Foureye: Defensive Deception Against Advanced Persistent Threats via Hypergame Theory

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 (CKC) 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 the attacker and defender, the system lifetime (i.e., mean time to security failure), and improved false-positive rates of intrusion detection.

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 [1]. 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 the 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 [2, 3].

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 [4]. 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 the best strategy [4].

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 [6].

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 are costly in contested environments. We partly addressed these challenges in our prior work [7]. In [7], we mainly considered a small-scale network with only four different strategies for each player, only one defensive deception strategy (i.e., fake patch), only three CKC stages, an attacker's deception detectability, limited modeling of strategy effects on attackers and systems, static utility functions (i.e., constants), no Hyper Nash Equilibrium (HNE) analysis, using Stochastic Petri Nets (SPNs) with simplified designs, limited performance metrics, and no comparative performance analysis. We summarized the detailed key

• *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.*

TABLE 1
THE KEY DIFFERENCES BETWEEN [7] AND THIS PAPER.

Design feature	Cho et al. [7]	This paper
Attack/defense strategies	4 strategies each	8 strategies each
# of defensive deception strategies	1	4
Attacker's deception detectability	No	Yes
# of CKC stages	3	6
Validation method	Stochastic Petri Nets (SPNs)	Extensive simulation
Strategy effects	System states in SPNs (implicit)	Explicit modeling in terms of vulnerability and system changes
Utility calculation	Static with constants	Dynamic utility functions
Analysis of HNE	No	Yes
# of metrics for evaluation	4	6
Comparative performance analysis	No	Yes (4 schemes)

differences between [7] and this paper in Table 1.

This paper has been substantially extended [7] with the additional contributions. We made the following **key contributions** in this work:

- We are the first that considers hypergame theory to model an attack-defense cybergame in order to reflect players' perceived uncertainty towards a given game more realistically. No prior work has considered uncertainty estimated by the dynamic interactions between an attacker and a defender in a cyber deception game.
- We consider an advanced persistent threat (APT) by modeling an attacker that can perform multiple attack strategies in the different stages of the cyber kill chain (CKC). Most existing approaches are limited to considering scanning attacks in the reconnaissance stage.
- We consider multiple, diverse defense strategies, including both conventional defense and defensive deception (DD) strategies, to investigate the key benefits of using DD strategies.
- We conduct extensive simulation experiments to compare the performance and security of a system under four different games, such as the games of perfect vs. imperfect information and the games of using DD vs. non-DD.
- We also investigate Hyper Nash Equilibrium (HNE) [8, 9, 10, 11] to study how players' NE solutions based on their perceived games can be different from the NE solutions based on a ground truth game.

2 RELATED WORK

Garg and Grosu [12] 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 [13] 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. [14] 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. [15] 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. [16] 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.

Xiao et al. [17] studied the APT attacks by proposing a detection game under uncertainty. They used the cumulative prospect theory for an attacker and defender to make subjective decisions on 'whether to attack or not' or what interval to use for performing scanning attacks. Fang et al. [18] introduced a game model to predict an APT attacker's move. They designed the attacker's strategies based on the APT attack path. Specifically, they separated the attacker's actions in terms of the means and the aim of an attack. The authors designed a game model to predict the optimal APT attack path so that the defender can effectively deal with the APT attackers. Pawlick et al. [19] proposed a framework by considering APT attackers within cloud-based systems. Specifically, they considered a three-player game with a cloud defender, an attacker, and a device. They employed a signaling game to model the interactions between the device and the cloud which may be compromised with a certain probability. However, the works above [17, 18, 19] assumed that all players play the same game with a single view, which is not realistic in practice.

Bakker et al. [20] applied a repeated hypergame to model interactions between an APT attacker and the defender within a cyber-physical system. The authors considered that the attacker can manipulate the control system, while the defender can sweep the system based on its monitoring scheme. The authors extended [20] in [21] by applying metagames and minimax hypergames in their hypergames and APT attacks. Hutchins et al. [22] proposed an intelligence-driven, threat-focused approach to assist the defender to mitigate intrusions by APT attackers. The authors modeled the APT attacks based on their cyber kill chain (CKC) and illustrated the stages in the CKC with several actual cases. Zhang and Thing [23] surveyed honeypots, honey tokens, and moving target defense techniques from the late 1980s to 2021. The authors first described the CKC and illustrated how the different deception technologies can disturb APT attacks following the CKC.

As similar proactive mechanisms to defensive deception in terms of their goal and effect of confusing attackers and mislead them to fail by changing attack surfaces, moving target defense (MTD) has been discussed in the literature. We extensively surveyed MTD in our prior work [24] and clarified the difference between MTD and DD. Although they are common in their goals and effects, DD uses false information (e.g., fake keys or honey information) or objects (e.g., decoys or honeypots) to mislead the attacker's beliefs and manipulate the attacker's perception. In addition, obfuscation has been used to deceive attackers. Ye et al. [25] proposed a cyber deception game where a defender uses differential privacy and obfuscation techniques aiming to strategically change the number of systems and the configurations of systems, respectively. In addition, they modeled powerful attackers that can infer real configurations of systems based on a Bayesian inference algorithm. However, this work did not consider players' perceived uncertainty which can affect their views towards the given game and

the moves of the opponent.

Unlike the above works [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 25], 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 beliefs, aiming to mislead them to adopt uncertain or non-optimized strategies. Although other game theories, aiming to derive Nash Equilibrium solutions, deal with uncertainty by considering probabilistic choices of strategies (e.g., mixed strategies) and probabilistic occurrences of events (e.g., the probability distribution of a player's type), they assume that all players play the same game [26] where the probability distribution of players' actions are known and their beliefs towards the moves of the players are correct. However, in a hypergame, these assumptions do not hold.

Hypergame theory has been used to solve decision-making problems in military and adversarial environments [27, 28, 29]. Vane and Lehner [27] explained how a decision-maker chooses actions when it has a belief context about another players' probable play. The authors claimed several insights and findings. For example, if the row player only knows little about the column player, the solution is close to the Nash Equilibrium Mixed Strategy for the entire game. To quickly protect the valuable targets and first responders, Vane [28] leveraged a hypergame by assisting the defender to plan purposes. The authors designed a defender who needs to dynamically plan countermeasures after immediate threats, such as mall fire or the corresponding additional threats. House and Cybenko [29] used a hypergame to mimic the interaction between the attacker and the defender. The attacker can compromise or infiltrate a computer network with three different attack methods, and the defender can decide to monitor the network with a junior sysadmin (who is less effective) or senior sysadmin. The defender can also deploy a decoy computer that lures the attacker and collects the attack evidence. The authors presented a hidden Markov model and maximum entropy for learning the unknown subgame probabilities.

Several studies [30, 31] 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. [8] studied an individual's belief in an evolutionary hypergame and how this belief can be modeled by interpreter functions. Sasaki [32] discussed the concept of *subjective rationalizability* where an agent believes that its action is the best response to the other agent's choices based on its perceived game. Putro et al. [33] proposed an adaptive, genetic learning algorithm to derive optimal strategies by players in a hypergame. The authors used hypergame and genetic algorithm (GA) to support a group of decision-makers to understand the nature. They developed three different learning procedures about nature. Ferguson-Walter et al. [34] 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. [35] designed a new model to analyze a first-level hypergame within the paradigm of graph model for conflict resolution (GMCR). Specifically, the authors developed a first-level hypergame in a graph form to analyze two decision-makers or n decision-makers conflicts. The authors categorized the first-level hypergame equilibrium into eight classes to understand the types of misperception involved in the dispute. Bakker et al. [20, 36] modeled a repeated hypergame in dynamic stochastic setting against APT attacks primarily in cyber-physical systems. One of their game moves was a 'sweep' where a defender had the capability to clear all malicious actors from the system for one time. They found that an advanced attacker can negate the effect of a higher frequency of sweep. In addition, the results showed that the worst-case cost (i.e., energy expenditure) for the defender is disproportionately higher than the average case.

