Defensive Deception in Enterprise Network

Mu Zhu

TABLE OF CONTENTS

Abstract

Defensive deception attracts much attention from research communities. Unlike traditional cybersecurity techniques, such as firewall and intrusion detection systems (IDS), defensive deception lets the defender participate in attack processes to lure and mislead the opponents' perspective. Therefore, defensive deception shows effectiveness when detecting or mitigating stealthy attacks, such as malicious reconnaissance and insider threats.

To better understand how defensive deception works, we first investigate different deception techniques and analyze their advantages and disadvantages. We survey current game theory (GT) and machine learning (ML) based defensive deception research to identify (1) what GT and ML are used, (2) what kinds of attack are discussed, (3) what defensive deception technique is proposed, and (4) how the authors evaluate their schemes.

As a popular defensive deception technique, honeyfile systems are commonly used to detect stealth threats, such as insider attacks. Although deployed honeyfiles can help uncover attackers, they can disrupt the work of legitimate users and generate false alarms, which may confuse administrators. To fill this gap, we propose a novel honeyfile system, named Mee, that relies on decentralized deployment and centralized control. In recent work, we have developed the Mee structure to model interactions between the attacker, the defender, and the users with a Bayesian game to assist the defender in producing an optimal strategy. We implement Mee in the testbed and evaluate it by considering 1) the defender payoff, 2) attackers payoff, and 3) true/false positive rate when detecting insider threats. The simulation results show that Mee has better performance than the traditional honeyfile system and disturbs regular users less.

In our first proposed work, we will expand the above research to use deep reinforcement learning (DRL) and adopt a more complicated environment (e.g., multiple users and attackers involved) to increase the realization. Our research objective is to increase the effectiveness of honeyfile and reduce false positive alarms from regular users.

In our second proposed work, we will investigate an approach to generate honey traffic, which represents fake network flows to defend against malicious reconnaissance (e.g., such as packets sniffer and banner grabbing). This research adopts the Generative Adversarial Network (GAN) architecture, which includes generator and discriminator components. We use the generator to learn the features of network flows that come from actual servers and produce fake flows. In addition, we use the discriminator to mimic an attacker, which needs to distinguish legitimate flows from fake flows.

Overall, this dissertation seeks to improve defensive deception by investigating reliable and practical designs.

Chapter 1

Introduction

Defensive deception research has grown from solving cybersecurity problems. Leverages on the deception thinking, the defender can involve attack processes by luring, misleading, or confusing the adversaries. Due to this nature, defensive deception creates a novel perspective to detect and mitigate stealthy threats, such as APTs and insider attacks. In this chapter, we first introduce the threats and deception techniques which we mainly concentrate on. Then, we discuss our research motivation and summarize our current processes.

1.1 Problem: Cyber Threats

Advanced Persistent Threat (APT)

An advanced persistent threat (APT) symbolizes well-trained attackers who perform multiple-year threats to exfiltrate valuable and sensitive economic, proprietary, or national security information. Currently, the number of APTs increases rapidly and challenges network or system security across the world [Bandla, 2019]. Therefore, APTs draw increasing attention from academics. Researchers [Chen et al., 2014] define six stages of an APTs' cyber kill chain:

- Reconnaissance: In this stage, adversaries gather information, such as users' email addresses and social relationships. With this information, the attacker identifies potential victims within the targeted organization.
- Delivery: The adversaries deploy their exploits to the targets in this stage. Two methods are commonly applied: 1) *Spear Phishing*, which represents the email carries malicious attachments or links; 2) *Watering Hole Attack*, which leverages third party application or website that the victim visits frequently to delivery malware. APT attackers typically use the information gathered during reconnaissance to make the attack more specific and personal to the victim.
- Initial intrusion: In this stage, APT attackers penetrate the target device and executes exploits. Some attackers can gather authorization through previous stages and access the device as legitimate users. This stage represents that the adversary establishes footholds, such as backdoor, in the target network.

- Command and control (C2): C2 mechanisms leverage controlled servers by the attacker to send commands to the compromised systems and receive data from a target network.
- Lateral movement: After successfully creating the connection between the target network and the C2 server, the attacker infiltrates the network and expands its control across the entire organization. This stage usually includes several objectives: 1) implement internal reconnaissance, 2) harvest additional credentials and compromise more systems, and 3) identify valuable data or assets.
- Data exfiltration: This stage involves collecting, encrypting, and extracting the stolen data from the victim environment.

Insider Attack

Salem et al. [2008] divide insider threats into two types: *traitors* and *masqueraders*.

Traitors are legitimate users within an organization who leverage their legal authorization to harvest and leak sensitive information. Further, using the legitimate status with the authorization of a system can perform attacks (e.g., illegal access, modification of confidential information) that can lead to loss of integrity and availability.

Masqueraders represent the attacker who gathers legitimate users' credentials to infiltrate the target environment and steal valuable data from the victims' devices. Note that APT attackers can become masqueraders after harvesting enough authorizations or create the backdoor in victim devices.

These two types of insider threats have different perspectives about the victims. Masqueraders can penetrate devices but are unfamiliar with the victim system, such as the file space. This character brings the opportunity to the defender to detect the attacker's behaviors. For example, a masquerader may have a higher probability of touching honeyfiles than the device owner.

Passive Monitoring

Whether passive or active, lawful or malicious, reconnaissance remains one of the most critical parts of any strategic cybersecurity operation. As a critical part of malicious reconnaissance, *Passive monitoring or scanning* mainly focuses on searching services on a network via observing traffic generated by servers and clients when it passes an observation point. Passive monitoring is difficult to detect and consumes fewer network resources than active probing. Therefore, it can perform on a long-term basis as part of regular operation, result in it can better detect active services running on transient hosts. The disadvantage of passive scanning is that it can only detect active services. In addition, it can not identify services that do not run on well-known ports. The popular passive monitoring methods includes *banner grabbing* and packet sniffing.

Banner grabbing is informative to network administrators, as well as the remote attackers. Adversary applies banner grabbing to reveal compromising information about the services that are running on a system.

Packet sniffing is the process of gathering traffic away from a network by capturing the data as they pass and store them for later analysis. A packet sniffer can reassemble the network packets by

capturing this network communication to view the information originally sent over the network. Any data sent in plaintexts, such as user names, passwords, IP addresses, and other sensitive data, are vulnerable to eavesdropping.

The research from [Montigny-Leboeuf and Massicotte, 2004] introduces the information (e.g., active nodes, OSs, and IP network configuration) that can be obtained through passive monitoring and describes how passive network discovery and persistent monitoring provide meaningful contextual information.

1.2 Defensive Deception

Deception has been applied broadly and with a variety of meanings. In many areas, such as the military, criminology, and economics, deception has played an important role. Generally, deception in the adversarial interaction is a strategy in which one party produces misinformation to confuse the other.

For cybersecurity, both adversaries and defenders can perform deception to mislead the opponents. Malicious deception, includes phishing emails, forged identities, and man-in-the-middle attacks, has been used as an attacker's strategy to deceive a defense system. For example, an attacker can spread phishing links to harvest legal authorization from victims, or the adversary pretends to be a legitimate user to infiltrate a system. Specifically, malicious deception is commonly seen in the reconnaissance and exploits of long-term attacks, such as advanced persistent threats (APTs).

Our research concentrates on defensive deception, representing the defender's leverage of misinformation to protect a given system against attackers. Through defensive deception, the defender can change the display of tangible assets to hide valuable information or expose the attacker's movements. Although attackers can perform strategies (e.g., reconnaissance) and uncover the deception, the deception slows down the attack processes and increase the attacker's payoff. Besides, the defender can lure attackers via mock objects exposed to adversaries and draw attackers' attention. Honeypots and honeyfiles are popular luring-based deception methods. The objective of luring is to waste adversaries' efforts and detect attackers' behaviors. These deception techniques can also monitor the adversaries to know attackers' interests. Based on the above discussion, the main objective of defensive deception is to intentionally produce misinformation to make adversaries acquire or continue to obtain a false belief rather than the actual perspective of the system.

Advantages of Defensive Deception:

- Defensive deception is a cost-effective security mechanism. Unlike some traditional cybersecurity techniques, such as firewall and IDS, defensive deception is relatively simple to deploy and maintain.
- Well-design defensive deception mechanisms provide a high accuracy rate to detect malicious behaviors. For example, the firewall or IDS may generate many false positive alarms while the alerts from honeypots represent a high probability of threat detection.
- Defensive deception shows effectiveness when against stealthy attacks. Specifically, traditional cybersecurity methods are insufficient to detect the hidden threats, such as malicious reconnaissance and insider threats. However, deception methods (e.g., honeyfile and honeypot) provide novel angels to observe attacker movement.

- Defensive deception provides additional defense services to supplement traditional security mechanisms. For example, honeypots can incorporate IDS and other monitoring mechanisms to enhance intrusion detection and understand attack features to protect a system from intrusions before launching attacks to actual targets.
- The variety of defensive deception methods can be applied at multiple layers of systems, such as network systems, application layers, and data layers. High deployability of defensive deception techniques also provides flexibility and adaptiveness with multiple layers of protection.

In this proposal, we contribute to the improvement of honeyfile systems and enhance high-interaction honeypots.

1.2.1 Honeypot

As a most popular defensive deception technique, honeypots show promising for luring, misleading, and detecting threats. The successfully deployed honeypots can distract attackers' attention from tangible assets and provide early alerts about new attacks and exploitation. Leverages on honeypots, the defender can expose the malicious actions, and in-depth examine adversaries during and after exploiting a honeypot. Honeypots can be classified based on the level of interaction.

Low-interaction honeypots emulate a limited amount of internet protocols and network services. For example, low-interaction honeypots can simulate protocols, such as TCP and IP, which deceive the attacker into trusting the connecting system is actual instead of a honeypot environment. The advantages of low-interaction honeypots include that they are simple to deploy and maintain. However, they only simulate limited information and are insufficient to confuse well-trained attackers or detect sophisticated threats, such as zero-day attacks.

High-interaction honeypots use the actual vulnerable service or software to lure attackers. Unlike low-level honeypots, the high-interaction honeypots provide much information to mimic the existing systems. They can assist the defender in identifying unknown vulnerabilities and detect zero-day attacks. But high-interaction honeypots occupy more resources of maintenance

1.2.2 Honeyfile

Honeyfiles, or decoy documents, symbolize fake files which use to mislead attackers' perspectives. Compare with other deception techniques (e.g., honeypots), the advantages of honeyfiles are lightweight, simple deployment, and easy maintenance. The main approaches of honeyfile research include 1) honeyfile generation and 2) honeyfile placement and amount. The first approach includes natural language processing to discuss how the system should generate appropriate honeyfile content. The second approach focuses on discussing the fair number of honeyfiles and other characters of honeyfiles to increase the attraction, such as placement and name.

1.3 Challenges in Defensive Deception

Although defensive deception confuses adversaries to slow down or mislead the malicious processes, it has disadvantages that inspire us to improve the deception techniques. Specifically, some defensive deception

methods also disturb legitimate users or administrators. For example, regular users may accidentally touch honeyfiles to cause false positive alarms, disturbing users and confusing the defender to correctly assess the system security situation. In addition, because of the nature of deception, the effectiveness of deception techniques requires continual configuration, reconfiguration, and implementation process. Otherwise, adversaries would easily distinguish deceptive devices unless deception is automated, like software obfuscating techniques. This character brings the unnecessary cost. For example, although a high-interaction honeypot provides much helpful information about the adversary when it successfully lures the attacker, it requires resources to maintain and increase the cost of a security system when there is no attacker. Furthermore, with well-trained attackers, there are growing demands for elaborated deception strategies (e.g., high-interactive honeypots) that attackers cannot simply identify. However, doing so brings a higher defense cost and more difficulty managing a defense system than a traditional system without deception mechanisms.

1.3.1 Research Questions

Due to the benefits and challenges of defensive deception, improving deception techniques is necessary and meaningful. The objectives of this research include the understanding of current defensive deception and enhancement of some deception techniques. Accordingly, we identify the following research questions:

Question 1: *how should the defender increase the deception attraction to the attacker?*

The main objective of defensive deception is to leverage false information to attract the attacker's attention from tangible assets. Therefore, increasing the deception attraction is the first target we need to address. This research question is challenging because it is difficult to understand the fundamental interests of an attacker.

Question 2: *how should the defender effectively allocate resources?*

Although defensive deception is a resource-effective security mechanism, the redundant deployment increases the defender's payoff and interrupts the system's normal processes. This task is challenging because efficacious deception techniques require the defender to successfully predict attackers' movement or behavior.

Question 3: *how should the defender reduce the impact from deception methods on the legitimate users?*

Some defensive deception (e.g., honeyfile) can confuse attackers but also disturb legitimate users. However, most of the deception research only discusses the interaction between attackers and the defender. We address this task by considering the regular users in the scenario and involve false positive rates as one of the metrics in our evaluation.

1.4 Summary of Approach and Results

Mee: Adaptive Honeyfile System

Abundant honeyfiles can remarkably increase the effectiveness when detecting insider threats. However, redundant honeyfiles increase the defender's payoff and disturb legitimate users. Many false-positive

alarms also confuse the defender to assess the system's security situation correctly. Therefore, how to assist the defender in deciding a fair number of honeyfiles is meaningful. Although current honeyfile research involves discussing how the number of honeyfiles influences the deception effectiveness, they only consider the static number of honeyfiles in a single device which is difficult to provide enough observation to defenders for adjusting the defense strategies.

To fill this gap, we contribute to improving honeyfile systems by considering multiple devices in the environment to assist the defender in better understanding the network security situation. Specifically, we propose a novel honeyfile system named Mee, which leverages centralized control and decentralized deployment to adjust the number of honeyfiles in the connected device rely on the security assessment across the network. Unlike some current deception research that discusses the interaction between attackers and defenders, we also consider legitimate users in the scenario. Thus, the defender needs to detect insider threats to decrease the system's risk and reduce the impact on regular users. We adopt the Bayesian game to model the interaction between insider attackers, legitimate users, and the defender. Our simulation results show that Mee can significantly increase the accuracy when detecting insider attackers and reduce the impact on regular users.

However, this research is regarding a relatively simple scenario. Because of the limitation of game theory, only two players participate in the game at one timeslot, and many users' and attackers' behaviors are ignored for simplification. To increase the complexity of the model, we adopt deep reinforcement learning to extend the honeyfile research. We treat the defender as the agent who can observe the misbehavior in the network through honeyfile alarms. The environment includes all connected devices, active users, and potential insider threats.

Honey Traffic generation

Generative Adversarial Networks (GANs) are generative modeling using deep learning methods, such as convolutional neural networks. Original GAN architecture includes *generator* and *discriminator*. Typically, a generator is trained to map from a latent space to data distribution, and the discriminator distinguishes candidates produced by the generator from the true data distribution. The objective of the trained generator is to increase the error rate of the discriminator. The generator network is utilized to produce additional data in this latent variable space. As is typical for Generative Adversarial Networks, a discriminator network is trained to classify actual and generated packets from their latent representations.

In this work, we model the defender by using the generator in GAN, which attempts to learn the network flow patterns from actual servers and deceive the discriminator by generating realistic network packets. In addition, we use a discriminator to model the attacker's passive reconnaissance processes. We assume that an attacker can obtain the traffic from a compromised switch and identify the hosts' background information through the gathered data. However, the defender may use honeypot and honey traffic to confuse the attacker.

We implement the GAN model, which includes a generator (i.e., the high-interaction honeypot) and a discriminator (i.e., the attack) within the CyberVAN testbed. The GAN generator learns network flow features from a server and produces honey traffic to confuse the attacker who performs passive monitoring via a compromised switch. Our simulation results show that the honey traffic can successfully disturb the attack's belief.

1.5 Plan of the Proposed Work

The following directions are the plan of our research within the next few months:

- Perform a complete and efficient simulation for our novel honeyfile system. The simulation includes a DQL-based agent and a more realistic environment.
- Enhance the testbed of honey traffic generation by considering a more complex network involving more servers and clients.

1.6 Organization

The rest of the proposal is organized as follows.

Chapter 2 introduces our survey paper about game theory and machine learning-based defensive deception. This chapter collects and analyzes current defensive deception research and applied game theory or machine learning thinking.

Chapter 3 describes Mee as a novel honeyfile system. We show how Mee increases the accuracy when detects insider attackers. We adopt the Bayesian game to assist the defender in searching for the optimal solution.

Chapter 4 is the extension work of Mee. We propose a more complete and realistic scenario and utilize deep reinforcement learning to guide the defender choose suitable actions.

Chapter 5 proposes a GAN-based honey traffic system, which helps the high-interaction honeypot learn from the actual server and generate fake network flow to confuse the malicious reconnaissance.

Chapter 6 summarizes our research approaches and discusses our future works.

Chapter 2

Survey of Game Theory and Machine Learning Based Defensive Deception

Abstract: Defensive deception is a promising approach for cyber defense. Via defensive deception, a defender can anticipate and prevent attacks by misleading or luring an attacker, or hiding some of its resources. Although defensive deception is garnering increasing research attention, there has not been a systematic investigation of its key components, the underlying principles, and its tradeoffs in various problem settings. This survey focuses on defensive deception research centered on game theory and machine learning, since these are prominent families of artificial intelligence approaches that are widely employed in defensive deception. This paper brings forth insights, lessons, and limitations from prior work. It closes with an outline of some research directions to tackle major gaps in current defensive deception research.

This chapter is joint work with: Ahmed H. Anwar, Zelin Wan, Jin-Hee Cho, Charles Kamhoua, and Munindar P. Singh MPS 1: cite the paper

2.1 Motivation and Contribution

2.1.1 Motivation

Conventional security mechanisms, such as access control and intrusion detection, help deal with outside and inside threats but inadequately resist attackers who can subvert controls or launch new attacks. Deception is a distinct line of defense aiming to thwart potential attackers. The key idea of deception is to manipulate an attacker's beliefs to mislead their decision making, inducing them to act suboptimally. Since the benefits of leveraging the core ideas of defensive deception (DD) have been realized in the cybersecurity research community, a good amount of research on DD has been explored.

Two main promising directions to develop DD techniques are observed in the literature. First, strategies of an attacker and defender have been commonly modeled based on game-theory where the defender takes DD strategies with the aim of creating confusion for attackers or misleading them so they would choose suboptimal or poor strategies. Second, DD techniques based on machine learning

Table 2.1: Comparison of our Survey Paper with the Existing Surveys of Defensive Deception

Key Criteria	Our Survey (2020)	Lu et al. [2020] (2020)	Pawlick et al. [2019] (2019)	Han et al. [2018] (2018)	Rowe and Rrushi [2016] (2016)	Almeshekah and Spafford [2014] (2014)
Provides concepts	✔	✔	✔	✔	✔	✔
Provides key taxonomies	✔	✔	✔	✔	✔	✔
Includes ML approaches	✔	✘	✘	✘	✘	✘
Includes game-theoretic approaches	✔	▲	✔	▲	▲	✘
Describes attack types considered	✔	✘	▲	▲	✔	▲
Describes metrics	✔	✘	▲	▲	▲	▲
Describes evaluation testbeds	✔	✘	✘	▲	▲	▲
Is not limited to specific application domains	✔	▲	✘	✔	✔	▲
Discusses pros and cons of relevant techniques	✔	▲	▲	✔	✔	▲
Discusses insights, lessons, and limitations	✔	▲	▲	✔	✔	▲
Discusses future research directions	✔	✔	✔	✔	▲	✘

✔: Fully addressed; ▲: Partially addressed; ✘: Not addressed at all.

(ML) have been proposed to create realistic fake information or decoy objects that mimic real objects to mislead or lure attackers.

The synergistic merit of combining GT and ML has been recognized in the cybersecurity literature [Kamhoua et al., 2021], such as using game-theoretic defenses against adversarial machine learning attacks [Dasgupta and Collins, 2019; Zhou et al., 2019] or generative adversarial models for creating deceptive objects [Kamhoua et al., 2021]. However, little work has explored the synergies between GT and ML to formulate various cybersecurity problems. In particular, since players' effective learning of their opponents' behavior is critical to the accuracy of their beliefs of the opponents' types or next moves, using ML for the players to form their beliefs can contribute to generating optimal plays under a certain environment. In addition, when developing DD techniques, ML-based approaches can provide better prediction of attackers or high similarity in creating deceptive object based on a large volume of data available. However, they may not provide effective strategic solutions under uncertainty which has been well explored in game-theoretic approaches. Therefore, this survey paper was motivated to facilitate future research taking hybrid DD approaches leveraging both GT and ML.

2.1.2 Comparison with Existing Surveys

We now distinguish the key contributions of our paper from existing survey papers on DD techniques. Several studies have conducted surveys of DD techniques [Almeshekah and Spafford, 2014; Han et al., 2018; Lu et al., 2020; Pawlick et al., 2019; Rowe and Rrushi, 2016].

Almeshekah and Spafford [2014] described how defensive deception is considered in cybersecurity defense. To be specific, they discussed the following three phrases in considering a defensive deception technique: planning, implementing and integrating, and monitoring and evaluating. In particular, Almeshekah and Spafford [2014] discussed the models of planning deceptions in terms of affecting an attacker's perception which can be misled for a defender to achieve a system's security goals. However, their work is limited in its contribution to modeling and integrating DD to a limited set of attackers. In

addition, this work did not consider the variety of network environments that should be considered in implementing DD techniques.

Rowe and Rrushi [2016] classified DD techniques in terms of impersonation, delays, fakes, camouflage, false excuses, and social engineering. They not only introduced the background on deception technologies but also explored the calculation of detectability and effectiveness of DD. However, their survey of game-theoretic DD is limited and lacks discussion of the state-of-the-art techniques.

Han et al. [2018] surveyed DD techniques based on four criteria, including the goal, unit, layer, and deployment of deception. They surveyed theoretical models used for DD techniques as well as the generation, placement, deployment, and monitoring of deception elements. Han et al. discussed the tradeoffs between various deceptive techniques in relation to whether they are deployed at the network, system, application, or data layers. Their discussion of game-theoretic deception, however, is not comprehensive.

Pawlick et al. [2019] conducted an extensive survey on DD taxonomies and game-theoretic DD techniques that have been used for cybersecurity and privacy. The authors discussed the main six different types of deception categories: perturbation, moving target defense, obfuscation, mixing, honey-X, and attacker engagement. Their paper surveyed 24 papers published over 2008–2018, and defined related taxonomies to develop their own classification of game-theoretic DD techniques. This work is interesting to treat moving target defense and obfuscation as subcategories under DD. This paper discussed the common game-theoretic approaches used for developing DD techniques, such as Stackelberg, Nash, and signaling game theories. However, the survey and analysis of the existing game-theoretic defensive techniques conducted in this paper are limited to game-theoretical analysis without considering realistic network environments where ML-based DD techniques or combination of these two (i.e., game theory and ML) may provide more useful insights and promising research directions.

Recently, Lu et al. [2020] conducted a brief survey based on the processes of DD consisting of three phases: planning of deception, implementation and deployment of deception, and monitoring and evaluation of deception. The authors discussed deception techniques based on information dissimulation to hide real information and information simulation to focus on attackers. This work briefly discussed game-theoretic DD and mainly focused on discussing challenges and limitations of the current research. However, only a small fraction of the literature was included. In addition, this paper did not discuss ML-based DD approaches.

Some survey papers mainly focused on DD techniques for a particular type of attacks or particular deception techniques. Carroll and Grosu [2011] investigated the effects of deception on the game-theoretic interactions between an attacker and defender of a computer network. They examined signaling games and related Nash equilibrium. However, this investigation only focused on honeypots technology while game-theoretic analysis of the deception is limited in examining the interactions between an attacker and a defender in the signaling games. Virvilis et al. [2014] surveyed partial DD techniques that can be used to mitigate Advanced Persistent Threats (APTs).

In Table 2.1, we summarized the key contributions of our survey paper, compared to those of the five existing survey papers [Almeshekah and Spafford, 2014; Han et al., 2018; Lu et al., 2020; Pawlick et al., 2019; Rowe and Rrushi, 2016] based on several key criteria.

2.1.3 Key Contributions

In this paper, we made the following **key contributions**:

1. We provided a novel classification scheme that characterizes a DD technique in its conceptual deception categories, presence of an object (i.e., physical or virtual), expected effects after applying deception, ultimate goal (i.e., for asset protection or attack detection), and activeness (i.e., active or passive or both). This provides an in-depth understanding of each deception technique and insights on how it can support a system's security goal.

2. We discussed key design principles of DD techniques in terms of what-attacker-to-deceive, when-to-deceive, and how-to-deceive. In addition, based on the key properties of DD techniques, we identified the key benefits and caveats when developing DD techniques leveraging game theory (GT) and ML.

3. We discussed GT and ML-based DD techniques based their types along with pros and cons. In addition, using the classification scheme in Section 2.2, we discussed both GT and ML algorithms.

4. We surveyed attacks handled by the existing game-theoretic and ML-based DD techniques. Accordingly, we discussed what attacks are more or less frequently considered in the literature by the DD techniques.

5. We surveyed how DD techniques are mainly considered to deal with the challenges of different network environments as application domains and discussed pros and cons of the deployed game-theoretic or ML-based DD techniques.

6. We examined what types of metrics and experiment testbeds are more or less frequently used in game-theoretic or ML-based DD techniques to prove their effectiveness and efficiency.

7. We extensively discussed lessons and insights learned and limitations observed from the DD techniques surveyed in this work. Based on these insights and limitations learned, we suggested promising future directions for DD research.

Note that the scope of this paper is mainly focused on surveying GT or ML-based DD techniques and discussing insights, limitations, or lessons learned from this extensive survey. Hence, some DD techniques that are not using either GT or ML are excluded in this survey paper.

2.1.4 Research Questions

We address the following **research questions** in this paper.

RQ Characteristics: What key characteristics of DD distinguish it from other defensive techniques?

RQ Metrics: What metrics are used to measure the effectiveness and efficiency of the existing game-theoretic or ML-based DD techniques?

RQ Principles: What key design principles help maximize the effectiveness and efficiency of DD techniques?

RQ GT: What are the key design features when a DD technique is devised using GT?

RQ ML: What are the key design features when a DD technique is developed using ML?

RQ Applications: How should different DD techniques be applied in different application domains?

We answered these questions in Section 2.9.1.

2.1.5 Structure of the Paper

The rest of this paper is structured as follows:

- Section 2.2 provides the concept of deception and taxonomies related to defensive deception (DD).
- Section 2.3 discusses the key principles of designing a DD technique. In addition, this section clarifies the key distinctive characteristics of the DD techniques, compared to other defense techniques that achieve a same defense goal.
- Section 2.4 explains the key components in using game-theoretic DD and surveys the existing game-theoretic DD techniques along with the discussions of their pros and cons.
- Section 2.5 discusses the key components in leveraging ML techniques to develop DD techniques. In addition, this section extensively surveys the existing ML-based DD techniques and addresses their pros and cons.
- Section 2.6 describes attack types countered by the existing game-theoretic and ML-based DD techniques.
- Section 2.8 presents metrics to measure effectiveness and efficiency of the existing DD techniques using game theory (GT) and ML. In addition, this section surveys evaluation testbeds used for validating those existing DD techniques surveyed in this work.
- Section 2.7 discusses how GT or ML-based DD techniques have been developed for different application domains, such as enterprise networks, cyberphysical systems (CPS), cloud web-based networks, Internet of Things (IoT), software-defined networks (SDNs), and wireless networks.
- Section 2.9 summarizes the insights and lessons learned by answering the key research questions raised in Section 2.1.4. In addition, this section discusses the limitations found from the DD techniques surveyed in this work and suggests promising future research directions.

2.2 Taxonomies of Defensive Deception

Deception has been heavily used by attackers which perform a variety of attacks at different levels of applications in both computer and social systems. In this section, we limit our discussions of deception in terms of a defender's perspective. This section mainly discusses formal models of deception, related common taxonomies used in developing defensive deception techniques, and their distinctive characteristics.

2.2.1 Formal Models of Cyber Deception

In this section, we discuss formal models of cyber deception that have been discussed in the literature. Although there are numerous formal models of cyber deception, we limit our discussion to formal modeling approaches based on logic, probabilistic logic, and hypergame theory.

Logic-based Cyber Deception

Deception planning has been studied based on logical reasoning of the key components of deception and their states (i.e., status). Deception modeling logic [Jafarian and Niakanlahiji, 2020; Takabi and Jafarian, 2017] was proposed to provide a formal deception model, $\mathcal{DM}$, as:

$$\mathcal{DM} = \{\text{facts},\ \text{beliefs},\ \text{actions},\ \text{causation rules},\ \text{attackers},\ \text{goal},\ \text{budget}\} \tag{2.1}$$

Each component of the DM is:

- *Facts*: The facts indicate a deception problem that can be described by (1) *attributes* of a system (e.g., configuration parameters, such as an operation system of a host, a server's criticality, existing routes between hosts), (2) data types of each attribute (e.g., Boolean, ordered, integral), and (3) actual values assigned to the attributes.
- *Beliefs*: These indicate an attacker's beliefs on the set of attributes, which may not be aligned with the true states of the attributes.
- *Actions*: An action refers to a defender's action to manipulate the attacker's belief on an attribute. Attributes can be *actionable* or *derivative*. Actionable attributes can be directly manipulated to influence the attacker's belief by taking a deceptive action. Derivable attributes cannot be directly manipulated but can be indirectly derived from the attacker's beliefs over other attributes. When modeling a set of actions, each action should be plausible to the attacker so the deception action is promising enough to deceive the attacker. In addition, the action should consider its cost.
- *Causation rules*: The causation rules apply to each attribute to determine what (e.g., facts or actions) can cause the manipulation of an attacker's belief on the attribute.
- *Attackers*: A set of attacker types can be modeled based on the probability distribution over the set when the probability of an attacker's particular type is available.
- *Goal & Budget*: The concrete benefit of successful or failed deception should be estimated in terms of defense effectiveness, defense cost (including both monetary budget), expected effect (i.e., a required level of achieved system performance/security) or other associated impact (e.g., interoperability with other defense or service mechanisms).

Rowe [2007] proposed a deception planning method to systematically provide logically consistent deceptions when users are detected as suspicious or malicious by an intrusion detection system. Rowe [2007] proposed an attack modeling technique that can consider realistic attackers characterized by their targets (e.g., files, a network, status such as log-in or administrator privileges), the status of the targets (e.g., existence, authorization, readiness, or operability), and their parameters (e.g., file size, network bandwidth, several sites, or a length of the password). By using an inference rule based on the status, an attacker can derive the status of a targeted resource. To prevent the attackers from achieving their goal, deception tactics can be used to deny the service the attacker requested by generating error messages, or respond with a credential request (e.g., password or other credential information). An error message can be generated in the form of 'a file not transferable', 'a file not recognizable', or 'protection violation.'

Probabilistic Cyber Deception

This type of formal models mainly studies how a developed DD technique can manipulate an attacker's belief and accordingly influence its action based on its belief in a probabilistic manner. Jajodia et al. [2017] proposed a probabilistic logic model for cyber deception. To maximize the probability of a successful deception, the defender decides to send true or fake responses to the attacker scanning probes. The effectiveness of deception is measured in terms of the accuracy of the attacker's belief on the system attributes (i.e., how much information the attacker has obtained towards a target system accurately). Crouse et al. [2015] proposed probabilistic models of reconnaissance-based defense with the aim of providing in-depth understanding on the effect of the defense on system security. The authors quantified attack success under various network conditions in terms of a network size, a size of deployment, and several vulnerable system components. The proposed deception models provide performance analysis of probabilistic triggering of network address shuffling and deployment of honeypots when these defense mechanisms are considered separately or in concert. In prior work [Cho et al., 2019a], we used Stochastic Petri Nets to model an attack-defense hypergame with the aim of examining how an attacker's beliefs manipulated by a DD can affect its decision making, resulting in attack failure. As Crouse et al. [2015]; Jajodia et al. [2017] and Cho et al. [2019a] observe, probabilistic models of deception games can effectively capture dynamic interactions between an attacker and a defender under uncertain environments.

Cyber Deception Hypergames

Much DD research uses game theory to formulate an attack-defense game. In particular, to effectively deal with uncertainty in real scenarios, a sequential attack-defense game where the defender uses DD has been formulated based on hypergame theory. Ferguson-Walter et al. [2019] formulated a cyber deception game as a sequential game as $G = (\mathcal{P}, \mathcal{M}, \Theta, u, \mathcal{T})$, where $\mathcal{P}$ refers to a set of players, $\mathcal{M}$ refers to a set of actions, such as $\mathcal{M} = \{\mathcal{M}_i\}$ for player i, Θ is a set of strategies, $\Theta = \{\Theta_i\}$ for player i, u is a utility function, $u = \{u_i\}$ for player i, and $\mathcal{T}$ is a sequence of moves when players take turns in sequence. By using the formulation of a regular, sequential game, a cyber deception can be formulated as (G, G^A, G^D) where G^A and G^D are derived games. In G, a move history of an attacker, $\mathcal{M}^A = \{m^A\} = \{m_1^A, m_2^A, \ldots, m_r^A\}$, and a defender, $\mathcal{M}^D = \{m^D\} = \{m_1^D, m_2^D, \ldots, m_s^D\}$, respectively, refers to the sequence of moves in the game where $r + s \leq \mathcal{T}$. For simplicity, the move histories of the attacker and defender can be denoted by $m = \{m^A, m^D\}$. Given that $\mathcal{M}^{X|Y} = m^{X|Y}$ refers to Y's perception of the set of X's actions (i.e., it is not necessarily true for $\mathcal{M}^{X|Y} = \mathcal{M}^X$), the attacker (A)'s beliefs about the move sequence, m, can be represented by $m^{*|A} = \{m^{A|A}, m^{D|A}\}$. Similar to the notation of the moves perceived by A, we can also obtain $u^{*|A} = (u^A|A, u^D|A)$, which is A's perception of $u = \{u^A, u^D\}$ and $\Theta^{*|A} = \{\Theta^{A|A}, \Theta^{D|A}\}$, which is A's perception of sets of strategies for Θ^A and Θ^D. Hence, the derived game can be denoted by $G^A = \{\mathcal{P}, \mathcal{M}^{*|A}, u^{*|A}, \mathcal{T}\}$, which is A's perception towards the game G, and $G^D = \{\mathcal{P}, \mathcal{M}^{*|D}, u^{*|D}, \mathcal{T}\}$, which is D's perception towards the game G. If an attack-defense game does not use the concept of hypergame, it simply relies on the game formulation of G.

