Advances in Information Security 89

Tiffany Bao
Milind Tambe
Cliff Wang *Editors*

Cyber Deception

Techniques, Strategies, and Human Aspects

Springer

Advances in Information Security

Volume 89

The purpose of the *Advances in Information Security* book series is to establish the state of the art and set the course for future research in information security. The scope of this series includes not only all aspects of computer, network security, and cryptography, but related areas, such as fault tolerance and software assurance. The series serves as a central source of reference for information security research and developments. The series aims to publish thorough and cohesive overviews on specific topics in Information Security, as well as works that are larger in scope than survey articles and that will contain more detailed background information. The series also provides a single point of coverage of advanced and timely topics and a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook.

Tiffany Bao • Milind Tambe • Cliff Wang
Editors

Cyber Deception

Techniques, Strategies, and Human Aspects

Springer

Editors
Tiffany Bao
Arizona State University
Tempe, AZ, USA

Milind Tambe
Harvard University
Cambridge, MA, USA

Cliff Wang
Army Research Office
Adelphi, MD, USA

ISSN 1568-2633 ISSN 2512-2193 (electronic)
Advances in Information Security
ISBN 978-3-031-16612-9 ISBN 978-3-031-16613-6 (eBook)
https://doi.org/10.1007/978-3-031-16613-6

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book introduces cutting-edge research works in cyber deception, a research topic that has been actively studied and significantly advanced in the past decade. With the focus on cyber deception, this book spans a wide variety of areas, including game theory, artificial intelligence, cognitive science, and cybersecurity. This book will address three core cyber deception research elements as follows:

1. Understanding of human's cognitive behaviors in decoyed network scenarios
2. Development of effective deceptive strategies based on human behaviors
3. Design of deceptive techniques enforcing deceptive strategies

The research introduced in this book will identify the scientific challenges, highlight the complexity, and inspire future research of cyber deception.

This book can be used as a professional book by cybersecurity practitioners and researchers, or a supplemental textbook for educational purposes. Readers will be able to learn the state of the art in cyber deception to conduct follow-up research or translate related research outcome to practice.

Tempe, AZ, USA — Tiffany Bao
Cambridge, MA, USA — Milind Tambe
Triangle Park, NC, USA — Cliff Wang

Acknowledgments

We would like to thank all the contributors for their dedication to this book. Special thanks go to Ms. Susan Lagerstrom-Fife and Ms. Shanthini Kamaraj for their kind support of this book. Finally, we thank the Army Research Office for their financial support under the grant numbers W911NF-17-1-0370.

Contents

Diversifying Deception: Game-Theoretic Models for Two-Sided Deception and Initial Human Studies

Mohammad Sujan Miah, Palvi Aggarwal, Marcus Gutierrez, Omkar Thakoor, Yinuo Du, Oscar Veliz, Kuldeep Singh, Christopher Kiekintveld, and Cleotilde Gonzalez

1 Introduction

Both civilian and military computer networks are under increasing threat from cyberattacks, with the most significant threat posed by Advanced Persistent Threat (APT) actors. These attackers use sophisticated methods to compromise networks and remain inside, establishing greater control and staying for long periods to gather valuable data and intelligence. These attackers seek to remain undetected, and estimates from APT attacks show that they are often present in a network for months before they are detected [31].

Cyber deception methods use deceptive decoy objects like fake hosts (honeypots), network traffic, files, and even user accounts to counter attackers in a variety of ways [1, 13, 28]. They can create confusion for attackers, make them more hesitant and less effective in executing further attacks, and can help to gather

An earlier version of some parts of the work was published in the proceedings of the 53rd Hawaii International Conference on System Sciences (HICSS), 2020, pp. 01–20.

M. S. Miah · M. Gutierrez · O. Veliz · C. Kiekintveld (✉)
Department of Computer Science, University of Texas at El Paso, El Paso, TX, USA
e-mail: msmiah@miners.utep.edu; mgutierrez22@miners.utep.edu; osveliz@utep.edu; cdkiekintveld@utep.edu

P. Aggarwal · Y. Du · K. Singh · C. Gonzalez
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: palvia@andrew.cmu.edu; yinuod@andrew.cmu.edu; kuldeep2g@andrew.cmu.edu; coty@cmu.edu

O. Thakoor
University of Southern California, Los Angeles, CA, USA
e-mail: othakoor@usc.edu

T. Bao et al. (eds.), *Cyber Deception*, Advances in Information Security 89,
https://doi.org/10.1007/978-3-031-16613-6_1

information about the behavior and tools of various attackers. They can also increase the ability of defenders to detect malicious activity and actors in the network. This deception is especially critical in the case of APT attackers, who are often cautious and skilled at evading detection [32]. Widespread and effective use of honeypots and other deceptive objects is a promising approach for combating this class of attackers.

However, the effectiveness of honeypots and other deceptive objects depends crucially on whether the honeypot creators can design them to look similar enough to real objects, to prevent honeypot detection and avoidance. This design goal especially holds for APT threats, which are likely to be aware of the use of such deception technologies and will actively seek to identify and avoid honeypots, and other deceptive objects, in their reconnaissance [32, 35]. A well-known problem with designing successful honeypots is that they often have characteristics that can be observed by an attacker that will reveal the deception [14]. Examples of such characteristics include the patterns of network traffic to a honeypot, the response times to queries, or the configuration of services which are not similar to real hosts in the network. However, with some additional effort, these characteristics can be made more effective in deception (e.g., by simulating more realistic traffic to and from honeypots).