Unlike the works using hypergame theory above [8, 17, 20, 27, 28, 29, 30, 31, 32, 33, 34, 35], 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 Hypergame Theory

We leverage the hypergame theory [37] in order to consider more realistic conditions of real-world applications where players' beliefs are not necessarily correct and the players may play differently depending on their perception about a game they play. In this section, we provide the background knowledge for readers to easily understand the key concepts of the hypergame theory. Hypergame theory offers two levels of hypergames that allow the analysis of games differently perceived by multiple players [37]. Although hypergame theory applies to multiple players, in this work, we consider a two-player game with a party of attackers and a defender system and adopt first-level hypergames.

A cyber deception game can be formulated as a regular, sequential game, $(G; G^A; G^D)$, where G is an original game and G^A and G^D are games perceived by an attacker and a defender, respectively [38]. When $G = G^A = G^D$, we obtain a conventional game as both players play the same game G. But when $G \neq G^A \neq G^D$ or $G^A \neq G^D$, the two players view the game differently and move according to their beliefs about the strategies taken by the opponent. Although the hypergame theory (HT) has been considered in the cybersecurity domain [38, 39], no prior work has studied HT with respect to a defender dealing with advanced persistent threat (APT) attackers.

3.1 First-Level Hypergame

Given two players, p and q, vectors of their preferences, denoted by V_p and V_q, define game G that can be represented by $G = \{V_p; V_q\}$ [37]. Note that V_p and V_q are player p's and player q's actual preferences (i.e., ground truth), respectively. If all players exactly know all other players'

preferences, all players are playing the same game because their view of the game is the same. However, in reality, the assumption may fail. Player p can perceive player q's preferences differently from what they are, leading to differences between p's view and q's view. A game perceived by player p based on its perceived preferences about q's preferences, V_{qp}, and the game perceived by player q based on its perceived preferences about p's preferences, V_{pq}, can be given by:

$$G_p = \{V_{qp}\}; G_q = \{V_{pq}\} \quad (1)$$

Hence, the first-level hypergame H perceived by each player is written by $H^1 = \{G_p; G_q\}$. In a first-level hypergame, analysis is performed at the level of each player's perceived game because each player plays the game based on its belief. Even if the player does not know all outcomes of the game, the outcome can be stable for the player because the player may not unilaterally change its belief. If a game includes an unknown outcome, the unknown outcome is caused by uncertainty. The stability of an outcome about a game is determined by each player's reaction to the action by the opponent. An outcome is *stable* for p's game if the outcome is stable in each of p's perceived preference vectors, i.e., in each V_{qp}. The equilibrium of p's game is determined by the outcome that p believes to resolve the conflict [37].

3.2 Hypergame Normal Form (HNF)

Vane [40] provides a hypergame normal form (HNF) that can succinctly model hypergames based on players' beliefs and possible strategies of their opponents. HNF is formulated, similar to the normal strategic form in game theory. HNF consists of the following four key aspects: (1) full game; (2) row-mixed strategies (RMSs); (3) column-mixed strategies (CMSs); and (4) belief contexts.

The **full game** is the grid form consisting of row and column strategies, which are associated with the utilities, $ru_{11}; \ldots; ru_{mn}$ and $cu_{11}; \ldots; cu_{mn}$ where m is the number of the row player's strategies and n is the number of the column player's strategies. The full game's grid form U can be represented by an $m \times n$ matrix with an element ru_{ij}, cu_{ij} for $i = 1; \ldots; m$ and $j = 1; \ldots; n$.

$$U = \begin{pmatrix} (ru_{11}, cu_{11}) & \cdots & (ru_{1n}, cu_{1n}) \\ \vdots & & \vdots \\ (ru_{m1}, cu_{m1}) & \cdots & (ru_{mn}, cu_{mn}) \end{pmatrix}; \quad (2)$$

where R_0 and C_0 denote the full-game strategies by the row and column players, respectively.

Row-mixed strategies (RMSs) are m strategies the row player considers based on its belief of the column player's strategies. A player's subgame is defined as a subset of the full game (i.e., a set of all possible strategies by all players) because the player may limit the number of strategies it wants to consider based on its belief. For example, if a defender perceives attacks by outside attackers, such as reconnaissance (e.g., scanning) attacks, it will not use a defense strategy using fake keys, which is mostly applicable to deal with inside attackers. Therefore, depending on the situation, the player can choose a subgame to play. RMSs for the ℓ-th subgame a player perceives are given by:

$$\text{RMS}_\ell = [r_{\ell 1}; \ldots; r_{\ell m}]; \text{ where } \sum_{i=1} r_{\ell i} = 1; \quad (3)$$

where each probability that a particular strategy i is chosen is estimated by player p's belief based on learning from past experience. Since a subgame consists of a subset of strategies in a full game, if a particular strategy i is not in the subgame ℓ, the probability for the row player to take strategy i at subgame ℓ is zero, i.e., $r_{\ell i} = 0$.

Column-mixed strategies (CMSs) are a column player's n strategies, believed by a row player for a ℓ-th subgame, which are denoted by:

$$\text{CMS}_\ell = [c_{\ell 1}; \ldots; c_{\ell n}]; \text{ where } \sum_{j=1} c_{\ell j} = 1; \quad (4)$$

where the probability that strategy j be chosen is obtained by player p's observations (or learning) towards q's strategies. Similar to the row-mixed strategies, if strategy j is not in subgame ℓ, we set $c_{\ell j} = 0$.

Belief contexts are the row player's belief probabilities that each subgame ℓ will be played and are represented by:

$$P = [P_0; \ldots; P_K]; \text{ where } \sum_{\ell=0} P_\ell = 1. \quad (5)$$

Here P_0 is the probability that the full game is played where the full game considers all possible strategies of a player based on the ground truth view of a situation. If the row player is not sure of what subgame ℓ to play due to perceived uncertainty, the unknown belief probability is treated simply for the probability that a full game is played, denoted by $P_0 = 1 - \sum_{\ell=1}^{K} P_\ell$.

The row player's belief towards the column player's strategy j, denoted by S_j, is computed by:

$$S_j = \sum_{\ell=0} P_\ell c_{\ell j} \text{ where } \sum_{j=1} S_j = 1. \quad (6)$$

The summary of the row player's belief on the column player's n strategies is represented by $C^P = [S_1; S_2; \ldots; S_n]$.

3.3 Hypergame Expected Utility

The hypergame expected utility (HEU) can be calculated based on EU( ) (see Eqs. (8) and (9)), and the uncertainty probability perceived by the row player, denoted by g, representing the level of uncertainty about what is guessed about a given game. The g affects the degree of the EU( ) of a given hyperstrategy (i.e., RMS in Eq. (3)) by the row player. The HEU for the given row player's strategy rs_i with the row player's perceived uncertainty g in a given context is given by [41]:

$$\text{HEU}(rs_i; g) = \quad (7)$$
$$(1 - g)\ EU(rs_i; C^P) + g\ EU(rs_i; \text{CMS}_w);$$

where $g = 0$ means complete confidence (i.e., complete certainty) perceived by the row player while $g = 1$ implies complete uncertainty perceived by the row player [40]. $\text{EU}(rs_i; C^P)$ refers to the row player's expected utility in choosing strategy i when the column player can take a strategy among all available strategies n. $\text{EU}(rs_i; \text{CMS}_w)$ indicates the row player's expected utility when choosing strategy i when the column player chooses strategy w which

gives the minimum utility to the row player. $\mathrm{EU}(rs_i; C^{\mathrm{P}})$ and $\mathrm{EU}(rs_i; \mathrm{CMS}_w)$ are computed by:

$$\mathrm{EU}(rs_i; C^{\mathrm{P}}) = \sum_{j=1}^{x} S_j \quad u_{ij}; \tag{8}$$

$$\mathrm{EU}(rs_i; \mathrm{CMS}_w) = n \quad S_w \quad u_{iw}: \tag{9}$$

The calculation of $\mathrm{EU}(rs_i; \mathrm{CMS}_w)$ is based on a pessimistic perspective in that when a player is uncertain, it estimates utility based on the worst scenario.