2.2.2 Concepts of Defensive Deception

The conventional concept of military deception refers to actions taken to intentionally mislead an enemy about one's strengths and weaknesses, intents, and tactics [Sharp, 2006]. The defender employs

deception to manipulate the enemy's actions to advance one's mission [Sharp, 2006]. Defensive deception (DD) applies to a wide spectrum of interactions between an attacker and a defender under conflict situations [Sharp, 2006]. The concept of DD initiated in military environments has been introduced to cyber domains with the name of cyberdeception. Almeshekah and Spafford [2016] refined the concept of cyberdeception in [Yuill, 2006] as "planned actions taken to mislead and/or confuse attackers and to thereby cause them to take (or not take) specific actions that aid computer-security defenses." Although the concept of deception is highly multidisciplinary [Guo et al., 2020a], the common idea is to mislead an entity so it forms a false belief and to control its behavior based on that false belief [Rowe and Rrushi, 2016]. Therefore, when deception is successfully executed, a deceivee responds suboptimally, which provides benefits for a deceiver and results in achieving the deceiver's goal. Rowe and Rrushi [2016] emphasized 'manipulation' as the core concept of deception where deception encourages a deceivee to do something a deceiver wants or discourages the deceivee from doing something the deceiver does not want.

As the discussion of deception in this work is limited to DD, we consider a defender's deception against an attacker to achieve the defender's goal.

Table 2.2: Taxonomies and Classification of Defensive Deception Techniques

Deception Tactic	**Presence of a Real Object**	**Expected Effect**	**Ultimate Goal**	**Activeness**
Masking	True	Blending, Hiding, Misleading	Asset Protection	Passive
Repackaging	True	Blending, Hiding, Misleading	Asset Protection	Passive
Dazzling	True	Hiding, Confusing, Misleading	Asset Protection	Passive
Mimicking	False	Luring, Misleading	Attack Detection	Passive
Inventing	False	Luring, Misleading	Attack Detection	Active
Decoying	True	Luring, Misleading	Attack Detection	Active
Bait	True	Luring, Misleading	Attack Detection	Active
Camouflaging	True	Blending, Hiding, Misleading	Asset Protection	Passive
Concealment	True	Hiding, Misleading	Attack Detection	Active
False Information	False	Luring, Confusing, Misleading	Asset Protection or Attack Detection	Active or Passive
Lies	False	Hiding, Misleading	Asset Protection	Passive
Display	True	Hiding, Misleading	Asset Protection	Passive

2.2.3 Taxonomies of Defensive Deception

Deception can be applied with certain intent, either malicious or non-malicious intent [Guo et al., 2020b]. In particular, deception with malicious intent has been used as an attacker's strategy to deceive a defense system. For example, spammers are paid from clicking deceptive advertisements [Nextgate, 2019], malicious users disseminate phishing links to obtain credentials from victim users [Vergelis et al., 2019], or social and political bots propagate false information to influence public opinions [Forelle et al., 2015].

As we consider deception in the context of DD, we limit our discussions to deception 'with intent' where the intent is to defend a system against attackers. In this section, we discuss taxonomies in terms of the following aspects: (i) what conceptual techniques are used; (ii) whether a deception uses a true object or not (i.e., a true object exists but is hidden to deceive or a fake object is created to deceive); (iii) what effects are expected when successfully deceiving an opponent; (iv) what is an ultimate goal of the deception, such as detecting an attacker or protecting a system; and (v) whether a deception is active (mainly for attack detection) or passive (mainly for attack protection) to deceive an opponent.

Conceptual Deception Tactics

In [Almeshekah and Spafford, 2016; Bell and Whaley, 1991; Bennett and Waltz, 2007; Dunnigan and Nofi, 2001], we found the following conceptual deception techniques:

- *Masking* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: This refers to hiding certain information or data in the background.

- *Repackaging* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: This means to hide a real object as something else to hide the real object. A defender can make a vulnerability look like something else. For example, a defender can create honey files [Yuill et al., 2004], such as creating trap files that look like regular files.

- *Dazzling* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: This is a different way to hide something that is hard to blend in with the background or to be repackaged as something else. A real object can be hidden by overshadowing it, aiming to be undetected as the real, true object. For example, an intruder can be confused by receiving many error messages.

- *Mimicking* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: This refers to imitating aspects of a real object. This is often used to look honeypots as real nodes or interfaces to attackers.

- *Inventing* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: A defender can create a new fake object to lure attackers. For instance, a software can be presented in a honeypot for attackers to download it, which allows collecting the attackers' personal data.

- *Decoying* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: A deceiver attracts an attacker's attention away from critical assets that should be protected in a system. For example, the deceiver can provide publicly available false information about the system configurations to hinder reconnaissance attacks.

- *Bait*: Deception can use truth [Bennett and Waltz, 2007] to earn trust from a deceiver. That is, a defender can present attackers with correct but low-grade information to lure them. The truth can be

the information that an attacker already has, or the deceiver needs to sacrifice some correct sensitive data to effectively lure attackers.

The military literature provides the following taxonomy for cyberdeception [Dunnigan and Nofi, 2001].

- *Concealment* [Dunnigan and Nofi, 2001]: This deception is similar to hiding in the classical taxonomy. For the defensive purpose, it helps for honeypots to conceal their user-monitoring software so they look more like normal machines.
- *Camouflage* [Dunnigan and Nofi, 2001]: This deceives an attacker by blending a real object into a background or environment. For example, valuable files and programs can be assigned misleading names to make it difficult for an attacker to find them.
- *False information* [Dunnigan and Nofi, 2001]: Fake information (or disinformation) can be planted to mislead an attacker in cyberspace. However, most false information about cybersystems is easy to be checked by testing or trying it out. Therefore, this kind of deception cannot fool attackers for a long time.
- *Lies* [Dunnigan and Nofi, 2001]: This provides deceptive information, which is false. However, lies are distinguished from disinformation because they are often provided as responses to questions or requests. In this sense, even if this deception uses false information, it implies a passive action because lies may not be used unless it is requested. Lies could be more effective than explicitly denying access as they encourage an attacker to continue wasting time trying to access the resource in different ways and at different times.
- *Displays* [Dunnigan and Nofi, 2001]: This deception is similar to *inventing* as it also aims at misleading objects or information. However, unlike the inventing, which creates a new, fake object, displays can be applied to a real object (i.e., an object that is present in the system) by displaying it differently.

Presence of Actual Objects or Information

Deception can be applied when an actual, true object exists or no actual object exists. To be more specific, a defender can use true objects or information but may want to hide it by lying or providing false information towards it. For example, even if a system uses Windows as an operating system but may lie it uses Unix. But the system is using a certain OS in deed. On the other hand, a fake object, which has not existed anywhere, can be created for the purpose of creating uncertainty or confusion, which can lead an opponent to reach a suboptimal or poor decision, such as honey tokens or honey files.

Expected Effects (or Intents) of Defensive Deception

Via defensive deception (DD), one or more effects are expected as the process of achieving the goal of deception, making an attacker choose a suboptimal choice in its strategies. We categorize these in the following five aspects:

- *Hiding*: This effect is addressed when deception is used to change the display of a real object into something else. Some obfuscation defense techniques also use hiding data to slow down the escalation of attackers or increase attack complexity [Almeshekah and Spafford, 2014; Rowe and Rrushi, 2016].

- *Luring* [Almeshekah and Spafford, 2016; Bell and Whaley, 1991]: This effect is shown when a fake object is demonstrated to an attacker where the object looks like a real object an attacker is interested in. The typical examples are honeypots to lure attackers or honey files to send false information to the attackers.

- *Misleading* [Daniel and Herbig, 1982]: Most deception techniques have this effect as the last step to achieve the ultimate goal of deception, misleading an attacker to choose a suboptimal action.

- *Blending*: This effect is made when a real object can be well blended into background or environments. Its effectiveness relies on an environment and in altering the appearance of things to hide. For instance, give password files a non-descriptive name to elude automated searches run by hackers looking for files named 'pass' [Yuill et al., 2006]. Additionally, blending hidden objects with the environment can be achieved through altering the environment instead. For example, a defender can create noise in the environment through adding bogus files to make it harder to find critical ones.

- *Confusing* [Almeshekah and Spafford, 2014]: This effect comes from perceived uncertainty caused by a lack of evidence, conflicting evidence, or failing in discerning observations (e.g., cannot pinpoint whether a car on the street is red or blue) [Almeshekah and Spafford, 2014; Jøsang et al., 2018]. A defender can confuse an attacker by presenting both truths and deceits [Almeshekah and Spafford, 2014].

Some deception techniques use more than one technique, such as bait-based deception including lies to increase attack cost or complexity [Keromytis and Stolfo, 2014]. Bowen et al. [2009] designed a trap-based defense technique to increase the detection of an insider attack using several deception techniques like camouflage, false information, and creating fake objects. First, the deception system embeds a watermark in the binary format of the document file to detect when the decoy is loaded. Moreover, the content of each decoy document includes several types of "bait" information, such as online banking logins, login accounts for online servers, and web-based email accounts. A "beacon" is embedded in the decoy document that signals a remote web site upon opening of the document.

Ultimate Goals of Defensive Deception in Cyberspace

The ultimate goal of a defender using deception techniques is either protecting assets or detecting attackers or both. Although some deception techniques allow the defender to achieve both asset protection and attack detection, deception by hiding is more likely to protect system assets while deception by using false objects or information is to catch attackers.

Activeness of Defensive Deception

Caddell [2004] categorized deception as *passive* vs. *active*. *Passive deception* uses hiding the valuable assets or their capabilities and information from attackers. On the other hand, *active deception* uses fake, false information aiming for the attacker to form false beliefs, leading to choosing suboptimal decisions. Even if false information is used, the defense goal can be either protecting assets or detecting attackers. Hence, the distinction of active deception and passive deception may not be always clear.

In Table 2.2, we summarized how a particular deception technique can be understood based on the four key aspects: the presence of actual objects or information used in deception, expected effects (or intents), ultimate goal(s), and the activeness (or passiveness) of the used deception technique.

2.3 Design Principles and Unique Properties of Defensive Deception

In this section, we discuss the four key design principles of DD techniques. In addition, we address unique properties of DD and their key merits and caveats when using DD techniques. Further, we discuss how the DD techniques differ from other similar defense techniques, such as moving target defense or obfuscation techniques.

2.3.1 Design Principles of Defensive Deception

In this section, we discuss the design principles of DD in terms of the four aspects: *what-attacker-to-deceive* (i.e., what type of an attacker to deceive), *when-to-deceive* (i.e., when deception can be used), *how-to-deceive* (i.e., what particular deception technique can be used to deceive an attacker). We discuss each principle as follows:

What-Attacker-to-Deceive

This design principle asks to determine what type of an attacker a defender wants to deceive. For example, if the defender targets to deceive attackers performing reconnaissance attacks as outside attackers, it may aim to deceive them by providing false information about a system configuration (e.g., saying it uses Windows operating system (OS) even if it actually uses Unix). In addition, if a valuable system asset should be protected from attackers aiming to exfiltrate confidential information to an outside network, the defender may deploy a honeypot which mimics a real node with highly confidential information (e.g., a database). Hence, determining *what-attacker-to-deceive* is to decide what attackers to target by the defender. Since developing a DD technique incurs a cost, there should be in-depth analysis and investigation into whether a specific attacker should be prevented or detected by a DD technique in terms of cost, effectiveness, and efficiency of the technique's deployment.

When-to-Deceive

This design principle refers to determining when a deception technique should be used in terms of the attack stage of an attacker in the cyber kill chain (CKC) [Chen et al., 2014; Okhravi et al., 2013]. The six CKC stages include reconnaissance, delivery, exploitation, command and control, lateral movement, and data exfiltration [Okhravi et al., 2013]. Outside attackers mainly perform attacks in the stage of reconnaissance and delivery stages with the aim of penetrating a target system. On the other hand, inside attackers aim to perform attacks to exfiltrate confidential information to the outside, unauthorized parties. For example, when more malicious scans are identified in the reconnaissance stage, the defender can use fake patches with false vulnerability information. As a result, this can mislead the attacker to target honeypots. After the attacker successfully penetrates the system as an inside attacker, the defender may use honey files or tokens to deceive the attacker to exploit them. However, there are also legacy defense mechanisms, such as access control, intrusion detection and prevention, or emerging technologies, such as moving target defense as the alternative mechanisms that deal with the same types of attackers. In such cases, there should be utility analyses based on losses and gains in terms of the timing of using a DD technique.

How-to-Deceive

The design decision on how-to-deceive in employing DD is related to what type of a DD technique to use. The conceptual deception techniques have been discussed in Section 2.2.3, such as masking, mimicking, decoying, false information, baits, and so forth. However, what specific technique to use for DD is more related to what technology to use to achieve deception. Some example DD technologies include honeypots, honey files, honey tokens, fake patches, fake network topologies, fake keys, or bait files [Han et al., 2018].

2.3.2 Benefits and Caveats of Defensive Deception Techniques

In this section, we discuss the key benefits as well as the caveats of developing and employing them.

Key Benefits of Defensive Deception Techniques

- DD is relatively cost-effective compared to other defense strategies because its deployment cost is relatively low while showing relatively high effectiveness to mislead attackers. For example, honey files or honey tokens are relatively simple to deploy and maintain, compared to other traditional cybersecurity mechanisms, such as access control or intrusion detection.
- DD provides complementary defense services to other legacy defense mechanisms, such as intrusion detection or prevention. For example, honeypots are well known as an effective monitoring mechanism that can provide additional attack features to enhance intrusion detection as well as proactively to protect a system from intrusions before they actually launch attacks to targets.
- Various types of DD techniques are deployable at multiple layers of systems, such as network, system, application, and data layers [Han et al., 2018], without significant changes of existing system architectures. High deployability of DD techniques also provides design flexibility as well as defense-in-depth (DiD) capability with multiple layers of protections.
- Automated cyberdeception techniques, such as obfuscating session information, data flow, and software's code obfuscation, have significantly improved security to counter automated attacks.

Key Caveats of Defensive Deception Techniques

- Since it is highly challenging to obtain an attacker's motivation, intent, and goal in launching and executing a particular attack, it is not trivial to choose an optimal DD strategy based on the estimated risk and benefit associated with a DD strategy.
- Most honey-X techniques (e.g., honeypots, honey files, honey tokens) aim to mislead attackers to choose suboptimal or poor choices in launching attacks by false information. This may introduce extra procedures or protocols for normal users or a defender not to be confused by them.
- Due to the nature of deception, the effectiveness of DD requires continual configuration, reconfiguration, and implementation process. Otherwise, adversaries would be able to easily distinguish deceptive devices unless deception is automated like software obfuscating techniques.
- With the emergence of clever attackers, there is a growing demand for high-interactive honeypots or elaborated deception strategies that cannot be easily identified by attackers. This brings a higher

defense cost and more difficulty in the management for a defense system than a traditional system without DD mechanisms.

2.3.3 Distinctions between Defensive Deception and Other Similar Defense Techniques

Defensive deception (DD) is often mentioned as moving target defense or vice versa. In addition, obfuscation has been used to achieve the same aim like DD. In this section, we discuss how the roles and natures of these three techniques differ from each other as well how they overlap in their functionalities and aims.

Distinction with Moving Target Defense (MTD)

MTD is similar to DD in terms of its aim to increase confusion or uncertainty of attackers, leading to deterring the escalation of their attacks to a next level or failing their attacks. However, the key distinction is that MTD does not use any false information to actively mislead attackers while DD often involves using false objects or information for the attackers to form false beliefs and be misled to make suboptimal or poor attack decisions. MTD relates its key functionality mainly with how to change system configurations more effectively and efficiently while DD more involves the exploitation of manipulating the attacker's perception. If DD is well deployed based on a solid formulation of manipulating the attacker's perception, it can be more cost effective than deploying MTD. Ward et al. [2018] considered MTD as part of DD while Cho et al. [2020] treated DD as part of MTD. The distinction between these two is not clear because DD can be deployed using MTD techniques, such as randomness or dynamic changes of system configurations (e.g., dynamically assigning decoys in a network) while it may not use false information all the time (e.g., baits or omission such as no response).

Distinction with Obfuscation

Obfuscation techniques have been used by an attacker to obfuscate malware [You and Yim, 2010], metamorphic viruses [Borello and Mé, 2008], or malicious JavaScript code [Xu et al., 2012]. In terms of a defender's perspective, obfuscation techniques have been used to obfuscate private information [Ardagna et al., 2007], or Java bytecode for decompiling [Chan and Yang, 2004]. Although obfuscation techniques are often mentioned as the part of DD techniques, they have been studied as a separate research area for decades. Chan and Yang [2004] conducted an extensive survey on obfuscation techniques enhancing system security and discussed the key aims of obfuscation techniques as: (i) increasing the difficulty of reverse engineering of the program logic; (ii) mitigating the exploitability of vulnerabilities by attackers; (iii) preventing modifications by unauthorized parties; and (iv) hiding data or information. Although the concept of DD has gained popularity of late while obfuscation techniques have been applied for decades, it is obvious that the aim of defensive obfuscation is well aligned with that of DD. However, a subtle difference lies in that DD involves cognitive aspects of an attacker's perception while defensive obfuscation is directly involved with achieving security goals (e.g., confidentiality, data integrity, availability). Defensive obfuscation techniques are also used in combination with diversification [Chan and Yang, 2004] (e.g., diversifying systems, network configurations, or components), which is a well-known concept used by MTD techniques.

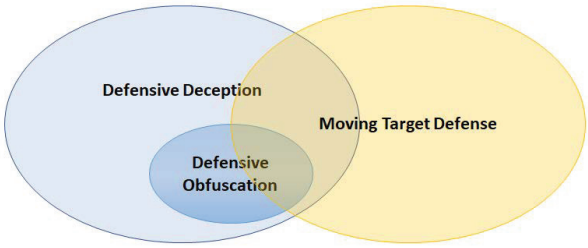


Figure 2.1: Relationships between defensive deception, moving target defense, and defensive obfuscation.

Based on our understanding on the functionalities and aims of DD, MTD, and defensive obfuscation, we would like to hold our view based on Fig. 2.1, showing the relationships among them. Our view is that a defensive obfuscation can belong to either MTD or DD. In addition, there are also an overlapping area of MTD and DD where some defensive techniques require using the features of both concepts.

2.4 Game-Theoretic Defensive Deception (GTDD) Techniques

In this section, we discuss the key component of modeling DD techniques using game theory. In addition, we discussed a wide range of DD techniques that have been considered based on various types of game theory with their pros and cons.

2.4.1 Key Components of Game-Theoretic Defensive Deception

Game theory has been used extensively to solve cybersecurity problems by modeling the following key elements: a defender and attacker as players, actions as attack and defense strategies, observability of a system and an opponent's strategies, or system dynamics. We discuss these key elements of cybersecurity games as follows.

Players

Most research taking game-theoretic DD approaches model a two-player game where the two players are an attacker and a defender using DD [Kiekintveld et al., 2015; Mao et al., 2019; Pawlick and Zhu, 2015]. However, even in a two-player game, some deception games modeled different types of players under each player. Pawlick and Zhu [2015] modeled a two-player signaling game where a defender as a sender can be either a normal system or a honeypot while an attacker as a receiver has one type. Depending on the type, the defender takes its strategy. Pawlick et al. [2015] introduced a three-player game between a cloud defender, an attacker and a device which is connected to the cloud where the defender can send false signals to deceive the attacker.

(In) Complete Information

A complete information game allows players to have full knowledge of the game parameters, such as possible actions to be taken by other players, a reward function, and the current state of the game in case of dynamic or multistage games. A game with complete information is sometimes considered to be impractical. The assumption of the defender knowing the set of attack actions to be taken by its adversary is not realistic due to inherent uncertainty in a context. On the other hand, an incomplete

information game represents a practical class of games where one or more players may or may not know some information about the other players [Kajii and Morris, 1997]. For instance, a player may not fully know other players' types, strategies, and payoff functions.

(Im) Perfect Information

An imperfect information game represents a game in which a player may not know what exact actions have been played by other players in the game. This makes it computationally prohibitive to track the history of actions in case of a multistage game. However, all players may know their opponents' types, strategies, and payoff functions, forming a complete, imperfect information game. This imperfect information game is often assumed in cybersecurity games between an attacker and a defender [Başar and Olsder, 1999; Phlips, 1988].

Partial Observability

Players acting upon dynamic systems are modeled as dynamic and stochastic games. In this case, the system/environment state changes over time and depends on the actions taken by all the players. However, in some scenarios, one or more players will not fully observe the state. This results in a partially observable game. If the system environment is affected by an exogenous factor, the state transition is stochastic, resulting in a partially observable stochastic game (POSG) [Hansen et al., 2004]. In the special case of POSG with only a single player, the game turns to a partially observable Markov dynamic process [Cassandra, 1998]. POSG model deals with the most general form of games that capture different game settings, such as incomplete information as well as imperfect information (i.e., imperfect monitoring). Therefore, solving such a game efficiently is still an open research question.

Bounded Rationality

Both the rational and bounded rational players are commonly applied in a game between an attacker and a defender. A rational player can always choose an optimal strategy to maximize its expected utility. However, a bounded rational player has only limited resources and cannot afford an unlimited search to find an optimal action [Tadelis, 2013].

Pure or Mixed Strategies

Two main strategies in game theory are pure strategy or mixed strategy [Tadelis, 2013]. A pure strategy means a player's strategy is determined with the probability 0 or 1. In contrast, a mixed strategy is a probability distribution of several pure strategies. For example, Clark et al. [2012] used pure strategies to perform deceptive routing paths to defend against jamming attacks in a two-stage game using Stackelberg game theory. Zhu et al. [2012] also used deceptive routing paths against jamming attacks but identified an optimal routing considering resource allocation based on mixed strategies.

Uncertain Future Reward in Utility

Due to the inherent uncertainty caused by multiple variables that may control the future states, a player's utility cannot be guaranteed [Tadelis, 2013]. In game-theoretic DD research, uncertainty is mainly caused by non-stationary environmental conditions and non-deterministic movements (or actions) by opponent

players. In a signaling game, uncertainty is caused by both the movement of an opponent player and its type [Mohammadi et al., 2016; Pawlick and Zhu, 2015] where a player can take an action after it knows its opponent's type.

Utilities

For a player to take an optimal action, it is critical to know both its own payoff (or utility or reward) function and its opponent's payoff function. In a static, one-time game, a reward per strategy is based on action profiles (i.e., a set of actions each player can take) of all players [Tadelis, 2013]. However, in a sequential game (i.e., a repeated game), a reward can be estimated based on the history of action profiles from the beginning to the end of the game. Hence, in such games, a player aims to maximize the expected average or accumulated reward over the period of the sequential game [Başar and Olsder, 1999].

Full Game or Subgame

A full game represents a game where each player considers a set of all possible actions to select one action to maximize its utility. A subgame indicates a subset of the full game where a player considers a subset of all possible actions when selecting one action to maximize its utility. Particularly, in sequential games, a subgame includes a single node and all its successors [Tadelis, 2013]. It is similar to a subtree in a tree-structure sequential game, which refers to an extensive form game. House and Cybenko [2010] used the hypergame theory to model the interactions between an attacker and a defender. Hypergame uses a subgame to model each player's different view on the game and corresponding strategies under the subgame. In addition, Zhang et al. [2019] and Xu and Zhuang [2016] used *Subgame Perfect Nash Equilibrium* (SPNE) [Mas-Colell et al., 1995] (i.e., a Nash equilibrium in every subgame) to deal with the resource allocation for both the defender and attacker in non-hypergames.

Static Game vs. Dynamic Game

In game theory, a static game is considered as one-time interaction game where players move simultaneously. Hence, a static game is a game of imperfect information. On the other hand, a dynamic game is considered as a sequential game assuming that players interact through multiple rounds of interactions as repeated games. Although most dynamic games are considered sequential games with perfect information, if a game considers partial (or imperfect) observability, the dynamic game can be sequential across rounds of games but within a subgame, it can be a game of imperfect information (i.e., players know other players' moves in previous rounds but do not know their moves in a current round).

Game theory has provided a powerful mathematical tool for solving strategic decision making in various dynamic settings. Since the powerful capability of game theory has been clearly proven based on the achievement of several game theorists as Nobel Prize Laureates [Tadelis, 2013], there is no doubt that game theory has substantially inspired methods for solving complex, dynamic decision-making problems to produce mathematically validated solutions. Although game theory has been substantially explored to provide solutions with solid theoretical proofs, one of its common limitations has been discussed in terms of too strong assumptions to derive Nash Equilibria, such as common knowledge about probability distributions of Nature's moves or players' correct beliefs towards opponents' moves. As game theory has a long history since 1700s [Bellhouse, 2007], its long evolution has made significant advancement

in considering increasingly realistic aspects of real-world problems. The exemplary efforts in addressing realistic scenarios include the consideration of bounded rationality, decision making under uncertainty using the concept of mixed strategies, games of imperfect or incomplete information, hypergame with different subjective views of players, games with partially observable Markov Decision Process (POMDP), subjective beliefs, subjective rationality, and so forth [Tadelis, 2013]. We believe these efforts have obviously opened a door for game theory to be highly applicable in real systems, including systems concerning cybersecurity. In the context of developing DD techniques, game theory can contribute to manipulating an attacker's beliefs and accordingly leading the attacker to miscalculate utilities under uncertainty with the aim of failing the attacker. In addition, how frequently a certain defense should be executed or what defense strategy to choose can be also determined based on game-theoretic decision methods.

2.4.2 Common Games Formulated for Defensive Deception

We now discuss popular game theories used to formulate various DD techniques. We will discuss the following games:

- *Bayesian games*: A Bayesian game is a game of incomplete information which means part of or all players do not know their opponents' types, actions, or payoffs where each player knows their own type, a set of actions, and corresponding payoffs. Hence, each player has a subjective prior probability distribution of their opponent's type [Harsanyi, 1967].

- *Stackelberg games*: Stackelberg game refers to a sequential two-player game where the first player is a leader and the second player is a follower. In a stage game, the leader moves first, then the followers observe the action of leader and select their strategy based on the observation [Von Stackelberg, 2010]. Hence, the first player takes the first-mover-advantage which can lead the second player's move to their intended direction [Tadelis, 2013].

- *Signaling games*: This is a sequential Bayesian game where one player sends a signal (i.e., information) to another player. When the player sends false information, it can be most costly [Tadelis, 2013].

- *Stochastic games*: This is a multistage game in which the game evolves from one stage (state) to another following a stochastic distribution. The transition probability matrix depends on a set of action profiles of both players and may also depends on random exogenous factors of the game environment. The game dynamics determine the payoffs of both players [Shapley, 1953].

- *Games with Partially Observable Markov Decision Process (POMDP)*: The POMDP considers to model an agent's decision process where the agent selects actions to maximize a reward of the system. However, the agent cannot directly perceive the system state, but it can obtain observations depending on the state. The agent can maintain a belief of a system state based on the observations. Formally, a POMDP model can be represented by a tuple (S, A, Ω, T, R, O) [Cassandra, 1998], where S represents a set of states, A represents a set of actions, Ω represents a set of observations, T is a state transition function, R is a reward function, and O is a set of conditional observation probabilities. As an extended version of the POMDP, the Interactive-POMDP (IPOMDP) presents a multiagent setting which allows an agent to model and predict behavior of

other agents [Gmytrasiewicz and Doshi, 2005]. A Bayes-Adaptive POMDP (BA-POMDP) assumes that transition and observation probabilities are unknown or partially known [Ross et al., 2007].

2.4.3 Game-Theoretic Defensive Deception

In this section, we discuss game-theoretic DD techniques used in the literature. We use the following classification to discuss game-theoretic DD techniques for asset protection: honeypots, honeywebs, honeynets, obfuscation, deceptive signals, multiple DD techniques, fake objects, honey patches, deceptive network flow. Some studies used empirical game-theoretic experiment settings where players are human subjects. We also discussed them to show how game theory has been used to develop DD in empirical experimental settings with human-in-the-loop.

Honeypots

A honeypot has been studied as the most common DD strategy in the literature. A honeypot is mainly used in two forms: *low interaction honeypots* (LHs) and *high interaction honeypots* (HHs) [Han et al., 2018]. The differences between them are based on different deception detectability and deployment cost. HHs provide lower detectability by attackers than LHs while incurring higher deployment cost than LHs [Han et al., 2018].

Pawlick and Zhu [2015] employed a signaling game to develop a honeypot-based defense system where an attacker is able to detect honeypots. The authors investigated multiple models of signaling games with or without evidence when complete information is available or not. In the cheap-talk games with evidence (i.e., a signaling game with deception detection), extended from cheap-talk games (i.e., costless communication signaling game between a sender and receiver), the receiver (i.e., an attacker) is modeled to detect deception (i.e., a honeypot) with some probability. They found that the attacker's ability to detect deception does not necessarily lower down the defender's utility.

The signaling game with evidence has been also applied to model the honeypot selection or creation in other works [Pawlick et al., 2018; Píbil et al., 2012]. Pawlick et al. [2018] used a signaling game with evidence where the evidence is estimated based on a sender type and a transmitted message. The authors extended the signaling game with a detector with probabilistic evidence of deception. However, this signal is vulnerable to attackers which can analyze the signal, leading to detecting the deception and causing the evidence leakage. Píbil et al. [2012] introduced a game-theoretic high-interaction honeypot to waste attackers' resources and efforts. The authors designed a *honeypot selection game (HSG)* as a two-player zero-sum extensive-form game with imperfect and incomplete information. A honeypot is designed to mimic servers with high importance to provide cost-effective DD.

La et al. [2016] proposed a two-player attacker-defender deception-based Bayesian game in an IoT network. They used honeypots as a deceptive defense mechanism. They studied the Bayesian equilibrium in one shot and repeated games. Their findings showed the existence of a certain frequency of attacks beyond which both players will primarily take deception as their strategy. The authors used a honeypot as a deceptive strategy where a game is considered under incomplete information.

Çeker et al. [2016] proposed a deception-based defense mechanism to mitigate Denial-of-Service (DoS) attacks. The defender deploys honeypots to attract the attacker and retrieve information about the attacker's real intent. They used signaling games with perfect Bayesian equilibrium to model the interactions between the defender and attacker. Basak et al. [2019] used cyber deception tools to identify an

attacker type as early as possible to take better defensive strategies. The attacker's type is reflected in actions and goals when planning an attack campaign. The authors leveraged on a multistage Stackelberg Security game to model the interaction between the attacker and defender. In the game, the defender is a leader taking strategies considering the attacker's strategy. As a follower, the attacker selects its strategy after observing the leader's strategy. Through a game-theoretic approach, the defender selects deception actions (e.g., honeypots) to detect specific attackers as early as possible in an attack.

Mao et al. [2019] used honeypots as a defense strategy in a non-cooperative Bayesian game with imperfect, incomplete information where an attacker is a leader and a defender is a follower. The authors considered a player's perception towards possible motivation, deceptions, and payoffs of the game. Kiekintveld et al. [2015] discussed several game models that address strategies to deploy honeypots, including a basic honeypot selection game. They first developed a honeypot selection game as a two-player zero-sum extensive-form game with imperfect and incomplete information in [Píbil et al., 2012]. Then they extended the game to allow additional probing actions by the attacker in which attack strategies are represented based on attack graphs [Kulkarni et al., July, 2021; Xi and Kamhoua, 2020]. The authors conducted experimental performance analysis to compare the performance of these model and discussed the advantages and disadvantages of the considered game-theoretic models in the DD research.

Durkota et al. [2015] leveraged a Stackelberg game where an attacker follows an attack graph and a defender can deploy a honeypot to deceive the attacker based on an efficient optimal strategy searching method using Markov Decision Processing (MDP) and sibling-class pruning. Game-theoretic cyberdeception techniques have been proposed using attack graphs [Kulkarni et al., 2020; Milani et al., 2020; Tsemogne et al., 2020]. Kulkarni et al. [2020] developed a zero-sum, hypergame of incomplete information for evaluating the effectiveness of the proposed honeypot allocation technique. Milani et al. [2020] developed a Stackelberg game to allocated defensive resources and manipulate a generated attack graph. Tsemogne et al. [2020] studied the malicious behavior through an epidemic model and solved a one-sided partially observable stochastic game for cyberdeception where a defender distributes a limited number of honeypots.

Wagener et al. [2009] addressed the issue of a honeypot being overly restrictive or overly tolerant by proposing a self-adaptive, game-theoretic honeypot. The authors modeled the game between an attacker and a defender by reusing the definitions proposed in [Greenwald, 2007]. They applied game theory to compute the optimal strategy profiles based on the computation of Nash Equilibrium. The game theory directs the configuration and a reciprocal action of the high-interaction honeypot. In their experiment, this technique produces an optimal strategy based on Nash equilibrium with rational attackers. Aggarwal et al. [2016, 2017] discussed a sequential, incomplete information game to model the attackers' decision-making in the presence of a honeypot and combined this model with instance-based learning (IBL). The authors introduced two timing-based deception by applying early or late deception in the rounds of games and investigated the effect of timing and the extent of deception. Their results showed that high amount of using deception and late timing of deception can effectively decrease attack actions on the network. However, using a different timing and amount of deception did not introduce any difference in attacking on a honeypot action.