In this chapter, we introduce a game-theoretic model of the problem of designing effective decoy objects that can fool even a sophisticated attacker. In our model, real and fake objects may naturally have different distributions of characteristic features than an attacker could use to tell them apart. However, the defender can make some (costly) modifications to *either* the real or the fake objects to make them harder to distinguish. This model captures some key aspects of cyber deception that are missing from other game-theoretic models. In particular, we focus on whether the defender can design convincing decoy objects, and what the limitations of deception are if some discriminating features of real and fake objects are not easily maskable.

We also present several analyses of fundamental questions in cyber deception based on our model. We analyze how to measure the informativeness of the signals in our model and then consider how effectively the defender can modify the features to improve the effectiveness of deception in various settings. We show how different variations in the costs of modifying the features can have a significant impact on the effects of deception. We also consider the differences between modifying only the features of deceptive objects and being able to modify both real and deceptive objects (two-sided deception). While this is not always necessary, in some cases, it is essential to enable effective deception. We also consider deception against naïve attackers, and how this compares to the case of sophisticated attackers.

Next, we present an exploratory study that looked into the effectiveness of a two-sided deception technique using a human-attackers trial. In the experiment, we used a network topology with an equal number of real machines and honeypots where we modify the features of a system using an experimental test bed (HackIT). We first categorized the adaptable features of both real machines and honeypots, then changed the features' characteristics and observed the attackers' behavior after modification. Finally, we discuss the results of our case study in no-deception, one-sided, and two-sided situations.

Later section in this chapter will discuss how our game model relates to work in adversarial learning and how this model could be applied beyond the case of honeypots to, for example, generating decoy network traffic.

2 Motivating Domain and Related Work

While the model we present may apply to many different types of deception and deceptive objects, we will focus on honeypots as a specific case to make our discussion more concrete and give an example of how this model captures essential features of real-world deception problems. Honeypots have had a considerable impact on cyber defense in the 30 years since they were first introduced [29].

Over time, honeypots have been used for many different purposes and have evolved to more sophisticated designs with more advanced abilities to mimic real hosts and to capture useful information about attackers [5, 18, 20]. The sophistication of honeypots can vary dramatically, from limited low-interaction honeypots to sophisticated high-interaction honeypots [9, 18, 22].

Here, we do not focus on the technological advancements of honeypots, but rather on the game-theoretic investigation of honeypot deception. There have been numerous works that emphasize this game-theoretic approach to cyber deception as well. Our work builds upon the Honeypot Selection Game (HSG), described by Píbil et al. [13, 21]. Much like the HSG, we model the game using an extensive form game. We extend the HSG model with the introduction of *features*, which are modifiable tokens in each host that enable more robust deceptions and allow to model more realistic settings. Several game-theoretic models have been established for other cyber defense problems [4, 17, 24, 26], specifically for deception as well [25, 33]; however, these consider attribute obfuscation as the means of deception rather than use of decoy objects.

Reference [34] notably investigate the use of honeypots in the smart grid to mitigate denial-of-service attacks through the lens of Bayesian games. Reference [16] also model honeypots mitigating denial-of-service attacks in a similar fashion but in the Internet-of-Things domain. Reference [8] tackle a similar "honeypots to protect social networks against DDoS attacks" problem with Bayesian game modeling. These works demonstrate the broad domains where honeypots can aid. This work differs in that we do not model a Bayesian incomplete information game.

A couple of works also consider the notion of two-sided deception, where the defender deploys not only *real*-looking honeypots but also *fake*-looking real hosts. Rowe et al. demonstrate that using two-sided deception offers an improved defense by scaring off attackers [23]. Caroll and Grosu introduced the signaling deception game where signals bolster a deployed honeypot's deception [6]. Our work differs in that we define specific features (signals) that can be altered and revealed to the attacker. Shi et al. introduce the mimicry honeypot framework, which combines real nodes, honeypots, and *fake*-looking honeypots to derive equilibria strategies to bolster defenses [27]. They validated their work in a simulated network. This notion

of two-sided deception is quickly becoming a reality; De Gaspari et al. provided a prototype proof-of-concept system where production systems also engaged in active deception [7].

3 Feature Selection Game

Feature Selection Game (FSG) models the optimal decisions for a player (the defender) who is trying to disguise the identity of real and fake objects so that the other player (the attacker) is not able to reliably distinguish between them. Each object in the game is associated with a vector of observable features (characteristics) that provides an informative signal that the attacker can use to detect fake objects more reliably. The defender can make (limited) changes to these observable features, at a cost. Unlike many models of deception, this game model considers the possibility that the defender can make changes to both the real and fake objects; we refer to this as 2-sided deception.

The original feature vector is modeled as a move by nature in a Bayesian game. Real and fake objects have different probabilities of generating every possible feature vector. How useful the features are to the attacker depends on how similar the distributions for generating the feature vectors are; very similar distributions have little information while very different distributions may precisely reveal which objects are real or fake. The defender can observe the features and may choose to pay some cost to modify a subset of the features. The attacker observes this modified set of feature vectors and chooses which object to attack. The attacker receives a positive payoff if he selects a real object, and a negative one if he selects a honeypot.

To keep the initial model simple, we focus on binary feature vectors to represent the signals. We will also assume that the defender can modify a maximum of one feature. Both of these can be generalized in a straightforward way, at the cost of a larger and more complex model.