In our work, we consider a realistic scenario that, when the player is uncertain, it will simply choose a random strategy among strategies in a given subgame. If the defender is not sure of what subgame the attacker played, it will simply play a full game. When players making a decision based on the probability distribution of HEU, the values of HEU will be normalized using the min-max normalization method [42].

4 System Model

In this section, we describe the network model, node model, and the assumptions made in this work.

4.1 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 [43]. 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 NIDS. SDN-enabled switches handle forwarding packets, where they encapsulate packets following 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.

In a given network, the nodes collect data and perform a periodic delivery of those collected data to the servers via multi-hop communications 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 [44] for the effective and efficient management of IoT nodes with the help of an SDN controller.

We assumed our network is an SDN-based IoT environment to leverage easy deployment of various defense strategies, particularly defensive deception techniques. However, our proposed attack-defense framework can be deployed in various network environments which have infrastructure playing the role of the centralized defender.

4.2 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:

$$c_i = \mathrm{importance}_i \quad \mathrm{reachability}_i; \tag{10}$$

where $\mathrm{importance}_i$ is given as an integer ranged in $[0; 10]$ during the network deployment phase. $\mathrm{reachability}_i$ is computed based on the faster betweenness centrality metric [45] by the SDN controller. Unlike the traditional betweenness centrality finding the shortest paths between each pair of nodes with the Floyd-Warshall algorithm, Brandes [45] proposed the faster betweenness (FB) centrality metric to reduce the complexity of betweenness. FB discovers minimum paths using breadth-first search (BFS) for unweighted graphs or Dijkstra's algorithm for weighted graphs to find all shortest paths starting from node s. In this process, the predecessor set of node v on the shortest paths starting from node s, denoted by $P_s(v)$, is discovered where $P_s(v) = \{u \in V : \{u, v\} \in E, d_G(s, u) + \omega(u, v) = d_G(s, v)\}$. Given the shortest path from s to u counted by $\sigma_{sv} = \sum_{u \in P_s(v)} \sigma_{su}$, faster betweenness of node v is calculated by $C(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where the $\sigma_{st}(v)$ is the number of shortest paths from s to t passes v. The running time for the faster betweenness is $O(|V||E|)$ for an unweighted graph and $O(|V||E| + |V|^2 \log |V|)$ for a weighted graph, which is much faster than betweenness centrality metric. 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 [46]. The $\mathrm{reachability}_i$ is estimated in the range of $[0; 1]$ as a real number. In [47], we conducted an extensive survey on centrality metrics and investigated how betweenness has been used in various networks. As shown in Table 10 of [47], betweenness has been commonly used in communication networks to select critical nodes for preventing or mitigating the spread of computer viruses or malware, or choose targeted attackers. To estimate a node's reachability, other types of centrality metrics can be used, as our prior work in [47] surveyed a rich volume of centrality metrics and their impact on network resilience. However, testing with multiple centrality metrics is out of the scope of this work.

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 $\mathrm{ev}_i = \mathrm{ev}_i \quad e^{\ 1/T_{\mathrm{rekey}}}$ and T_{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) [48] with the severity value in $[0; 10]$ as an

TABLE 2
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]

integer. We measure the average vulnerability associated with node i being vulnerable by:

$$\text{vulnerability}_i = \frac{\sum_{v_j \in V_i} v_j}{|V_i|}; \quad (11)$$

where V_i is a set of vulnerabilities associated with node i (e.g., $\{sv_0; sv_1; sv_2; ev_0; ev_1; ev_2; uv_0\}$) and v_j refers to one of vulnerabilities, associated with node i where v_j is measured as an integer ranged in $[0; 10]$ following the CVSS. We denote $P_i^v = \text{vulnerability}_i/10$ as a normalized vulnerability probability. The P_i^v is used as the probability to exploit (i.e., compromise) node i by an attacker.

Network topology changes caused by node 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 the same number of neighbors (nodes being directly connected).

Table 2 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 7. We select each attribute value at random based on the 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., $cp_i = 1$ for compromise; 0 otherwise). We assume that attackers may know the vulnerabilities of adjacent nodes with limited observability (i.e., no perfect knowledge is assumed), but do not know the vulnerabilities of non-adjacent nodes. However, the defender is aware of vulnerabilities and the importance of nodes in Table 2.

4.3 Assumptions

We assume that the SDN controller and control channel are trusted. Scott-Hayward [49] designed secure, robust, and resilient SDN controllers, which consist of secure controller design, secure controller interfaces, and controller security services. Raghunath and Krishnan [50] also analyzed several types of research about SDN control plane security. They introduced existing works on protecting SDN from different attacks, such as DDoS, TCP SYN, and packet injection attacks. To achieve the security of the SDN controller and control channel, we assumed to use the SDN security services provided by [49, 50]. 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 security vulnerabilities (see Section 4.2) 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}; ;) where refers to false negatives (FN) and is true positives (TP) and the expected value (mean) [51] $E[P_{fn}]$ would be: $E[P_{fn}] = FN/(TP + FN)$. Similarly, as more attack intelligence is forwarded to NIDS via defensive deception-based monitoring, (TP) increments by 1 per monitoring interval. Similarly, false positives will be reduced as defensive deception techniques are more frequently used where $E[P_{fp}] = FP/(TN + FP)$ and TN increments 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 becomes 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 through passing authentication process and providing a secret key or using the private key of a legitimate user.

4.4 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 [52]. 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 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 of 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. An attacker will first select strategy k based on the probability distribution of strategies derived from the HEU of its attack strategies, then it will select target node i among adjacent nodes according to the normalized vulnerability probability, P_i^v, and attack cost metric of selected strategy, ac_k, as described in Eq. (12).

Moreover, if the attacker finds an adjacent node already compromised by the previous attacker, it can leverage the compromised nodes, which does not require additional effort. That is, the attacker will aim to select a node with the least cost with the highest vulnerability. We estimate this by the so-called Attack Path Vulnerability (APV), denoted by $APV(i;j;k)$, which refers to the value of intermediate node i in an attack path to reach target node j where k refers to attack strategy ID. Highest $APV(i;j;k)$ will be added to the attacker's attack path to the target node j. The $APV(i;j;k)$ is given by:

$$APV(i;j;k) = \begin{cases} (3 - ac_k)\ P_i^v & \text{if } cp_i == 0 \wedge path(i;j) > 0; \\ 3 & \text{if } cp_i == 1 \wedge path(i;j) > 0; \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

Here ac_k is a predefined attack cost ranged in $[1;3]$ as an integer (see 'Attack Strategy Attributes' below). The P_i^v is estimated by $P_i^v = vulnerability_i/10$ where $vulnerability_i$ is estimated in Eq. (11). The $path(i;j)$ returns 1 when there exists a path between nodes i and j; 0 otherwise. If $cp_i == 1 \wedge path(i;j) > 0$ (i.e., node i is compromised and a path exists between i and j), the attacker may add it to the attack path at no cost with $APV(i;j;k) = 3$. The attacker may need to compromise more than one intermediate node 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. (10)). 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:

$$ai_k = \frac{\sum_{j \in C_k} c_j}{N}; \quad (13)$$

where C_k is a set of compromised nodes by given attack k and N is the total number of nodes. If node j is already compromised, then there is no additional attack impact, $ai_k = 0$, introduced by attack strategy k. Compromising more important nodes with highly confidential information leads to early system failure (see Eq. (16)).