Various honeypot allocation methods for deception over attack graphs have been studied based on game-theoretic approaches that considers uncertainty using partially observable MDP (POMDP) and stochastic games (POSG) [Anwar and Kamhoua, 2020; Kamhoua, 2018]. Anwar et al. [2019] developed a static game model to protect a connected graph of targeted nodes against an attacker through honeypot

placements based on node features and significance. The game model is extended to study the game dynamics as a stochastic allocation game in [Anwar et al., 2020]. To resolve high complexity of attack graph-based honeypot allocation games, the authors proposed a heuristic approach.

Nan et al. [2019] and Nan et al. [2020a] provided Nash equilibrium-based solutions for a defender to intelligently select the nodes that should be used for performing a computation task while deceiving the attacker into expending resources for attacking fake nodes. Nan et al. [2019] investigated a two-player static game of complete information with mixed-strategy, where the system selects a node to place the deception source and the attacker selects a node to compromise. Nan et al. [2020a] constructed a static game with mixed-strategy, where both players select one strategy (target node) with probability distribution. The defender selects a node to install defense software while the attacker selects a target node to compromise and avoid being detected by the anti-malware software.

Pros and Cons: A honeypot is the most popular DD technology which has matured over decades. Since its aim is not to introduce any additional vulnerabilities, if an attacker is successfully lured by the honeypots, the honeypots can protect the existing system components (i.e., system assets) while collecting additional attack intelligence which can lead to improving intrusion detection with new attack signatures. However, maintaining honeypots incurs extra cost. In addition, there is still a potential performance degradation due to the existence of the honeypots, such as additional routing paths or resource consumption. However, the drawbacks of honeypots have been little investigated. In addition, as more intelligent, sophisticated attackers have been emerged recently, it becomes more challenging to develop realistic honeypots for effectively deceiving attackers in terms of both complexity and cost.

Honeywebs

El-Kosairy and Azer [2018] proposed a web server framework that provides a web application firewall with a honeyweb to detect malicious traffics and forward suspicious traffics to honeypot servers. This work formulated the interactions between the attacker (i.e., malicious traffics) and the defender (i.e., honey servers) as a game of imperfect information (i.e., players cannot not monitor each other's moves) but under complete information setting (i.e., players know their opponent's type, possible actions, and payoff function) considering simultaneous moves. Typically, this research combines deception technologies, including honey token, honeypot, and decoy document. In the game definition, the main action of a defender is implementing a virtual machine (VM) to consume the attacker's effort and resources.

Pros and Cons: Honeywebs are rarely used particularly based on game-theoretic approaches. They are used to assist honeypots along with honey tokens or honey files. Developing fake webpages incurs less cost while its role is more like an assistant to support the honeypots. Hence, honeywebs may not be used without other defensive technologies.

Honeynets

Garg and Grosu [2007] considered an attacker and a honeynet system (i.e., a defender) as players in a strategic non-cooperative game. The framework is based on extensive games of imperfect information. In their game model, the defender makes deception about the placement of a honeypot while the attacker can probe the target host to identify their true roles with some probability. The attacker's utility considers the cost of both probing and compromising a host or honeypot. They studied the mixed strategy equilibrium solutions of these games and showed how they are used to determine the strategies

of the honeynet system.

Dimitriadis [2007] proposed a honeynet architecture based on an attacker-defender game. This architecture is called 3GHNET (3G-based Honeynets) and aimed to enhance the security of the 3G core network by defending against DDoS and node compromise attack. The main components of this 3GHNET are two gateways with a set of strategies to control and capture the data flow between two nodes. In their emulation, each gateway and attacker are considered as individual players. The 3GHNET-G is a two-player, non-cooperative and zero-sum game. The authors identified the optimal strategy based on Nash Equilibrium.

Pros and Cons: The honeynet is a system architecture designed to deal with inside attackers and allow a system manager to monitor/learn threats. The Honeynet Project [Spitzner, 2003] is a typical example of a honeynet consisting of multiple honeypots applied in practice. Since the honeynet works with multiple honeypots, how to optimally deploy and work together among them needs more investigation based on both the effectiveness of deception and additional deployment costs.

Obfuscation

Shokri [2015] modeled a leader-follower game (i.e., Stackelberg game) between a designer of an obfuscation technique and a potential attacker. Authors designed adaptive mechanisms to defend against optimal inference attacks. They assumed that the users plan to protect sensitive information when sharing their data with untrustful entities. This allows the users to obfuscate the data with noises before sharing it. The attacker can observe valuable information of users and noises caused by obfuscation. They provided linear program solutions to search for optimal solutions that provably achieves a minimum utility loss under those privacy bounds.

Hespanha et al. [2000] proposed a DD framework based on non-cooperative, zero-sum, stochastic games with partial information. They analyzed how a defender in a competitive game can manipulate information available to its opponents to maximize the effectiveness of the DD. They found that there exists an optimal amount of information to be presented to effectively deceive an attacker.

Pros and Cons: Compared to other DD techniques, such as honey-X techniques, the key benefit of data obfuscation is easy deployability with low cost. However, adding noise into normal information can also confuse a defender or a legitimate user. On the other hand, most data obfuscation research mainly aims to develop a technique of how to hide real information rather than how to detect an attacker.

Deceptive Signals

In many game-theoretic approaches that consider DD as a defense strategy, deception is simply used as a signal to deceive an opponent player. Pawlick et al. [2015] modeled a game of three players, consisting of a cloud defender, an attacker, and a device, in a cloud-based system that can deal with advanced persistent threats (APTs). The authors designed the so-called *FlipIt* game to model the interactions between an attacker and a defender where signaling games are used to model the interactions between the device and the cloud which may be compromised with a certain probability. Xu and Zhuang [2016] designed a game between an attacker and a defender where the attacker can investigate the vulnerability of a target and the defender can apply defensive technologies to change the vulnerability of a certain device. This work mainly studied how the attackers' preparation for launching an attack affects the effectiveness of DD strategies. As the result, the authors applied the subgame perfect Nash Equilibrium (SPNE) to

analyze the strategic interactions of the terrorist's costly learning and the defender's counter-learning.

Yin et al. [2013] leveraged a fake resource or converted a real resource to secrete recourse to mislead attackers. They applied a Stackelberg game to model the interactions between an attacker and a defender. The authors conducted the mathematical analysis of a game using both pure and mixed strategies played by the defender. They provided an optimal strategy for the defender and calculated the accuracy rate of identifying when the attacker can deploy unlimited surveillance before attacking. Horák et al. [2017] used one-sided POSG (partially observable stochastic games), in which a defender has perfect information while other players have imperfect information. In this work, an attacker and defender take sequences of actions to either deceive an opponent or attempt to obtain true actions taken by the opponent. This work assumed that a rational attacker has knowledge towards its opponent and its actions to take. However, the attacker has no knowledge of a network topology which introduces a hurdle to identify a target and whether the defender is aware of its presence in the network.

Ferguson-Walter et al. [2019] proposed a hypergame-based DD scenario where players have imperfect and incomplete information. Based on the key concept of a hypergame, the authors modeled an individual player's own game where the game structure and payoffs may be manipulated by an opponent player. However, this work assumed an asymmetric view by the attacker and defender in that the attacker does not know the defenders taking DD strategies while the defender can know the attacker's true payoff and game structure.

Bilinski et al. [2019] introduced game-theoretic deception procedures based on a masking game in which a defender masks the true nature of a device. In this game, an attacker can ask the defender whether a specific device is real or fake at each round of the game. The defender needs to pay cost if it lies. After several rounds, the attacker chooses a target to attack. The authors investigated this scenario by conducting a mathematical analysis for non-adaptive and adaptive game models. They also designed a Stackelberg game model where the attacker is a leader and the defender is a follower. Through a Markov Decision Process (MDP) simulation, they investigated the potential behavior of an attacker at noncritical points.

Pros and Cons: This deceptive signal is an abstract way to apply a game theory concept to model a deception game between an attacker and a defender. Due to the nature of the abstracted game formulation, it has a high flexibility that can use various types of deception techniques. On the other hand, it is quite challenging to model a cybersecurity problem in a particular context as the proposed deception game framework does not use specific DD techniques.

Multiple Defensive Deception Techniques

Some existing approaches leverage a set of cyberdeception technologies to model a set of strategies by a defender. Chiang et al. [2018] discussed DD to improve system security and dependability based on an SDN environment by utilizing a set of honey technologies. The authors discussed what are the critical requirements to realize effective DD and identified promising evaluation methods including metrics and evaluation testbeds. Chiang et al. [2018] constructed a dynamic game of complete information where the attacker does not know the strategies chosen by defender. Huang and Zhu [2019] discussed several DD techniques, such as honeypots, fake personal profiles, or multiple game models. They used a static Bayesian game to capture stealthy and deceptive characteristics of an attacker and advance this model by applying asymmetric information one-shot game. They demonstrated the performance of their proposed techniques under APT based on the Tennessee Eastman (TE) process as their case study.

Pros and Cons: Since multiple DDs are combined and used as a set of a defender's strategies, the defender can make more choices to defend against attackers based on different merits that can introduce to different types of attacks. In particular, to deal with APT attackers which can perform multi-staged attacks, using various types of DD can provide more relevant resources to deal with them. However, combing more than one deception techniques introduces additional deployment costs. In addition, it is not trivial to identify a optimal combination of multiple DD techniques.

Fake Objects

Mohammadi et al. [2016] analyzed two-player signaling games with a defensive fake avatar for selecting an optimal strategy under various scenarios. A defender can use an avatar in the signaling game as a fake internal user to identify an external attacker by interacting with external users. This game considered the payoff of two players and derived an optimal threshold of raising alarms. In their experiment, the expected payoff is used a metric to guide actions of the avatar or the defender. Unlike other works using signaling games [Pawlick and Zhu, 2015; Pawlick et al., 2015], their signaling games put the defender as a second mover (i.e., a receiver) while an attacker is a first mover (i.e., a sender) where the games are played with incomplete information. The key idea of using the signaling games is creating uncertainty for the attacker because the defender uses a fake identity which can make the attacker doubtful about whether the receiver is a real user.

Casey et al. [2015] discussed interactions between an insider attacker and a defender of an organization. They considered a hostile agent that may obtain organization surface and harm the whole system. To mitigate this kind of threats, the authors proposed a *honey surface* to confuse the malicious agent by designing a basic compliance signaling game to model the interaction between agents and the organization. Casey et al. [2016] discussed the insider threat in an organization and applied a signaling game to model the interactions between agents with learning intelligence and a defender.

Thakoor et al. [2019] introduced a general-sum game, named *Cyber Camouflage Games* (CCGs), to model the interactions between a defender and an attacker performing reconnaissance attacks. The defender can mask the machine in the network with fake information, such as an operating system, to mitigate the effect of reconnaissance attacks. To identify an optimal strategy, they introduced the Fully Polynomial Time Approximation Scheme given constraints and applied Mixed Integer Linear Program to search for an optimal strategy.

Zhu et al. [2013] simulated the infiltration of social honeybots-based defense into botnets of social networks. The authors proposed a framework, so-called SODEXO (SOcial network Deception and EXploitation) consisting of Honeybot Deployment (HD), Honeybot Exploitation (HE), and Protection and Alert System (PAS). HD is composed of a moderate number of honeybots. The HE considered the dynamics and utility optimization of honeybots and botmaster by a Stackelberg game model. The PAS chose an optimal deployment strategy based on the information gathered by honeybots. The results showed that a small number of honeybots can significantly decrease the infected population (i.e., a botnet) in a large social network.

Pros and Cons: Although using fake identities or avatars as a DD technique is less costly and relatively simple, it can introduce confusion to normal users and accordingly increase false alarms. However, little work has investigated the adverse effect of using fake identities, such as introducing confusion for legitimate users or producing extra vulnerabilities. In addition, the effect of intelligent attackers with high deception detectability has not been studied.

Honey Patches

A honey patch mainly has two components: (1) A traditional patch to fix known software vulnerabilities; and (2) additional code to mislead an attacker to fake software vulnerabilities [Avery and Spafford, 2017]. In the literature, fake patches are also called *honey patches* [Araujo et al., 2014] or *ghost patches* [Avery and Spafford, 2017].

Araujo et al. [2014] coined the term *honey patches* that can function the same as a regular patch; but can efficiently redirect an attacker to a decoy, and allow the attacker to achieve a fake success. The game used in [Araujo et al., 2014] is a dynamic game of incomplete information where the payoff of attacker is unknown to defender. In addition, the authors provided a strategy allowing a web server administrator to transfer regular patches into honey patches. Enterprise scale systems are often exposed by the vulnerabilities due to the lack of boundary checking and subtle program misbehavior [Avery and Spafford, 2017]. A common way to eliminate the vulnerabilities is releasing security patches. However, as the security patches contain the location and types of vulnerability to be patched, an attacker could obtain a blueprint of system vulnerabilities by analyzing the security patches from the past. As a solution, Avery and Spafford [2017] provided a fake patch technique for misleading the attackers that try to analyze the historical patches. In particularly, this fake patch technique is designed to protect the patch that fixes input validation vulnerabilities. This fake patch can contain two parts. One is creating a decoy patch file to seduce the attacker and attract the attacker's attention away from the real patch file and the other is to alert defenders of potential intrusions or exfiltration attempts. This decoy file is generated by a modified technique [Bowen et al., 2009]. Another part is a bogus control flow for programs. This alternation misleads the reverse engineering, but does not change the output of the program. In their experiment, this technique was evaluated in [Cadar et al., 2008] where the program runtime and program analysis measure the effectiveness of this technique. The game and its components in [Avery and Spafford, 2017] were not rigorously presented since it mainly focused on the technicalities of developing fake honeypots. However, the deception and the evaluation of its reward represents a normal form one-shot game. Such games were clearly presented in [Avery and Wallrabenstein, 2018] where the authors evaluated the effectiveness of three deceptive patches (i.e., faux, obfuscated, and active response) and applying them into the proposed game-based module which provides the security guidelines. In addition, they found that many techniques are unable to meet the proposed security definition while emphasizing the importance of assessing deceptive patches based on a clear and meaningful security definition.

Cho et al. [2019a] modeled a deception game based on hypergame theory in which an attacker and a defender have different perceptions of the game. This work examined how a player's (mis)perception can affect its decision making to choose strategies to take, which can affect the player's utility in the game. This work used Stochastic Perti Nets to build a probabilistic model of the hypergame where the defender uses fake patches as a deception strategy for misleading the attacker.

Pros and Cons: Honey patches are known as effective deception techniques to deceive attackers with low cost. However, as discussed in [Avery and Wallrabenstein, 2018], when some systematic and meaningful security assessments are conducted to validate the security of the honey patches, some honey patches fail to pass those security criteria, such as indistinguishablity between real and fake patches, vulnerabilities of fake patches, or cryptographic breakability for obfuscated messages [Avery and Wallrabenstein, 2018].

Table 2.3: Summary of Game-Theoretic-based Defensive Deception Techniques

Ref.	DD technique	Goal	Tactic	Expected effect	Main attacks	Game type	Domain
[Cho et al., 2019a]	Fake patch	Asset protection	Mimicking; False information; Lies	Luring; Confusing; Misleading	APT	Hypergame	No domain specified
[Ferguson-Walter et al., 2019]	Deceptive signal	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Hypergame	No domain specified
[Kiekintveld et al., 2015]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC/ probing	General-sum	CPS
[Mao et al., 2019]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	APT	Bayesian	SDN
[Pawlick and Zhu, 2015]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Signaling	No domain specified
[Pawlick et al., 2015]	Deceptive signal	Asset protection	Mimicking; Masking; Lies; Decoying	Misleading; Luring; Confusing	APT	Three-player signaling	Cloud Web
[Clark et al., 2012]	Dummy packet generation	Asset protection	Mimicking; Masking	Misleading; Hiding; Confusing	Jamming	Stackelberg	Wireless
[Zhu et al., 2012]	Deceptive network flow	Asset protection	Mimicking; Masking	Misleading; Hiding; Confusing	Jamming	Multi-stage stochastic	No domain specified
[Mohammadi et al., 2016]	Fake avatar	Asset protection	Mimicking; Decoying	Asset protection	NC	Signaling	No domain specified
[Pawlick et al., 2018]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Signaling	No domain specified
[Píbil et al., 2012]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	General-sum	No domain specified
[La et al., 2016]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Bayesian	IoT
[Çeker et al., 2016]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	DoS	Signaling	No domain specified
[Basak et al., 2019]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	APT	General-sum	No domain specified
[Xi and Kamhoua, 2020]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	APT	Static	IoT
[Durkota et al., 2015]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Stackelberg	No domain specified

[Wagener et al., 2009]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	General-Sum	No domain specified
[Aggarwal et al., 2016]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC/ probing	Non-cooperative sequential; incomplete information	No domain specified
[Aggarwal et al., 2017]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC/ probing	Sequential	No domain specified
[Anwar and Kamhoua, 2020]	Deceptive signal	Asset protection	Mimicking; Masking; Decoying	Misleading; Luring; Confusing	NC/ probing	Stochastic with POMDP	IoT
[Anwar et al., 2019]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Stochastic	IoT
[Anwar et al., 2020]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Stochastic	IoT
[Nan et al., 2019]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	NC	Static	IoT
[El-Kosairy and Azer, 2018]	Honeywebs	Asset protection	Decoying; Mimicking	Blending; Misleading; Luring	Web attack	Complete, imperfect game	Cloud Web
[Garg and Grosu, 2007]	Honeynet	Asset protection; Attack detection	Decoying; Mimicking	Asset protection; Attack detection	NC	Zero-sum static	No domain specified
[Dimitriadis, 2007]	Honeypot	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	DDoS	Zero-sum static	Wireless
[Hespanha et al., 2000]	Obfuscation	Asset protection	Masking	Misleading; Confusing	NC	Non-cooperative stochastic	No domain specified
[Yin et al., 2013]	Deceptive signal	Asset protection	Masking; Camouflaging; Mimicking	Hiding; Blending, Hiding; Misleading	Reconnaissance	Stackelberg	No domain specified
[Horák et al., 2017]	Deceptive signal	Asset protection; Attack detection	Misleading; Hiding; Mimicking; Decoying	Misleading; Hiding; Luring; Confusing; Blending	NC	Stochastic	No domain specified
[Bilinski et al., 2019]	Deceptive signal	Asset protection; Attack detection	Misleading; Hiding; Mimicking; Decoying	Asset protection; Attack detection	NC	Stackelberg	No domain specified

[Chiang et al., 2018]	Honey-X technologies	Asset protection; Attack detection	Misleading; Hiding; Mimicking; Decoying	Misleading; Hiding; Luring; Confusing; Blending	APT	General-sum	SDN
[Huang and Zhu, 2019]	Deceptive signal	Asset protection; Attack detection	Misleading; Hiding; Mimicking; Decoying	Misleading; Hiding; Luring; Confusing; Blending	APT	Dynamic Bayesian	No domain specified
[Casey et al., 2015]	Honey surface	Asset protection	Mimicking; Decoying	Hiding; Misleading	Insider	Signaling	No domain specified
[Casey et al., 2016]	Honey surface	Asset protection	Mimicking; Decoying	Hiding; Misleading	Insider	Signaling	No domain specified
[Thakoor et al., 2019]	Fake objects	Asset protection	Mimicking; Decoying	Hiding; Misleading	Recon.	General-sum	No domain specified
[Zhu et al., 2013]	Social honeypots	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	Social bots	Stackelberg	No domain specified
[Sayin and Başar, 2019]	Deceptive network flow	Asset protection	Mimicking; Masking	Misleading; Hiding; Confusing	Recon./ probing	Non-zero-sum Stackelberg	CPS
[Al Amin et al., 2019a]	Honeypot	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	APT	POMDP	SDN
[Al Amin et al., 2020]	Honeypot	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	APT	POMDP	SDN
[Nan et al., 2020b]	Deception signal	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	NC	Stackelberg	Wireless
[Wang et al., 2020]	Honeypot	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	APT	Q-learning-based	No domain specified
[Al Amin et al., 2019b]	Fake nodes	Asset protection; Attack detection	Decoying	Hiding; Misleading	APT	POMDP	No domain specified
[Rahman et al., 2013]	Fake objects	Asset protection	Mimicking; Decoying	Hiding; Misleading	Recon.	Signaling	No domain specified
[Shi et al., 2020]	Feature deception	Asset protection	Lies	Misleading	Reconnaissance	RL-based feature deception	No domain specified

Deceptive Network Flow

Clark et al. [2012] proposed the deceptive network flow, representing the flow of randomly generated dummy packets. They assumed that both of real and deceptive packets are encrypted. Hence, an attacker cannot distinguish between them and may spend limited resources on targeting a false flow. This work used this deceptive flow to lure the attacker and waste its resource to assist those real packets to protect the network from jamming attacks. The main challenges of developing defensive network flow are: (1) applying deceptive packets may increase the risk of congestion and incur extra delay for the delivery of

real packets; and (2) the source node has a limited capacity to generate and transfer packets, requiring a balance between real and fake flows. To mitigate this adverse effect, the authors designed a two-stage game model to obtain deception strategies at pure-strategy Stackelberg equilibrium. They considered two types of source nodes: (1) *selfish* nodes aiming to maximize its own utility; and (2) *altruistic* nodes considering the congestion of other sources when choosing a flow rate. Their results proved that altruistic node behavior improves the overall utility of the sources. Similarly, Zhu et al. [2012] considered a single source selecting routing paths for real and deceptive flows. Before sending a deceptive flow, the source node is allowed to choose the rate of deceptive and real flow as well as the path of the deceptive flow. This work introduced the solution concepts, such as the path Stackelberg equilibrium, the rate Stackelberg equilibrium, and their mixed strategy counterparts of the game. Their results proved that there exist such equilibria. Miah et al. [2020] designed a deceptive network flow system, called *Snaz*, to mislead the attacker performing reconnaissance attacks. They model the interaction between the attacker and the defender with a two players non-zero-sum Stackelberg game. Sayin and Başar [2019] proposed a deceptive signaling game framework to deal with APT attacks in cyber-physical systems. Under the attacks aiming to obtain system intelligence via scanning or reconnaissance attacks, this work crafted bait information to lure the attackers. This work leveraged the concept of game-theoretic hierarchical equilibrium and solved a semi-definite programming problem where the defender does not have perfect information by considering partial or noisy observations or uncertainty towards the attacker's goal.

Pros and Cons: Deceptive network flow is a new DD approach to effectively mitigate some malicious actions which cannot be defended by traditional DD technologies, such as malicious network flow scanning and fingerprint. However, it is highly challenging to balance fake and real network flows that minimizes any additional network performance degradation.

In Table 2.3, we summarized the five key aspects (i.e., deception techniques, categories, expected effects, attacks considered, and application domain) of each game-theoretic DD technique surveyed in this work.

2.4.4 Empirical Game Experiments Using Human Subjects

As we observed in Section 2.4.3, most deception game-theoretic works have been studied based on simulation testbeds. However, cognitive scientists (specializing in decision making) have conducted empirical experiments using human subjects which are assigned as attackers or defenders to consider a deception game. Cranford et al. [2020, 2019] conducted an empirical experiment to realize a signaling game between an attacker and a defender where the defender uses deceptive signals. In this experiment, the players are humans with limited cognition, reflecting bounded rationality in game theory. To measure the effectiveness of DD techniques, Ferguson-Walter et al. [2018] and Ferguson-Walter [2020] also conducted an empirical experiment with 130 red team members as participants in a network penetration study at different deception scenarios. For instance, different types of decoy devices are described explicitly for the existence of deceptive defensive techniques to the participant. The results showed that when the participants are aware of the deception, they took much more time before taking any action. This shows that deception made them slower to move and attack. Aggarwal et al. [2019] developed a simulation tool, called *HackIt*, to study attacks in a network reconnaissance stage where participants studied the effect of introducing deception at different timing intervals. Their results showed that the attacker performed attacks on the honeypots more often than real machines.

Pure game theorists may argue that this type of empirical game experiments is not a game-theoretic

deception game. However, we discuss these empirical deception game studies in this paper to understand the role of human subjects-based empirical game experiments. Such games can partly follow the structure of traditional game-theoretic models. For instance, it considers two or more players, with bounded rationality, opponent strategies, and corresponding utilities.

2.5 Machine-Learning-Based Defensive Deception (MLDD) Techniques

ML-based applications become more popular than ever in various domains, including cybersecurity. ML techniques have been extensively adopted for automating attacks and learning system behaviors in the cyberdeception domain [Al-Shaer et al., 2019]. In this section, we first discuss the key steps to implement ML-based approaches and conduct an extensive survey on ML-based defensive deception (DD) techniques in the literature.

2.5.1 Key Steps of Implementing ML-Based Defensive Deception

Like other ML-based applications, ML-based deception techniques also commonly used the following steps to detect malicious activities:

- *Dataset Generation*: ML-based approaches have been mainly used in detecting attackers or identifying malicious activities based on the information collected in honeynets or honeypots [Song et al., 2011].

- *Dataset Collection*: The success of ML techniques hugely depends on whether reliable datasets are available or not. However, obtaining reliable cybersecurity datasets is not trivial based on the following reasons: (i) such datasets are often confidential due to security reasons of a company or an organization owning the datasets; and (ii) there are inherent unknown attacks which make it difficult to obtain accurate and complete annotated datasets. Cybersecurity related datasets have been more available as more security applications are using ML-based approaches [Snijders et al., 2012].

- *Pre-Processing*: Datasets should be carefully pre-processed ahead of the learning step to eliminate unwanted noises from the dataset. This includes data cleaning, editing, and reduction [Han et al., 2011].

- *Feature Extraction*: It is challenging to analyze complex data because of the potentially numerous variables involved. The goal of feature extraction is to construct a combination of the relevant variables to describe the data with sufficient accuracy.

- *Training*: Training dataset is a dataset of examples used during the learning process and is used to learn the parameters of the detector [Han et al., 2011; Ron and Foster, 1998].

- *Testing*: A testing dataset is a set of examples used to evaluate the performance (i.e, detection accuracy) of a trained detector [Ron and Foster, 1998].

- *Evaluation*: Independent and new data are used to evaluate the performance of a detector to avoid overfitting problems in the data [Ron and Foster, 1998].

2.5.2 Common ML Techniques Used for Defensive Deception

We discuss popular ML techniques used to develop various DD techniques in the literature. For the easy understanding of readers who may not have enough background on ML, we brief the overview of the following ML techniques:

- *Support Vector Machine (SVM)*: SVMs can deal with a set of supervised learning models and be applied to solve classification and regression problems [Cortes and Vapnik, 1995]. SVM uses the hypothesis space of linear functions in a high dimensional feature space to analyze data. SVM is popular due to its simple training process. However, SVM is not efficient for large or highly noisy datasets.

- *K-Means*: This uses an iterative refinement to partition n observations into k clusters and guarantee that each observation belongs to a cluster with the nearest mean [contributors].

- *Expectation Maximization (EM)*: This statistically searches for maximum likelihood parameters with two major steps: (1) an expectation step, which creates a function for the expectation of the likelihood evaluated using the current estimate for the parameters; and (2) a maximization step, which computes a new estimate of the parameters [Moon, 1996].

- *Hierarchical Grouping*: This is a well-established classification method for cluster analysis which seeks to build a hierarchy of clusters [Ward Jr, 1963]. For example, plants and animals may be grouped into a smaller number of mutually exclusive classes with respect to their genetic characteristics.

- *Bayesian Network (BayesNet)*: This is a classifier that learns the conditional probability of each attribute from training data while assuming strong independence [Friedman et al., 1997]. That is, the underlying probability model would be an independent feature model such that the presence of a particular feature of a class is unrelated to the presence of any other features. For example, BayesNets could represent the probabilistic relationships between diseases and symptoms.

- *Decision Tree (DT)*: DT is widely used for multi-stage and sequential decision making. The key idea of DT lies in breaking up a complex decision into a union of several simpler decisions, leading to the intended desired solution [Safavian and Landgrebe, 1991].

- *C4.5 Algorithm*: This generates a decision tree based on the concept of information entropy in which the DT generated by C4.5 is widely used for data classification [Quinlan, 2014].

- *Naïve-Bayes Algorithm*: This is a widely applied classification algorithm based on Bayesian Theory and statistical analysis [Domingos and Pazzani, 1997]. This algorithm uses the prior and posterior probabilities of the training data to calculate the probability of type (posterior probability) of a specific sample.

- *Deep Neural Networks*: DNN is a neural network with at least one hidden layer and has been used for deep learning. For each node (neuron), its output is a non-linear function of a weighted sum of its inputs. The DNN model is trained by back propagation, which changes the weight of each node. DNNs have been used for natural language processing, image recognition, and disease diagnosis.

2.5.3 ML-Based Defensive Deception

This section discusses ML-based DD techniques discussed in the literature. We discuss the following DD techniques along with what ML techniques are used to develop them: Social honeypots, honey files, honeypots or decoy systems, bait-based deception, fake services, obfuscation, and decoy data.

Social Honeypots

In online social networks (OSNs), the so-called *social honeypots*, as good bots in contrast to bad bots, have been studied by creating social network avatars.

Lee et al. [2010] proposed an ML-based mechanism to discover social spammers. The authors used social honeypots as a DD technique to monitor spammer activities. The data from spam and legitimate profiles are used to improve an ML-based classifier. Thus, the system can classify spam profiles automatically. The paper compared true positive rate (TPR) and false negative rate (FNR) of various ML-based classifiers to show the effectiveness of the proposed mechanism. Lee et al. [2011] developed social honeypots to detect content polluters in Twitter. Sixty social honeypot accounts are created to follow other social honeypot accounts and post four types of tweets to each other. This work categorized the collected user features into nine classes based on the Expectation-Maximization (EM) algorithm. They classified the content polluters based on Random Forest and enhanced the results by using standard boosting and bagging and different feature group combinations.

El Idrissi Younes et al. [2016] developed social honeypots to discover malicious profiles. The authors employed feature-based strategies as well as honeypot feature-based strategies to collect data of the malicious profiles and analyzed the collected data based on a suite of ML algorithms provided by Weka ML toolkit (e.g., logistic regression, Bayes classifier, support vector machine (SVM), K-Means, Expectation Maximization, or hierarchical grouping). Zhu [2015] introduced the concept of 'active honeypots' as active Twitter accounts that can capture more than 10 new spammers every day. They identified 1,814 accounts from the Twitter and examined the key features of active honeypots. In addition, using a suite of ML algorithms, such as SVM, logistic regression, J48 Tree, Bagging, and AdaBoost M1, they examined the impact of unbalanced datasets on the detection accuracy of the different ML algorithms.

Badri Satya et al. [2016] also used social honeypot pages to collect fake likers on Facebook where fake likes are provided by paid workers. The authors obtained four types of user profiles and behavioral features as the distinctive patterns of fake likers. This work evaluated the robustness of their ML-based detection algorithms based on synthetic datasets which have been modified to reflect individual and coordinated attack models. Yang et al. [2014] proposed a passive social honeypot to collect a spammer's preferences by considering various behaviors of the social honeypot. The considered tweet design features include tweet frequency, tweet keywords, and tweet topics as well as the behaviors of famous users' accounts and application installation. The authors conducted in-depth analysis on what types of social honeypot features can introduce a higher rate of collecting social adversaries and enhanced the social honeypots based on the result. The improved honeypot has shown 26 times faster than a normal social honeypot in collecting social adversaries based on the evaluation using a random forest classifier.

Pros and Cons: The works discussed in [El Idrissi Younes et al., 2016; Lee et al., 2010, 2011; Yang et al., 2014] mainly relied on luring deception techniques to attract spammers in OSNs and collect more attack intelligence particularly in terms of novel behavioral characteristics of spammers. Some limitations of the existing social honeypots include: (i) the current efforts are mainly to detect spammers, not

other social network attacks (e.g., cybergroomers, human trafficking, cyberstalking, or cyberbullying); (ii) most features are based on user behaviors, but not on network topological features as an attacker's characteristics; (iii) most detectors are applied on static datasets to evaluate the detection of spammer. This implies that there is no theoretical simulation framework that considers dynamic interactions between an attacker and a defender; and (iv) there is a lack of studies investigating the effectiveness of social honeypots in terms of how quickly more attack intelligence can be collected and what types of attack intelligence are collected.

Honey Profiles

Stringhini et al. [2010] analyzed 900 honey profiles for detecting spammers in three online social networks, including MySpace, Facebook, and Twitter. The authors collected users' activity data for a year where the user datasets include both spammers' and legitimate users' profiles and the honey profiles are spread in different geographic networks. Further, this work captured both spam profiles and spam campaigns based on the shared URL using machine learning techniques (e.g., SVM).

Pros and Cons: Honey profiles allow identifying a large set of attackers, such as a large-scale spam bot farms, leading to detecting large-scale and coordinated campaigns. However, the existing spam detectors using honey profiles is not generic and needs to be tailored depending on a different social media platform.

Honeypots or Decoy Systems

Nanda et al. [2016] used a suite of machine learning algorithms to train historical network attack data in software-defined networks for the development of quality honeypots. This ML-based method is to identify potential malicious connections and attack destinations. The authors employed four well-known ML algorithms, including C4.5, Bayesian Network (BayesNet), Decision Table (DT), and Naïve-Bayes to predict potential victim hosts based on the historical data. To prevent call fraud and identity theft attacks, Krueger et al. [2012] proposed a method called *PRISMA* (PRotocol Inspection and State Machine Analysis). This method is developed to infer a functional state machine and a message format of a protocol from only network traffic. The authors experimented to validate the performance of the PRISMA based on three real-world network traces datasets, including 10K to 2 million messages in both binary and textual protocols. Their experiments proved that the PRISMA provides the capability to simulate complete and correct sessions with the learned models and execute different states for malware analysis.