3.1 Formal Definition of Feature Selection Game

We now define the Feature Selection Game (FSG) formally by the tuple $G = (K^{\mathrm{r}}, K^{\mathrm{h}}, N, v^{\mathrm{r}}, v^{\mathrm{h}}, C^{\mathrm{r}}, C^{\mathrm{h}}, P^{\mathrm{r}}, P^{\mathrm{h}}, \tau, \chi)$.

- K^{r} denotes the set of real hosts and K^{h} denotes the set of honeypots. Altogether, we have the complete set of hosts $K = K^{\mathrm{r}} \cup K^{\mathrm{h}}$. We denote the cardinalities of these by $k = |K|$, $r = |K^{\mathrm{r}}|$, $h = |K^{\mathrm{h}}|$.
- $[n]$ is the set of features that describe any given host. The sequence of feature values of a host is referred to as its *configuration*. Thus, the set of different possible configurations is $\{0, 1\}^n$.
- v^{r}, v^{h} denote the importance values of the real hosts and honeypots, respectively.

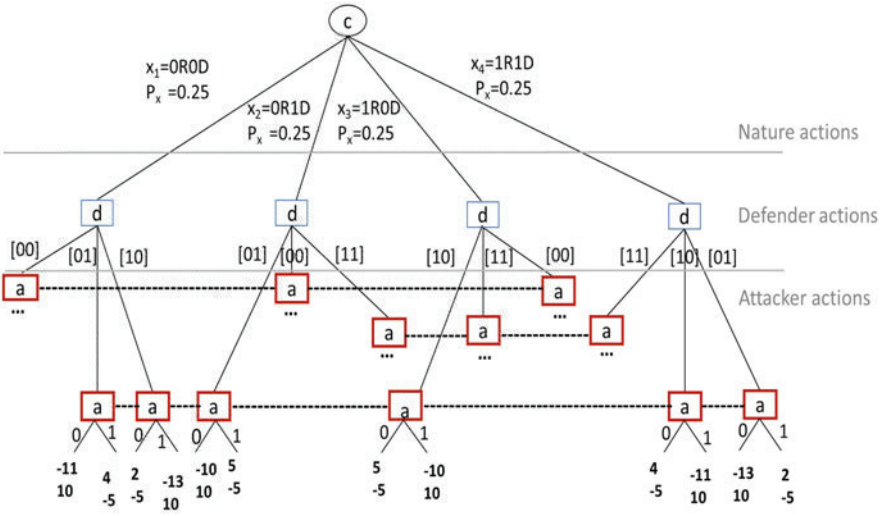

Fig. 1 The extensive form game tree with one real host, one honeypot and 1 feature in each host. The importance value of real host is 10, whereas the modification cost of a feature is 3. The same values for the honeypot are 5, 1, respectively

- C^{r}, C^{h} denote the cost vectors associated with modifying a single feature of a real host and a honeypot, respectively, and are indexed by the set of features N. Thus, C_i^{r} is the cost of modifying the ith feature of a real host.
- $P^{\mathrm{r}} : \{0, 1\}^n \rightarrow [0, 1]$ is probability distribution over feature vectors for real hosts.
- $P^{\mathrm{h}} : \{0, 1\}^n \rightarrow [0, 1]$ is the probability distribution over feature vectors for honeypots.
- The collection of all possible information sets is denoted by τ.
- $\chi : \{0, 1\}^{kn} \times D \rightarrow \tau$ is a function that given the initial network and a defender action, outputs the attacker's resultant information set $I \in \tau$. Here, D is the set of defender actions.

An example of a small FSG with 1 real host, 1 honeypot, and 1 feature for each host is shown in Fig. 1. The probability distributions $P^{\mathrm{r}}(0) = P^{\mathrm{r}}(1) = 0.5$ and $P^{\mathrm{h}}(0) = P^{\mathrm{h}}(1) = 0.5$ are randomly generated for each feature combination.

3.2 *Nature Player Actions*

We assume that both players know the probability distributions P^r and P^h that define how the feature vectors are selected by nature for real and honeypot hosts, respectively. Nature generates the network configurations as per the distributions P^{r} and P^{h}. Thus, the network state $x = (x_1, \ldots, x_k)$ is generated as per the joint

distribution P^{x} where $P^{\mathrm{x}}(x) = \prod_{i=1}^{r} P^{\mathrm{r}}(x_i) \times \prod_{i=r+1}^{k} P^{\mathrm{h}}(x_i)$. Both players can compute the distribution P^{x}. For example, in Fig. 1 $P^{\mathrm{x}} = 0.25$ for network 0R1D is calculated from $P^{\mathrm{r}}(0) = 0.5$ and $P^{\mathrm{h}}(1) = 0.5$. Here, 0R1D refers to the 1st feature's status of real host and decoy object (honeypot).

3.3 Defender Actions

The defender observes the network configuration $x \in X$, selected by nature as per probability distribution P^{x}. Then he chooses an appropriate action $d \in D$, which is to change at most one feature of any single host. Thus, D has $nk + 1$ different actions. This action results in a configuration $x' \in \{0, 1\}^{nk}$ that the attacker observes, defining his information set $I \in \tau$ as described previously. In the example of Fig. 1, given the initial network configuration 0R0D, the defender can alter a feature which results into 0R1D or 1R0D, or make no change leading to 0R0D as the attacker's observation.