Attack Strategies: Attackers in IoT environments have their 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 3). Each attack strategy is characterized by (1) in which CKC stage the attacker is in; (2) what attack cost ac_k and attack impact ai_k are associated with each attack strategy k; (3) whether the attacker will compromise other nodes in an attack path to reach a target (e.g., 'AS_3 - Botnet-based attack' does not consider APV); 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. (13). Note that an attacker can select a non-compromised adjacent victim with the highest APV value (see Eq. (12)) to maximize the attack success probability while minimizing the attack cost. We describe each attack strategy as follows:

TABLE 3
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 indicated 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).

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 the R to DE stages. In this attack, no node compromise process is involved and accordingly, its attack cost is low, $ac_1 = 1$.

AS_2 – *Social engineering:* The typical examples of this attack include email phishing, pretexting, baiting, or tailgating [53]. 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, $ac_2 = 3$.

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 [54]. This attack can be used from E to DE stages, with high attack cost, $ac_3 = 3$.

AS_4 – *Distributed Denial-of-Service (DDoS)*: A set of compromised nodes can form a botnet and perform DDoS [54]. 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 $_1$% for UV). This attack can be performed from E to DE stages, with high attack cost, $ac_4 = 3$.

AS_5 – *Zero-day attacks:* This attack can be performed to exploit unknown vulnerabilities of software, which are not patched yet. When AD_5 is performed, the attacker can leverage unpatched vulnerabilities to attack its adjacent nodes and obtain their root permission. We model this attack behavior by the attacker that can compromise its adjacent nodes using the probability based on the normalized uv_j, represented by uv_j=10. This attack can be performed from E to DE stages at low cost, $ac_5 = 1$. The effect of launching this attack can be shown in terms of attack impact in terms of how much confidential information has been compromised by the zero-day attack.

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 the 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 succeeds, the attacker can intercept all the information to be sent to a victim node whose private key is compromised. This attack may exploit vulnerabilities ev_i associated with encryption keys and involve high attack cost, $ac_6 = 3$. 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 [55]. This attack involves compromising a node with a fake ID. This attack can be performed from E to DE stages with cost, $ac_7 = 2$. This attack increases the encryption vulnerabilities of its adjacent nodes (e.g., increasing $_1$% 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 the DE stage. Then, if the accumulated importance of compromised data exceeds a certain threshold (i.e., $\sum_{j \in C_A} \text{importance}_j > Th_c$), the attacker can decide whether to exfiltrate the collected intelligence to the outside. This attack costs high with $ac_8 = 3$.

We summarize the characteristics of all attack strategies considered in terms of the CKC stages involved, attack cost, node compromise, and exploited the vulnerability in Table 3. Except for 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 $\text{vulnerability}_i \; e^{\,1=T_A}$, 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 scanned the targeted system longer and finds 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 try other strategies to escalate its attack to the next level.

An Attacker's Deception Detectability: Depending on an attacker's capability, it 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 honeypot, honey information, fake key, and hidden edge as D_5 D_8 described in the following section.

4.5 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 4.1). Note that we do not count triggering an NIDS as one of the defense strategies in order to meet a high standard of 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 Th_{risk}, which is a real number ranged in $[0; 1]$. The decision to evict node i, which is detected as compromised, can be given by:

$$\text{Evict}_i = \begin{cases} 1 \text{ if } c_i > Th_{risk} \\ 0 \text{ otherwise.} \end{cases} \quad (14)$$

Here $\text{Evict}_i = 1$ means evicting node i while $\text{Evict}_i = 0$ 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 Th_{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 [46]. To deal with the attackers (or compromised nodes) residing in the system, the defender system can take several defense strategies. Each strategy k will be represented by: (i) defense cost (dc_k) in time/complexity and expense, where $dc_k \; 2 \; [0; 3]$ 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:

$$di_k = 1 \quad ai_k; \tag{15}$$

where ai_k is the attack impact introduced by strategy k in Eq. (13). 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: We consider the four well-known conventional defense strategies and four popular deception defense (DD) strategies. The conventional four defense strategies include patch management, firewall, rekeying cryptographic keys, and eviction of detected compromised nodes, which aim to prevent, detect, mitigate, and respond to intrusions, respectively. We also chose the four DD strategies, which deploy honeypots, honey information, fake keys, and hiding network topology edges to mislead attackers, waste their resources, and ultimately result in attack failure. Each defense strategy is described as follows:

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. When the defender launches DS_1, the firewall is updated with new rules that can cover the latest vulnerabilities. Since our work concerns the SDN with IoT mobility, we assume to use the stateful firewall to filter the TCP flow and maintain the state of each IoT device until it connects to the system [56]. We model the effectiveness of firewalls by lowering down unknown vulnerabilities (uv_i) all over the network by a certain percent (i.e., $_2$%).

DS_2 – *Patch Management*: Known vulnerabilities can be patched by a given defense system [57]. 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., $_2$%).

DS_3 – *Rekeying Cryptographic Keys*: Cryptographic keys used for all nodes in the network are rekeyed, which lowers the encryption vulnerability ev_i. We modeled this by formulating $ev_i = ev_i \; e^{\;1=T_{rekey}}$, which is similarly modeled as in a system reliability model [58] where $1=T_{rekey}$ is used as a failure rate. T_{rekey} increments by 1 per game to count how long the key has been used in terms of the number of games. When this strategy is taken, it means a new key is issued, so we reset $T_{rekey} = 1$.

DS_4 – *Eviction*: Recall that an attacker with low risk (see Eq. (14)) 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. (16)), 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)* [59]: 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. When DS_5 is selected, LHs and HHs will be activated. This will enable them to be connected to highly vulnerable nodes based on $vulnerability_i$ 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 an LH and $ad{=}2$ for an HH. When a honeypot strategy is taken by the defender, it will last for the duration of a given subgame.

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, which ultimately will result in attack failure.

DS_7 – *Fake keys* [2]: 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.

We excluded moving target defense (MTD) techniques in the defense strategies because their goals, effects, and roles may overlap those of DD strategies [24, 60]. Hence, we focus our work on investigating the pure effect of DD strategies.

All defense strategies will have corresponding defense costs (dc_k's) and are believed useful when the attacker is 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 4.

4.6 System Failure Conditions

We define a system failure (SF) state in terms of failing three security goals, such as *loss of confidentiality*, *loss of*

TABLE 4
CHARACTERISTICS OF DEFENSE STRATEGIES

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

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).

integrity, and *loss of availability*. The loss of confidentiality occurs when a certain amount of important information is compromised. The loss of integrity occurs when the system has too many compromised nodes. Lastly, the loss of availability occurs when the system is malfunctioning due to too many nodes being evicted. Based on the three conditions of the security breach, we define SF by:

$$SF = \begin{cases} 1 \text{ if } {}_1 \dfrac{\sum_{p \in G} \mathrm{cp}_i \ \mathrm{Importance}_i}{\sum_{i \in G} \mathrm{Importance}_i} \ jj \ {}_2 \dfrac{jG_t j}{jGj} \\ 0 \text{ otherwise.} \end{cases} \quad (16)$$

Here cp_i is the compromised status of node i, returning 0 for non-compromise; 1 otherwise. $\mathsf{Importance}_i$ is the characteristic of each node shown in Table 2. G_t refers to a network at time t which does not include nodes being evicted while G is an original network. Hence jGj and $jG_t j$ are the number of the original nodes and the number of the current nodes in the system at time t, respectively. The ${}_1$ 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. The ${}_2$ is a threshold to determine whether a system can functionally operate based on a sufficient number of active nodes at time t. Therefore, in the SF condition, the first term (i.e., ${}_1 \dfrac{\sum_{p \in G} \mathrm{cp}_i \ \mathrm{Importance}_i}{\sum_{i \in G} \mathrm{Importance}_i}$) checks the loss of confidentiality and integrity while the second term (i.e., ${}_2 \dfrac{jG_t j}{jGj}$)) checks the loss of availability.