Hofer et al. [2019] discussed the attributes of cyber-physical decoys to design a system with the attributes to develop deception decoys. They integrated the deception decoys into real systems to make them harder to detect and more appealing as targets. To increase the fidelity of the added decoy devices to the physical system, three attributes, a protocol, variables, and a logic of the deployed decoy devices, are maintained. The authors trained a recurrent neural network (RNN) using a dataset collected for a year to learn such system attributes.

Some DD games have been studied by considering game-theoretic reinforcement learning (RL) where RL is considered in formulating players (i.e., attackers or defenders)'s utilities. That is, in the RL-based game formulation, players use an RL to identify an optimal strategy where the RL's reward function considers gain and loss based on the player's belief towards an opponent's move. RL-based

deception games are studied in [Al Amin et al., 2019b; Wang et al., 2020]. Al Amin et al. [2019b] proposed an online deception approach that designs and places network decoys considering scenarios where a defender's action influences an attacker to dynamically change its strategies and tactics while maintaining the trade-off between availability and security. The defender maintains a belief consisting of a security state while the resultant actions are modeled as Partially Observable Markov Decision Process (POMDP). The defender uses an online deception algorithm to select actions based on the observed attacker behavior using a POMDP model. This embedded reinforcement learning (RL)-based model assumes that the defender belief about the attacker's progress is observed through an network-based intrusion detection system (NIDS). The defender hence takes actions that attract the attacker toward decoy nodes. Wang et al. [2020] identified an optimal deployment strategy for deception resource, such as honeypots. The authors developed a Q-learning algorithm for an intelligent deployment policy to dynamically place deception resources as the network security state changes. They considered an attacker-defender game by analyzing an attacker's strategy under uncertainty and a defender's strategies with several deployment location policies. They used RL with a Q-learning training algorithm that identifies an optimal deployment policy for deception resources.

Pros and Cons: In honeypots, ML is mainly used to detect attacks based on the attack intelligence gathered in honeypots. Hence, this research direction is like developing an intrusion detection mechanism, rather than creating better quality DD techniques. Various ML-based techniques can be highly leveraged to create real-like honeypots and its effectiveness can be measured based on the volume of attackers caught in honeypots and the diversity of attacker types.

Bait-based Deception

Ben Salem and Stolfo [2012] developed a bait-based deception to improve the detection of impersonation attacks by using the features combining a user behavior profiling technique with a baiting deception technique. This work used highly crafted and well-placed decoy documents to bait attackers and deployed a detector that uses SVM to model the user's search behavior. Whitham [2016] used bait-based deception to study and evaluate honey files in military networks. The main challenge in military networks is the need to utilize third party resources such as cloud services. Third party servers are directly connected to the Internet and may store sensitive material and information which represents a security challenge. The author proposed three new automated honey file designs that minimize the replication of classified material, yet remain enticing to malicious software or user driven searches. Moreover, this work presented an NLP-based content generation design for honey files.

Pros and Cons: Although bait-based deception can contribute to increasing intrusion detection by collecting more intelligence during the time of the deception, there is no performance guarantee based on bait-based deception because the attackers may not be interested in the bait. In addition, to attract attackers more effectively, if more important information is used as a bait, the bait itself introduces risk when intelligent attackers can derive clues of system vulnerabilities based on the baits they have examined. Thus, mixing real with fake information to avoid too high a risk can provide an alternative, such as semi-bait-based deception.

Table 2.4: Summary of Machine Learning-based Defensive Deception Techniques

Ref.	DD technique	Goal	Tactic	Expected effect	Main attacks	ML technique	Domain
[Nanda et al., 2016]	Honeypot	Attack detection	Mimicking	Attack detection	Anomaly network traffic	C4.5, Bayesian Network, Decision Table, and Naïve-Bayes	No domain specified
[Badri Satya et al., 2016]	Social honeypot	Attack detection	Mimicking	Luring	Fake liker (crowdturfing)	Supervised ML	OSN
[El Idrissi Younes et al., 2016]	Social honeypot	Attacker identification	Mimicking	Luring	Malicious profile	ML toolkit	No domain specified
[Krueger et al., 2012]	Honeypot	Asset protection; Attack detection	Mimicking; Decoying	Luring; Misleading	Call fraud; Identity theft	ML toolkit	No domain specified
[Lee et al., 2010]	Social honeypot	Identifying spammers	Mimicking; Decoying	Luring	Social spammers	ML toolkit	OSN
[Lee et al., 2011]	Social honeypot	Identifying spammers	Mimicking; Decoying	Luring	Social spammers	ML toolkit	OSN
[Hofer et al., 2019]	Decoy	Asset protection; Attack detection	Decoying; Mimicking	Blending; Misleading; Luring	APT	RNN	CPS
[Ben Salem and Stolfo, 2012]	Bait-based Deception	Asset protection; Attack detection	Mimicking; Lies	Luring; Confusing	Insider	SVM	No domain specified
[Whitham, 2016]	Bait-based Deception	Attack protection	Mimicking; Lies	Luring; Confusing	Zero-day	NLP	No domain specified
[Stringhini et al., 2010]	Honey profile	Asset protection	Mimicking; Decoying	Hiding; Misleading	Spamming	SVM	OSN
[Whitham, 2017]	Bait-based Deception	Attack detection	Mimicking	Luring	Zero-day	NLP	No domain specified
[Abay et al., 2019]	Decoy data	Attack detection	Lies	Misleading	Data leakout	DL	No domain specified

Fake Services

To increase the effectiveness and efficiency of DD techniques, real systems have been used to deceive attackers. ML-based fake system modification is used as a DD mechanism [Al Amin et al., 2019a, 2020] using POMDP to consider uncertainty in learning attack behavior. The authors designed a deception server that can modify IP addresses of fake network via DHCP server, DNS server, and forwarding ARP requests by appropriate flow rules to the deception server. In addition, the deception server is designed to modify the requests and sent it back to the requesting host. Shi et al. [2020] considered the changes of system configurations to deceive an attacker after learning its preferences and behavior from attack data.

Pros and Cons: Proving fake services to attackers can effectively increase attack cost or complexity which can delay launching attacks or make the escalation of attack fail. However, this technique can be deployed based on the assumption of accurate detection of intrusions. As the current intrusion detection mechanisms suffer from high false positives in practice, if a normal user is treated as an intrusion and fake services are provided to the user, it hinders normal availability of service provisions to legitimate users.

Decoy Data

Machine and deep learning (DL) algorithms have played a key role in developing decoy data, creating decoy objects or honey information. Abay et al. [2019] took a decoy data generation approach to

fool attackers without degrading system performance by leveraging DL. To be specific, their approach employs unsupervised DL to form a generative neural network that samples HoneyData.

Pros and Cons: Since the defender launches false information injection (or data disruption) attack while an attacker stays in a system, this technique will 'scare off' attackers performing highly detectable attacks. However, it can also be disruptive to ordinary users if fake or decoy information is allowed to remain in the system.

As we observed in the extensive discussions of ML-based DD techniques above, ML-based DD has been addressed mainly with the following aims: (1) Protecting system components by hiding real components or creating fake things (i.e., honey-X) that look like real; and (2) Detecting attacks (or identifying attackers) by creating fake objects (e.g., honeypots) and accordingly attract attackers to collect attack intelligence. Therefore, the questions of what ML techniques are used and how are closely related to the quality of DD techniques whose success is closely related to how well it can deceive attackers, representing the quality of deception.

In Table 2.4, we summarized the five key aspects (i.e., deception techniques, categories, expected effects, attacks considered, and application domain) of each ML-based DD techniques surveyed in this work.

2.6 Attacks Counteracted by Defensive Deception Techniques

A variety of attacks have been countered by existing defensive deception (DD) techniques as follows:

- *Scanning (or reconnaissance) attacks* [Al Amin et al., 2019a; Avery and Wallrabenstein, 2018; Bilinski et al., 2019; Chiang et al., 2018; Mao et al., 2019; Miah et al., 2020; Sayin and Başar, 2019; Xu and Zhuang, 2016]: An outside attacker may aim to obtain a target system's information and identify vulnerable system component to penetrate the target system. This attack is often considered as part of the cyber kill chain in APT attacks. In the literature, the scanning attacks are categorized as two types as follows:
  - *Passive monitoring attacks* [Bilinski et al., 2019; Miah et al., 2020; Xu and Zhuang, 2016]: Leveraging compromised nodes (e.g., routers or switches), attackers can scan traffics passing through the nodes and analyze packets to gather information about hosts. Passive monitoring can make the attackers achieve active nodes' discovery, OS, roles, up-time, services, supporting protocols, and IP network configuration [Montigny-Leboeuf and Massicotte, 2004]. However, passive monitoring cannot identify services that do not run on well-known ports or are misdirected without protocol-specific decoders [Bartlett et al., 2007].
  - *Active probing attacks* [Al Amin et al., 2019a; Avery and Wallrabenstein, 2018; Chiang et al., 2018; Mao et al., 2019; Sayin and Başar, 2019]: Through sending probing packets to a potential target, the attacker can obtain the machine's information, includes OS, opened ports, and installed application version. Although active probing can allow the attackers to collect more system configurations, it is relatively easy to be detected by traditional defensive technologies, such as IDS [Bartlett et al., 2007].
- *Network fingerprinting* [Rahman et al., 2013]: To analyze a target network and identify ideal target nodes, an attacker uses fingerprint tools to identify specific features of a network or a device, such as

a network protocol or OS type. Common fingerprint tools include Nmap [Lyon], P0f [Zalewski, 2014], and Xprobe [Arkin and Yarochkin, 2002]. Some deception techniques, such as deceptive network flows and honeypots, focuses on addressing this kind of attacks.

- *(Distributed) Denial-of-Service* (DoS) [Çeker et al., 2016; Dimitriadis, 2007]: DoS attacks are mainly to make legitimate users be denied from proper services. This attack can overwhelm or flood network flow to a targeted machine to disrupt normal functions or communications. A common DoS attack is ping flooding by leveraging ICMP (ping) packets to overwhelm a target [Peng et al., 2007]. In distributed environments, distributed DoS (DDoS) flooding attacks can be performed with the intent to block legitimate users from accessing network resource(s). With flooding packets to a target, DDoS attackers can exhaust network bandwidth or other resources, such as a server's CPU, memory, or I/O bandwidth. Unlike DoS attacks, DDoS attackers can remotely control several devices as its botnet can continuously send a large amount of traffics to a target to block the normal functions and services [Zargar et al., 2013].

- *Malware* [Krueger et al., 2012]: Malware refers to any software aiming to introduce some damage to a system component, such as server, client, or network. Honeypot-based deception approaches have been popularly used for malware analysis [Krueger et al., 2012].

- *Privacy attacks* [Shokri, 2015]: An attacker can identify private information while training and processing data to develop DD techniques. Obfuscation is often used to defend against privacy attacks.

- *Advanced persistent threats* (APT) [Cho et al., 2019a; Hofer et al., 2019; Huang and Zhu, 2019; La et al., 2016; Pawlick et al., 2015, 2018; Shi et al., 2020]: An APT attack is the most well-known sophisticated attack performing different types of attacks in the stages of the cyber kill chain (CKC) [Okhravi et al., 2013]. In particular, game-theoretic DD research has considered the APT attack but mostly focused on reconnaissance attacks.

- *Spamming* [Lee et al., 2010, 2011; Stringhini et al., 2010; Yang et al., 2014; Zhu, 2015]: Online users may receive unsolicited messages (spam), ranging from advertising to phishing messages [Rathore et al., 2017].

- *Malicious or fake profiles* [El Idrissi Younes et al., 2016] (a.k.a. Sybil attacks): OSN attackers can create numerous fake identities to achieve their own selfish goal based on others' personal information, such as e-mail, physical addresses, date of birth, employment date, or photos.

- *Fake likers by crowdturfing* [Badri Satya et al., 2016]: Human attackers, paid by malicious employers, can disseminate false information to achieve the malicious employers' purposes via crowdsourcing systems [Wang et al., 2012]. This is called *crowdturfing* and mostly aimed to spread fake information to mislead people's beliefs.

- *Loss of confidentiality, integrity, and availability (CIA)* [Casey et al., 2016; Cranford et al., 2020, 2019]: An insider attacker, called a *traitor* [Salem et al., 2008], can leak out confidential information to the outside as a legitimate user. Further, by using its legitimate status with the authorization of a target device, it can perform attacks (e.g., illegal access, modification of confidential information) that can lead to loss of integrity and availability.

- *Impersonation attacks* [Ben Salem and Stolfo, 2012]: An inside attacker can masquerade a legitimate user's identity to access resources in a target system component [Salem et al., 2008].
- *Jamming attack* [Clark et al., 2012; Nan et al., 2020b; Zhu et al., 2012]: A jamming attack can be seen as a subset of DoS attacks. The difference is that jamming attacks mainly focus on incurring traffics in wireless networks while the DoS attacks can be applicable in all networks. The jamming attack is relatively simple to archive. By keeping flooding packets, a jamming attacker can effectively block the communication on a wireless channel, disrupt the normal operation, cause performance issues, and even damage the control system [Grover et al., 2014].
- *Node compromise (NC)* [Aggarwal et al., 2016, 2017; Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; Basak et al., 2019; Bilinski et al., 2019; Durkota et al., 2015; Ferguson-Walter et al., 2019; Garg and Grosu, 2007; Horák et al., 2017; Kiekintveld et al., 2015; La et al., 2016; Nan et al., 2019, 2020a; Pawlick and Zhu, 2015; Pawlick et al., 2018; Píbil et al., 2012; Wagener et al., 2009; Xi and Kamhoua, 2020]: Some research does not specify the details of attack process. The authors only use "device compromising" to represent an attack. Some research discusses that an attacker can probe a target before attacking [Aggarwal et al., 2016, 2017; Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; Ferguson-Walter et al., 2019; Garg and Grosu, 2007; Kiekintveld et al., 2015; La et al., 2016; Nan et al., 2019, 2020a; Píbil et al., 2012; Wagener et al., 2009; Xi and Kamhoua, 2020] while others only discuss the attacking actions [Basak et al., 2019; Bilinski et al., 2019; Durkota et al., 2015; Horák et al., 2017; Pawlick and Zhu, 2015; Pawlick et al., 2018].
- *Web attack* [El-Kosairy and Azer, 2018]: An attacker can leverage existing vulnerabilities in web applications to gather sensitive data and gain unauthorized access to the web servers.
- *Zero-day attack* [Whitham, 2016, 2017]: An attacker can exploit the vulnerability period where an unknown vulnerability is not mitigated by a defense system. For example, before a vulnerability is patched by the system, the attacker can exploit the vulnerability to launch the attack, which is called zero-day attack. Often, DD along with moving target defense (e.g., network address or topology shuffling [Djamaluddin et al., 2018]) can mitigate this attack effectively.

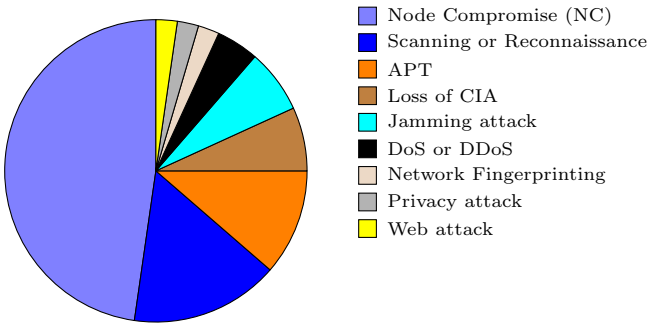


Figure 2.2: Types and frequency of attacks considered in the existing game-theoretic defensive deception approaches.

In Figs. 2.2 and 2.3, we summarized the number of game-theoretic or ML-based DD techniques that handled each type of attacks discussed in this paper. Note that the surveyed papers in this work

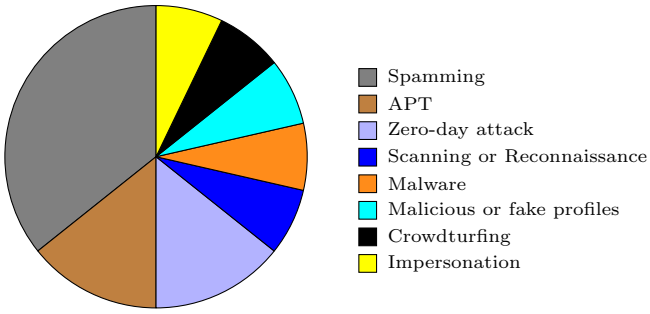


Figure 2.3: Types and frequency of attacks considered in the existing machine-learning defensive deception approaches.

were collected based on the following keywords: cyber deception, DD, game-theoretic DD, and machine learning-based DD. The surveyed papers in this work were published from 1982 to 2020. As shown in Fig. 2.2, a majority of game-theoretic DD techniques considered node capture or compromise most and scanning (or reconnaissance) attack as the second most. As illustrated in Fig. 2.3, ML-based DD techniques considered spamming or malicious/fake profile attacks most as most ML-based approaches are mainly used to detect attacks in honeypots.

2.7 Application Domains for Defensive Deception Techniques

In this section, we provide how existing DD techniques have been studied in different network environments or platforms. To be specific, we described the characteristics of each network environment and corresponding DD techniques using game-theoretic or ML-based approaches, attacks considered, and metrics and experiment testbeds used.

2.7.1 Non-Domain Specific Applications

An enterprise network is a common system that is homogeneously configured to be operated in a static configuration [Kandula et al., 2009]. Hence, an attacker can easily plan and perform its attack successfully and accordingly penetrate the system without experiencing high difficulty but using dynamic strategies to defeat defense strategies.

DD Techniques

The common defensive deception (DD) techniques used in this network environment include various types of honey information, such as fake honey files [Whitham, 2016] or honeypots [Çeker et al., 2016; Pawlick and Zhu, 2015; Pawlick et al., 2018; Píbil et al., 2012; Rowe et al., 2006; Zhu et al., 2013].

Main Attacks

In enterprise networks, the following attacks have been countered by game-theoretic (GT) or ML-based DD, including insider threat and sophisticated adversaries, such as APTs [Horák et al., 2017], zero-day attacks [Whitham, 2016, 2017], reverse engineering attacker using security patch Avery and Spafford [2017], or DoS attack [Çeker et al., 2016].

Key GT and ML Methods

For game-theoretic DD techniques deployed in this network environment, signaling games [Çeker et al., 2016; Pawlick and Zhu, 2015; Pawlick et al., 2018] or Stackelberg game [Zhu et al., 2013] have been commonly used. Some different game types, such as Metagames/Expected Utility [Píbil et al., 2012] or Bayesian Game using Bayesian equilibrium [Çeker et al., 2016] have been also considered to develop DD techniques in this network context. For ML-based DD techniques, Natural Language Processing (NLP) techniques [Whitham, 2016, 2017] have been used to develop honey files.

Pros and Cons

A majority of game-theoretic DD approaches have considered an enterprise network without providing the details of a network model with the aim of examining a proposed game-theoretic framework with solid theoretical analysis, such as identifying optimal strategies or solutions based on Nash Equilibria. However, highly theoretical game-theoretic DD approaches may not provide concrete details on how to design DD techniques considering the characteristics of each network environment in terms of resource availability, system security and performance/services requirements, or network and environmental dynamics. In addition, some general game-theoretic DD approaches do not specify a DD technique, but can provide a general idea of modeling a DD technique. However, this lacks details, showing limited applicability in real systems. Further, enterprise networks are mostly highly complex systems which may need to deal with a wide range of attack types. Due to the abstract nature of game-theoretic approaches proposed for the enterprise networks, they do not provide how to simulate specific types of attacks.

2.7.2 General Cyber-Physical Systems (CPSs)

A CPS is a system that can provide communications between humans via cyber capabilities and physical infrastructure or platforms [Poovendran, 2010]. The CPS has been advanced with the cyber capabilities of communications, networking, sensing, and computing as well as physical capabilities with materials, sensors, actuators, and hardware. The CPS is uniquely distinguished from other platforms due to the presence of both cyber and physical aspects of systems and their coordination [Poovendran, 2010]. CPSs include wireless sensor networks, Internet of Things (IoT), software-defined networks (SDNs), and industrial control systems (ICSs) [Kathiravelu and Veiga, 2017; Serpanos, 2018]. Although we discuss each of them in separate sections, we include this section particularly to discuss DD techniques that are designed for 'general CPSs' without specifying a particular platform.

DD Techniques

Although DD techniques have been deployed in CPS environments, their applicability seems not limited to only CPS environments, rather applicable in any environments. We found bait information and honeypots are applied to the CPS environments for introducing confusion or uncertainty to attackers or luring them to honeypots for collecting attack intelligence [Hofer et al., 2019; Kiekintveld et al., 2015; Sayin and Başar, 2019].

Main Attacks

Game-theoretic or ML-based DD techniques in this domain mainly handled node compromise [Kiekintveld et al., 2015], scanning/reconnaissance [Sayin and Başar, 2019], or APT [Hofer et al., 2019]

attacks.

Key GT and ML Methods

A signaling game theory was used for a defender to identify an optimal strategy using DD techniques [Sayin and Başar, 2019]. In addition, how to select honeypots to maximize confusion or uncertainty to attackers is also studied in [Kiekintveld et al., 2015]. To create decoy devices that look like real, RNN (Recurrent Neural Networks) was also used based on observations of device behaviors for a year in a CPS [Hofer et al., 2019].

Pros and Cons

Since a CPS is a popular network environment and commonly used in real systems, developing DD techniques for the CPS can have high applicability in diverse CPS contexts. However, the DD techniques developed for the CPS did not consider much about its unique characteristics, such as the features of having both cyber and physical aspects, and challenges of the environment itself. Hence, we do not observe much differences between the CPS's DD and the enterprise network's DD techniques. In addition, physical honeypots are sometimes recommended when dealing with highly intelligent attackers. However, such honeypot deployment and maintenance often come with higher deployment and management cost.

2.7.3 Cloud-based Web Environments

Cloud-based web applications become more common and popular than ever to deal with critical tasks which bring security as a prime concern. DD is one of promising directions to defend against web attacks [Efendi et al., 2019]. DD techniques can be utilized to defend against advanced web attacks that cannot be well handled by existing signature or anomaly-based intrusion detection or prevention mechanisms [Efendi et al., 2019].

DD Techniques

In this environment, honey-X [El-Kosairy and Azer, 2018] or fake signals [Pawlick et al., 2015] are used as DD techniques, such as honeywebs, honey files, honey tokens, or honeypots.

Main Attacks

Game-theoretic or ML-based DD techniques in cloud environments have mainly considered general web attacks [El-Kosairy and Azer, 2018], or multi-staged APT attacks [Pawlick et al., 2015].

Key GT and ML Methods

Signaling games with complete, imperfect information have been used to model attack-defense games [El-Kosairy and Azer, 2018; Pawlick et al., 2015].

Pros and Cons

While game-theoretic approaches proposed a generic game-theoretic DD framework [El-Kosairy and Azer, 2018; Pawlick et al., 2015], ML-based obfuscation technique is unique as it aims to defend against

privacy attacks to deal with adversarial ML examples. However, we do not observe concrete design features of each DD technique only for the cloud computing.

2.7.4 Internet-of-Things (IoT)

IoT network environments become highly popular and have been recognized as one of CPSs with the capability to provide effective services to users. IoT has been also specialized for particular domains such as Internet-of-Battle-Things (IoBT) [Kott et al., 2016], Industrial-Internet-of-Things (IIoT) [Pinto et al., 2017], or Internet-of-Health-Things (IoHT) [Rodrigues et al., 2018]. In addition, IoT embraces ideas of several CPS network environments, such as wireless sensor networks (WSNs), mobile ad hoc networks (MANETs), or smart city environments consisting of sensors and other intelligence machines [Kocakulak and Butun, 2017]. IoT has been popular considered as a main platform of deploying cyberdeception technologies [Alshammari et al., 2020].

DD Techniques

As in other environments, honeypot enabled IoT networks have been mainly considered by solving the optimal deployment of honeypots [Çeker et al., 2016; La et al., 2016].

Main Attacks

In IoT environments, game-theoretic or ML-based DD techniques aimed to defend against reconnaissance and probing attacks, DoS attacks [Çeker et al., 2016], packet dropping attacks [Çeker et al., 2016], or APT attacks [La et al., 2016; Xi and Kamhoua, 2020].

Key GT and ML Methods

Bayesian games with incomplete information and meta games are considered for a player who is unsure of a type of attackers [La et al., 2016]. IoTs in battlefields is referred to IoBTs where deception games introduced in [Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; Nan et al., 2019; Xi and Kamhoua, 2020]

Pros and Cons

Since honeypots are fake nodes mimicking the behavior of a regular node, adding a honeypot does not change the hierarchy of the IoT network or the interface of IoT gateways. A game-theoretic honeypot technique is the only technique applied to this domain. However, the IoT naturally can generate a large amount of data for ML-based DD techniques which can create real-like decoys or enhance detecting inside and outside attackers.

Table 2.5: Application Domains of game-theoretic or Machine Learning-based Defensive Deception Techniques

Application domain	**Defensive deception techniques**	**Main attacks**	**Key GT/ML techniques**	**Pros**	**Cons**

Table 2.5: Application Domains of game-theoretic or Machine Learning-based Defensive Deception Techniques

Application domain	**Defensive deception techniques**	**Main attacks**	**Key GT/ML techniques**	**Pros**	**Cons**
Non-domain specific environment: GT-Based [Aggarwal et al., 2016, 2017; Al Amin et al., 2019b; Basak et al., 2019; Bilinski et al., 2019; Casey et al., 2016, 2015; Çeker et al., 2016; Cho et al., 2019a; Durkota et al., 2015; Ferguson-Walter et al., 2019; Garg and Grosu, 2007; Hespanha et al., 2000; Horák et al., 2017; Huang and Zhu, 2019; Mohammadi et al., 2016; Pawlick and Zhu, 2015; Pawlick et al., 2018; Píbil et al., 2012; Rahman et al., 2013; Shi et al., 2020; Thakoor et al., 2019; Wagener et al., 2009; Wang et al., 2020; Yin et al., 2013; Zhu et al., 2012, 2013]; ML-Based [Abay et al., 2019; Badri Satya et al., 2016; Ben Salem and Stolfo, 2012; El Idrissi Younes et al., 2016; Hofer et al., 2019; Krueger et al., 2012; Lee et al., 2010, 2011; Nanda et al., 2016; Stringhini et al., 2010; Whitham, 2016, 2017]	Honey files, honeypots, honeywebs, honeynets, honey patch, honey profiles, honey surface, honeybots, HMAC, obfuscation, deceptive signal, Fake Identities, deceptive network flow, social honeypots, bait-based deception, fake services	Zero-day attacks, APTs, masquerade attacks, reverse engineering for security patch, DoS attack, optimal inference attacks, reconnaissance attack, spam, social spam, malicious profiles	Signaling games, Stackelberg game, Metagames/-Expected Utility, Bayesian Game using Bayesian equilibrium, subgame perfect Nash Equilibrium (SPNE), NLP techniques, hypergame, Stackelberg game, one-sided POSG, BayesNet, Decision Table, Naïve-Bayes, Reinforcement Learning, SVM	Most game-theoretic DD approaches propose a general deception game framework without specifying a certain domain environment, which can have high applicability regardless of domain environments.	Due to the nature of a general approach, there is high overhead to cover the general approach to a domain-specific approach.
General Cyber-Physical Systems: GT-Based [Kiekintveld et al., 2015; Sayin and Başar, 2019]; ML-based [Hofer et al., 2019]	Crafted bait information, honeypot	Reconnaissance attacks, node compromise, APT	Signaling game theory, RNN	Due to a wide range of CPS applications, developing DD techniques will have high values and applicabilities in diverse CPS environments.	DD techniques for the CPS do not reflect unique challenges of the CPS environments. Physical honeypot deployment and maintenance often come with higher deployment and management cost.

Table 2.5: Application Domains of game-theoretic or Machine Learning-based Defensive Deception Techniques

Application domain	**Defensive deception techniques**	**Main attacks**	**Key GT/ML techniques**	**Pros**	**Cons**
Cloud web-based environments: GT-Based [El-Kosairy and Azer, 2018; Pawlick et al., 2015]	Honeywebs (using honeytokens, honey files, decoy resources, honeypots), deceptive signals, and obfuscation	General web attacks, APT attacks, privacy attacks	Signaling game theory, a suite of ML classifiers	Multiple DD frameworks are provided with high applicability to deal with a wide range of web attacks.	Some detailed designs should be considered to deal with unique challenges of cloud environments.
Internet-of-Things: GT-Based [Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; La et al., 2016; Nan et al., 2019; Xi and Kamhoua, 2020]	Honeypot	Reconnaissance and probing attacks, DoS attacks, packet dropping attacks, APTs	Bayesian games, meta games, signaling game with perfect Bayesian equilibrium	Since honeypots are fake nodes mimicking the behavior of a regular node, adding a honeypot does not change the hierarchy of the IoT network or the interface of IoT gateways.	A game-theoretic honeypot technique is the only technique that applied to this domain. However, the IoT naturally can generate a large amount of data for ML-based DD techniques which can create decoys that looks like real or enhance detect inside and outside attackers.

Table 2.5: Application Domains of game-theoretic or Machine Learning-based Defensive Deception Techniques

Application domain	**Defensive deception techniques**	**Main attacks**	**Key GT/ML techniques**	**Pros**	**Cons**
Software-defined networks: GT-Based [Al Amin et al., 2019a, 2020; Chiang et al., 2018; Mao et al., 2019]	Honeypot	Reconnaissance attacks	Bayesian game	The key advantage of using SDN-based DD approaches is their easy deployability. For example, the existence of an SDN controller enables easily camouflaging a network topology or hiding vital nodes through flow traffic control.	Most existing approaches use a single SDN controller, which expose a single point of failure.
Wireless networks: GT-Based [Clark et al., 2012; Dimitriadis, 2007; Nan et al., 2020b]	Deceptive network flow, honeynet	Jamming attacks, 3G core network attacks	A non-cooperative non-zero-sum static game	Specific design features to deal with key concerns of wireless networks are provided.	If a honeynet architecture is heavily dependent upon the accuracy of deployed honeypots, the formulated game may not work under different deployment settings.
Online social networks: ML-Based [Badri Satya et al., 2016; Lee et al., 2010, 2011; Stringhini et al., 2010]	Social Honeypot, honey profiles	Fake liker (crowdturfing), social spammers, spamming	Supervised ML, ML toolkit, SVM	Learning the characteristics of attack behavior and attract spammers in OSN	Detectors are applied on static datasets to evaluate the detection of spammer and honey profiles are not generic and depend on the platform type.

2.7.5 Software-Defined Networks (SDNs)

The SDN paradigm separates data plane processing (e.g., packet forwarding) from control-plane processing (e.g., routing decisions) [MacFarland and Shue, 2015]. The OpenFlow protocol [Benton et al., 2013] acts as an API between network switches and a logically centralized decision maker, called the *OpenFlow controller*. In this protocol, network switches cache data-plane flow rules. When a switch receives a packet and does not know how to forward it according to its cached rules, it sends an "elevation" request containing the original packet and a request for the guidance to the controller. The controller examines the packet and sends a set of rules that the switch should add to the data plane cache for forwarding packets [Dixit et al., 2013]. deception techniques proposed in [Al Amin et al., 2019a, 2020] were designed to secure SDN.

DD Techniques

Honeypots are popularly used as a defense strategy in game-theoretic DD framework [Chiang et al., 2018; Mao et al., 2019] for taking dynamic, adaptive defense strategies.

Main Attacks

Common attack behaviors considered include scanning or reconnaissance attacks to obtain information and intelligence towards a target system by scanning network addresses (e.g., IP or port numbers) aiming to obtain defense information, network mapping, and inside information [Chiang et al., 2018]

Key GT and ML Methods

Bayesian game theory has been popularly adopted under various conditions [Mao et al., 2019], such as imperfect information, incomplete information, information sets, and perfect Bayesian equilibrium.

Pros and Cons

The key advantage of using SDN-based DD approaches is their easy deployability. For example, the existence of an SDN controller enables easily camouflaging a network topology or hiding vital nodes through flow traffic control. Most existing approaches use a single SDN controller, which exposes a single point of failure.

2.7.6 Wireless Networks

Wireless communications are everywhere these days and more common than wired communications due to their easy deployment and efficiency. However, when network resources are scarce, bandwidth constraints or unreliable wireless medium become main issues to be resolved. Game-theoretic or ML-based DD techniques are also proposed to defend against various types of attacks exploiting vulnerabilities of wireless network environments.

DD Techniques

A deceptive network flow is proposed to generate the flow of random dummy packets in multihop wireless networks [Clark et al., 2012]. A honeynet architecture address mobile network security (3G networks) [Dimitriadis, 2007] was developed to enhance the security of the core network of mobile

telecommunication systems. In [Dimitriadis, 2007], a gateway was designed to control and capture network packets as well as investigate and protect other information systems from attacks launched from potentially compromised systems inside the honeynet. DD is used to transmit fake information over fake channels to mitigate jamming attacks in wireless networks [Nan et al., 2020b].

Main Attacks

Game-theoretic or ML-based DD techniques have defended against jamming attacks [Clark et al., 2012] in multihop wireless networks and 3G core network. The goal of adversaries is to compromise servers or gateways that support nodes by providing general packet radio services [Dimitriadis, 2007].