3.4 Attacker Actions

The attacker observes the set of feature vectors for each network but does not directly know which ones are real and which are honeypot. Thus, any permutation of the host configurations is perceived identically by the attacker. Hence, the attacker's information set is merely characterized by the combination of the host configurations and thus represented as a multiset on the set of host configurations as the Universe. For example, in Fig. 1, the networks 0R1D and 1R0D belong to the same information set. Given the attacker's information set, he decides which host to attack. When indexing the attack options, we write the information set as an enumeration of the k host configurations, and we assume a lexicographically sorted order as a convention. Given this order, we use a binary variable a_i^I to indicate that when he is in the information set I, the attacker's action is to attack host $i \in K$.

3.5 Utility Functions

A terminal state t in the extensive form game tree is characterized by the sequence of actions that the players (nature, defender, attacker) take. The utilities of the players can be identified based on the terminal state that the game reaches. Thus, given a terminal state t as a tuple (x, j, a) of the player actions, we define a function $U(t) = U(x, j, a)$ such that the attacker gains this value while the defender loses as much. That is, this function serves as the zero-sum component of the player

rewards. In particular, if the action a in the information set $\chi(x, j)$ corresponds to a real host, then $U(x, j, a) = v^{\mathrm{r}}$, whereas if it corresponds to a honeypot, then $U(x, j, a) = -v^{\mathrm{h}}$. Intuitively, the successful identification of a real host gives a positive reward to the attacker otherwise gives a negative reward that is equal to the importance value of a honeypot. The expected rewards are computed by summing over the terminal states and considering the probabilities of reaching them. Finally, the defender additionally also incurs the feature modification cost C_i^{r} or C_j^{h} if his action involved modifying ith feature of a real host or jth feature of a honeypot, respectively.

3.6 Solution Approach

We solve this extensive form game with imperfect information using a linear program. For solving this game in sequence form [15], we create a path from the root node to the terminal node that is a valid sequence and consists of a list of actions for all players. Then we compute defender's behavioral strategies on all valid sequences using a formulated LP as follows, where U_d and U_a are the utilities of the defender and the attacker. To solve the program, we construct a matrix $X[0 : 2^{kn}]$ of all possible network configurations, and then the defender chooses a network $x \in X$ to modify. In network x, any action d of the defender leads to an information set I for the attacker. Different defender's actions in different networks can lead to the same information set $I \in \tau$. Then, in every information set I, the attacker chooses a best response action to maximize his expected utility.

$$max \sum_{x \epsilon X} \sum_{j \epsilon D} \sum_{i \epsilon K} U_d(x, j, i)\ d_j^x P^x a_i^{\chi(x,j)} \tag{1}$$

$$s.t. \sum_{(x,j):\chi(x,j)=I} U_a(x, j, i) d_j^x P^x\ a_i^I \geq \sum_{(x,j):\chi(x,j)=I} U_a(x, j, i')\ d_j^x P^x\ a_i^I$$

$$\forall i, i' \in K \quad \forall I \in \tau \tag{2}$$

$$d_j^x \geq 0 \quad \forall x \in X \quad \forall j \in D \tag{3}$$

$$\sum_{j \epsilon D} d_j^x = 1 \quad \forall x \in X \tag{4}$$

$$\sum_{i \epsilon K} a_i^I = 1 \quad \forall I \in \tau \tag{5}$$

The program's objective is to maximize the defender's expected utility, assuming that the attacker will also play a best response. In the above program, the only unknown variables are the defender's actions D (the strategies of a defender in a network $x \in X$) and the attacker's actions a^I. The inequality in Eq. 2 ensures that the attacker plays his best response in this game, setting the binary variable a_i^I to 1 only for the best response i in each information set. Equation 3 ensures that the defender strategies in a network x is a valid probability distribution. Equation 4 makes sure that all probability for all network configurations sum to 1. Finally, Eq. 5 ensures that the attacker plays pure strategies.

4 Empirical Study of FSG

The FSG game model allows us to study the strategic aspects of cyber deception against a sophisticated adversary who may be able to detect the deception using additional observations and analysis. In particular, we can evaluate the effectiveness of cyber deception under several different realistic assumptions about the costs and benefits of deception, as well as the abilities of the players. We identify cases where deception is highly beneficial, as well as some cases where deception has limited or no value. We also show that in some cases, using two-sided deception is critical to the effectiveness of deception methods.

4.1 Measuring the Similarity of Features

One of the key components of our model is that real hosts and honeypots generate observable features according to different probability distributions. The similarity of these distributions has a large effect on the strategies in the game, and the outcome of the game. Intuitively, if out-of-the-box honeypot solutions look indistinguishable from existing nodes on the network the deception will be effective without any additional intervention by the defender. However, when the distributions of features are very dissimilar the defender should pay higher costs to modify the features to disguise the honeypots. In some cases this may not be possible, and the attacker will always be able to distinguish the real hosts and honeypots.

Measuring the similarity of the feature distributions is a somewhat subtle issue, since the defender can make changes to a limited number of features. Standard approaches such as Manhattan distance or Euclidean distance do not provide a good way to compare the similarity due to these constraints. We use a measure based on the Earth Mover's Distance (EMD) [19], which can be seen as the minimum distance required to shift one pile of earth (probability distribution) to look like another. This measure can be constrained by the legal moves, so probability is only shifted between configurations that are reachable by the defender's ability to change features.