5 ATTACK-DEFENSE HYPERGAME

5.1 Procedures and Structure of a Game

First, the attacker will select strategy AS_1 to monitor a target system in the reconnaissance (R) stage, aiming to penetrate it as a legitimate user. If the attack is successful in the stage of the R, it means it successfully identified a vulnerable node during the period of the reconnaissance, T_A. Hence, the attacker is successful in identifying node i based on the success probability, $\mathsf{vulnerability}_i \ e^{\ 1=T_A}$. After then, 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. (14). 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 $\mathsf{ai}_i > 0$ (see Eq. (13)). 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. (16).

The attacker-defender game considered in this work is a repeated game of complete information and imperfect information where each game session will be a multi-stage game as an APT attacker performs multi-stage attacks, following the CKC stages. Since we assume that an attacker and a defender know their opponent's type, action space, and payoff, they play a game of complete information. However, although players play sequential games in the given repeated game, within each game session where an attacker and a defender take an action, they take actions simultaneously, which forms a game of imperfect information. In addition, they are limited in accurately observing opponent player's actions, generating the level of perceived uncertainty, g, as discussed in Section 3. In Fig. 1, we summarized the key ideas of a player's choice of the strategy at each CKC game and the overall structure of the repeated game considered in this work.

5.2 Utilities

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

$$u^A_{ij} = G^A_{ij} \quad L^A_{ij}; \ G^A_{ij} = \mathsf{ai}_i + \mathsf{dc}_j; \ L^A_{ij} = \mathsf{ac}_i + \mathsf{di}_j; \quad (17)$$

where the attack and defense cost (i.e., ac_i and dc_j) and the attack and defense impact (i.e., ai_i and di_j) are discussed in Sections 4.4 and 4.5, respectively.

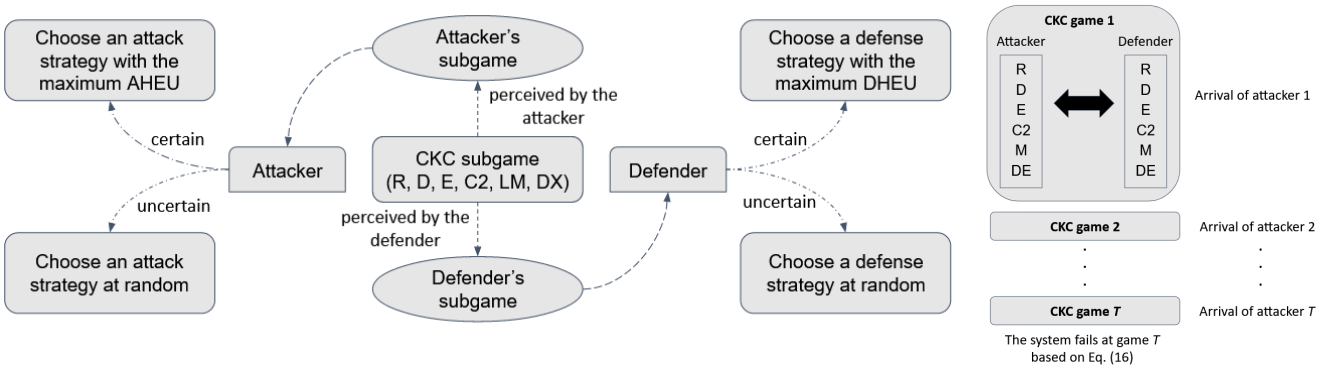
A defender's utility (u^D_{ij}) by selecting DS_j when the attacker takes AS_i can be computed based on the difference between the gain and loss by:

$$u^D_{ji} = G^D_{ji} \quad L^D_{ji}; \ G^D_{ji} = \mathsf{di}_j + \mathsf{ac}_i; \ L^D_{ji} = \mathsf{dc}_j + \mathsf{ai}_i: \quad (18)$$

Similar to u^A_{ji}, the attack and defense cost (i.e., dc_j and ac_i) and the attack and defense impact (i.e., di_j and ai_i) are computed. We consider a zero-sum game between the attacker and defender (i.e., $u^A_{ij} + u^D_{ji} = 0$).

5.3 Estimation of Uncertainty

As shown in Eq. (7) in Section 3, an attacker's and defender's hypergame expected utilities (HEUs) are estimated based on the level of uncertainty, g, perceived by each



(a) A player's choice of the strategy at a single CKC game (b) The structure of a repeated game

Fig. 1. The description of overall game: (a) describes how the attacker and defender choose its strategy at each CKC stage. The overall game consists of multiples of the CKC games as shown in (b), which shows an example case. Each CKC game consists of multiple plays of the attacker and defense and the whole game consists of multiples of CKC games until the system fails. The modeled game is a repeated game where each session is a CKC game.

TABLE 5
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

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 on 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 the effectiveness of defensive deception strategies taken (df), the attacker's uncertainty (g^A) is estimated by:

$$g^A = 1 \quad \exp(\quad (1 + (1 \quad \text{ad}) \quad \text{dec})=T_A); \tag{19}$$

where is a parameter of representing an amount of initial knowledge towards a given system configuration (higher increases uncertainty, and vice versa) and ad refers to the probability that an attacker can detect defensive deception deployed by the defender and dec indicates whether a defensive deception (DD) strategy is taken. That is, dec = 1 when DD strategies (i.e., DS_5–DS_8) are taken by the defender while setting dec = 0 when non-DD strategies (i.e., DS_1 DS_4) are taken by the defender. 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 ad and the defender takes a DD strategy.

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

$$g^D = 1 \quad \exp(\quad \text{ad}=T_D); \tag{20}$$

where is a parameter representing an amount of initial knowledge towards an attacker (i.e., higher 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.

5.4 Estimation of HEUs

In order to calculate the HEU for each player (see Eq. (7)), we need to obtain P (i.e., the probability a row player chooses subgame), r_i (i.e., the probability of a row player taking strategy i in subgame), and c_j (i.e., the probability of a column player taking strategy j in subgame based on a row player's belief) because S_j is estimated based on P and c_j (see Eq. (6)) while r_i is needed when a row player considers strategy i.

5.4.1 Computation of P

Recall that P refers to the probability that subgame is played by a row player. We notate this for an attacker and a defender by P^A and P^D, 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 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 based on its uncertainty g^D. Thus, the defender can know the CKC stage of the attacker with 1 g^D (i.e., 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 5.

5.4.2 Computation of r_i and c_j

The r_i is the probability that a row player will play strategy i. We denote this for the attacker and defender by r^A_i and r^D_j for attack strategy j and defense strategy j, respectively. The

c_j is the probability that a column player will take strategy j based on a row player's belief. We also denote this for the attacker and defender by c^A_i and c^D_j attack strategy i and defense strategy j, respectively. In the very beginning, since no historical information is available, each player will use a uniform probability by choosing one of the available strategies in a chosen subgame with an equal probability, meaning that a strategy is chosen 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 [61] 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_i, r^D_j, c^A_i, and c^D_j by:

$$r^A_i = \frac{^A_i}{\sum_{i2\mathbf{AS}} {}^A_i}; \; c^A_j = \frac{^D_j}{\sum_{j2\mathbf{DS}} {}^D_j}; \tag{21}$$

$$r^D_j = \frac{^D_j}{\sum_{j2\mathbf{DS}} {}^D_j}; \; c^D_i = \frac{^A_i}{\sum_{i2\mathbf{AS}} {}^A_i}: \tag{22}$$

Note that $\mathbf{AS}$ and $\mathbf{DS}$ are a set of attack strategies and defense strategies, respectively. ^D_j (^A_i) is the number of times the defender (attacker) will take strategy j (or i) based on the attacker's (defender's) belief up to time $(t \quad 1)$ 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_j and c^D_i (as shown in the equations above) will be only detected with the probability $(1 \quad g^A)$ and $(1 \quad g^D)$ 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 at random due to the uncertainty.