Key GT and ML Methods

A non-cooperative, non-zero-sum static game is used to model the interactions between an attacker and a defender [Dimitriadis, 2007]. Deceptive routing paths are also designed based on a two-stage game using Stackelberg game theory [Clark et al., 2012]. Deceptive power transmission allocation based on game theory is applied to defender wireless networks against jamming attacks [Nan et al., 2020b].

Pros and Cons

Specific design features to deal with key concerns of wireless networks, such as multihop communications or mobile wireless security, are helpful to implement real systems based on the given game-theoretic DD technologies. However, if a honeynet architecture heavily depends upon the accuracy of deployed honeypots, the formulated game framework may not guarantee the same level of effectiveness under different deployment settings as in Dimitriadis [2007].

2.7.7 Online Social Networks

Due to the large scales of OSNS and their significant influence in social, economic, and political aspects, AI and ML communities have recognized high challenges in developing DD techniques in the OSN platforms. Hence, there have been significant efforts made to secure OSNS from malicious users (e.g., fake users and spammers) by developing various social DD techniques [Badri Satya et al., 2016; Lee et al., 2010, 2011; Stringhini et al., 2010].

DD Techniques

The authors in [Badri Satya et al., 2016; Lee et al., 2010, 2011] specifically used social honeypots primarily to attract and identify attackers. Social honeypots can learn the behavioural models of malicious users and collect a spammer's preferences. Stringhini et al. [2010] analyzed 900 honey profiles for detecting spammers across MySpace, Facebook, and Twitter for classification and identification purposes.

Main Attacks

OSNS are mainly targeted by fake likers and spammers that would like to exploit the existing dynamics of OSNS to achieve a specific goals [Badri Satya et al., 2016; Lee et al., 2010, 2011; Stringhini et al., 2010].

Key GT and ML Methods

Traditional ML tool kits such as SVM have been used to analyze the data collected by social honeypots and honey profiles.

Pros and Cons

Social honeypots allows learning the characteristics of attack behavior and honey files can attract spammers to protect real files from them in OSN. However, detectors are applied on static datasets to evaluate the detection of spammers. In addition, honey profiles are not generic and should be customized to a specific platform type.

We summarized our discussions of this section in Table 2.5.

2.8 Evaluation of Defensive Deception Techniques: Metrics and Testbeds

In this section, we survey what types of metrics are used to measure the effectiveness and efficiency of DD techniques. In addition, we address what types of testbeds are employed to evaluate the effectiveness and efficiency of DD techniques.

2.8.1 Metrics

In the literature, the following metrics are used to measure the **effectiveness** of existing DD techniques:

- *Detection accuracy* [Abay et al., 2019; Badri Satya et al., 2016; Basak et al., 2019; Ben Salem and Stolfo, 2012; Chiang et al., 2018; El Idrissi Younes et al., 2016; Lee et al., 2010, 2011; Pawlick et al., 2018, 2019; Xu et al., 2012; Yang et al., 2014]: This is often measured by the AUC (Area Under the Curve) of ROC (Receiver Operating Characteristic). AUC is used to measure the accuracy of detection by showing TPR (True Positive Rate) as FPR (False Positive Rate) increases. Abay et al. [2019] used a classifier accuracy as a metric to evaluate the effectiveness of the developed Honey data to deceive the attacker. Badri Satya et al. [2016] and Chiang et al. [2018] used an algorithm to discover fake Liker in social networks. The authors in [Ben Salem and Stolfo, 2012; Xu et al., 2012] evaluated a masquerade attack detector based on AUC. ROC is also used to measure detection accuracy of social honeypot-based approach against social spammers [Lee et al., 2010, 2011] and malicious account [El Idrissi Younes et al., 2016; Yang et al., 2014] in online social networks. Pawlick et al. [2018, 2019] evaluated their deception mechanism based on different metrics, including the detector accuracy developed by the adversary to discover deception.

- *Mean time to detect attacks* [Basak et al., 2019]: The effectiveness of a detection mechanism is also measured by how early their deception technique can capture the attacker.

- *Utility* [Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; Casey et al., 2016; Clark et al., 2012; Garg and Grosu, 2007; Horák et al., 2017; Huang and Zhu, 2019; Kiekintveld et al., 2015; Milani et al., 2020; Mohammadi et al., 2016; Nan et al., 2019, 2020a,b; Pawlick et al., 2018, 2019; Píbil et al., 2012; Thakoor et al., 2019; Tsemogne et al., 2020; Wang et al., 2020; Xi and Kamhoua, 2020; Zhang et al., 2019; Zhu et al., 2012]: In game-theoretic DD techniques, a player (either an attacker or a defender)'s utility (or payoff) is considered as one of key metrics to evaluate a deception game

between the attacker and defender. Commonly, the utility of taking a DD strategy is formulated based on the deployment cost and the confusion increased to an attacker. Some signaling games model the interactions between an attacker, a benign client, and a defender. As a result, some of these studies considered the impact to normal as another cost of deception technology [Rahman et al., 2013; Wang et al., 2020]. A defender's expected utility is a common metric to be maximized as the system's objective, as shown in game-theoretic or ML-based DD techniques in [Al Amin et al., 2019a, 2020; Garg and Grosu, 2007; Horák et al., 2017; Huang and Zhu, 2019; Kiekintveld et al., 2015; Milani et al., 2020; Nan et al., 2020a; Pawlick et al., 2018, 2019; Píbil et al., 2012; Thakoor et al., 2019; Tsemogne et al., 2020; Xi and Kamhoua, 2020; Zhang et al., 2019; Zhu et al., 2012].

- *Probabilities of an attacker taking certain actions* [Bilinski et al., 2019; Cranford et al., 2020, 2019; Nan et al., 2019; Rahman et al., 2013; Wagener et al., 2009]: An attacker's actions in proceeding attacks are also considered as a metric to measure the effectiveness of DD (e.g., honeypots). For example, in [Wagener et al., 2009], the following probabilities of an attacker's action are considered: a probability of an attacker retrying a failure command, a probability of an attacker choosing an alternative strategy, and a probability of an attacker leaving a game. The probability of an attacker to successfully control the network and overcome the deception is used to evaluate the effectiveness of cyberdeception in [Bilinski et al., 2019; Rahman et al., 2013] and the probability of launching an attack as in [Cranford et al., 2020, 2019].

In addition, we also found the following metrics to capture the **efficiency** of the surveyed DD techniques:

- *Round trip-time* [Aggarwal et al., 2019; Araujo et al., 2014; Basak et al., 2019; Borello and Mé, 2008; Cai et al., 2009; Chan and Yang, 2004]: The key role of DD is to delay the adversary processes [Araujo et al., 2014]. Increasing the time required to crack an obfuscated code is the evaluation metric adopted in [Cai et al., 2009; Chan and Yang, 2004], as well as the complexity of the deobfuscation process [Borello and Mé, 2008]. The HackIt deception tool in [Aggarwal et al., 2019] successfully increased the time taken by hackers to exploit a system. An attacker's deception detection time is of great significance in evaluating the effectiveness of cyber deception [Basak et al., 2019].

- *Runtime* [Avery and Spafford, 2017; Avery and Wallrabenstein, 2018; Shi et al., 2020]: This evaluates the cost of automated deception and obfuscation techniques. The performance of the deception technique based on fake patches [Avery and Spafford, 2017; Avery and Wallrabenstein, 2018] was evaluated using this metric. Similarly, the learning time also evaluates the learning the attacker's behavior model from attack data introduced in [Shi et al., 2020].

In Figs. 2.4 and 2.5, we summarized the types and frequency of metrics measuring performance and security of the surveyed DD techniques. As shown in Fig. 2.4, a player's utility is the most common metric among all metrics used for the surveyed game-theoretic DD techniques. On the other hand, in Fig. 2.5, ML-based DD approaches heavily relied on detection accuracy using such metrics as AUC, TPR, and FPR metrics.

2.8.2 Evaluation Testbeds

In this section, we classify evaluation testbeds of the existing game-theoretic or ML-based DD techniques surveyed in this work in terms of the four classes: those based on probability, simulation, emulation, and

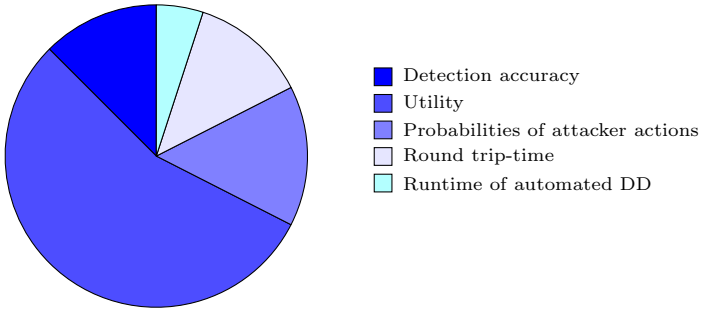


Figure 2.4: Types and frequency of metrics measuring performance and security of game-theoretic defensive deception techniques.

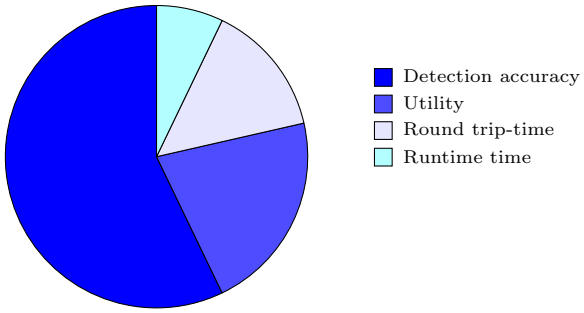


Figure 2.5: Types and frequency of metrics measuring performance and security of ML-based defensive deception techniques.

real testbed. We discuss each of the evaluation testbeds and discuss the general trends observed from the survey.

Probability Model-Based Evaluation

Stochastic Petri Nets (SPN) probability models was used to evaluate the performance of an integrated defense system that combines multiple deceptive defense techniques [Cho and Ben-Asher, 2018]. In addition, the SPN is used to measure the performance of the deceptive defender and the attacker in a hypergame model [Cho et al., 2019a].

Simulation Model-Based Evaluation

To simulate benign network traffic, Rahman et al. [2013] generated network flow by the Internet Traffic Archive [Danzig et al., 2008], using *Nmap* to test their deception technology. Chiang et al. [2016, 2018] simulate their deception system and combine it with their proposed work. Thakoor et al. [2019] used the CyberVan [Chadha et al., 2016; Pham et al., 2020] as a network simulation testbed to simulate their game-theoretic cyberdeception techniques. Al Amin et al. [2019b] and Bilinski et al. [2019] proposed a deceptive system which was simulated using a Markov decision processes. Several studies also simulated ML-based deceptive algorithms or classifiers to examine its accuracy [Abay et al., 2019; Al Amin et al., 2019a,b, 2020; Al-Shaer et al., 2019; Badri Satya et al., 2016; Ben Salem and Stolfo, 2012; Shi et al., 2020]. A masquerade attack is simulated to evaluate the developed detection techniques based on deception [Ben Salem and Stolfo, 2012; Xu et al., 2012]. Various types of game-theoretic based deception

framework have been evaluated based on simulation models [Anwar et al., 2020; Anwar and Kamhoua, 2020; Anwar et al., 2019; Avery and Spafford, 2017; Basak et al., 2019; Bilinski et al., 2019; Cadar et al., 2008; Chiang et al., 2018; Clark et al., 2012; Garg and Grosu, 2007; Horák et al., 2017; House and Cybenko, 2010; Huang and Zhu, 2019; Kiekintveld et al., 2015; Kulkarni et al., 2020; La et al., 2016; Milani et al., 2020; Mohammadi et al., 2016; Nan et al., 2019, 2020a; Pawlick et al., 2018; Píbil et al., 2012; Rahman et al., 2013; Thakoor et al., 2019; Tsemogne et al., 2020; Xi and Kamhoua, 2020; Yin et al., 2013; Zhang et al., 2019]. Simulation-based models showing a proof of concept have been developed to demonstrate the runtime of the proposed automated deception algorithm [Avery and Spafford, 2017; Cadar et al., 2008].

Emulation Testbed-Based Evaluation

Emulation network environments were proposed to dynamically place deception resources [Wang et al., 2020], deploy code obfuscation techniques on real-like decompilers [Chan and Yang, 2004], and deploy DD techniques in emulated SDN-based data center network testbed.

Real Testbed-Based Evaluation

An application for data collection has been developed [Aggarwal et al., 2016, 2017, 2019]. Experiments with human-in-the-loop with human participants (e.g., an attacker or a defender) have been conducted [Aggarwal et al., 2016, 2017, 2019], such as HackIt to evaluate the effectiveness of deception on participating hackers under different deception scenarios. Socialbots on Twitter real networks are significantly leveraged to detect and study content polluters and understand how spammers choose their spamming targets to ultimately create a malicious profile classifier [El Idrissi Younes et al., 2016; Lee et al., 2010, 2011; Yang et al., 2014]. In addition, various DD techniques are validated under real settings, such as honey-patches-based deception system [Araujo et al., 2014], location obfuscation for preserving users' privacy [Ardagna et al., 2007], and PRISMA (PRotocol Inspection and State Machine Analysis) tool capable of learning and generating communication sessions for deception [Krueger et al., 2012].

In Figs. 2.6 and 2.7, we summarized the frequency of particular testbeds used in developing game-theoretic and ML-based DD techniques surveyed in this paper, respectively. The overall trends observed were that simulation-based evaluation was dominant in both GT and ML-based DD approaches. Although ML-based DD techniques are less explored than GT-based DD techniques, it is noticeable that more than 40 percent of the ML-based DD techniques surveyed here used evaluation based on a real testbed. This is natural that ML-based approaches basically need actual datasets to analyze. Another noticeable finding is that there has been a lack of probability model-based approaches. This is because it is highly challenging to model a complex system environment characterized by many design parameters based on mathematical models.

2.9 Conclusions and Future Work

In this section, we discussed insights and lessons learned and limitations from the extensive survey conducted in this work. In addition, based on the conducted survey, we provided a set of promising future research directions.

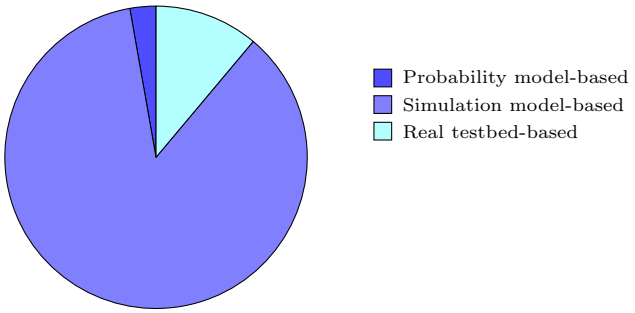


Figure 2.6: Evaluation testbeds for game-theoretic defensive deception techniques.

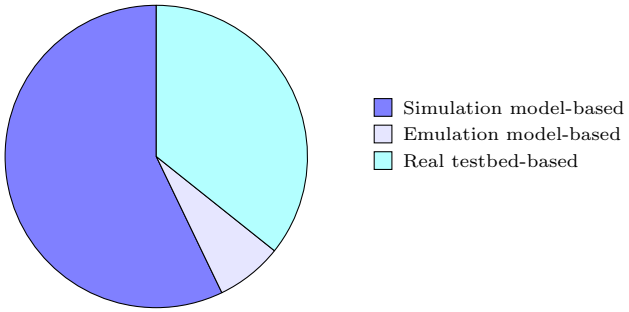


Figure 2.7: Evaluation testbeds for ML-based defensive deception techniques.

2.9.1 Insights and Lessons Learned

We now revisit and answer the research questions raised in Section 2.1.4.

RQ Characteristics: *What key characteristics of DD distinguish it from other defensive techniques?*

Answer: Unlike traditional defense mechanisms, deception requires a certain level of risk as it requires some interactions with attackers with the aim of confusing or misleading them. Particularly, if the goal of defense requires long term deception, it is inevitable to face risk. In this case, DD can be used with other legacy defense mechanisms such as intrusion prevention or detection mechanisms to avoid too high risk. Moving target defense (MTD) or obfuscation techniques share a similar defense goal with DD, such as increasing confusion or uncertainty for attackers. However, unlike MTD or obfuscation which changes system configuration or information based on the existing resources of a system, DD can create false objects or information with the aim of misleading an attacker's cognitive perception or forming a misbelief for the attacker to choose a sub-optimal or poor attack strategy.

RQ Metrics: *What metrics are used to measure the effectiveness and efficiency of the existing game-theoretic or ML-based DD techniques?*

Answer: As surveyed in Section 2.8 along with Figs. 2.4 and 2.5, the effectiveness of GT-based deception is mainly measured based on utility. On the other hand, ML-based deception is mainly evaluated according to the accuracy of classifiers or intrusion detectors. The existing metrics observed

in the surveyed literature for game-theoretic or ML-based DD approaches do not capture the direct impact of DD. That is, they do not quantify the uncertainty or confusion induced by the different deceptive techniques. Moreover, new metrics should be developed to measure the effectiveness of deception in misleading the attackers, such as lightweight deception with minimal or low defense cost in deploying or maintaining a specific deception technique.

According to Section 2.8 along with Figs. 2.4 and 2.5, GT-based deception techniques have been mainly evaluated in terms of the defender's utility. However, ML-based deception techniques focused on improving the accuracy of classifiers or intrusion detectors. However, the existing evaluation of GT-based DD techniques rarely consider utility analyses based on the losses and gains in terms of the timing of using different types of deceptions. In addition, although it is important to use metrics capturing deception cost introduced by its deployment and maintenance and performance degradation caused by running deception techniques, there has been much less effort to use those metrics. Furthermore, measuring the extent of attackers deviating from the optimal course of actions can be another meaningful metric to evaluate the effectiveness of games with DD.

RQ Principles: *What key design principles help maximize the effectiveness and efficiency of DD techniques?*

Answer: It is critical to make three key design decisions, which are what-attacker-to-deceive, when-to-deceive, and how-to-deceive. These three design decisions should be determined based on critical tradeoffs between effectiveness and efficiency of a developed DD technique. The what-attacker-to-deceive question should be answered based on what attacks are targeted by a defender and what attacks should be commonly handled in each application domain because each application domain generates distinctive challenges. When-to-deceive is related to the fact that depending on when a DD technique is used to deal with attackers in a certain stage of the cyber kill chain, the extent of deployability, effectiveness, efficiency, and cost of using a DD technique can be affected. For example, when-to-deceive can be decided according to how complex the deployment of such DD technique or how effective the DD technique is in terms of the timing. In addition, it can consider how costly deception is in terms of the availability of the resources. In some settings, deception is needed only temporarily to delay an attack until a defense system needs to take enough time for its reconfiguration or deploying/performing a certain defense operation (e.g., deceive an attacker until the MTD is executed to fully reconfigure an OS). Each type of a DD technique has its own unique characteristics and corresponding expected outcomes. Given what attack to deal with by determining what-to-deceive and when to deceive the attacker (i.e., in the attack stage), the how-to-deceive question is related to answering (1) what DD techniques are better than other DD techniques under the available resources and (2) what quality of deception is most appropriate to deal with specific attacks? For example, if a defense system can afford to deploy high-quality honeypots to deal with highly intelligent attackers, high-interaction honeypots would be a good choice. However, if the defense system has a limited budget or resources available and wants to deploy a lightweight DD technique temporarily (e.g., before a new setting of MTD is fully reconfigured), a honey file or honey token can be used with low cost but they can be detected by intelligent attackers soon. Therefore, depending on the defense cost budget, the system's resource availability, or the expected outcome for a DD technique to be used, a different type of DD techniques with a different level of deception quality can be adopted to meet the expected effectiveness and efficiency.

RQ GT: *What are the key design features when a DD technique is devised using game theory?*

Answer: Game-theoretic DD techniques mainly adopt an attack-defense framework where the attacker and defender interact with a conflict of interest scenario, which often use a form of zero-sum games. However, as multiple players in each party (i.e., multiple attackers for collusive attacks or multiple defenders for cooperative defense) can be involve in a attack-defense game, general-sum games are often considered as well. To be more specific, we also summarize a game type used in each paper that proposes game-theoretic DD in Table 2.3. Game-theoretic DD focuses on identifying optimal defense strategies to confuse or mislead an attacker by increasing uncertainty. Three design features are: (i) each player needs to have a clear goal and a corresponding clear utility function that reflects the player's intent, tactics, and resources; (ii) modeling and simulating how to increase confusion or uncertainty via different types of DD techniques should be elaborated; and (iii) metrics to measure effectiveness and efficiency should be articulated.

RQ ML: *What are the key design features when a DD technique is developed using ML?*

Answer: ML-based DD techniques learn and detect attack behaviors. However, ML can improve DD by considering the following: (i) We need to consider what types of datasets use to develop DD techniques. To develop believable fake objects presupposes good datasets for mimicking real objects and evaluating fake objects. Along these lines, targeted attack datasets should be used to model attackers and develop honey resources; (ii) ML-based DD approaches should adopt relevant metrics to capture their effectiveness and efficiency. So far, classification accuracy is the only metric to capture ML-based honeypots. Additional metrics should be developed to capture the quality of ML-based DD techniques; and (iii) A fully automated deception is highly desirable for ML-based DD techniques, such as generating deceptive traffics and network topologies.

RQ Applications: *How should different DD techniques be applied in different application domains?*

As the enterprise network is a complex domain, multiple honey-X techniques, possibly in combination, are applicable. For simpler domains, an easy-to-deploy honeypot is effective and widely used. However, developing high quality DD techniques often incurs more cost or higher complexity. For example, deploying more honeypots may increase the chance to mislead an attacker. However, too many honeypots also increase unnecessary costs and confuse the administrator or legitimate users. Depending on applications, DD techniques should be applied in an automated manner. For example, online social networks require deception for detecting attacks or collecting attack intelligence since social spammers do not aim to gain control but to influence other users in the network.

As you can observe in our extensive survey, many more game-theoretic DD studies have been conducted while ML-based DD studies are limited to developing honey-X-based approaches aiming to detect more attacks. Via our extensive survey, we could clearly observe the main benefit of using game-theoretic DD techniques is their capability to allow players to make strategic decisions and take various strategies upon observed dynamics of a game over time. In addition, since game theory provides a mathematical framework to model the interactions between attackers and defenders, it can be more extensively applied in developing various types of DD techniques as a general framework. However, as we also observed in existing work, although game-theoretic DD has powerful capability with solid mathematical proofs in developing DD techniques, their validation in security and performance has been heavily limited to the theoretical or simulation-based verification. However, ML-based DD approaches have been applied to

mainly honey-X applications, mostly developing honeypots. Unlike other DD techniques, the main goals of honeypots are two-fold: protecting system assets by creating fake objects, which are honeypots and detecting attackers by luring them to the honeypots. Although the original purpose of DD more focuses on protecting the system assets, which is more like passive defense, the effectiveness of honeypots has been mainly measured its role in detecting attacks in terms of the detection accuracy. That is, ML-based honeypots have been developed by creating a fake object which looks like real with the aim of attracting more attackers. Compared to GT-based DD approaches, ML-based DD approaches are less dynamic because ML techniques are mainly used to improve the quality of deception in its development time and as a classifier to detect attacks. Although reinforcement learning as one of promising ML techniques as been used to develop a DD technique along with game theory [Al Amin et al., 2019b; Wang et al., 2020], hybrid approaches incorporating ML into game theory to develop DD techniques are rarely taken. We discuss examples of how these can be incorporated into a hybrid DD technique in the future research directions discussed in Section 2.9.3.

2.9.2 Limitations of Current Defensive Deception

From our extensive survey on the state-of-the-art defensive deception (DD) techniques using game theory and ML, we found the following limitations of the existing approaches:

- Although many DD techniques have used game-theoretic approaches, they have mainly focused on theoretical validation based on Nash equilibrium and the identification of optimal solutions, in which attacks are not often detailed enough but simply considered attacks compromising other nodes. Often, attacks are complicated and performed with more than one episode or multistage over time, such as APT attacks. Therefore, although we can benefit from modeling simple attack processes, such as active reconnaissance or insider attack, it is still challenging to deploy the game-theoretic DD in real systems and model attacks based on a complex cyber kill chain.

- We observed that an enterprise networking is the only domain that is well developed. Research effort in other domains has been limited and mainly focused on reconnaissance attacks. IoT and SDN environments can generate large amounts of traffic flow data, which can be used to train ML models to identify attackers. However, current DD approaches for those domains lack using ML.

- Due to high complexity of action spaces, most game-theoretic DD has dealt with a limited set of actions for an attacker and defender, which cannot accurately model an attack or defensive process when the attacker can perform multiple attacks following the multiple stages of attack processes, such as cyber kill chain in APT attacks.

- When multiple attacks arrive in a system, considering the interactions only between one attacker and one defender is not sufficient to capture the practical challenge. How to interpret a response towards a certain action (e.g., who is responding to what action when there are multiple attackers) is not clear although it is critical to model the interactions between the attacker and defender.

- A majority of game-theoretic DD approaches are studied at an abstract level by using game-theoretical analysis without explicitly addressing design challenges derived from the network environment or platform the DD technique is deployed. When some ideas are adopted and applied in a certain network environment, it is quite challenging to deploy a game-theoretic work onto a specific platform due to a lack of detail regarding the deployment process.

- The accuracy of ML-based deception techniques depends on data availability regarding the attacker identity, techniques, and targets. Practically, such data are not available for the defender, which significantly limits training ML-based classifiers or detectors. Moreover, such models tend to assume that the attacker is acting normally toward its target. However, if an attacker decided to deceive a defender to remain stealthy, the effectiveness of DD techniques may not be clearly measured.

- Compared to game-theoretic DD techniques, ML-based DD has been much less studied. The majority of ML techniques in DD are used to detect attacks in honeypots. Only a few works have used ML to mimic real objects or information for developing honey objects or information.

- Although game-theoretic or ML-based DD has been studied, there have been a few studies that combine both to develop hybrid DD techniques, particularly incorporating reinforcement learning (RL) into players' utility functions and using RL to identify the players' optimal strategies.

- Evaluation metrics of DD are limited. Most game-theoretic DD approaches mainly used metrics in game theory, such as utility or probability of taking attack strategies. Most ML-based DD approaches were mainly studied to achieve attack detection, which leads to using detection accuracy as a major metric. These metrics, utility and detection accuracy, do not fully capture the effectiveness and efficiency of DD approaches.

- Evaluation testbeds are mostly based on simulation models. Although some ML-based DD approaches used real testbeds, they are mostly social honeypots deployed in social media platforms. In particular, game-theoretic DD approaches remain highly theoretical analysis.

2.9.3 Future Research Directions

We suggest the following future research directions:

- **Need more metrics to measure the quality of honey-X technologies**: The quality of honey-X has been captured mainly based on detection accuracy (see Figs. 2.4 and 2.5) even if the key role of honey-X is to protect assets as well as to detect attacks. Hence, there should be more diverse metrics to measure the roles of both protecting assets and detecting attacks. The example metrics can include an attacker's perceive uncertainty level, the amount of missing vulnerable assets by attackers, the amount of novel attack vectors collected via honeypots, or the amount of important system components attacked. In addition, the efficiency of running honey-X technologies (i.e., defense cost) or performance degradation due to the deception deployment should be considered for a defender to choose an optimal strategy based on multiple criteria including those multiple metrics as objectives.

- **Consider more sophisticated, intelligent attackers**: The attacker behaviors in the literature mostly consider simple attacks which are not more intelligent than the defender behaviors, which are not realistic in practice. In particular, a highly intelligent attacker may be able to detect the deception deployed by the defender. Further, in most existing game-theoretic approaches using Stackelberg games, the defender is a leader while the attacker is a follower where there is a first-mover advantage in such sequential games. This may not be true in real-world cyber settings. When APT attacks are considered, most existing approaches consider only reconnaissance attacks.

In future work, we need to consider the APT attackers performing multistaged attacks following the cyber kill chain (CKC) stages (i.e., including reconnaissance, delivery, exploitation, command and control (C2), lateral movement, and data exfiltration).

- **Measure the quality of DD**: The quality of a DD technique should be determined based on how well it deceives attackers. That is, the quality of DD should be measured by the attacker's view and actions based on its belief towards the defender's moves. However, existing work mainly uses system metrics as a proxy to measure the quality of a DD technique deployed by the defender. Some example metrics can be the degree of uncertainty perceived by attackers, the degree of the discrepancy between what the attackers perceived towards the defense system and the ground truth system states, or service availability without being disrupted by deployed DD techniques.

- **Develop hybrid DD techniques based on both game theory and ML**: Approaches that combine machine learning and game theory (e.g., game theory with deep reinforcement learning) have been considered in developing other defense techniques [Liu et al., 2020; Xu et al., 2020]. However, only a few studies [Al Amin et al., 2019b; Wang et al., 2020] used hybrid approaches of incorporating reinforcement learning (RL) with game theory where we treat RL as one of ML techniques. RL's reward functions can be used to formulate players' utility functions and let an RL agent identify an optimal strategy as other attack-defense games [Liu et al., 2020; Xu et al., 2020] leverage RL or deep RL (DRL). In addition, ML can be used for players to estimate their own beliefs to decide what strategy to take or opponents' beliefs to predict their moves. In addition, when players use a game-theoretic approach to take an optimal strategy, a set of DD techniques can be developed using ML technologies to increase the quality of the deception by mimicking real objects or information.

- **Measure cost or service availability under the deployment of DD techniques**: As discussed in Section IX-B, most game-theoretic DD studies used utility and probabilities of taking strategies while ML-based DD approaches mainly used detection accuracy as the major metric. However, the drawbacks introduced by deploying DD techniques have not been sufficiently discussed. In addition, some strong assumptions, such as no impact introduced to the defender system, are made while deploying DD techniques. By considering metrics to measure performance degradation by a used DD, we can measure a broader aspect of the quality of the DD deployed in a system based on the classical tradeoff between security and performance.

- **Build a realistic testbed to evaluate game-theoretic DD techniques**: Game-theoretic DD approaches have been dominantly studied based on the theoretical validation via mathematical proof or experimental analysis via simulation models. More realistic emulation or real testbeds should be reflected to validate game-theoretic DD approaches considering highly intelligent attackers and highly uncertain network environments. To this end, there should be clear designs on how to deal with uncertainty in terms of how much knowledge is known for an attacker, what system information (i.e., a defender's information) can be known to the attacker, and what network environment information or dynamics are known to both the defender and the attacker. How much and how accurate information is available to players and how their beliefs are formulated under real settings are critical points to tackle because strong assumptions on common knowledge in game theory cannot hold in real environments. The first step to tackle this problem is to build a game

where two players may have asynchronous information with imperfect or incomplete information, such as a hypergame [Bakker et al., 2021].

- **Develop a domain-specific game-theoretic framework that can provide DD as a security service**: As observed in Table 2.3, we find many game-theoretic frameworks are developed as general frameworks without specifying a domain application. This would be helpful for general understanding about the core idea of the method itself. However, it misses how to design DD techniques for a specific domain environment which may pose challenges in terms of resources available (e.g., bandwidth, computational power), environmental dynamics, or other characteristics of common adversarial attacks. Providing more specific applications and corresponding system designs for developing and deploying DD techniques will lead to their improved applicability into the real environments.

- **Identify highly compatible configurations of using both DD and legacy defense services**: It is not necessarily always true to use DD techniques to handle all types of attacks. In addition, it is not always true that DD techniques should be always deployed with legacy defense services, such as intrusion prevention, detection, and response systems. This is because some combinations of deploying DD techniques with legacy defense services, such as using honeypots with intrusion detection and response systems, may introduce overlapping effect which is not efficient. We should develop a more systematic method to figure out how to synergistically leverage both defense services to provide cost-effective defense services.

- **Development of DD technologies based on cooperative effort between industry and academia:** DD technologies have been discussed widely in the academic research community. We observe that some cybersecurity companies developed DD products to detect or mitigate threats, mainly honeypots [ill; tra]. Current DD-based security products mainly focus on counter ransomware, credential theft, or insider attack. In addition, most of the products consider enterprise networks or cloud-based networks. However, since those companies have not publish their core technologies used in their products, it was not possible to check if approaches based on game theory or machine learning have bee used in those products. Hence, there should be more constructive collaboration between industry and academia to develop more innovative DD technologies based on the state-of-the-art approaches based on game theory and machine learning.

Chapter 3

Mee: Adaptive Honeyfile System for Inside Attacker Detection

Abstract: Insider attacks cause significant computer breaches security and challenge current cybersecurity approaches. Specifically, in advanced persistence threats (APTs), an adversary becomes an insider attacker by masquerading as a legitimate user and carries out malicious actions, such as stealthy exploration or data exfiltration. Traditional cybersecurity techniques, such as firewalls, cryptography, and intrusion detection systems, are inadequate against such attacks. Defensive deception offers a new way to detect and mitigate insider attacks. A popular defensive deception approach is placing honeyfiles on devices to confuse and detect insider attackers. Unfortunately, although honeyfiles can detect an attacker's anomalous action, they can confuse legitimate users by generating too many false-positive alarms. We develop a novel honeyfile approach named Mee that adjusts the number and placement of honeyfiles according to the risk profiles of the devices. We model our approach as a honeyfile game to assist the defender in searching for an optimal strategy that determines the number of honeyfiles in each connected device, considering the probability distribution of the attacker's targets. Our experimental results show that Mee reduces false positive alarms from legitimate users and increases attacker cost.

3.1 Introduction

Insider threats have recently attracted much attention. Traditional defensive approaches are insufficient to detect and respond to insider attacks. Following Salem et al. [2008], insider threats are divided into two types: *traitors*, who misuse their legitimate credentials, and *masqueraders*, who impersonate a legitimate user (and generally know less than a traitor about where the victim's valuable information resides).

