster algorithm for betweenness centrality," *Jour. mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[9] J. W. Caddell, "Deception 101-primer on deception," DTIC Document, Tech. Rep., 2004.

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

[11] W. Casey, A. Kellner, P. Memarmoshrefi, J. A. Morales, and B. Mishra, "Deception, identity, and security: The game theory of Sybil attacks," *Comms. of the ACM*, vol. 62, no. 1, pp. 85–93, 2018.

[12] J.-H. Cho, M. Zhu, and M. P. Singh, *Modeling and Analysis of Deception Games based on Hypergame Theory*. Cham, Switzerland: Springer Nature, 2019, ch. 4, pp. 49–74.

[13] K. Ferguson-Walter, S. Fugate, J. Mauger, and M. Major, "Game theory for adaptive defensive cyber deception," in *Proc. 6th Annual Symp. on Hot Topics in the Science of Security*. ACM, 2019, p. 4.

[14] N. Garg and D. Grosu, "Deception in honeynets: A game-theoretic analysis," in *Proc. IEEE Information Assurance and Security Workshop (IAW)*. IEEE, 2007, pp. 107–113.

[15] B. Gharesifard and J. Cortés, "Evolution of the perception about the opponent in hypergames," in *Proc. 49th IEEE Conf. Decision and Control (CDC)*, Dec. 2010, pp. 1076–1081.

[16] ——, "Evolution of players' misperceptions in hypergames under perfect observations," *IEEE Trans. Automatic Control*, vol. 57, no. 7, pp. 1627–1640, Jul. 2012.

[17] J. T. House and G. Cybenko, "Hypergame theory applied to cyber attack and defense," in *Proc. SPIE Conf. Sensors, and Command, Control, Comms., and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, vol. 766604, May. 2010.

[18] T. Kanazawa, T. Ushio, and T. Yamasaki, "Replicator dynamics of evolutionary hypergames," *IEEE Trans. Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 1, pp. 132–138, Jan. 2007.

[19] N. S. Kovach, A. S. Gibson, and G. B. Lamont, "Hypergame theory: A model for conflict, misperception, and deception," *Game Theory*, 2015, article ID 570639, 20 pages.

[20] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Jour. Information Security and Applications*, vol. 22, pp. 113–122, 2015.

[21] S. Kyung, W. Han, N. Tiwari, V. H. Dixit, L. Srinivas, Z. Zhao, A. Doupé, and G. Ahn, "Honeyproxy: Design and implementation of next-generation honeynet via SDN," in *2017 IEEE Conf. Comms. and Network Security (CNS)*, Oct. 2017, pp. 1–9.

[22] O. Leiba, Y. Yitzchak, R. Bitton, A. Nadler, and A. Shabtai, "Incentivized delivery network of IoT software updates based on trustless proof-of-distribution," in *2018 IEEE European Symp. on Security and Privacy Workshops (EuroS PW)*, Apr. 2018, pp. 29–39.

[23] Y. Liu, Y. Kuang, Y. Xiao, and G. Xu, "SDN-based data transfer security for Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 257–268, Feb. 2018.

[24] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira, "Programmable networks—from software-defined radio to software-defined networking," *IEEE Comms. Surveys Tutorials*, vol. 17, no. 2, pp. 1102–1125, Second Quarter 2015.
[25] M. E. J. Newman, *Networks: An introduction*, 1st ed. Oxford University Press, 2010.
[26] H. Okhravi, M. A. Rabe, W. G. Leonard, T. R. Hobson, D. Bigelow, and W. W. Streilein, "Survey of cyber moving targets," Lexington Lincoln Lab, MIT, TR 1166, 2013.
[27] U. S. Putro, K. Kijima, and S. Takahashi, "Adaptive learning of hypergame situations using a genetic algorithm," *IEEE Trans. Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 30, no. 5, pp. 562–572, Sep. 2000.
[28] Y. Sasaki, "Subjective rationalizability in hypergames," *Advances in Decision Sciences*, vol. 2014, no. Article ID 263615, p. 7 pages, 2014.
[29] A. Schlenker, O. Thakoor, h. Xu, F. Fang, M. Tambe, L. Tran-Thanh, P. Vayanos, and Y. Vorobeychik, "Deceiving cyber adversaries: A game theoretic approach," in *Proc. 17th Int'l Conf. Autonomous Agents and MultiAgent Systems*, 2018, pp. 892–900.
[30] W. L. Sharp, "Military deception," Joint War-Fighting Center, Doctrine and Education Group, Norfolk, VA, Pub. 3-13.4, 2006.
[31] S. Tadelis, *Game Theory*. Princeton University Press, 2013.
[32] Y. W. Teh, *Dirichlet Process*. Boston, MA: Springer US, 2010, pp. 280–287.
[33] R. Vane, "Planning for terrorist-caused emergencies," in *Proc. Winter Simulation Conf.*, Dec. 2005.
[34] R. Vane and P. E. Lehner, "Using hypergames to select plans in adversarial environments," in *Proc. 1st Workshop on Game Theoretic and Decision Theoretic Agents*, 1999, pp. 103–111.
[35] Wikipedia. (2018) Foureye butterflyfish. Available at https://en.wikipedia.org/wiki/Foureye_butterflyfish.
[36] Y. Yin, B. An, Y. Vorobeychik, and J. Zhuang, "Optimal deceptive strategies in security games: A preliminary study," in *Proc. AAAI Conf. Artificial Intelligence*, 2013.