5.4.3 Hypergame Expected Utilities

An attacker's HEU (AHEU) is computed with: (1) attack utilities (i.e., u^A_{ij}'s in Eq. (17)); (2) the attacker's belief about defense strategy j (i.e., S^A_j in Eq. (6)); and (3) the attacker's perceived uncertainty (i.e., g^A in Eq. (19)). Similarly, a defender's HEU (DHEU) is estimated using: (1) defense utilities (i.e., u^D_{ji}'s in Eq. (18)); (2) the defender's belief about attack strategy i (i.e., S^D_i in Eq. (6)); and (3) the defender's perceived uncertainty (i.e., g^D in Eq. (20)). Both AHEU and DHEU can be obtained based on Eq. (7). Since the row player selects strategy i based on r_i for given subgame , we calculate AHEU and DHEU by:

$$\begin{aligned} \text{AHEU}(rs^A_i; g^A) &= \text{HEU}(rs^A_i; g^A); \\ \text{DHEU}(rs^D_j; g^D) &= \text{HEU}(rs^D_j; g^D): \end{aligned} \tag{23}$$

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

The use of subjectively perceived uncertainty by each player (see Eqs (19) and (20)) in determining the defender's subgame and calculating the attacker's and defender's HEUs (see Eq. (23)) shows how the concept of bounded rationality is applied in our formulated cyber deception hypergame.

6 Experimental Setting

In this section, we describe comparing schemes for performance analysis, the metrics, and the network setting that are considered in our experiments.

6.1 Metrics

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. (19) and (20), respectively.
Hypergame Expected Utility (HEU): This metric measures the HEU of played strategies profile based on Eq. (7).
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 3 and 4, respectively. For a given scenario consisting of a series of games until the system fails based on Eq. (16), 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. (16).
TPR of an NIDS: This measures the true positive rate of the NIDS to observe how much DD contributes to enhancing the quality of the NIDS by increasing the TPR.
FPR of an NIDS: This measures the false positive rate of the NIDS in order to observe how much DD contributes to enhancing the quality of the NIDS by reducing the FPR.
HNE Hitting Ratio: This measures the fraction that the attacker's HNE ($HNE(G^A)$) is matched with the defender's HNE ($HNE(G^D)$), where G^A and G^D are the games viewed by the defender and attacker, respectively.

6.2 Comparing Schemes

We compare the performance of the 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., $g^A = g^D = 0$) 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., $g_A > 0; g^D > 0$) when the defender uses DD strategies. This is our proposed scheme.
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., $g_A > 0; g^D > 0$) when the defender does not use DD strategies.

6.3 Network Setting

We consider 500 nodes in a given network where 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 [46]. 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 does not connect 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. 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 the top 75 vulnerable nodes are connected to 25 HHs and the 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 = 0:8 and = 8 in Eqs. (19) and (20), respectively.

Table 6 summarizes the notations of key design parameters, their meaning, and default values used in our experiments. We selected the default values used in Table 6 based on the following rationale. Based on our investigation, the effects of varying , , ad, and Th_{risk} are not significant. On the other hand, we used = 8 and Th_{risk} = 0:3 to maximize MTTSF under the DD-IPI scheme. In addition, following the concept of Byzantine failure [62] assuming that a system cannot operate properly if more than one-third of system components are compromised, we set $_1$ = 1=3. As a system requirement to ensure service availability, we assume at least half of the nodes in the network should be available and accordingly set $_2$ = 0:5.

The simulation is coded in Python 3.8, and the network is constructed with the NetworkX package. We run our code in the TinkerCliffs cluster of VT-ARC where we requested 500 threads from 5 AMD EPYC 7702 CPUs for computing. The system is Red Hat 7.7. The simulation source code is available at https://github.com/Wan-ZL/Foureye-1-Simulation.

TABLE 6
NOTATIONS OF KEY DESIGN PARAMETERS AND THEIR DEFAULT VALUES

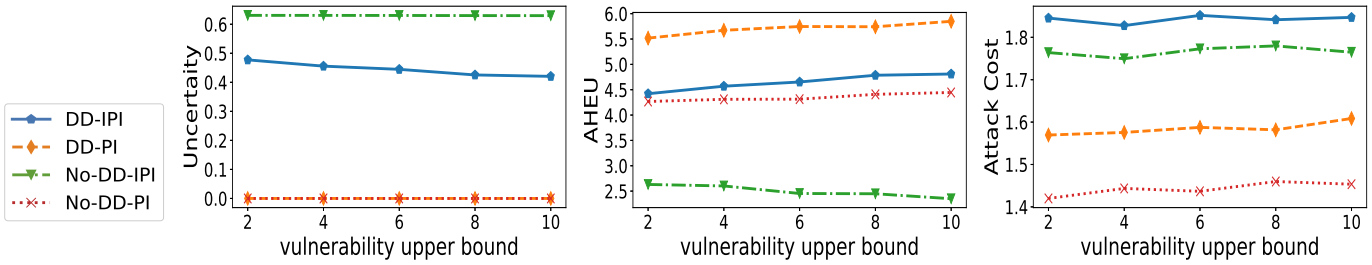
Symbol	Meaning	Default
ac_k	Cost of attack strategy k	Table 2
dc_k	Cost of defense strategy k	Table 3
$_1$, $_2$	Thresholds for SF in Eq. (16)	1/3, 1/2
N_{LH}	Number of low-interaction honeypots deployed but not activated in a network	50
N_{HH}	Number of high-interaction honeypots deployed but not activated in a network	25
N_{WS}	Number of Web servers deployed in a network	25
N_{DB}	Number of databases deployed in a network	25
N_{IoT}	Number of IoT nodes deployed in a network	450
N	Total number of nodes	500
nv_i	Total number of security vulnerabilities of node i, including encryption (5), software (5), and unknown (1)	11
P^r	Probability of two nodes being connected in an Erdös-Rényi random network	0.05
ad	An attacker's deception detectability	[0; 0:5]
,	A constant for normalization for the attacker's uncertainty and defender's uncertainty, respectively	1, 8
P_{fp}, P_{fn}	Probabilities of false positives and false negatives in the NIDS	0.01, 0.1
Th_{risk}	Risk threshold used in Eq. (14)	0.3
Th_c	The threshold used in AS_8 (Data exfiltration)	150
$_1$; $_2$	Increased or decreased percent of a given vulnerability probability by taking attack strategies (i.e., AS_5; AS_7) or defense strategies (i.e., DS_1 DS_3)	10, 1
P_{fake}	Probability the attacker obtains a fake key	1 ad
c_{NT}	Percentage (%) of edges that are hidden by defense strategy DS_8	20

7 RESULTS & ANALYSES

Fig. 2a shows the average uncertainty perceived by the attacker when varying the vulnerability upper bound of nodes in a given network where each data point was shown based on 100 times simulation runs. Since DD-PI and No-DD-PI have zero uncertainty (i.e., $g^A = g^D = 0$), the players have perfect information towards each other. This means the attacker and defender can accurately know the moves of their opponent, respectively. Therefore, zero uncertainty with games of perfect information is reasonable. However, under imperfect information, when defensive deception (DD) strategies are used (i.e., DD-IPI), low-risk attackers are allowed to stay in the system and they can decrease their perceived uncertainty due to their longer staying time in the system. On the other hand, under No-DD-IPI, no use of DD strategies does not allow any attackers in the system. This makes the attackers hard to reduce their perceived uncertainty towards the defender (i.e., the given system).