An advanced persistent threat (APT) is how an attacker becomes a masquerader. Chen et al. [2014] define six stages of an APT attack: (1) reconnaissance; (2) delivery; (3) initial intrusion; (4) command and control (C2); (5) lateral movement; (6) data exfiltration. In the first three stages, an attacker gathers user information, such as accounts and passwords via phishing, and create backdoors in a compromised device. Then, the attacker can remotely control and access the victim's device without being detected by traditional cybersecurity technologies, such as IDS and firewall. Recent research focuses on blocking an APT [Marchetti et al., 2016; Shan-Shan and Ya-Bin, 2018; Tankard, 2011]. In contrast, our research

focuses on the detection of masquerader-type insider attacks.

Defensive deception [Lu et al., 2020] is an effective method to mitigate and detect attacks, such as reconnaissance [Achleitner et al., 2016; Anjum et al., 2021] and insider attacks [Ben Salem and Stolfo, 2012; Bowen et al., 2009]. Where traditional cybersecurity focuses on attacker actions, defensive deception focuses on anticipating such actions [Almeshekah and Spafford, 2014]. Deception has two goals: *confusion*, i.e., wasting the adversary's effort and hiding sensitive information, and *detection*, i.e., identifying malicious actions.

Honeyfiles, or decoy documents, are a lightweight defensive deception technology [Bowen et al., 2009; Voris et al., 2015; Yuill et al., 2004] comprising two major tasks: 1) generate content in a honeyfile [Abay et al., 2019; Whitham, 2016, 2017], and 2) decide number and placement of honeyfiles [Bowen et al., 2009; Gómez-Hernández et al., 2018; Voris et al., 2015]. Bowen et al. [2009] propose the *Decoy Document Distributor (D^3) system*, which generates and places decoy documents in a file system.

Salem and Stolfo [2011] analyze legitimate users and adversaries are affected by the number and location of honeyfiles. Raising the number of honeyfiles on a device increases the probability of the adversary being confused and detected but at the cost of resource consumption and increasing the false-positive rate of detecting an attacker by confusing legitimate users.

How can a defender decide how many and where to place honeyfiles to increase the effectiveness of detecting and disrupting attackers while reducing the impact on regular users? We introduce a honeyfile approach named Mee geared toward a large-scale enterprise network. Mee demonstrates **decentralized deployment**: by any user. It also demonstrates **centralized control**: the defender analyzes suspicious behavior across the network to determine the number and placement of honeyfiles for each device. This approach helps a defender accomplish the following.

- Detect an attack in progress.
- Detect a compromised device because an attacker who enters a device's filespace needs to explore it to locate valuable data and is likelier to touch a honeyfile than legitimate users who are familiar with their file systems.

We consider the attacker a masquerader-type adversary who can penetrate devices but does not identify the locations of valuable files on that device. Thus, the attacker needs to explore the victim's device and search for valuable files. Assume that the attacker is aware of the existence of the honeyfile system but cannot distinguish honeyfiles from regular files.

Game theory applies well in defensive deception [Pawlick et al., 2019; Zhu et al., 2021]. Pawlick et al. [2018] use signaling games to model interactions between the defender and APT attackers. La et al. [2016] propose a two-player game to model attacker and defender in an IoT environment with honeypots used for defensive deception. Cho et al. [2019b] use a hypergame model to show how differences in perceptions of the players affect decision-making.

Our contributions include the follows:

- We propose Mee as a novel honeyfile system for detecting insider attackers in enterprise networks that dynamically adjusts the number of honeyfiles placed on each device.
- We describe a Bayesian game model to analyze the optimal strategies for the attacker and the defender.

- We simulate and compare Mee with the traditional honeyfile approach and show that Mee is more effective at detecting insider attackers and has a more negligible impact on legitimate users.

Organization

The rest of this paper is organized as follows. Section 3.2 describes our problem in terms of details about insider attacker and the threat model. Section 3.3 introduces the design of our honeyfile system. Section 3.4 describes our scenario and model design. Section 3.6 compares Mee with the traditional honeyfile system. Section 3.7 describe the conclusions and our future directions.

3.2 Problem Statement

We focus on masquerader-type insider attackers in an enterprise network environment. The attacker can gather authorization of device owners and search for the valuable files in target devices without triggering traditional defensive technologies, such as IDS. With considering deployed honeyfiles, the attacker may have three results after penetrating a device.

Success: Viewing or transferring valuable files from a device.

Invalid: Not finding a valuable file, i.e., suffering wasted effort but no additional loss.

Defeat: Triggering alarms by touching honeyfiles. The defender would update its belief about the network's security level while cleaning or replacing the compromised device, as a result of which the attacker would lose a compromised device and have wasted its effort.

An adversary does reconnaissance before attacking to investigate whether a device contains valuable files to save energy. Thus, an insider attacker has clear intent regarding what information is valuable and targets devices accordingly. For example, an attacker who sought a final exam in a university would target several specific professors in a particular department, not a random device. Therefore, we name the device owner's identity, such as professor, student, and CEO, as *organizational role*. One insider attacker may pay more attention to several specific organizational roles.

3.3 Design of Mee System

The use of honeyfile has two purposes: to confuse the attacker with false information and detect any unauthorized access to connected devices. The number of honeyfiles in one device can significantly change the effectiveness of a honeyfile system. However, current honeyfile research mainly focuses on the honeyfile system in a single device, which cannot provide adequate observation to assist the defender in estimating the security situation and changing the number of honeyfiles. In contrast, we consider all devices in an entire enterprise network. Specifically, we have a central controller, named *Mee controller*, that receives information (e.g., about accesses to honeyfiles) from and instructs clients, named *Mee client*, which resides on individual devices. Following instructions from the Mee Controller, Mee client increases or decreases the number of honeyfiles.

3.3.1 Mee Client Design

The Mee client is an application endpoint that resides on each device. A Mee client generates and delete honeyfiles, and detect suspicious behaviors on a honeyfile, include opening, modifying, deleting, and transferring. When anyone acts on a honeyfile, the corresponding Mee client sends the alarm to the Mee controller and receives the controller's instructions.

Honeyfiles need to be placed and named well to avoid confusing legitimate users and attract attention from the adversary. Salem and Stolfo [2011] empirically studied decoy documents. They invited 52 students to download and install decoy documents and monitored the students' behaviors to figure out the suitable placement and names of decoy documents that minimize false positives. Although we mainly focus on the number of honeyfiles in each device, we incorporate some current honeyfile research, such as which honeyfile names the attacker interests more. Thus, we assign a *sensitivity*, $H_s \in [1, 3]$, to each honeyfile. A higher value represents a greater attraction for both the adversary and a legitimate user.

Legitimate users may touch honeyfile by accident. However, even if the users accidentally act on, such as open or read, honeyfiles, they have a lower chance to edit or transfer honeyfiles. To capture this feature, we assign *seriousness* of action on a honeyfile to reflect how much of a security threat the action can be. We consider two levels of seriousness.

Weak: open or close a honeyfile.

Strong: edit, transfer, or apply tools like `zip` and `tar`.

When anyone (e.g., the user or attacker) touches honeyfiles, the Mee client collects corresponding sensitivity and seriousness values and sends them to the Mee controller.

3.3.2 Mee Controller Design

The Mee controller represents the defender across the network. It receives alarms from Mee clients and analyzes them to update its beliefs about the security level of each device. Then, it instructs each client to create or delete a specified number of honeyfiles on its device. Upon receiving an alarm, the controller determines whether the described access is by a legitimate user or an attacker. We define HN_j as the number of honeyfiles on device j. This number is adjusted based on the defender's beliefs about how secure a device is and the attacker's goals. To assist the Mee Controller in making the optimal decision, we describe the following measures of the alarm analysis to assess network security situations.

Device Group and Group Risk Level

We assume that the insider attacker targets files placed on devices belonging to specific organizational roles, which we model as *groups*. For example, in a university, we may have groups for professors, students, and accountants. Meanwhile, one device can be included in multiple groups. For example, one device may belong to groups "professor" ($G_{\text{professor}}$) and "engineering" ($G_{\text{engineering}}$). Below, G_i is the set of devices in the group i, and HG_i denotes the number of honeyfiles in the group i, overloading the notation for the number of honeyfiles in device d. We use R_i to represent the *Risk Level* of group i. Higher R_i represents a higher probability of being the target group of an attacker. If someone touches a honeyfile on a device in group i, the defender raises the corresponding R_i.

Group Risk Level Update

Recall the sensitivity of a honeyfile and the seriousness of an action from Section 3.3.1. Equation 3.1 computes the change of risk level of group g due to a single action a on a specific honeyfile, h.

We define the group risk updating ($\Delta risk_g(h, a)$) as the product of the honeyfile sensitivity and the action seriousness divided by the number of honeyfiles in the group.

$$\triangle \text{risk}_g(h, a) = \frac{\text{sensitivity}_h * \text{seriousness}_a}{HG_g} \tag{3.1}$$

Through the group risk level updating, the Mee controller evaluates the security situation for each group. To compare a group's security situation to the rest of the network, we introduce R_{-i} with a negative in the subscript to denote the average group risk level of all groups except group i. Equation 3.2 shows how we calculate R_{-i}, where NG denotes the number of groups in a network.

$$R_{-i} = \frac{\sum_{j \neq i} R_j}{NG - 1} \tag{3.2}$$

Based on R_{-i} and R_i, we use the classification as *Dangerous*, *Medium*, and *Safe* to separate groups (as shown in Equation 3.3).

$$\textit{Classification} = \begin{cases} \text{Dangerous} & \text{if} \quad R_i > R_{-i} * 2 \\ \text{Medium} & \text{if} \quad R_{-i} < R_i < R_{-i} * 2 \\ \text{Safe} & \text{if} \quad R_i < R_{-i} \end{cases} \tag{3.3}$$

Via the group risk level, the Mee Controller can evaluate the security situation of all the groups. If the insider attacker tends to compromise the device based on the roles of the device owner, the Mee controller increases the number of honeyfiles in each device whose group obtains a high group risk level.

3.3.3 Communication between Mee Client and Controller

As we discussed above, Mee clients and controller exchange messages by which the controller evaluates the network security situation and asks Mee clients to adjust the number of honeyfiles in the corresponding device. Therefore, we name *Honeyfile Alarm* as the messages that transfer from all Mee clients to the Controller. Each honeyfile alarm contains a tuple $\langle CN, HN, AS \rangle$, where CN represents the client name, HN represents the honeyfile that are acted on, and AS represents the action upon the triggered honeyfile. In contrast, *Command* is the messages that is from Mee controller to each Mee client, which contains a tuple as $\langle CN, I \rangle$. Here, CN represents the target host of the command, and I represents the instructions that the Mee client needs to follow with.

The Mee system is shown in Figure 3.1. If anyone acts on a honeyfile, the Mee client sends honeyfile alarm to Mee Controller. The alarm contains information, includes the location of the triggered honeyfile, corresponding honeyfile sensitivity, and action seriousness, which are used to update the belief of the Mee Controller, such as each group's risk level. All alarms are stored in Mee Controller as well. Based on the group risk and alarm history, the Mee controller sends the command to Mee clients to adjust the number of honeyfiles in the device or check the device if necessary.

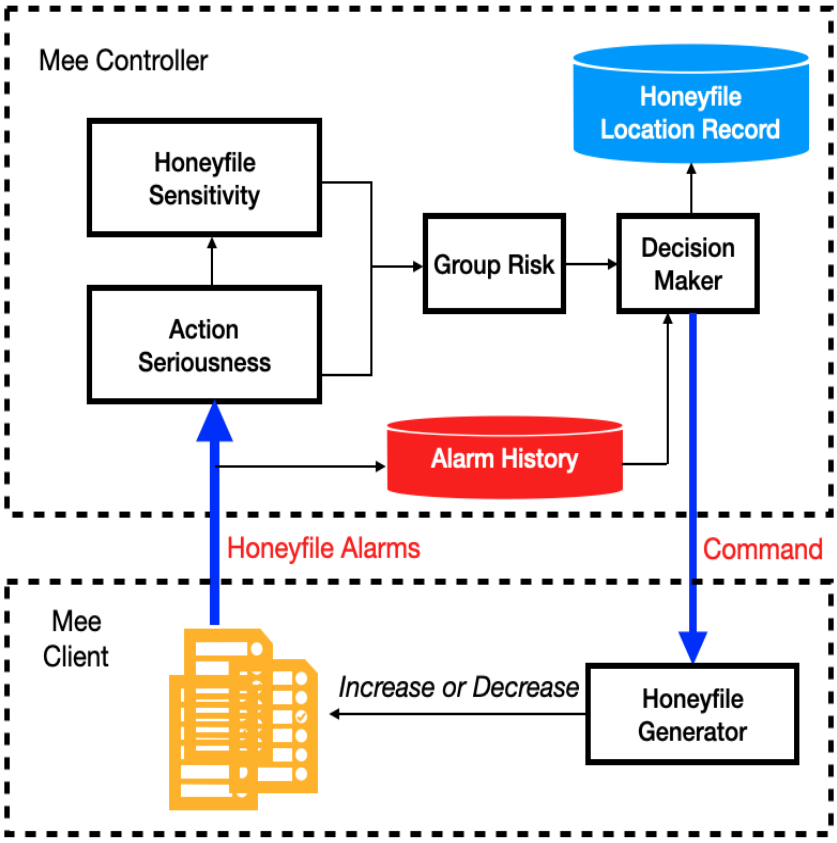


Figure 3.1: Mee System Structure

3.4 Scenario and Model

In this section, we describe the model of insider attackers, the defender, and users, and also the interaction between them.

3.4.1 Network and Node Model

An enterprise network environment includes connected end-user devices, such as computers and laptops. As introduced in Section 3.3.2, these devices are separated into groups based on their respective owners' roles. Each group's group risk level indicates its security situation.

All connected personal devices (e.g., laptop) have an installed Mee client, which can generate or delete honeyfile. Let NF_i denote the number of honeyfiles in device i. Following the concept of file sensitivity, each created honeyfile has a value to represent the attraction for attackers and regular users.

3.4.2 Attacker Model

We assume an attacker conducts reconnaissance before it attempts to penetrate into a device. Specifically, the adversary gathers full perception of all the users' basic information, include their organizational roles (such as CEO, professor, administrator), to guide it to select target devices. We also assume an insider attacker can access and explore a victim device without triggering traditional defensive technologies, such as IDSs. The attacker is rational and has knowledge of the existence of the honeyfile system, but

cannot distinguish a honeyfile from a regular file. However, as the insider attacker is unfamiliar with the file location in the victim device, it needs to explore a penetrated device to search for valuable files.

Attacker Actions

We consider the following action set:

Penetrate Device: Leverage legitimate authorization to penetrate a device (but without knowledge of its file system).

Search: Explore device i to search for valuable files.

File Read: Opens, read, or close a file in device i.

File Transfer or Modify: Transfer, edit, or delete a file.

Attacker Payoff

We separate the attacker payoff into *Effectiveness (EA):* the benefit which the attacker gets from an action; *Action Cost (AC):* cost to attacker to deploy action i; and *Impact of Failure (IF):* cost to attacker if the attacker touches a honeyfile.

3.4.3 Defender Model

Mee combines decentralized deployment of honeyfiles with centralized control to adjust the number of honeyfiles in each connected device. Specifically, the Mee controller monitors the devices and decides how many honeyfiles to place on each device. The defender's goal is to maintain group risk levels and detect a compromised device.

Defender Actions

We design four actions for the defender.

Check: Inform a device's Mee client to check for existing backdoors or update OS and application to avoid vulnerabilities. Upon doing so, set the number of honeyfiles on the device resets to the initial value and the group risk level value to the R_{-i} (defined in Section 3.3.2) value.

Increase number of honeyfiles on a device.

Decrease number of honeyfiles on a device.

No change: Maintain current strategy and save resources.

Note that in Mee, the defender chooses an action based on alarms from deployed honeyfiles as well as the network security situation. For example, if the defender receives an alarm from a device in the dangerous group, the defender has a higher probability to choose the Check action.

Defender Payoff

The defender's payoff has four parts.

Effectiveness (ED) Reward when a malicious action is detected, e.g., by recovering the compromised device and misleading the attacker with honeyfile.

Defence Cost (DC) Cost of deploying an action.

Failure in Protecting Real File (FR): Punishment upon failing in protecting real files.

Impact to Legitimate User (IN): False positive if a user accidentally opens, closes, transfers, or modifies a honeyfile.

3.4.4 Model of Legitimate User

We model the behaviors of legitimate users to capture Mee's impact on them. Therefore, we design the user action set as:

Login: Access a device.

Search: Explore a device, e.g., open a folder, search for a file.

Read a file.

Transfer, modify, or delete a file.

No matter what organizational roles are, the users may access their devices and read or modify their target files at any time. Therefore, given our threat and system models, there are two differences between insider attackers and legitimate users.

First, the login behaviors from regular users across the network relatively are randomly distributed. As a result, if there is no insider attacker in the environment, all devices have the same chance to be logged in by their owner. In contrast, the insider attacker would compromise the device based on the roles of the device owner. Second, users can quickly locate a target file as they are familiar with the file system, but a masquerader, lacking such knowledge, would have a relatively random movement to search for valuable files. Therefore, an insider attack has a greater chance to act on a honeyfile than a legitimate user.

3.5 Honeyfile Game with Mee

We consider the honeyfile game as a two-player dynamic Bayesian game. The two players update their beliefs according to the game's evolution. One player, named by *player a*, is the defender who can deploy honeyfiles within connected devices to detect insider attackers. Another player, denoted by *player b*, has two potential types: insider attacker and regular user. Although the type of the defender is common knowledge to the players, the defender does not know another player's type. However, it can generate its belief by observing honeyfile alarms from each device. Let $t \in T = [0, 1]$ represent the player b's type, where $t = 0$ represents a legitimate user and $t = 1$ represents an attacker.

In the defender's belief, we use $\sigma_a^s = p$ to denote the probability that player b is an attacker ($t = 1$), and $\sigma_n^s = 1 - p$ to denote the probability that player b is a legitimate user ($t = 0$). An alarm from a

Mee client represents an observation from player b. The defender has no knowledge of which device is compromised. Therefore, the honeyfile alarms may represent the detection of an attacker or false alarms from legitimate users. To make an optimal decision, the defender needs to estimate player b's type based on its belief, including the corresponding group risk level and the history of previous alarms. With the definition of action seriousness and the two players' action sets, Figure 3.2 explains the decision tree between the two players.

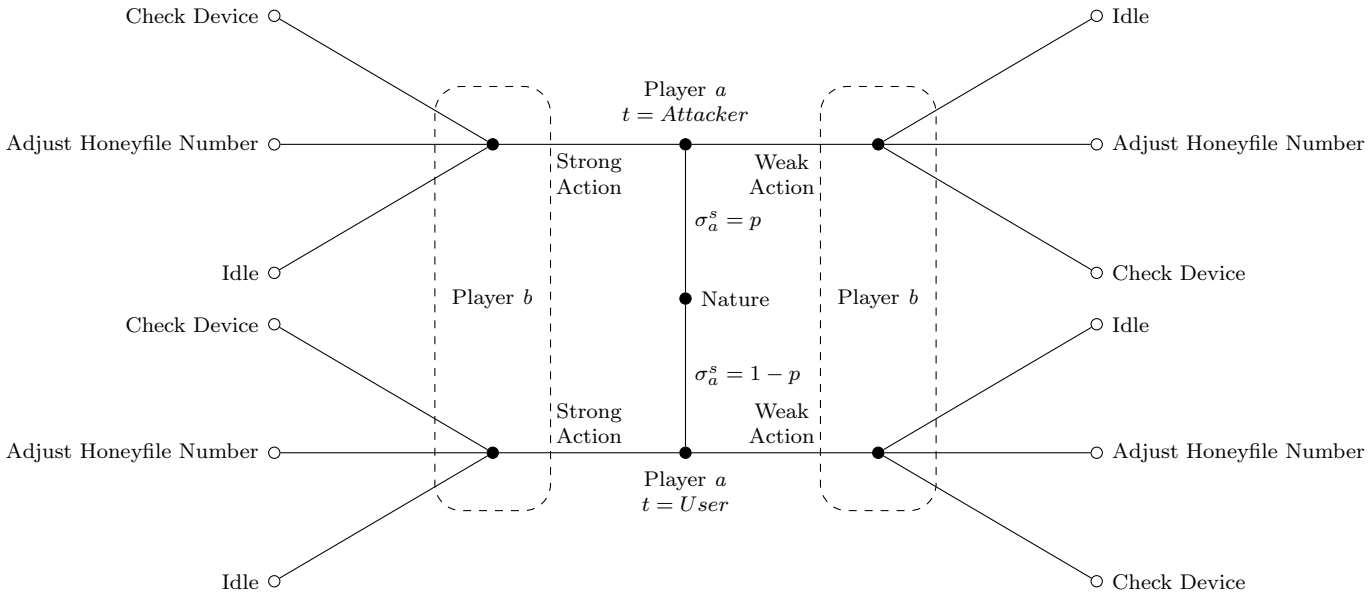


Figure 3.2: Decision tree of the honeyfile game: *Step 1:* player a obtains a type from nature, which is the root of the decision tree, as its private information. *Step 2:* After player a triggering a honeyfile alarm, if player a is an attacker, the path of decision tree goes to the upside. On the other hand, the path goes down. *Step 3:* the defender (player b) chooses its action based on the perspective of player a's type.

Utility Function

Let $w \in [1, 3]$ denote the worth of a regular file, where a higher number represents a more valuable file. For simplicity, we stipulate that an insider attacker obtains a gain of w if it reads a file and obtains $2 * w$ if it transfers or modifies a file. Let c_c denote the cost of compromising a device, c_r represents the cost of reading a file, and c_t represents the cost of transferring or modifying a file. Recall the definition of honeyfile sensitivity and action seriousness. We use h_p to denote the punishment of an attacker when it acts on a honeyfile. For simplicity, we stipulate that the punishment of reading a honeyfile is h_p, and the punishment of transferring or modifying a honeyfile is $2 * h_p$. Let $\beta \in [0, 1]$ represent the probability that an insider attacker estimates an actual file. The attacker calculates β based on its history, such as how many honeyfiles and actual files it touches.

We use c_a to represent the cost when the defender adjusts (e.g., increases and decreases) the number of honeyfiles. Let c_{cd} represent the cost of checking device action, and r_{cd} represents the reward if the

defender successfully recovers a compromised device. Meanwhile, $-r_{cd}$ represents the loss of an insider attacker if the defender recovers a compromised device. Let α represent the probability that the defender thinks the device is compromised.

Table 3.1 shows the expected payoff when an insider attacker acts on a file. Note that the payoff in Table 3.1 does not include the cost of compromising a device. As a rational player, an insider attacker expects that the reward from real files in a device is larger than the cost of compromising the device. In other words, if NR represents a set of real files that an attacker acts on (e.g., reads, and transfers), it expects $c_c < \sum_{nr \in NR}(w_{nr} - c_{r(nr)}) + \sum_{nr \in NR}(2w_{nr} - c_{t(nr)})$.

Table 3.1: Player a is an insider attacker. The tuples below include ⟨defender's payoff, attacker's payoff ⟩.

	Read a file	Transfer or Modify a file
Check Device	$\alpha * r_{cd} - c_{cd}, \beta w - (1-\beta)h_p - r_{cd} - c_r$	$\alpha * r_{cd} - c_{cd}, 2\beta w - (1-\beta)h_p - r_{cd} - c_t$
Increase Honeyfile	$-c_a, \beta w - (1-\beta)h_f - c_r$	$-c_a, 2\beta w - 2(1-\beta)h_f - c_t$
Decrease Honeyfile	$-c_a, \beta w - (1-\beta)h_f - c_r$	$-c_a, 2\beta w - 2(1-\beta)h_f - c_t$
No Change	$0, \beta w - (1-\beta)h_f - c_r$	$0, 2\beta w - 2(1-\beta)h_f - c_t$

Table 3.2: Player a is a legitimate user. The tuples below include ⟨defender's cost, user's cost ⟩.

	Read a file	Transfer or Modify a file
Check Device	$\alpha * r_{cd} - c_{cd}, 0$	$\alpha * r_{cd} - c_{cd}, 0$
Increase Honeyfile	$-c_a, 0$	$-c_a, 0$
Decrease Honeyfile	$-c_a, 0$	$-c_a, 0$
No Change	$0, 0$	$0, 0$

Dynamic Game

The honeyfile game is repeatedly played between player a and player b. Let k_t represent the timeline of one game, where $t = 0, 1, 2, \ldots$. Here, at any one time, only one player b (e.g., an insider attacker or a legitimate user) is active. Player b firstly obtains its type from nature as private information. Upon one player b finishing its objective, such as an attacker gathering enough valuables files or a user obtaining the target file, this player b exits the game, and another player b comes in to participate. The player a (e.g., the defender) keeps accumulating its beliefs throughout the game. Specifically, the defender updates its belief about the type of player b after receiving honeyfile alarms. It chooses actions based on all the gathered information, such as group risk levels and honeyfile locations, from the beginning of the game.

3.6 Implementation and Evaluation

We adopt these metrics to quantify performance.

Attacker payoff ($\langle EA, AC, IF \rangle$): We evaluate the attacker's payoff based on the definition in Sec-

tion 3.4.2.

Defender payoff ($\langle ED, DC, FR, IN \rangle$): The defender's payoff is calculated using the definition in Section 3.4.3.

Accuracy measures: To compare the accuracy of Mee with the traditional honeyfile system, we calculate false positive rate (FPR), true positive rate (TPR), and the area under Receiver Operating Characteristic (ROC) curve.

3.6.1 Simulation Settings

We compare Mee with the traditional honeyfile system via a simulation. In the testbed, we deploy 114 devices and separate them into 20 groups. The installed Mee client in each connected device can create or remove honeyfiles following the Mee controller's command. Meanwhile, the Mee controller can estimate the network security situation via receiving honeyfile alarms from Mee clients.

Adversary Setting

The action set of an insider attacker is {*Penetrate*, *Search*, *Read*, *Transfer*} (as introduced in Section 3.4.2). The insider attacker selects several groups as its *target groups* and has a higher probability of penetrating the devices within the target groups rather than randomly selects a device to attack. Specifically, in all following tests, an attacker has a ten percent probability of randomly choosing a target device to compromise (without considering the target group) and a ninety percent probability of selecting a device in the target group(s).

Assume that an insider attacker can penetrate any device across the network and explore the compromised machine to search for valuable files. However, not every compromised device contains a valuable file. In each simulation, the attacker starts from an initial budget, representing the expected cost for searching and gathering the valuable files on every device. Suppose the cost of searching for valuable files within a compromised device is more than the initial budget. The attacker can choose to abandon the current device and penetrate another device instead.

User Setting

To calculate the defender's cost and false positive rate of the honeyfile alarm, we model the behaviors of legitimate users in our simulation. A user has an action set [*Login*, *Search*, *Read File*, *Transfer or Modify File*]. Besides, the user obtains a full map of their file systems, which can assist the legitimate user in accessing a target file faster than the attacker. However, every user has a ten percent probability of choosing an incorrect action or action target. Thus, the user may act on a honeyfile and generate a false alarm.

Although the last three actions of the user model are the same as the corresponding actions in the attacker model, we emphasize the differences between legitimate users and the adversary. At first, each legitimate user has a clear target file and has complete knowledge of the file system. Thus, the user can locate any file in the device without a random search. And then, users access the corresponding devices at any time, so the distribution of login behaviors across the network is random. In contrast, the insider attacker tends to perform the compromise action relying on its target group(s) but randomly explores the file system.

Defender Setting

We simulate both the traditional honeyfile system, in which the number of honeyfiles in each device is static, and Mee, which dynamic adjusts the number of honeyfiles in each device. The defender has an action set [*Check Device*, *No Change*] for the traditional honeyfile system, and an action set [*Check Device*, *No Change*, *increase honeyfile decrease honeyfile*] for Mee (as introduced in Section 3.4.3).

Honeyfile Generation

Following Gómez-Hernández et al. [2018], we monitor honeyfiles using `inotify-tools`, which is a library and a set of command-line programs for Linux that monitors and acts upon file system events. Specifically, our client uses `inotify-tools` to detect actions, such as open, close, and modify, on a honeyfile. Then, the Mee client sends such information to the Mee controller. For simplicity, we consider only `txt` file in our research, i.e., files with the `txt` extension. For the traditional honeyfile system simulation, we deploy the same number of honeyfiles in each device. Then, we change the number of honeyfiles in each device and observe the defender's performance, the attacker's cost, and the false/true positive rate. For Mee, we deploy 40 honeyfiles in each device as the initial statement. The defender can modify the number of honeyfiles based on its belief of network security situation.

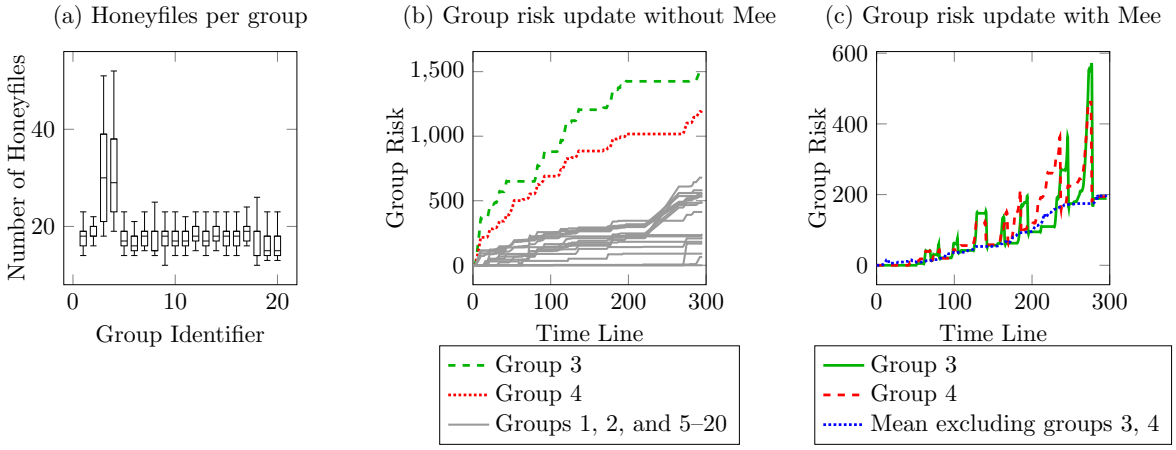


Figure 3.3: Mee's Performance: (a) Numbers of honeyfiles in different groups; (b) Group risk updates without Mee; (c) Group risk updates with Mee

3.6.2 Comparing Mee with the Traditional Honeyfile System

A traditional honeyfile system does not adjust the number of honeyfiles in each device and maintain group risk levels. To simulate a traditional honeyfile system, we deploy fixed number of honeyfiles in each connected device. As discussed in Section 3.6.1, we separate the devices into 20 groups and posit that the insider attacker prefers to attack Group 3 (G3) and 4 (G4). We evaluate the performance of the traditional honeyfile system and Mee based on various numbers of honeyfiles and the group risk level update.

Adjusting the Number of Honeyfiles with Mee

Mee changes the number of honeyfiles in each connected device to reduce unnecessary overhead and the impact on legitimate users. Figure 3.3(a) records the variance of the number of honeyfiles in each group. We assign ten insider attackers and 200 legitimate users in this test. Because the attacker's target groups are G3 and G4, Mee automatically adjusts and deploys more honeyfiles in these two groups than others. The average numbers of honeyfiles in G3 and G4 are 29 and 30, whereas the average number of honeyfiles in other groups is 18.

Group Risk Level of Traditional Honeyfile System and Mee

Figure 3.3(b) shows the result of group risk level update with the traditional honeyfile system. The X-axis represents the timeline in one test, and the Y-axis represents the group risk level. Each player joins in and performs as a legitimate user or an insider attacker in each time slot. The group risk levels of G3 and G4 (dotted red and dashed green lines) are much higher than other groups (solid grey lines), representing the tendency of the attacker's movement to be captured via the group risk level updating.

With the same setting, Mee controller receives and analyzes honeyfile alarms from Mee clients to maintain the group risk level. Figure 3.3(c) shows the group risk level updating with Mee. Anyone who triggers the honeyfile alarm makes group risk levels rise. Suppose Mee controller confirms that a group has an abnormal (e.g., much higher than other groups) group risk level. It can apply *check* action and reset the corresponding group risk level to R_{-i} (introduced in Section 3.4.3). In Figure 3.3(c), the red dash line and solid green line represent the group risk levels of G3 and G4 (e.g., the attacker's target groups). We use the blue dotted line for the rest of the group's risk levels to denote their average value.

Defender Payoff with the Traditional Honeyfile System and with Mee

We then calculate the defender's payoff by considering ten insider attackers and 200 legitimate users. The players' cost calculations follow the setting of Section 3.4. For the traditional honeyfile system, we increase the number of honeyfiles in each device from 0 to 100.As shown in Figure 3.4(a), the defender payoff (blue dashed line) keeps increasing before the number of honeyfiles in each device is less than forty and decreasing later. The results show that grow the number of honeyfiles can assist the defender in increasing its payoff because the defender can detect the insider attackers more effectively. But the redundancy of honeyfiles may disrupt legitimate users—and disrupt them more and more as the number of honeyfiles is increased, thereby generating more false positive alarms that only confuse the defender to check a safe device. The straight (orange) line represents the defender payoff when we deploy Mee with the same attacker and user settings.

Attacker Cost with Traditional Honeyfile System and Mee

With the same setting as above, Figure 3.4(b) records the attacker's payoff given a certain number of honeyfiles in the environment. The X-axis represents the number of honeyfiles in each device, and the Y-axis represents the attacker's payoff. With a traditional honeyfile system, more honeyfiles can significantly reduce the attacker's payoff. However, compare with Figure 3.4(a), the overhead of honeyfiles also disturbs the defender's performance. Also, the straight orange line represents the attacker payoff when we apply Mee.