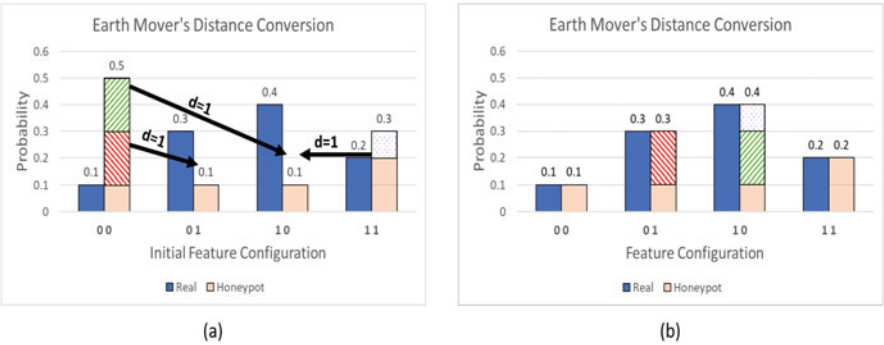

Fig. 2 Earth Mover's Distance process. (**a**) Displays the initial feature configuration probability distributions P_r and P_h and where to move slices of the distribution from P_h and (**b**) Shows the updated P_h after the conversion, resulting in a final EMD of 0.5

In the experiments, we allow the defender to modify only a single feature in the network and the EMD determines the minimum cost needed to transform a weighted set of features to another where the probability of each feature configuration is the weight. The ground dissimilarity between two distributions is calculated by the Hamming distance. This distance between two distributions of equal length is the number of positions at which the comparing features are dissimilar. In other words, it measures the minimum number of feature modification or unit change required to make two sets of feature indistinguishable. We model the distance from moving the probability of one configuration (e.g., turning $[0, 0]$ into $[0, 1]$) to another by flipping of a single bit at a time with a unit cost of 1. This can be seen visually in Fig. 2 where we calculate the EMD of moving the honeypot's initial distribution into that of the real node's initial distribution.

In our experiments we will often show the impact of varying levels of similarity in the feature distributions. We generated 1000 different initial distributions for the features using uniform random sampling. We then calculated the similarities using the constrained EMD and selected 100 distributions so that we have 10 distributions in each similarity interval. We randomly select these 10 for each interval from the ones that meet this similarity constraint in the original sample. This is necessary to balance the sample because random sampling produces many more distributions that are very similar than distributions that are further apart, and we need to ensure a sufficient sample size for different levels of similarity. we present the results by aggregating over the similarity intervals of 0.1 and average ten results in each interval.

4.2 *Deception with Symmetric Costs*

Our first experiment investigates the impact of varying the similarity of the feature distributions. We also vary the values of real host and honeypot. As the similarity of the distributions P^r and P^h decreases, we would expect a decrease in overall expected defender utility. We can see this decrease in Figs. 3a and b as we vary

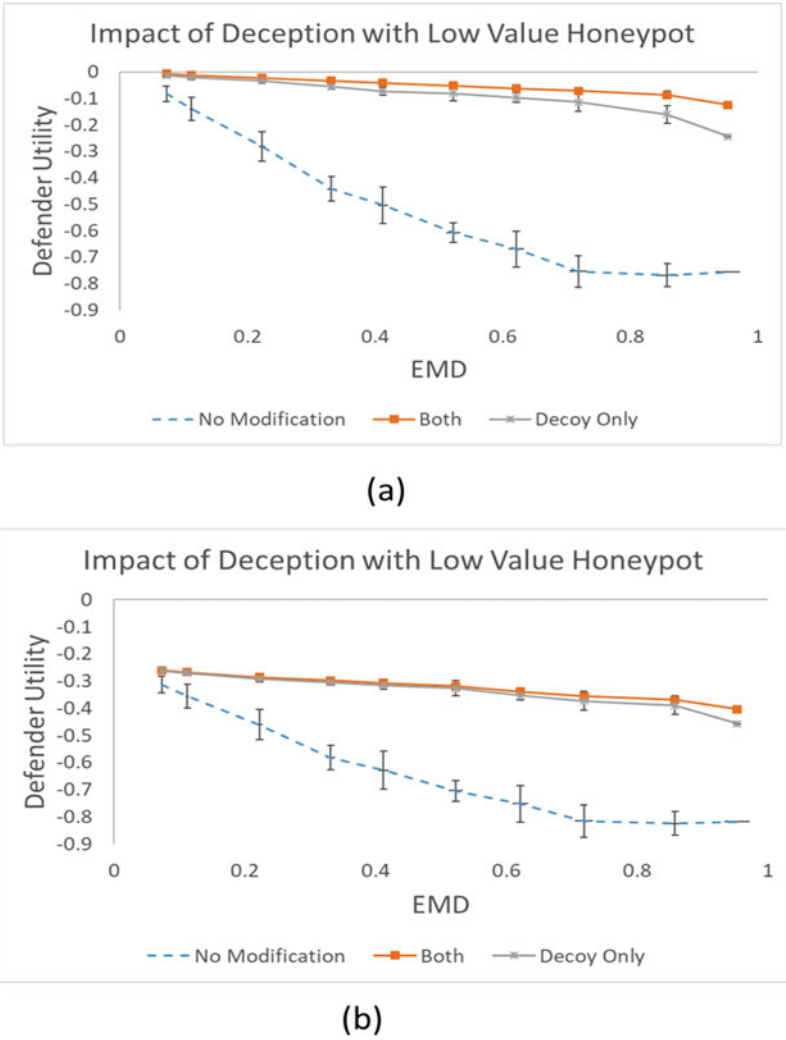

Fig. 3 Comparison of defender utility when the real host's importance value (**a**) doubles that of the honeypot and (**b**) equals that of the honeypot. Here we see one-sided deception provides a comparable defense despite a high initial dissimilarity

the similarity measured using EMD. In Figs. 3a and b, we compare the utility differences between an optimal defender that can only modify the features of the honeypot (one-sided deception), an optimal defender that can modify features of *both* the honeypot and real host (two-sided deception), and a baseline defender that cannot make any modifications against a fully rational best response attacker.