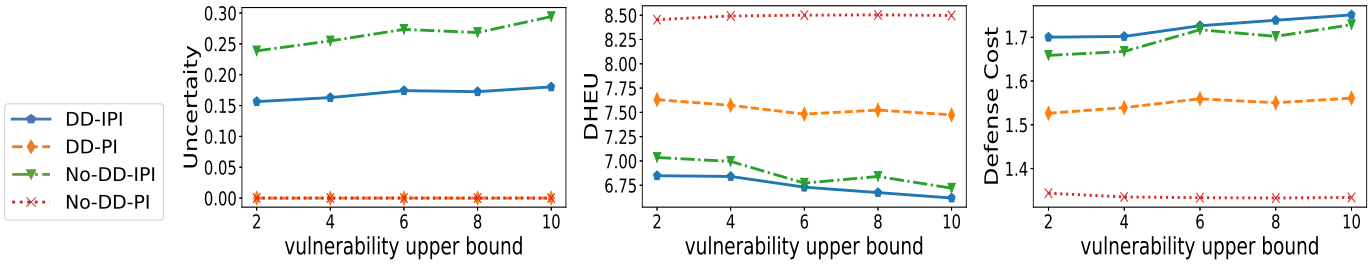
In Figs. 2b and 2c, under varying the vulnerability of nodes in the network, we plotted AHEU and attack cost for each defense scheme. We did not observe any noticeable sensitivity with respect to varying the extent of node vulnerability. This is because all three metrics, including uncertainty, AHEU, 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. 2b, overall the attacker performs better under DD-based schemes than No-DD-based schemes. The reason is that under the 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. 2c. Under uncertainty, the attacker cannot choose its optimal, cost-effective strategy. Moreover, the attacker paid a 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 A of the supplement document with the discussions of the observed trends.

In Figs. 3a, 3b and 3c, we showed the defender's uncertainty, DHEU, and defense cost when the vulnerability upper bound of nodes (U_v) in the network varies. In Fig. 3a,



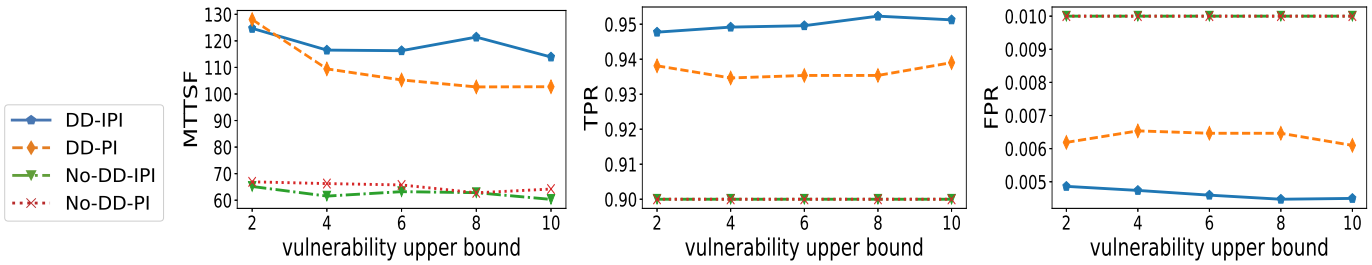
(a) Attacker's uncertainty (b) Attacker's HEU (c) Attack cost

Fig. 2. 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, where U_v is given as an integer ranged in [0; U_v].



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

Fig. 3. 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, where U_v is given as an integer ranged in [0; U_v].



(a) MTTSF (b) TPR of an NIDS (c) FPR of an NIDS

Fig. 4. System lifetime (i.e., MTTSF), true positive rate (TPR), and false positive rate (FPR) of an NIDS under varying the level of system vulnerability.

DD-PI and No-DD-PI have zero uncertainty due to the perfect information available. Under DD-IPI, the defender's uncertainty is much lower than No-DD-IPI. This is because the defender can observe the same attacker for a longer period and collect more attack intelligence because using DD allows low-risk attackers to stay in the system.

In Fig. 3b, 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 and results in 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. 3c, 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 A of the submitted supplement document.

In Fig. 4a, 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 a reduction in false positives while improving the 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 for 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 4b. TPR can be improved under DD-IPI due to the high effectiveness of DD under IPI. In Fig. 4c, we observed: (1) No-DD-IPI and No-DD-PI

TABLE 7
PAYOFF MATRICES FOR HNE CALIBRATION

(a) Defender's Game G^D

	AS_x	AS_y
DS_x	DHEU(rs_x^D), AHEU(rs_x^A)	DHEU(rs_x^D), AHEU(rs_y^A)
DS_y	DHEU(rs_y^D), AHEU(rs_x^A)	DHEU(rs_y^D), AHEU(rs_y^A)

(b) Attacker's Game G^A

	DS_x	DS_y
AS_x	AHEU(rs_x^A), DHEU(rs_x^D)	AHEU(rs_x^A), DHEU(rs_y^D)
AS_y	AHEU(rs_y^A), DHEU(rs_x^D)	AHEU(rs_y^A), DHEU(rs_y^D)

schemes have the highest FPR because those schemes do not use the defensive deception to obtain additional attack intelligence that can decrease the FPR of the NIDS; and (2) DD-IPI scheme has a lower FPR than DD-PI scheme because the defender can have better chances to monitor attackers by taking the benefit of using DD strategies.

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

8 HYPER NASH EQUILIBRIUM

Nash equilibrium is a solution concept for game theory where each player has complete information and the same view about the game. However, in hypergame theory, each player has a unique view of the game and may not have a correct belief about the strategy taken by the opponent. Therefore several studies [8, 9, 10, 11] introduced the Hyper Nash Equilibrium (HNE) for discovering the strategy equilibrium based on the belief of each player. The HNE is defined as:

For N rational players in a game $\hbar N; fA_i g_{i=1}^N; fv_i g_{i=1}^N i$, each player i have a set of action, $A_i = fa_i^1; a_i^2; : : : ; a_i^m g$ where m actions available. We define a_i as HNE if the following meet:

$$v_i(a_i; a_{\ ii}) \quad v_i(a_i; a_{\ ii}) \text{ for } 8i; 8a_i; \tag{24}$$

where i refers to the players except player i. a_{ji} means player j's action in player i's belief, A_{ji} means player j's strategy space in player i's belief, and $a_{ji}; a_{ji}\ 2\ A_{ji}$. The $v_i(\ )$ means player i's payoff function.

The HNE is based on each player's belief about a game G. We defined the attacker's view as G^A, and the defender's view as G^D, where G^A, G^D, and G may not be the same. In addition, the $G^A = G^D$ does not mean the belief of attacker and defender are correct based on the ground truth game G, because attacker and defender may have the same misbelief. In our work, the $v_i(a_i; a_{ji})$ is AHEU for the attacker and DHEU for the defender. Since player need to estimate opponent's AHEU/DHEU based on the observed action record, defender can estimate AHEU by replacing $r_p^{A_t} = c_p^{A_t}$ and $c_q^{D_t} = r_q^{D_t}$, and attacker can estimate DHEU by replacing $r_q^{D_t} = c_q^{D_t}$ and $c_q^{D_t} = r_q^{D_t}$. The player's belief is computed by Eq. (21)–(22). For easy understanding, we show Table 7 as a simple example, where each player has a different view about the game.

Based on the concept of Hyper Nash Equilibrium, we investigate the HNE hitting ratio to show the probability that the attacker's HNE matches the defender's HNE in our proposed game. Fig. 5 is the result under a game with fixed

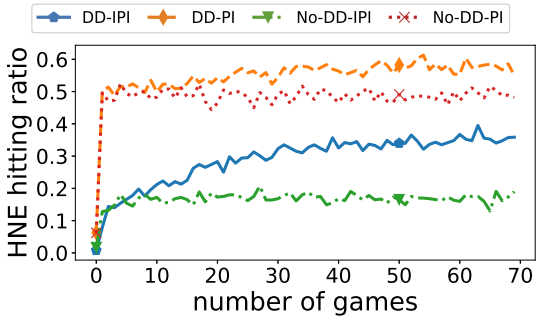


Fig. 5. Hyper Nash Equilibrium (HNE) Hitting Ratio.

values from Table 6. According to the result, we found: (1) The schemes with imperfect information (IPI) have lower ratios than the schemes with perfect information because the existence of uncertainty diverges players' beliefs about the game; (2) DD-IPI scheme has a low ratio at the beginning but gradually increased because the uncertainty for both attacker and defender decreases as their monitoring time increases; and (3) No-DD-IPI performs worse than DD-IPI because both the attacker and defender do not have many chances to observe each other as detected attackers will be evicted from the system immediately.