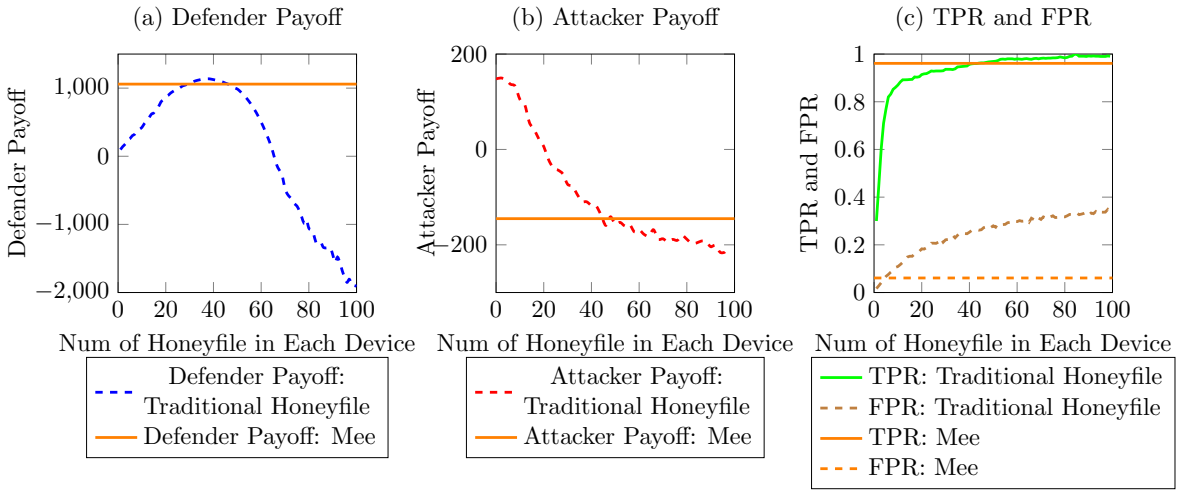


Figure 3.4: Comparison Between the Traditional Honeyfile System and Mee

TPR and FPR with the Traditional Honeyfile System and with Mee

We calculate the false positive rate (FPR), true positive rate (TPR), and the area under the ROC curve for the traditional honeyfile system and Mee. The area under the ROC curve is define as $ROC\ area = TPR * (1 - FPR)$.

We consider positive, negative, true positive, false positive as metrics for the defender.

Positive: Defender detects that a insider attacker is present.

Negative: Believes that the honeyfile alarm is triggered by a legitimate user.

True positive: Detection of the insider attacker rather than a legitimate user.

False positive: Misunderstanding of honeyfile alarm triggered by a user.

With the same test settings as above, Figure 3.4(c) shows the TPR and FPR of the traditional honeyfile system and Mee. The solid green line and dashed brown line show the TPR and FPT of the traditional honeyfile system, while the orange solid and dashed lines represent Mee. With the traditional honeyfile system, increasing the number of honeyfiles in each device can significantly raise the TPR but also brings the high FPR. Mee maintains the TPR at a high level and reduces the FPR.

We then test the performance of the traditional honeyfile system and Mee with the different number of insider attackers. Following the definition of action seriousness in Section 3.3.1, with the different number of actions on honeyfile, we define several situations for insider attacker detection with a traditional honeyfile system. Figure 3.5 shows the ROC obtained with different definitions of detection. For example, the red marks represent the defender detection is triggered if anyone acts on the honeyfile in one device with at least two weak actions or one strong action. Within each detection definition, we increase the number of insider attackers in one test from 1 to 100. As shown in Figure 3.5, when weak action=2 and strong action=1, the ROC value concentrates within high TPR and FPR, which indicates that the detection is extra sensitive. Even if a traditional approach detects insider attackers, it generates many false positive alarms that confuse the defender and impact legitimate users. Meanwhile, increasing the number of attackers does not significantly influence the performance of the traditional honeyfile

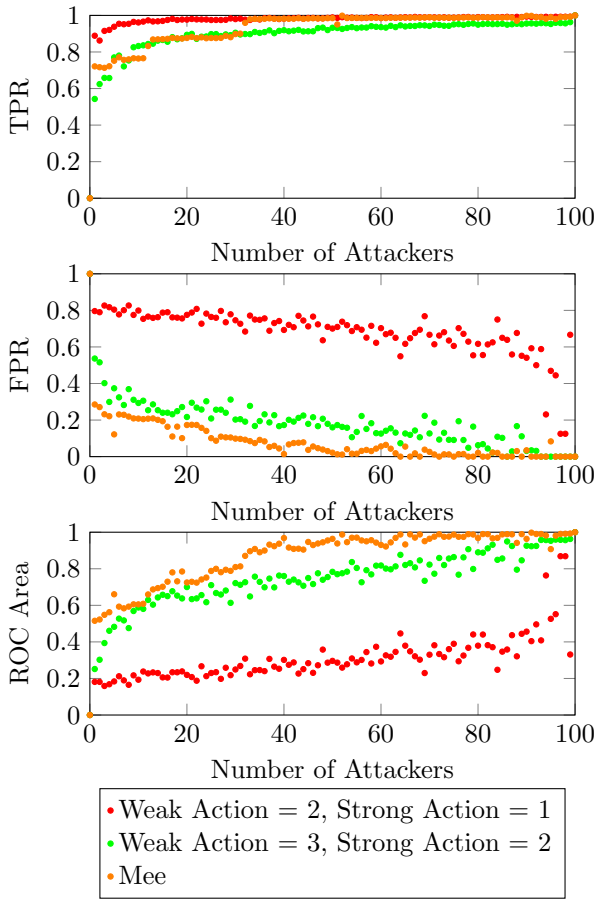


Figure 3.5: True and False Positive Rate of Traditional and Mee Honeyfile System

system. When weak action=3 and strong action=2 (shown as green marks in Figure 3.5), although the FPR reduces, the TPR also decreases, which represents that a large number of insider attackers escape. Orange marks in Figure 3.5) represent the performance of Mee, showing that Mee can reduce the FPR and maintain a high level of the TPR. With the figure of ROC area, it is clear that Mee has a better performance.

T-Test and Cohen's d

To evaluate the difference of performance between the traditional honeyfile system and Mee statistically, we apply the t-test and calculate Cohen's d values. The corresponding Cohen's d value between each pair of distributions can be found in Table 3.3 .

3.7 Conclusion and Future Work

Defensive deception technologies are commonly used to delay and detect stealthy attacks. Honeyfile systems are a simple and lightweight deception technology that is widely applied. We design Mee as a novel honeyfile system to confuse and detect the insider attacker. Mee leverages *centralized* control and *decentralized* deployment to adjust the number of honeyfiles in each connected device. With Mee, the

Table 3.3: Cohen's d Value for Each Pair of Distributions

Comparison	TPR	FPR	ROC Area
(Weak Action = 2, Strong Action = 1) and (Weak Action = 3, Strong Action = 2)	0.28	3.56	3.05
(Weak Action = 2, Strong Action = 1) and Mee	0.37	4.54	3.61
(Weak Action = 3, Strong Action = 2) and Mee	0.69	0.80	0.75

defender can reduce the overhead and the false positive alarms from the legitimate users. We use game theory to model the interaction between the defender, the attacker, and legitimate users. To measure the performance of Mee, we simulate and compare it with the traditional honeyfile system. The results show that Mee can significantly reduce the average number of honeyfiles in the whole network. As a result, Mee avoids deploying unnecessary honeyfiles and decreasing the impact on legitimate users.

Future Work

This research assumes each device has an equal workload. Hence, all the honeyfiles in the network obtain the same probability that legitimate users act on them. However, in the real world, some users may have a higher likelihood of touching a honeyfile, such as those using the device more frequently than others. Therefore, we will consider the environment with a more realistic model and model the legitimate user with more details in our future work.

Chapter 4

Mee 2: deep reinforcement learning based adaptive honeyfile system

Abstract:

4.1 Introduction

Defensive Deception

Defensive deception brings a novel angel to detect or mitigate the stealthy threats that cannot be countered by traditional cybersecurity techniques, such as IDS and firewall. Zhu et al. [2021] suvery the game theory and machine learning based defensive deception research to show the advance of the deception technologies. Salem and Stolfo [2011] empirically studied decoy documents. They invited 52 students to download and install decoy documents and monitored the students' behaviors to figure out the suitable placement and names of decoy documents that minimize false positives. Lu et al. [2020]

Most current defensive deception research concentrates on how the defender should mislead the adversary's perspective rather than discuss the impact on legitimate users. In this paper, we not only focus on how to increase the effectiveness of the honeyfile system and but also decrease the disturbance to regular users.

Deep Q Learning

Cyber attacks are complex and dynamic, resulting in the security mechanisms being responsive, adaptive, and scalable. Traditional cybersecurity methods are insufficient to counter well-trained attackers. Deep reinforcement learning (DRL) has been widely applied to improve the intelligence of defensive techniques to address these issues. Incorporating reinforcement learning and deep learning, a DQL agent produces its learning experience by the direct interaction with *environment*. Similar with reinforcement learning, the description of an DQL agent involves *state*, *action space*, and *reward function*. At each time slot, the agent selects an action based on its state to change the environment and receives an immediate reward.

4.1.1 Deep Q-Learning

One popular RL process is Q-Learning, whose objective is to maximize the accumulative reward based on the Bellman function:

$$Q(s_t, a_t) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} ... | s_t, a_t] \tag{4.1}$$

where s_t, a_t, and r_t represent the state, action, and reward of the agent at time t. The discounted factor $\gamma \in [0, 1]$ resolves the importance of future rewards. Through the learning processes, the agent generate a *Q-table* for collect the expected reward (Q-value) of an action given a set of state. However, the Q-table requires large memory to store increasing states and actions. Specifically, a real-world model often includes continues states and action space, which can not be mimic within a Q-table. Deep learning brings opportunity to improve RL for learning complicated environment.

In our previous work Zhu and Singh, we propose a novel honeyfile system, named *Mee*, which leverages *centralized control* and *decentralized* deployment to adjust the number of honeyfiles in the connected devices based on the security situation across the network. We apply a Bayesian game model to assist the defender in searching for the optimal solution. This paper extends Mee by considering more complex network environments, such as multiple anticipating users and insider threats.

4.2 Problem Statement and Assumption

Masquerader type insider attacker can gather a device owner's authorizations and penetrate the device (e.g., laptop or computer) without triggering the traditional cybersecurity alarms, such as IDS or firewall. The attacker desires to search for resided valuable files with minimum cost. Thus, an insider attacker has relatively clear intent regarding what information or file is valuable. The adversary does reconnaissance before its attack to investigate whether the device contains valuable files. For instance, an attacker who sought a final exam in a university would target several specific professors in a particular department, not a random device. To capture this feature, we name the device owner's identity, such as professor, student, and CEO, as *organizational role*. Within an enterprise network, there are multiple organizational roles, and some of them may have overlap. One insider attacker may pay more attention to several specific organizational roles. Assume the insider attacker knows the existence of honeyfiles but cannot distinguish honeyfiles from regular files.

For the defender, we assume honeyfile alarms are the only observation of the insider threats. Upon receiving honeyfile alarms, the defender needs to judge whether the alarms represent the detection of insider threats or false positive alarms from legitimate users.

4.3 Design of Mee2

However, honeyfile alarms may represent insider threats or false positive alarms from legitimate users. Via Mee Zhu and Singh, the defender analyzes the device security level and adjusts the number of honeyfiles in each device. This paper extends the previous work and concentrates on differentiating true positive alarm and false positive alarm with applying Deep Reinforcement Learning.

4.3.1 Structure

We develop Mee2 and separate the whole system to two parts: Mee2 Client, and Mee2 Controller.

- **Mee2 Client** represents an application endpoint that resides on each device. A Mee client monitors generated honeyfiles to detect suspicious behaviors, include opening, modifying, deleting, and transferring. When anyone acts on a honeyfile, the corresponding Mee client sends the alarm to the Mee controller and receives the controller's instructions. We assign HN_j as the number of honeyfiles on device j.

- **Mee2 Controller** represents the defender of the network. It receives alarms from Mee clients and analyzes them to update its beliefs about the security level of each device. Upon receiving an alarm, the controller determines whether the described access is by a legitimate user or an attacker.

- **Communication between Mee2 Client and Controller** includes all of the exchange messages. We name *Honeyfile Alarm* as the messages that transfer from all Mee clients to the Controller. Each honeyfile alarm contains a tuple $\langle CN, HN, AS \rangle$, where CN represents the client name, HN represents the honeyfile that are acted on, and AS represents the action upon the triggered honeyfile. *Command* is the messages that is from Mee controller to each Mee client, which contains a tuple as $\langle CN, I \rangle$. Here, CN represents the target host of the command, and I represents the instructions that the Mee client needs to follow with.

4.3.2 Risk Assessment

To assist the controller in making the optimal decision, we describe the following measures to assess network security situations.

Honeyfile Sensitivity

Salem and Stolfo [2011] investigate which honeyfile name and location attract adversaries more. We are inspired by this research and assign a *sensitivity*, $HS \in [1, 3]$, to each honeyfile. A higher sensitivity value represents a more significant attraction for both the adversary and a legitimate user.

Action Seriousness

Legitimate users may touch honeyfile by accident. However, even if the users accidentally act on, such as open or read, honeyfiles, they have a lower chance to edit or transfer honeyfiles. To capture this feature, we assign *seriousness*, $AC \in [1, 3]$, of action on a honeyfile to reflect how much of a security threat the action can be. We consider two levels of seriousness: *1) Weak Action*, which represents opening or closing a honeyfile, and *2) Strong action*, which represents transferring, editing, or applying tools like `zip` and `tar`.

When anyone (e.g., the user or attacker) touches honeyfiles, the Mee2 client collects corresponding sensitivity and seriousness values and sends them to the Mee2 controller.

Group

Recall the definition of organization role (introduced in Sec 3.2). The insider attacker targets files placed on devices belonging to specific organizational roles, which we model as *groups*. Meanwhile,

one device can be involved in multiple groups. For example, in a university, we may have groups for professors, students, and accountants, and one device may belong to groups "professor" ($G_{\text{professor}}$) and "engineering" ($G_{\text{engineering}}$). Below, G_i is the set of devices in the group i.

To differentiate the groups, we assign the *Importance* value ($I \in [1, 5]$) to each group based on how important this group is to the defender. Meanwhile, the higher importance value also represents more valuable for the adversaries.

Group and Device Risk

We use R_i to represent the *Risk Level* of group i. Higher R_i represents a higher probability of being the target group of an attacker. If someone touches a honeyfile on a device in group i, the defender raises the corresponding R_i. We assign $HG_i = \sum_{j \in i} HN_j$ as the number of honeyfiles in the group i, overloading the notation for the number of honeyfiles in device $j(HN_j)$.

With considering the sensitivity of a honeyfile, the seriousness of an action, and the importance of the group, Equation 4.2 computes the change of risk level of group g due to a single action a on a specific honeyfile, h, within the group g. We define the group risk updating ($\Delta risk_g(h, a)$) as the product of the honeyfile sensitivity and the action seriousness divided by the number of honeyfiles in the group.

$$\triangle \text{risk}_g(h, a, g) = \frac{\text{sensitivity}_h * \text{seriousness}_a * \text{importance}_g}{HG_g} \tag{4.2}$$

Through the group risk level updating, the Mee controller evaluates the security situation for each group. To compare a group's security situation to the rest of the network, we introduce R_{-i} with a negative in the subscript to denote the average group risk level of all groups except group i. Equation 4.3 shows how we calculate R_{-i}, where NG denotes the number of groups in a network.

$$R_{-i} = \frac{\sum_{j \neq i} R_j}{NG - 1} \tag{4.3}$$

Via the group risk level, the Mee Controller can evaluate the security situation of all the groups. If the insider attacker tends to compromise the device based on the roles of the device owner, the Mee controller increases the number of honeyfiles in each device whose group obtains a high group risk level.

Table 4.1: Summary of notation

Symbol	Meaning	Range
	Environment	
NF_i	Number of honeyfiles in device i	None
FS	Honeyfile sensitiveness	[1,3]
AC	Action seriousness	[1,3]
SW	strong warning when someone triggers a honeyfile alarm	None
	Attacker	
$A1_i$	attacker explores the device i	None
	Defender	
DE	reward of the defender when it correctly recovers a compromised device or misleads attacker by honeyfile	[1,10]
DC	cost of each defender's action	[1,5]
DF_i	impact if the defender fail in protecting the real file	AE_i

4.4 DQL Scheme

For deep reinforcement learning, an agent interacts with the environment E by actions ($a \in A$) and states ($s \in S$). The agent searches for the policy $\pi(a_t|s_t)$ that guides it to maximize the expected payoff according to a reward function $r(s, a)$ at time t. In this section, we define the agent, state, and environment. We illustrate how we use DQL to assist the defender in searching for the optimal policy.

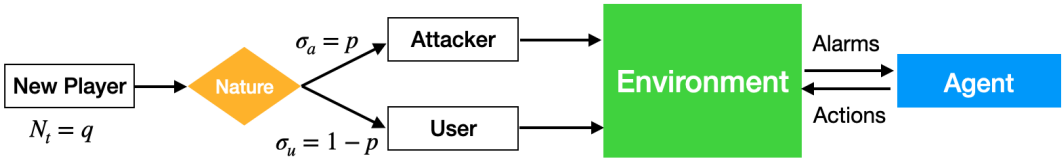


Figure 4.1: Reinforcement Learning Process

4.4.1 Environment

We consider an enterprise network that includes connected end-user devices, such as computers and laptops. As introduced in Section 4.3.2, these devices are separated into groups based on their respective owners' roles. Each group's group risk level indicates its security situation. All connected personal devices (e.g., laptop) have an installed Mee client, which can monitor honeyfiles and send honeyfile alarms to the Mee controller. Let NF_i denote the number of honeyfiles in device i. Following the concept of file sensitivity, each created honeyfile has a value to represent the attraction for attackers and regular users.

Assume any legitimate user and insider attacker can log in or penetrate the connected device anytime. Specifically, at each time $t \in [0, 1, 2, ...]$, we assign $N_t = q$ to denote the probability that one *player* participates in the environment at time t. Before interacting with the environment, the player gets *type* (e.g., regular user and insider attacker) from nature as its private information. we use $\sigma_a = p$ to denote the probability that the player is an attacker, and $\sigma_u = 1 - p$ to denote the probability that the player is a legitimate user. To mimic the behavior of legitimate users and the insider threat, we model them as follows.

Insider Attacker

The action set of an insider attacker is {*Penetrate*, *Search*, *Read*, *Transfer*}. The insider attacker selects several groups as its *target groups* and has a higher probability of penetrating the devices within the target groups rather than randomly selects a device to attack. Specifically, in all following tests, an attacker has a ten percent probability of randomly choosing a target device to compromise (without considering the target group) and a ninety percent probability of selecting a device in the target group(s).

Assume that an insider attacker can penetrate any device across the network and explore the compromised machine to search for valuable files. However, not every compromised device contains a valuable file. The attacker starts from an initial budget (IB), representing the expected cost for searching and gathering the valuable files on every device. Suppose the cost of searching for valuable files within a

compromised device is more than the initial budget. The attacker can choose to abandon the current device and penetrate another device instead.

Legitimate User

A user has an action set [*Login*, *Search*, *Read File*, *Transfer or Modify File*]. Besides, the user obtains a full map of their file systems, which can assist the legitimate user in accessing a target file faster than the attacker. However, users have probability of choosing an incorrect action or action target. Thus, the user may act on a honeyfile and generate a false alarm.

Although the last three actions of the user model are the same as the corresponding actions in the attacker model, we emphasize the differences between legitimate users and the adversary. At first, each legitimate user has a clear target file and has complete knowledge of the file system. Thus, the user can locate any file in the device without a random search. And then, users access the corresponding devices at any time, so the distribution of login behaviors across the network is random. In contrast, the insider attacker tends to perform the compromise action relying on its target group(s) but randomly explores the file system.

4.4.2 Agent

Mee2 controller is the agent of the reinforcement learning scheme. Following the previous settings, the controller has an action set [*Check Device*, *No Change*]. In each time slot, the agent selects an action a_t and receives a reward r_t. The agent's observation comes from the alarms of honeyfiles. However, an alarm may represent the detection of an insider attacker or mistouching of a legitimate user. The agent accumulates its belief and search for a policy guiding itself to choose the optimal action.

Utility

The agent's payoff has four parts.

Effectiveness (ED): Reward when an insider attacker is detected.

Defence Cost (DC): Cost of deploying an action.

Failure in Protecting Real File (FR): Punishment upon failing in protecting real files, e.g., the insider attacker successfully touches valuable files and escapes.

Wrong Detection (WD): False positive detection after a user mistouching a honeyfile.

After an action a_t at time t, the agent receive a reward $r_t = ED - DC - FR - WD$. The goal of the agent is to interact with the environment by selecting action and maximize its future rewards. Let $\pi(a_t|a_t)$ represent the policy to guide the agent select action A when it is in state S at time t. We use deep convolutional neural network to approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2}...|s_t = s, a_t = a, \pi] \tag{4.4}$$

which is the maximum sum of rewards r_t discounted by γ at each time step t.

State

B = $\Delta(S)$ represents the continuous belief state space, which includes the group risk level and the honeyfile alarm history.

To fully observe and understand the situation, the agent accumulate a sequence of actions and observations $s = x_1, a_1, x_2, a_2...$, and learn strategies based on these sequences.

4.5 Implementation and Simulation

We compare Mee 2 with the traditional honeyfile system via a simulation.

Adversary Setting

The insider attacker's action space is {*Penetrate*, *Search*, *Read*, *Transfer*} (as introduced in Section 4.4). The insider attacker selects several groups as its *target groups* and has a higher probability of penetrating the devices within the target groups rather than randomly selects a device to attack. Specifically, in all following tests, an attacker has a ten percent probability of randomly choosing a target device to compromise (without considering the target group) and a ninety percent probability of selecting a device in the target group(s).

Assume that an insider attacker can penetrate any device across the network and explore the compromised machine to search for valuable files. However, not every compromised device contains a valuable file. In each simulation, the attacker starts from an initial budget, representing the expected cost for searching and gathering the valuable files on every device. Suppose the cost of searching for valuable files within a compromised device is more than the initial budget. The attacker can choose to abandon the current device and penetrate another device instead.

User Setting

To calculate the defender's cost and false positive rate of the honeyfile alarm, we model the behaviors of legitimate users in our simulation. A user has an action set [*Login*, *Search*, *Read File*, *Transfer or Modify File*]. Besides, the user obtains a full map of their file systems, which can assist the legitimate user in accessing a target file faster than the attacker. However, every user has a ten percent probability of choosing an incorrect action or action target. Thus, the user may act on a honeyfile and generate a false alarm.

Although the last three actions of the user model are the same as the corresponding actions in the attacker model, we emphasize the differences between legitimate users and the adversary. At first, each legitimate user has a clear target file and has complete knowledge of the file system. Thus, the user can locate any file in the device without a random search. And then, users access the corresponding devices at any time, so the distribution of login behaviors across the network is random. In contrast, the insider attacker tends to perform the compromise action relying on its target group(s) but randomly explores the file system.

Defender Setting

We simulate both the traditional honeyfile system, in which the number of honeyfiles in each device is static, and Mee, which dynamic adjusts the number of honeyfiles in each device. The defender has

an action set [*Check Device*, *No Change*] for the traditional honeyfile system, and an action set [*Check Device*, *No Change*, *increase honeyfile decrease honeyfile*] for Mee.

Honeyfile Generation

Following Gómez-Hernández et al. [2018], we monitor honeyfiles using `inotify-tools`, which is a library and a set of command-line programs for Linux that monitors and acts upon file system events. Specifically, our client uses `inotify-tools` to detect actions, such as open, close, and modify, on a honeyfile. Then, the Mee client sends such information to the Mee controller. For simplicity, we consider only `txt` file in our research, i.e., files with the `txt` extension. For the traditional honeyfile system simulation, we deploy the same number of honeyfiles in each device. Then, we change the number of honeyfiles in each device and observe the defender's performance, the attacker's cost, and the false/true positive rate. For Mee, we deploy 40 honeyfiles in each device as the initial statement. The defender can modify the number of honeyfiles based on its belief of network security situation.

Chapter 5

GAN Based Honeyfile Traffic Generation for Passive Monitoring

Abstract: Enterprises networks are increasingly concerned about adversaries that slowly and deliberately exploit resources over months or even years. A critical step in this cyber kill chain is network reconnaissance, which has been active (e.g., network scans) and therefore detectable. However, new networking technology increases the possibility of *passive* network reconnaissance, which is challenging to be detected by defenders. In this paper, we propose *honey traffic*, as a technique that uses deceptively crafted network flow to confound the knowledge gained through passive network reconnaissance. We present a Generative Adversarial Network (GAN) to learn the features of actual servers and guide the honey traffic generation. In doing so, we demonstrate that honey traffic assists the defender to either dissuade an adversary from discovering the existence of real vulnerabilities or reveal the adversary's presence when it attempts to attack an intrusion detection node.

5.1 Introduction

5.1.1 Passive Monitoring

Banner Grabbing

The banner grabbing can be defined as connecting to a remote applications and observing the feedback. This can be informative to network administrators, as well as the remote attackers. Adversary applies banner grabbing to reveal compromising information about the services that are running on a system.

Packet Sniffing

Packet sniffing is the process of gathering traffic away from a network by capturing the data as they pass and store them for later analysis. By capturing this network communication, a packet sniffer can reassemble the network packets to view the information originally sent over the network. Any data sent in plaintext, such as user names, passwords, IP addresses, and other sensitive data, are all vulnerable to eavesdropping.

5.1.2 Generative Adversarial Networks (GANs)

Original GAN architecture includes *generator* and *discriminator* (Fig. 5.1). Typically, a generator is trained to map from a latent space to a data distribution, and the discriminator distinguishes candidates produced by the generator from the true data distribution. The objective of trained generator is to increase the error rate of the discriminator.

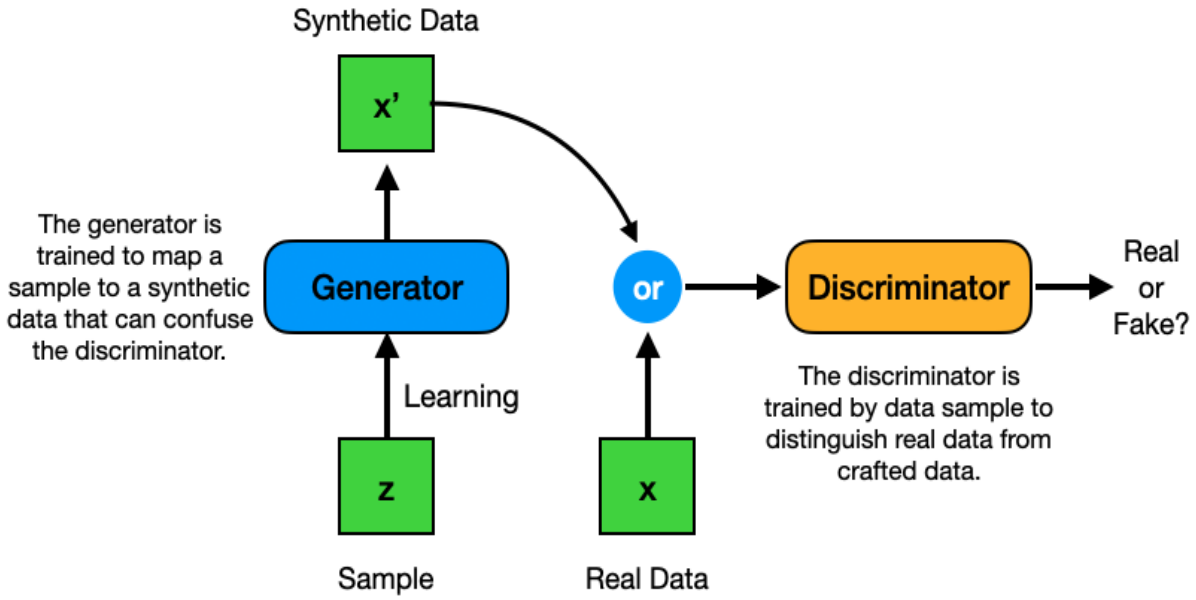


Figure 5.1: Original GAN Architecture

5.1.3 Defensive Deception

High interactive honeypot is a common used defensive deception technology. The main objective of honeypot is attract attackers' attention and protect real hosts. However, an adversary may do reconnaissance before its further malicious actions, such as compromising a target. One challenge of honeypot is how to create a real enough fake host. In this paper, we consider passive reconnaissance as the threat model. Meanwhile, we introduce *Honeyflow* to mimic the real network traffic and confuse the attacker's reconnaissance process.

5.2 Problem Statement and Assumption

The adversary aims to gather information about enterprise network resources and their vulnerabilities through passive reconnaissance (e.g., packet sniffer). We assume the adversary has gained a foothold on one or more network switches and can inspect all packets flowing through the compromised switches. As a result, the adversary can learn about (1) partial connectivity of the network, (2) client and server identity, (3) banner information and other fingerprints, and (4) other sensitive data (without encryption).

We assume the adversary has the necessary processing power to analyze the abovementioned knowl-

edge to identify its target information, including banner information and IP address. After that, the attacker analyzes the traffic pattern and packet headers to identify possible vulnerable or critical targets. We assume the attacker obtains at least one compromised switch as the vantage point to deploy reconnaissance as the initial situation. Meanwhile, we also assume the attacker can only capture packets through the compromised switches. We focus on passive traffic monitoring without discussing Openflow switch compromise methods and other malicious actions in the SDN environment. Specifically, our trust base includes the SDN controller and the communication between Openflow Switch and the controller.

We assume the defender's network contains actual hosts with exploitable vulnerabilities. For example, the defender's inventory system may indicate the existence of an unpatched and vulnerable server, but due to production requirements, the server has not been patched yet. Finally, we assume that the network utilization is not near-maximum capacity during regular operation. However, exceeding honey traffic may cause congestion and cause network degradation.

Our trusted computing base (TCB) includes the system defender (SDN controller or a separate trusted server) and the southbound network between the SDN controller and SDN switches. We assume the SDN switches are configured to either use out-of-band communication or in-band communication protected by TLS. We do not assume SDN switches are trustworthy; however, we assume not all the switches are compromised. The fake hosts (physical and logical) initiating honey-traffics are also part of TCB.

5.3 Deception Architecture

Network vulnerabilities constantly evolve, leaving weaknesses in the network for the adversaries to intrude on the system. Monitoring and patching vulnerabilities for all the nodes in the network require an intense amount of resources and might not be cost-effective. To protect the valuable hosts in the network, the defender can deploy fake hosts (e.g., honeypot) to attract the attacker's attention. Meanwhile, one challenge is how to create natural enough fake hosts to mislead an attacker's behavior. This work proposes honey traffic to mimic the TCP-based network traffic for deceiving the attacker's reconnaissance processes and assist the fake hosts in mimicking the corresponding host.

Servers have different individuality or performance. For example, a server may receive more requests in the morning than at night. To produce realistic honey traffic, a generator needs to learn the pattern of a real server. For doing so, we present the GAN generator to learn from the actual traffic from a server as the instruction of honey traffic creation and generate corresponding fake traffic.

However, the GAN model can only process continuous input attributes, but flow-based network data consist of continuous and categorical attributes (e.g., *IP addresses or port numbers*). Therefore, we consider network dataflow attributes as **utterances** in natural language. We adopt deep learning approaches for natural language generation tasks in this work to overcome the above challenge. Donahue et al. Donahue and Rumshisky [2018] proposed LaTextGAN to utilize an autoencoder to learn a low-dimensional representation of sentences and generate sentences with Improved Wasserstein GAN (WGAN) Gulrajani et al. [2017]. We adopt the autoencoder model for representing the attributes of the network traffic packets. Moreover, we utilize the autoencoder to encode attributes of network data (e.g., *IP addresses and port number*) into smooth representations. The generator network is then trained to generate its flow-based representations in the learned latent space. Each network flow-based data vector produced by the generator is then passed through the decoder, which decodes to the nearest network

attributes.

GAN architecture

The generator network produces additional points in this latent variable space, which decode to valid network packets. As is typical for Generative Adversarial Networks, a discriminator network is trained to classify actual and generated packets from their latent representations. In this work, we model the defender by using the generator in GAN, which attempts to fool the discriminator by generating more realistic packets representations. We adopt synthetic flow-based network traffic generators, which are based on Improved WGAN Gulrajani et al. [2017] network. The WGAN improved the original model by using gradient penalty as a soft constrain to enforce the Lipschitz constraint. Furthermore, it applied the training objective to large ResNet architectures He et al. [2016] to mitigate gradient instability associated with the randomly-initialized fully connected layers.

We use the discriminator to model the attacker's passive reconnaissance processes. We assume that an attacker can obtain the traffic from a compromised switch and identify the hosts' background information through the gathered data. However, the defender may use honeypot and honey traffic to confuse the attacker.

5.4 Models

Node Model

The research from Montigny-Leboeuf and Massicotte [2004] introduces a game-theoretic model to analyze the optimal strategy of a defender when it applies honeypot as the deception system. They utilize a variable, named *importance*, to represent the host and honeypot value. In our research, we apply a similar definition and extend this idea with a new parameter, named *Vulnerability*.

We apply the Importance value ($I \in [1, 10]$) to represent this host's worth for the defender to differentiate the hosts. The hosts may obtain different values based on the importance of the whole network. For example, the server or database in SDN should own larger I than the laptops.