In Fig. 3a, the honeypot has the same importance value as the real host, while in Fig. 3b, the honeypot value is half of the real host. The first observation is that in both cases the value of deception is high relative to the baseline with no deception, and this value grows dramatically as the feature distributions become more informative (higher EMD). In general, the defender does worse in cases where the hosts have different values. Two-sided deception does have a small advantage in cases with highly informative features, but the effect is small. Here, the costs of modifying the features are symmetric, so there is little advantage in being able to modify the feature on either the honeypot or the real host, since the defender can choose between these options without any penalty.

To further investigate the issue of one-sided and two-sided deception, we fix the honeypot features modification costs and increased real host modification costs as reflected in Table 1. Here, we compare how increasing the real host's feature modification negatively affects the defender's expected utility. As the cost for modifying the real hosts increases relative to the cost of modifying honeypots, the defender must make more changes on honeypots in order to maximize his utility. Altering the real system in this case is not feasible and does not provide a good return on investment.

Traditionally network administrators avoid altering features in their real hosts on the network and simply employ one-sided deception, attempting to alter the honeypot to look like a real host. In the case where modifying a real host to look *less believable* might be too costly or even impossible, one-sided deception is an obvious

Table 1 Parameters used in HFSG experiments. RIV denotes real system's importance value, RMC denotes real system's feature modification cost, HpIV denotes importance value of honeypot, and HpMC denotes feature modification cost of honeypot. All numbers are normalized to 1

		RMC		HpMC		
Figure	RIV	F 1	F 2	F 1	F 2	HpIV
3a	1.0	0.25	0.1	0.1	0.25	0.5
3b	1.0	0.25	0.1	0.2	0.1	1.0
4 (Both (A))	1.0	0.25	0.1	0.1	0.2	0.5
4 (Both (B))	1.0	0.5	0.2	0.1	0.2	0.5
4 (Both (C))	1.0	1.0	0.5	0.1	0.2	0.5
5 (Exp-1)	1.0	0.1	∞	0.1	∞	1.0
5 (Exp-2)	1.0	0.1	∞	∞	0.1	1.0
6 (Exp-1)	1.0	0.2	0.2	0.2	0.2	1.0
6 (Exp-2)	1.0	0.15	0.25	0.25	0.15	1.0
6 (Exp-3)	1.0	0.1	0.3	0.3	0.1	1.0
6 (Exp-4)	1.0	0.05	0.35	0.35	0.05	1.0
6 (Exp-5)	1.0	0.0	0.4	0.4	0.0	1.0
8	1.0	0.25	0.1	0.2	0.1	1.0

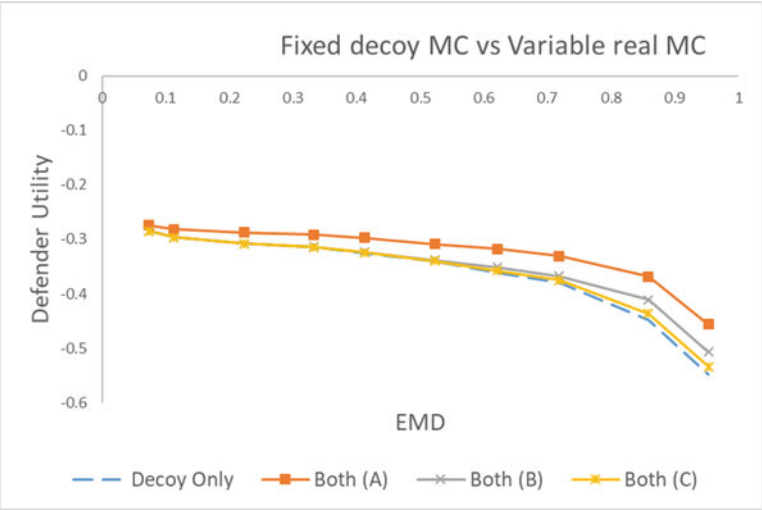

Fig. 4 Comparison of defender utility when the cost of modifying the real host features is different than modifying the honeypot features

choice as demonstrated in Fig. 4. However, when these real feature modifications are not too costly, we see that two-sided provides a noticeable increase in defenses when the feature distributions are increasingly dissimilar.

4.3 Deception with Asymmetric Costs

While the results so far have suggested that one-sided deception may be nearly as effective as two-sided deception, they have all focused on settings where the costs of modifying features are *symmetric* for real and fake hosts. We now investigate what happens when the costs of modifying different features are asymmetric. We start with the extreme case where some features may not be possible to modify at all.

In our examples with two features, we can set the unmodifiable features for the real and honeypot hosts to be the same or to be opposite. In Fig. 5, we show the results of the game when we set the modification costs of some features to infinity. If the same feature for the real host and honeypot are unmodifiable, then there is little the defender can do to deceive an intelligent attacker when they are highly dissimilar. However, when the features that cannot be modified are different for the real and honeypot hosts, we see a very different situation. In this case the defender benefits greatly from being able to use two-sided deception, since he can avoid the constraints by modifying either the real or fake hosts as needed.