9 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 significantly reduced by collecting attack intelligence via DD while allowing the attacker to be in the system. This also naturally led to reducing false detection of intrusions. 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.

Even if DD strategies mainly aim to mislead the attacker's perception and make them poor attack decisions, they also introduce a positive effect to the system security by increasing TPR of the NIDS due to the benefit of the attack intelligence collected through the DD strategies. This implies that DD strategies play key roles in both defending against attackers and protecting existing assets by increasing intrusion detection.

In a hypergame of imperfect information where DD strategies are used, even under nodes with high-security vulnerabilities, we observed high resilience in maintaining high MTTSF and high TPR of the NIDS. This is mainly interpreted as the benefit of using DD under uncertainty which can effectively manipulate the attacker's perception towards a defender's move and the system.

We observed that the HNE hitting ratio is lower under hypergames with imperfect information than under games with perfect information due to the subjectively perceived uncertainty by the attacker and defender. When the defender uses DD techniques, it allows both players to monitor each other for a longer period of time than the setting with no DD strategies, leading to a higher HNE hitting ratio.

This work brings up the following future research directions: (1) considering multiple attackers arriving in a system simultaneously in order to consider more realistic scenarios; (2) developing a strategy selection method that employs resources by each player in a distributed manner to maximize its respective reward; (3) estimating each player's belief based on machine learning in order to more correctly predict a next move of its opponent; (4) dynamically adjusting a risk threshold, i.e., Eq. (14), depending on a system's security state; (5) introducing a recovery mechanism to restore a compromised node to a healthy node allowing the recovery delay; (6) 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 (7) considering another intrusion prevention mechanism, such as moving target defense, as one of the defense strategies.

ACKNOWLEDGMENTS

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] W. L. Sharp, "Military deception," Joint War-Fighting Center, Doctrine and Education Group, Norfolk, VA, Pub. 3-13.4, 2006.

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

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

[4] 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.

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

[6] Wikipedia. (2018) Foureye butterflyfish. Accessed: 10-01-2020. [Online]. Available: https://en.wikipedia.org/wiki/Foureye_butterflyfish

[7] 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.

[8] 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.

[9] Y. Sasaki, N. Kobayashi, and K. Kijima, "Mixed extension of hypergames and its applications to inspection games," in *Proc. 51st Annual Meeting of the ISSS-2007, Tokyo, Japan*, 2007.

[10] Y. Sasaki, "Preservation of misperceptions–stability analysis of hypergames," in *Proc. 52nd Annual Meeting of the ISSS-2008, Madison, Wisconsin*, 2008.

[11] K. Kijima, "Intelligent poly-agent learning model and its application," *Information and Systems Engineering*, vol. 2, pp. 47–61, 1996.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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. on Autonomous Agents and Multiagent Systems*, 2018, pp. 892–900.

[17] L. Xiao, D. Xu, N. B. Mandayam, and H. V. Poor, "Attacker-centric view of a detection game against advanced persistent threats," *IEEE Trans. on Mobile Computing*, vol. 17, no. 11, pp. 2512–2523, 2018.

[18] X. Fang, L. Zhai, Z. Jia, and W. Bai, "A game model for predicting the attack path of APT," in *Proc. 2014 IEEE 12th Int'l Conf. on Dependable, Autonomic and Secure Computing*, 2014, pp. 491–495.

[19] J. Pawlick, S. Farhang, and Q. Zhu, "Flip the cloud: Cyber-physical signaling games in the presence of advanced persistent threats," in *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2015, pp. 289–308.

[20] 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.

[21] ——, "Metagames and hypergames for deception-robust control," *ACM Trans. on Cyber-Physical Systems*, vol. 5, no. 3, pp. 1–25, 2021.

[22] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[23] L. Zhang and V. L. Thing, "Three decades of deception techniques in active cyber defense-retrospect and

outlook," *Computers & Security*, p. 102288, 2021.
[24] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Comms. Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.
[25] D. Ye, T. Zhu, S. Shen, and W. Zhou, "A differentially private game theoretic approach for deceiving cyber adversaries," *IEEE Trans. on Information Forensics and Security*, vol. 16, pp. 569–584, 2021.
[26] S. Tadelis, *Game Theory*. Princeton University Press, 2013.
[27] 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.
[28] R. Vane, "Planning for terrorist-caused emergencies," in *Proc. Winter Simulation Conf.*, Dec. 2005.
[29] 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.
[30] 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.
[31] ——, "Evolution of players' misperceptions in hypergames under perfect observations," *IEEE Trans. Automatic Control*, vol. 57, no. 7, pp. 1627–1640, Jul. 2012.
[32] Y. Sasaki, "Subjective rationalizability in hypergames," *Advances in Decision Sciences*, vol. 2014, p. 7 pages, 2014.
[33] 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.
[34] 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.
[35] 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.
[36] C. Bakker, A. Bhattacharya, S. Chatterjee, and D. L. Vrabie, "Metagames and hypergames for deception-robust control," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 3, Mar. 2021.
[37] N. M. Fraser and K. W. Hipel, *Conflict Analysis: Models and Resolutions*. North-Holland, 1984.
[38] 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*, 2019, pp. 1–8.
[39] C. N. Gutierrez, M. H. Almeshekah, J. Avery, S. Bagchi, and E. H. Spafford, "Modeling deception in information security as a hypergame: A primer," in *Proc. 16th Annual Information Security Symp.*, ser. CERIAS. West Lafayette, IN: CERIAS - Purdue University, 2015, pp. 41:1–41:1.
[40] R. Vane, "Advances in hypergame theory," in *Proc. AAMAS Workshop on Game-Theoretic and Decision Theoretic Agents*, 2006.
[41] ——, *Hypergame theory for DTGT agents*. AAAI, 2000.
[42] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques," *The Morgan Kaufmann Series in Data Management Systems*, p. 105, 2011, Section 3.5.2.
[43] 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 *Proc. 2015 27th Int'l Teletraffic Congress*, Sep. 2015, pp. 219–227.
[44] 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.
[45] U. Brandes, "A faster algorithm for betweenness centrality," *Jour. of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.
[46] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, 2010.
[47] Z. Wan, Y. Mahajan, B. W. Kang, T. J. Moore, and J.-H. Cho, "A survey on centrality metrics and their network resilience analysis," *IEEE Access*, pp. 1–1, 2021.
[48] CVSS, "Common vulnerability scoring system," accessed: 07-18-2020. [Online]. Available: https://www.first.org/cvss/
[49] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN controllers," in *Proc. 2015 1st IEEE Conf. on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–5.
[50] K. Raghunath and P. Krishnan, "Towards a secure SDN architecture," in *Proc. 2018 9th Int'l Conf. on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018, pp. 1–7.
[51] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions, volume 2*. John Wiley & Sons, 1995, vol. 289.
[52] 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.
[53] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Jour. of Information Security and Applications*, vol. 22, pp. 113–122, 2015.
[54] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017.
[55] Y. Liu, Y. Kuang, Y. Xiao, and G. Xu, "SDN-based data transfer security for Internet of Things," *IEEE Internet of Things Jour.*, vol. 5, no. 1, pp. 257–268, Feb. 2018.
[56] S. Prabakaran and R. Ramar, "Stateful firewall-enabled software-defined network with distributed controllers: A network performance study," *Int'l Jour. of Communication Systems*, vol. 32, no. 17, p. e4237, 2019.
[57] 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 *Proc. 2018 IEEE European Symp. on Security and Privacy Workshops (EuroS PW)*, Apr. 2018, pp. 29–39.