In our next experiment, we investigate less extreme differences in the costs of modifying features. We set the costs so that they are increasingly different for

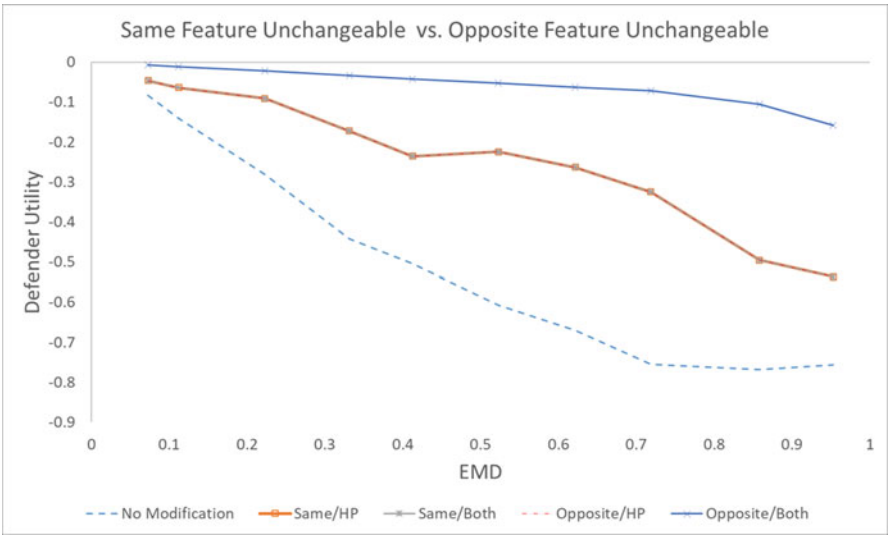

Fig. 5 Comparison of defender utility when some features cannot be modified

real and honeypot hosts, so modifying one feature is cheap for one but expensive for the other, but not impossible. We show the results of using either one or two-sided deception for varying levels of initial feature distribution similarity in Fig. 6. The specific costs are given in Table 1. We see that there is very little difference when the initial distributions are similar; this is intuitive since the attacker has little information and deception is not very valuable in these cases. However, we see a large difference when the initial distributions are informative. As the difference in the feature modification costs increases, the value of two-sided deception increases, indicating that this asymmetry is crucial to understanding when two-sided deception is necessary to employ effective deception tactics.

We also expect that the number of features available to the players will have a significant impact on the value of deception. While the current optimal solution algorithm does not scale well, we can evaluate the differences between small numbers of features, holding all else equal. Figure 7 presents the results of the modeling HFSG with variable number of features. We found that when the number of features is increased two-sided deception becomes more effective than one-sided deception. The defender in this case has more opportunity to alter the network by changing the features and make it the more confusing network to the attacker. However, the defender payoff decreases with more features due to the constraint on how many features he can modify and the total cost of modifying these features.

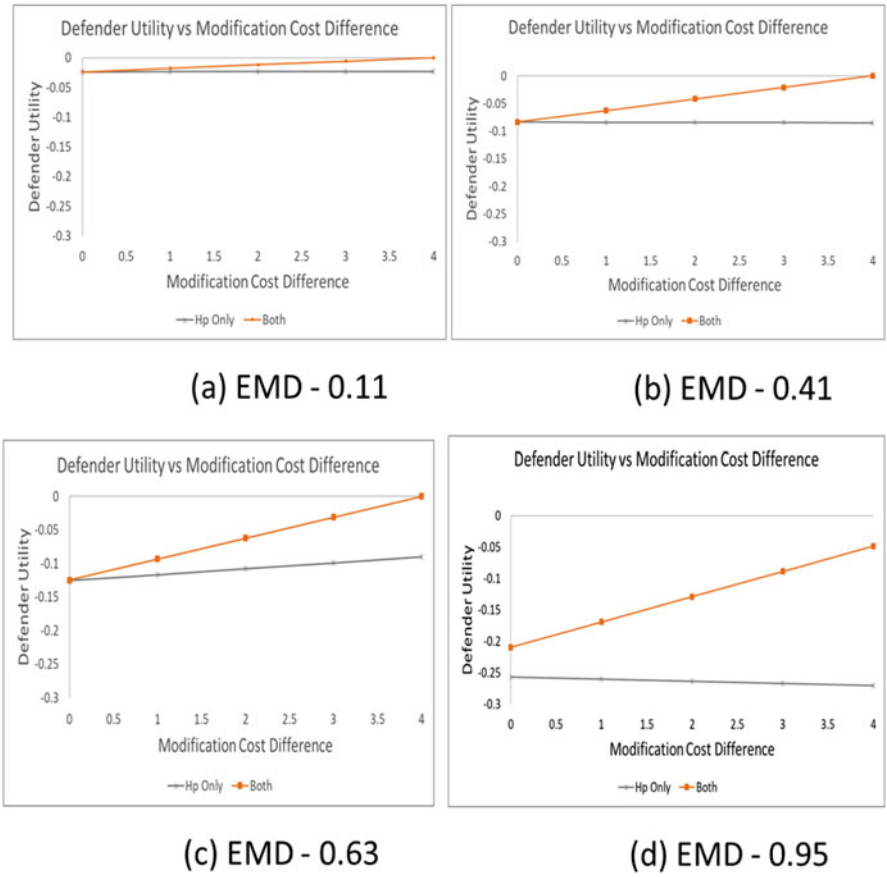

Fig. 6 Impact of modification cost over various initial similarity parameters

4.4 Deception with Naïve Attackers

The previous empirical results all assumed a cautiously rational attacker who actively avoided attacking honeypots. This is a common practice, because fully rational actors present the highest threat. In cybersecurity, these fully rational attackers might be an experienced hacker or APT. However, these are not the only threats faced in cybersecurity and we cannot assume that these attacking agents are always cautious and stealthy. For example, many attacks on networks may be conducted by worms or automated scripts that are much simpler and may be much more easily fooled by deceptive strategies.

We now consider a more naïve attacker that does not consider the defender's deception. He observes the hosts on the network and assumes no modifications were made. Based on all observations for a particular network he calculates his

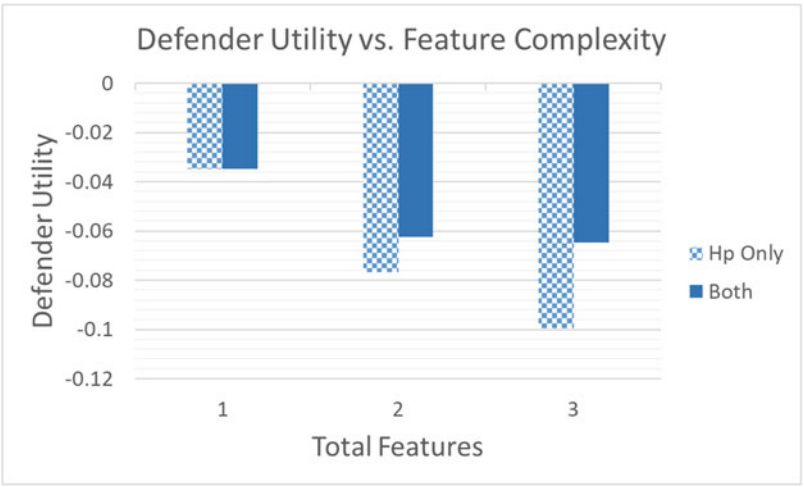

Fig. 7 Comparison of defender utility when increasing the number of features

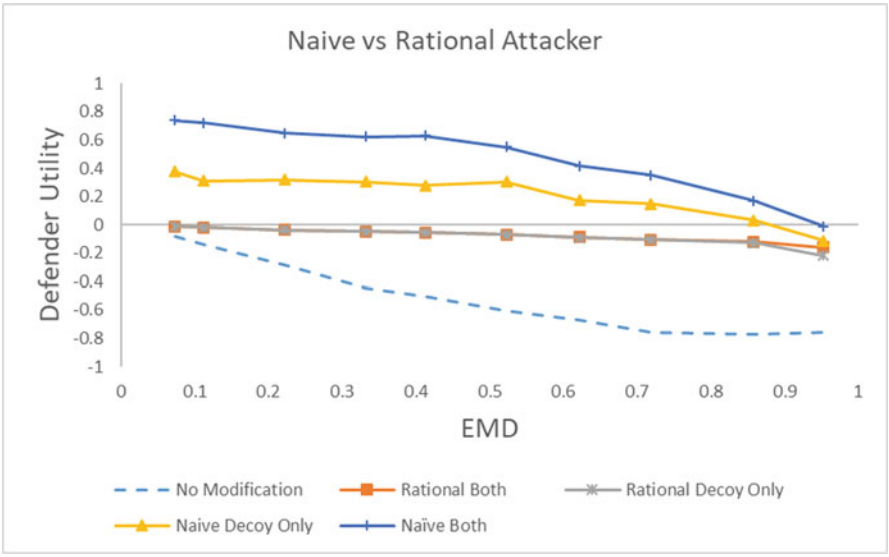

Fig. 8 Comparison of defender utility of a naïve attacker versus a fully rational attacker. Here, the naïve attacker does not consider the defender's utility or strategy at all

best response but does predict the defender's optimal strategy. The results of the experiment are shown in Fig. 8 and the costs given in Table 1.

The best case is when the defender can perform two-sided deception against a naïve attacker and the worst case is when the defender performs no deceptive actions against a fully rational attacker. These two cases form an upper- and lower-bound as

seen in Fig. 8. Two-sided deception is more effective in this case when the feature distributions are similar, while the opposite was true for a rational attacker. Overall, deception strategies are much more effective against naïve attackers.

5 Human Experiment

In this section, we explain the human experiment conducted using an experimental testbed HackIT [2]. HackIT provides various capabilities such as network creation with different sizes, configurable machine features, and manipulation of topologies for conducting human experiments. This experiment was conducted with a network size of 40 machines, including 20 honeypots and 20 real machines. These machines were used with different configurations to achieve deception based on the experimental conditions.

5.1 Experimental Design

In this experiment, we study the effectiveness of a two-sided deception strategy. We define two between-subject conditions: *default*, where both real machines and honeypots have default configurations; and *2-sided deception*, where the configuration of both real and honeypot is modified. The real machines were configured with the default honeypot configuration, and the honeypots were configured with the default real configuration shown in Table 2.

In each condition, there are two identical rounds of 25 min each. The proportion of honeypots (20 out of 40 machines) remains the same across conditions and rounds. The honeypots were allocated randomly in the network. The real and honeypots used in the study have multiple features as shown in Table 2. These features were configured based on the experimental conditions.

Table 2 The configuration of Real machines and Honeypots

Features	Default honeypot	Default real machine
OS	Obsolete OS	Up to date
Ports & services	Normal ports, honeypot ports	Normal ports
Exploit information	Obsolete	Up to date
Exploit success rate	100%	40%
Link latency	1 ms	0.2 ms
Virtual/physical machine	VM	Physical
Running processes	2 processes	10 processes
File system (1 user folder with 5∼6 sub-folders)	4∼5 empty or access-deny folders	1 empty or access-deny folder