Additional, we utilize *Vulnerability value (*$V \in [1, 10]$*)* to represent how difficult to compromise a host for attacker. Every host in the network obtains a Vulnerability value based on its actual situation, such as the OS type and application version. For example, suppose a host owns an old version of OS or applications. In that case, it will get a relatively high Vulnerability value, which means the attacker can use less effort or resource to compromise it. Via the reconnaissance, the attacker can obtain these two values of some hosts and choose a target to compromise. However, the defender can apply honey traffic and fake host(s) to confuse the attacker's observation.

For example, we have three real hosts (H1, H2, and H3). The importance vector(IV) of these three hosts is [3, 8, 10], and the vulnerability vector(VV) is [2, 9, 1]. Without honey traffic, the attacker observes these hosts' actual values and compromises H2 as its optimal choice. However, with honey traffic and fake host, the defender can change the Importance vector and Vulnerability vector by adding another element in each of these two vectors. For example, if attacker observes honey traffic from fake host(s), its observation may become to such as $IV = [3, 8, 10, 9]$ and $VV = [2, 9, 1, 10]$. In this situation, the attacker may prefer to compromise the fake host rather than H2.

Attacker Model

We consider passive monitoring (e.g., packets sniffer) as the threat model. Attackers can gather traffic from a network by capturing the packets when they pass through the compromised switch(es) and store the data for later analysis. For example, a packet sniffer can reassemble the network packets by capturing the network communication to view the information initially sent over the network.

In order to model the attacker behaviors, we introduce the **Attacker Actions** as:

- Scanning Traffic (AA_1): This action can only be applied in a compromised switch. Through scanning traffic, an attacker can update its belief, such as hosts vulnerabilities.
- Compromise End Host (AA_2): This action represents that the adversary tries to intrude on a host rely on its belief in a particular host, such as a vulnerable OS or application.
- Do Nothing (AA_3): If the adversary can not determine a target to compromise or decide to save its energy, it can select this action to keep its state.

Based on the attacker action space, there are three possible **results**:

- Success: This result represents the attacker capturing the target device's correct vulnerability and successfully penetrating this device.
- Invalid: This result denotes the attacker does not obtain helpful information to compromise the device. The attacker can keep monitoring the traffic to collect information to compromise the target.
- Defeat: This result represents the attacker trusting the false information inside honey traffic and attacking a fake host result in expose the attacker and waste its efforts.

5.4.1 Defender Model

The defender assigns core values to each node based on the vulnerability scores, ease of exploits, previous attack records, and other parameters. Many tools are available to rank nodes based on the risk factor. Via the vulnerability score, a defender prioritizes nodes and implements deception to reduce the chance of exploiting critical hosts. Our research applies partial concepts from Han et al. [2016] and modified them based on our threat model. **Observation of Defender** Network data observation is essential to keep the network smooth and efficient. Monitoring and capturing specific traffics passing through the network, especially vulnerable servers, is vital before implementing honey flow. Various sniffing tools can be deployed to achieve the goal. By using sniffing tools, protected and vulnerable traffic can be sorted.

Defender Observation includes:

- Normal Traffic Flows: This observation contains all the information that the SDN controller and network administrator can know, such as the flow rules and the amount of network flows.
- Honey Traffic: The defender has knowledge about all the information about the honey traffic, includes the path, amount and contained misinformation of all the honey traffic.
- Partial Misbehavior: we assume if attacker trust the misinformation in honey traffic and deploy attacks based on the misinformation, the defender can detect them.

- System Security Situation: the defender can know the security situation that is introduced in the node model.

Actions can be taken based on vulnerability value, normal traffic statistics and partial misbehavior situation.

- Change honey traffic for creating delusion towards hosts' states.

5.5 Implementation and Evaluation

We deploy our scenario with CyberVAN testbed by considering an SDN environment. Figure 5.2 shows the topology in the simulation. We create a pair of actual sever and client and establish a TCP-based channel between them. Similarly, the fake server and client are built for honey traffic. Specifically, the GAN generator is deployed in the fake sever to learn the real server's pattern for instructing the fake sever to produce the crafted network flow. All the hosts connect with an Openflow switch which is a

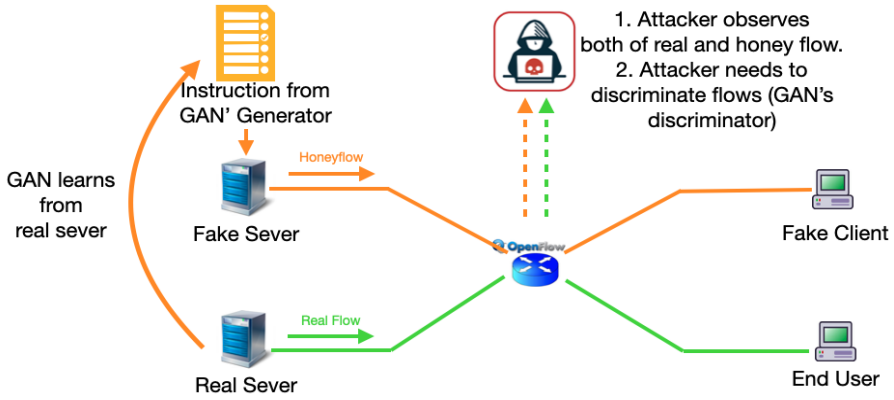


Figure 5.2: Overview of Honeyflow and GAN Model

vantage point of the adversary. Recall the definition of passive monitoring attack. The attacker observes all the traffic passing through the compromised switch and determines its target to intrude. We adopt the GAN discriminator to model the attacker processes, include analyze the network flows and decide which are the actual flows.

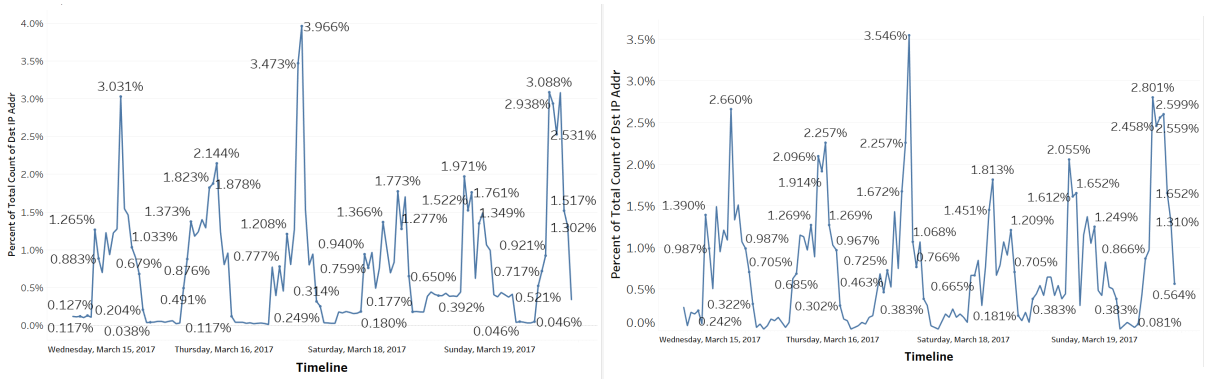
5.5.1 Dataset and Textual Autoencoder

We adopt the CIDDS-001 dataset Ring et al. [2019] which includes flow-based network packets represented with network attributes. From the dataset, we select attributes in Table 5.1 as the features of legitimate network flow.

We utilize an autoencoder component that learns a dense low-dimensional representation of network flow-based data. The encoder network is used to compress information about each attribute into a finite vector. The decoder network is tasked with reconstructing the input representation from the vector. Long-Short Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Furthermore, we aim to learn the pattern

Table 5.1: Attributes overview.

Attribute	Type	Example
data first seen	timestamp	2018-03-13
duration	continuous	0.12
transport protocol	categorical	TCP
source IP address	categorical	192.168.100.5
source port	categorical	52128
destination IP address	categorical	8.8.8.8
destination IP port	categorical	80
bytes	numeric	2391
packets	numeric	12
TCP flags	binar/categorical	.A..S.



(a) Real server

(b) Fake server

Figure 5.3: Amount of network flows of real and fake severs

of real network traffic. Therefore, we utilize Long-Short Term Memory (LSTM) network for both the encoder and the decoder. The LSTM network reads one attribute at a time. During the sentence reconstruction, the decoder takes latent representation and the previous hidden state as input and produces the probability distribution where we select the highest probability word sequentially.

In Figure 5.3a, the X-axis represents the timeline, and Y-axis is the percentage of network flow at a particular time slot to the total traffic amount in a week. The GAN generator learns the feature and guides the fake sever to produce the network flow following the same pattern, as shown in Figure 5.3b).

Chapter 6

Plan of Work

This chapter summarizes our research approaches and plans. We concern defensive deception as the security scheme to detect and mitigate threats. We mainly focus on stealthy attacks as the threat model, such as insider attacks and passive reconnaissance. Due to the nature of these attacks, traditional security methods (e.g., firewall and IDS) are insufficient against them. Therefore, defensive deception proposes a prospective perspective by generating false information to mislead the attack processes. In addition, game theory and machine learning have been commonly adopted to increase the intelligence of deception techniques and assist the defender in searching for the optimal solution. We adopt game theory and machine learning to improve the deception techniques and help the defender search for the optimal solution. Table 6.1 includes our partial projects and the corresponding timelines.

Table 6.1: Time line for research approach

Schedule	Project
Complete	A Survey of Defensive Deception: Approaches Using Game Theory and Machine Learning
Complete	Mee: Adaptable honeyfile system based on Bayesian game
November 2021	Mee v.2: deep reinforcement learning-based adaptive honeyfile system
January 2022	GAN Based Honey Traffic Generation for Passive Monitoring

A Survey of Defensive Deception: Approaches Using Game Theory and Machine Learning

We investigate the current game theory and machine learning-based defensive deception research. We analyze each research with considering:

- the key characteristics of deception techniques that distinguish it from other defensive techniques
- (dis)advantage of each deception technique
- the commonly used metrics when measuring the effectiveness and efficiency of deception techniques.

This paper assists us in an in-depth understanding of defensive deception techniques and the corresponding application of game theory and machine learning. Furthermore, we recognize the limitation of current research, which help us to explore the future research direction.

Mee: Adaptive Honeyfile System for Inside Attacker Detection

We present Mee as a novel honeyfile system. Relies on centralized control and decentralized deployment, Mee assists the defender in obtaining the suitable number of honeyfiles in each device based on the assessment of the network security situation. Specifically, Mee includes a Mee controller and Mee clients. The client is installed in each device and aims to maintain honeyfiles following the instruction from the controller. This work discusses how Mee detects insider attackers, which can penetrate devices and search for valuable files. In addition, our scenario involves legitimate users who may touch the honeyfiles accidentally. As a result, Mee needs to distinguish the honeyfile alarm is from an attacker or a user. We adopt the Bayesian game to model the interaction between insider attackers, legitimate users, and the defender. We consider the game as a two-player dynamic Bayesian game, in which player a is the defender and player b has two potential types (i.e., insider attacker or legitimate user). At the beginning of each game, the player b obtains its type from nature as its private information. To make an optimal decision, the defender needs to estimate player b's type based on its belief, including the corresponding group risk level and the history of previous alarms.

We implement Mee in the testbed by considering an enterprise network and compare it with the traditional honeyfile system follow three metrics: 1) the defender payoff, 2) the attacker payoff, and 3) the true/false positive rate when detecting insider attackers. The simulation results show that Mee is more helpful when detecting insider threats and disturb legitimate users less.

Mee v.2: deep reinforcement learning-based adaptive honeyfile system

This work is an extension of the previous research. In the Mee project, although we involve insider attackers, legitimate users, and the defender in the scenario and model the interaction between them by the Bayesian game, we ignore many details and make assumptions for simplification. In addition, due to the limitation of game theory, we can only model two players at one timestamp.

To increase the complexity and realization, we propose a complete scenario, which includes multiple active users and insider attackers. We use deep reinforcement learning to guide the agent (i.e., the defender) to select actions based on the environment and its states. The environments include all the connected devices, active users, and potential threats. After receiving the honeyfile alarm, the defender needs to determine whether this alarm represents the insider threat detection or is a false positive alarm from the legitimate user. We have finished the honeyfile system design and corresponding settings in the DRL processes. Our next step is to settle the testbed for implementation and evaluation. We expect Mee 2 can assist the defender in differentiating true and false-positive alarms to increase the accuracy when detecting insider attackers.

GAN Based Honey Traffic Generation for Passive Monitoring

We concentrate on dealing with malicious passive monitoring in this work. Adversaries can discover victim devices' information through passive monitoring, such as OS, active services, and vulnerabili-

ties. Unlike active scanning, which is usually detectable and sporadically, passive monitoring is slow, persistent, and undetectable.

We achieve honey traffic as the deception technique that mimics actual network flow and involves deceptive crafted false information to mislead attackers' beliefs and slow down the attack processes. However, attackers can distinguish the honey traffic from actual network traffic by continually observing traffics and analyzing corresponding device features, such as the duration of connection and packet size. For dealing with this, we adopt GANs to learn from the actual traffic and assist the defender in generating more realistic honey traffic. A GAN model involves a generator and a discriminator, which the generator aims to create crafted data, and the discriminator needs to distinguish the fake data from actual data. Due to this nature, we utilize the generator to learn from the existing network for producing realistic honey traffic and adopt the discriminator to mimic the attacker. Our current testbed includes two servers and two clients. The simulation results show that the honey traffic can successfully disturb attackers' beliefs. The next step of this work is to complement the implementation, such as increasing the complexity of the network environment. In addition, we are designing a game model to demonstrate optimal defender strategies that will either prevent an adversary from acting on the existence of real vulnerabilities or reveal the adversary's presence when it attempts to intrude on nodes.

References

Illusive. URL https://illusive.com/. Accessed: 07-18-2021.

TRAPX SECURITY. URL https://www.trapx.com/resources/. Accessed: 07-18-2021.

Nazmiye Ceren Abay, Cuneyt Gurcan Akcora, Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham. Using deep learning to generate relational HoneyData. In *Autonomous Cyber Deception*, pages 3–19. Springer, 2019.

Stefan Achleitner, Thomas La Porta, Patrick McDaniel, Shridatt Sugrim, Srikanth V. Krishnamurthy, and Ritu Chadha. Cyber deception: Virtual networks to defend insider reconnaissance. In *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, pages 57–68, 2016.

Palvi Aggarwal, Cleotilde Gonzalez, and Varun Dutt. Cyber-security: Role of deception in cyber-attack detection. In *Advances in Human Factors in Cybersecurity*, pages 85–96. Springer, 2016.

Palvi Aggarwal, Cleotilde Gonzalez, and Varun Dutt. Modeling the effects of amount and timing of deception in simulated network scenarios. In *Proc. 2017 Int'l Conf. On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–7. IEEE, 2017.

Palvi Aggarwal, Aksh Gautam, Vaibhav Agarwal, Cleotilde Gonzalez, and Varun Dutt. HackIt: A human-in-the-loop simulation tool for realistic cyber deception experiments. In *Proc. Int'l Conf. on Applied Human Factors and Ergonomics*, pages 109–121. Springer, 2019.

Md Al Amin, Sachin Shetty, Laurent Njilla, Deepak Tosh, and Charles Kamouha. Attacker capability based dynamic deception model for large-scale networks. *EAI Endorsed Transactions on Security and Safety*, 6(21), 2019a.

Md Ali Reza Al Amin, Sachin Shetty, Laurent Njilla, Deepak K Tosh, and Charles Kamhoua. Online cyber deception system using partially observable Monte-Carlo planning framework. In *Proc. Int'l Conf. on Security and Privacy in Communication Systems*, pages 205–223. Springer, 2019b.

Md Ali Reza Al Amin, Sachin Shetty, Laurent L Njilla, Deepak K Tosh, and Charles A Kamhoua. Dynamic cyber deception using partially observable monte-carlo planning framework. *Modeling and Design of Secure Internet of Things*, pages 331–355, 2020.

Ehab Al-Shaer, Jinpeng Wei, Kevin W. Hamlen, and Cliff Wang. Towards intelligent cyber deception systems. In *Autonomous Cyber Deception*, pages 21–33. Springer, 2019.

Mohammed H Almeshekah and Eugene H Spafford. Planning and integrating deception into computer security defenses. In *Proc. New Security Paradigms Workshop*, pages 127–138, 2014.

Mohammed H Almeshekah and Eugene H Spafford. Cyber security deception. In *Cyber Deception*, pages 23–50. Springer, 2016.

A. Alshammari, D. B. Rawat, M. Garuba, C. A. Kamhoua, and L. L. Njilla. *Deception for Cyber Adversaries: Status, Challenges, and Perspectives*, pages 141–160. Wiley Online Library, 2020.

Iffat Anjum, Mu Zhu, Isaac Polinsky, William H. Enck, Michael K. Reiter, and Munindar P. Singh. Role-based deception in enterprise networks. In *Proceedings of the 11th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 65–76, Online, April 2021. ACM. doi: 10.1145/3422337.3447824.

A. H. Anwar, C. Kamhoua, and N. Leslie. Honeypot allocation over attack graphs in cyber deception games. In *Proc. 2020 Int'l Conf. on Computing, Networking and Communications (ICNC)*, pages 502–506, 2020.

Ahmed H Anwar and Charles Kamhoua. Game theory on attack graph for cyber deception. In *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2020.

Ahmed H Anwar, Charles Kamhoua, and Nandi Leslie. A game-theoretic framework for dynamic cyber deception in Internet of Battlefield Things. In *Proc. 16th EAI Int'l Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 522–526, 2019.

Frederico Araujo, Kevin W. Hamlen, Sebastian Biedermann, and Stefan Katzenbeisser. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proc. ACM SIGSAC Conf. on Computer and Communications Security*, pages 942–953, New York, NY, USA, 2014. ACM.

C. A. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Location privacy protection through obfuscation-based techniques. In Steve Barker and Gail-Joon Ahn, editors, *Proc. Data and Applications Security XXI*, pages 47–60, Berlin, 2007. Springer.

Ofir Arkin and Fyodor Yarochkin. Xprobe v2. 0. *Fuzzy" Approach to Remote Active Operating Systems Fingerprinting," http://www. xprobe2. org*, 2002.

Jeffrey Avery and Eugene H. Spafford. Ghost patches: Fake patches for fake vulnerabilities. In Sabrina De Capitani di Vimercati and Fabio Martinelli, editors, *Proc. 32nd IFIP TC 11 Int'l Conf. on ICT Systems Security and Privacy Protection*, pages 399–412, Rome, 2017. Springer Int'l Publishing.

Jeffrey Avery and John Ross Wallrabenstein. Formally modeling deceptive patches using a game-based approach. *Computers & Security*, 75:182–190, 2018.

Prudhvi Ratna Badri Satya, Kyumin Lee, Dongwon Lee, Thanh Tran, and Jason Jiasheng Zhang. Uncovering fake likers in online social networks. In *Proc. 25th ACM Int'l on Conf. on Information and Knowledge Management*, pages 2365–2370. ACM, 2016.

Craig Bakker, Arnab Bhattacharya, Samrat Chatterjee, and Draguna L Vrabie. Metagames and hypergames for deception-robust control. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–25, 2021.

Kiran Bandla. Aptnotes. https://github.com/kbandla/APTnotes.git, 2019.

Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Understanding passive and active service discovery. In *Proc. 7th ACM SIGCOMM Conf. on Internet measurement*, pages 57–70, 2007.

Anjon Basak, Charles Kamhoua, Sridhar Venkatesan, Marcus Gutierrez, Ahmed H. Anwar, and Christopher Kiekintveld. Identifying stealthy attackers in a game theoretic framework using deception. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 21–32. Springer, 2019.

Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*, volume 23. Siam, 1999.

J. Bowyer Bell and Barton Whaley. *Cheating and Deception*. Transaction, New York, 1991.

David Bellhouse. The problem of Waldegrave. *Electronic Journal for the History of Probability and Statistics*, 3(2):1–12, 2007.

Malek Ben Salem and Salvatore Stolfo. Combining baiting and user search profiling techniques for masquerade detection. *Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications*, 3(1), Mar. 2012.

Michael Bennett and Edward Waltz. *Counterdeception Principles and Applications for National Security*. Artech House, 2007.

Kevin Benton, L Jean Camp, and Chris Small. Openflow vulnerability assessment. In *Proc. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pages 151–152, 2013.

Mark Bilinski, Kimberly Ferguson-Walter, Sunny Fugate, Ryan Gabrys, Justin Mauger, and Brian Souza. You only lie twice: A multi-round cyber deception game of questionable veracity. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 65–84. Springer, 2019.

Jean-Marie Borello and Ludovic Mé. Code obfuscation techniques for metamorphic viruses. *Journal in Computer Virology*, 4(3):211–220, 2008.

Brian M Bowen, Shlomo Hershkop, Angelos D Keromytis, and Salvatore J Stolfo. Baiting inside attackers using decoy documents. In *Proc. Int'l Conf. on Security and Privacy in Communication Systems*, pages 51–70. Springer, 2009.

Cristian Cadar, Daniel Dunbar, and Dawson R Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *Operating Systems Design and Implementation*, volume 8, pages 209–224, 2008.

Joseph W. Caddell. Deception 101-primer on deception. Technical report, DTIC Document, 2004.

Jin-Yi Cai, Vinod Yegneswaran, Chris Alfeld, and Paul Barford. An attacker-defender game for honeynets. In *Proc. Int'l Computing and Combinatorics Conf.*, pages 7–16. Springer, 2009.

Thomas E. Carroll and Daniel Grosu. A game theoretic investigation of deception in network security. *Security and Communication Networks*, 4(10):1162–1172, 2011.

William Casey, Jose Andre Morales, Evan Wright, Quanyan Zhu, and Bud Mishra. Compliance signaling games: Toward modeling the deterrence of insider threats. *Computational and Mathematical Organization Theory*, 22(3):318–349, 2016.

William Austin Casey, Quanyan Zhu, Jose Andre Morales, and Bud Mishra. Compliance control: Managed vulnerability surface in social-technological systems via signaling games. In *Proc. 7th ACM CCS Int'l Workshop on Managing Insider Security Threats*, pages 53–62, 2015.

Anthony R Cassandra. A survey of POMDP applications. In *Proc. Working notes of AAAI 1998 fall Symp. on planning with partially observable Markov decision processes*, volume 1724, 1998.

Hayreddin Çeker, Jun Zhuang, Shambhu Upadhyaya, Quang Duy La, and Boon-Hee Soong. Deception-based game theoretical approach to mitigate DoS attacks. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 18–38. Springer, 2016.

Ritu Chadha, Thomas Bowen, Cho-Yu J Chiang, Yitzchak M Gottlieb, Alex Poylisher, Angello Sapello, Constantin Serban, Shridatt Sugrim, Gary Walther, Lisa M Marvel, et al. CyberVAN: A cyber security virtual assured network testbed. In *Proc. MILCOM 2016-2016 IEEE Military Communications Conf.*, pages 1125–1130. IEEE, 2016.

Jien-Tsai Chan and Wuu Yang. Advanced obfuscation techniques for Java bytecode. *Journal of Systems and Software*, 71(1):1–10, Apr. 2004.

Ping Chen, Lieven Desmet, and Christophe Huygens. A study on Advanced Persistent Threats. In *Proc. IFIP Int'l Conf. on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

Cho-Yu J Chiang, Yitzchak M Gottlieb, Shridatt James Sugrim, Ritu Chadha, Constantin Serban, Alex Poylisher, Lisa M Marvel, and Jonathan Santos. ACyDS: An adaptive cyber deception system. In *Proc. 2016 IEEE Military Communications Conf.*, pages 800–805. IEEE, 2016.

Cho-Yu J Chiang, Sridhar Venkatesan, Shridatt Sugrim, Jason A. Youzwak, Ritu Chadha, Edward I. Colbert, Hasan Cam, and Massimiliano Albanese. On defensive cyber deception: A case study using SDN. In *Proc. IEEE Military Communications Conf. (MILCOM)*, pages 110–115. IEEE, 2018.

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 Communications Surveys Tutorials*, pages 1–1, 2020. Early access.

Jin-Hee Cho and Noam Ben-Asher. Cyber defense in breadth: Modeling and analysis of integrated defense systems. *The Journal of Defense Modeling and Simulation*, 15(2):147–160, 2018.

Jin-Hee Cho, Mu Zhu, and Munindar P. Singh. *Modeling and Analysis of Deception Games based on Hypergame Theory*, chapter 4, pages 49–74. Springer Nature, Cham, Switzerland, 2019a.

Jin-Hee Cho, Mu Zhu, and Munindar P. Singh. Modeling and analysis of deception games based on hypergame theory. In *Autonomous Cyber Deception*, pages 49–74. Springer, 2019b.

Andrew Clark, Quanyan Zhu, Radha Poovendran, and Tamer Başar. Deceptive routing in relay networks. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 171–185. Springer, 2012.

Wikipedia contributors. K-means clustering. URL https://en.wikipedia.org/wiki/K-means$_$clustering. Accessed: 07-18-2021.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Edward Cranford, Cleotilde Gonzalez, Palvi Aggarwal, Sarah Cooney, Milind Tambe, and Christian Lebiere. Adaptive cyber deception: Cognitively informed signaling for cyber defense. In *Proc. 53rd Hawaii Int'l Conf. on System Sciences*, 2020.

Edward A Cranford, Cleotilde Gonzalez, Palvi Aggarwal, S Cooney, M Tambe, and C Lebiere. Towards personalized deceptive signaling for cyber defense using cognitive models. In *Proc. 17th Annual Meeting of the ICCM, Montreal, CA*, volume 56, 2019.

Michael Crouse, Bryan Prosser, and Errin W Fulp. Probabilistic performance analysis of moving target and deception reconnaissance defenses. In *Proc. 2nd ACM Workshop on Moving Target Defense*, pages 21–29, 2015.

Donald Charles Daniel and Katherine Lydigsen Herbig. *Strategic Military Deception*. Pergamon, 1982.

P. Danzig, J. Mogul, V. Paxson, and M. Schwartz. The Internet traffic archive, 2008. URL http://ita.ee.lbl.gov/html/traces.html.

Prithviraj Dasgupta and Joseph Collins. A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. *AI Magazine*, 40(2):31–43, Jun. 2019.

Christos K Dimitriadis. Improving mobile core network security with honeynets. *IEEE Security & Privacy*, 5(4):40–47, 2007.

Advait Dixit, Fang Hao, Sarit Mukherjee, TV Lakshman, and Ramana Kompella. Towards an elastic distributed SDN controller. In *Proc. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pages 7–12, 2013.

Basirudin Djamaluddin, Ahmed Alnazeer, and Farag Azzedin. Web deception towards moving target defense. In *Proc. 2018 Int'l Carnahan Conf. on Security Technology (ICCST)*, pages 1–5. IEEE, 2018.

Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, 1997.

David Donahue and Anna Rumshisky. Adversarial text generation without reinforcement learning. *ArXiv*, abs/1810.06640, 2018.

J. F. Dunnigan and A. A. Nofi. *Victory and deceit*. Writers Club Press, 2nd edition, 2001.

K. Durkota, V. Lisỳ, B. Bošanskỳ, and C. Kiekintveld. Optimal network security hardening using attack graph games. In *Proc. 24th Int'l Joint Conf. on Artificial Intelligence*, 2015.

A. I. M. Efendi, Z. Ibrahim, M. N. A. Zawawi, F. Abdul Rahim, N. A. M. Pahri, and A. Ismail. A survey on deception techniques for securing web application. In *Proc. 2019 IEEE 5th Int'l Conf. on Big Data Security on Cloud (BigDataSecurity)*, pages 328–331, 2019.

E. B. El Idrissi Younes, E. M. Fatna, and M. Nisrine. A security approach for social networks based on honeypots. In *Proc. 2016 4th IEEE Int'l Colloquium on Information Science and Technology (CiSt)*, pages 638–643, 2016.

A. El-Kosairy and M. A. Azer. A new web deception system framework. In *Proc. 1st Int'l Conf. on Computer Applications & Information Security (ICCAIS)*, pages 1–10. IEEE, 2018.

K. Ferguson-Walter, T. Shade, A. Rogers, M. C. S. Trumbo, K. S Nauer, K. M. Divis, A. Jones, A. Combs, and R. G. Abbott. The tularosa study: An experimental design and implementation to quantify the effectiveness of cyber deception. Technical report, Sandia National Lab (SNL-NM), Albuquerque, NM (United States), 2018.

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*, pages 1–8, 2019.

K. J. Ferguson-Walter. *An empirical assessment of the effectiveness of deception for cyber defense*. PhD thesis, University of Massachusetts Amherst, 2020.

Michelle Forelle, Phil Howard, Andrés Monroy-Hernández, and Saiph Savage. Political bots and the manipulation of public opinion in venezuela. *arXiv Preprint arXiv:1507.07109*, 2015.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29 (2):131–163, 1997.

Nandan Garg and Daniel Grosu. Deception in honeynets: A game-theoretic analysis. In *Proc. IEEE SMC Information Assurance and Security Workshop*, pages 107–113. IEEE, 2007.

Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro. R-Locker: Thwarting ransomware action through a honeyfile-based approach. *Computers & Security*, 73:389–398, 2018.

A Greenwald. Matrix games and nash equilibrium. *Lecture in Game-theoretic Artificial Intelligence, Brown University CS Dep. Providence, US*, 2007.

K. Grover, A. Lim, and Q. Yang. Jamming and anti–jamming techniques in wireless networks: A survey. *Int'l Journal of Ad Hoc and Ubiquitous Computing*, 17(4):197–215, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5769–5779, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Z. Guo, J.-H. Cho, I.-R. Chen, S. Sengupta, M. Hong, and T. Mitra. Online social deception and its countermeasures for trustworthy cyberspace: A survey, 2020a.

Zhen Guo, Jin-Hee Cho, Ray Chen, Srijan Sengupta, Michin Hong, and Tanushree Mitra. Online social deception and its countermeasures: A survey. *IEEE Access*, 2020b.

J. Han, J. Pei, and M. Kamber. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

Wonkyu Han, Ziming Zhao, Adam Doupé, and Gail Joon Ahn. HoneyMix: Toward SDN-based intelligent Honeynet. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, co-located with CODASPY*, pages 1–6. ACM, March 2016. ISBN 9781450340786. doi: 10.1145/2876019.2876022.

X. Han, N. Kheir, and D. Balzarotti. Deception techniques in computer security: A research perspective. *ACM Computing Surveys*, 51(4), Jul. 2018.

E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

John C. Harsanyi. Games with incomplete information played by "Bayesian" players, I–III Part I. the basic model. *Management Science*, 14(3):159–182, 1967.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

J. P. Hespanha, Y. S. Ateskan, H. Kizilocak, et al. Deception in non-cooperative games with partial information. In *Proc. 2nd DARPA-JFACC Symp. on Advances in Enterprise Control*, pages 1–9, 2000.

W. Hofer, T. Edgar, D. Vrabie, and K. Nowak. Model-driven deception for control system environments. In *Proc. IEEE Int'l Symp. on Technologies for Homeland Security (HST)*, pages 1–7. IEEE, 2019.

K. Horák, Q. Zhu, and B. Bošanskỳ. Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 273–294. Springer, 2017.

James Thomas House and George Cybenko. Hypergame theory applied to cyber attack and defense. In *Proc. SPIE Conf. on Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, volume 766604, May. 2010.

L. Huang and Q. Zhu. Dynamic Bayesian games for adversarial and defensive cyber deception. In *Autonomous Cyber Deception*, pages 75–97. Springer, 2019.

Jafar Haadi Jafarian and Amirreza Niakanlahiji. A deception planning framework for cyber defense. In *Proc. 53rd Hawaii Int'l Conf. on System Sciences*, 2020.

Sushil Jajodia, Noseong Park, Fabio Pierazzi, Andrea Pugliese, Edoardo Serra, Gerardo I. Simari, and V. S. Subrahmanian. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security*, 12(11):2532–2544, 2017.

A. Jøsang, J. Cho, and F. Chen. Uncertainty characteristics of subjective opinions. In *Proc. 2018 21st Int'l Conf. on Information Fusion (FUSION)*, pages 1998–2005, 2018.

Atsushi Kajii and Stephen Morris. The robustness of equilibria to incomplete information. *Econometrica: Journal of the Econometric Society*, pages 1283–1309, 1997.

C. A. Kamhoua. Game theoretic modeling of cyber deception in the Internet of Battlefield Things. In *Proc. 2018 56th Annual Allerton Conf. on Communication, Control, and Computing (Allerton)*, pages 862–862. IEEE, 2018.

C.A. Kamhoua, C.D. Kiekintveld, F. Fang, and Q. Zhu. *Game Theory and Machine Learning for Cyber Security*. Wiley, 2021.

S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. *SIGCOMM Comput. Commun. Rev.*, 39(4):243–254, Aug. 2009.

Pradeeban Kathiravelu and Luís Veiga. Sd-cps: taming the challenges of cyber-physical systems with a software-defined approach. In *Proc. 2017 Fourth Int'l Conf. on Software Defined Systems (SDS)*, pages 6–13. IEEE, 2017.

A. D. Keromytis and S. J. Stolfo. Systems, methods, and media for generating bait information for trap-based defenses, Aug. 2014. US Patent 8,819,825.

C. Kiekintveld, V. Lisỳ, and R. Píbil. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer, 2015.

M. Kocakulak and I. Butun. An overview of wireless sensor networks towards Internet of Things. In *Proc. IEEE 7th Annual Computing and Communication Workshop and Conf. (CCWC)*, pages 1–6. IEEE, 2017.

A. Kott, A. Swami, and B. West. The Internet of Battle Things. *IEEE Computer*, Dec. 2016.

T. Krueger, H. Gascon, N. Krämer, and K. Rieck. Learning stateful models for network honeypots. In *Proc. 5th ACM workshop on Security and artificial intelligence*, pages 37–48, 2012.

Abhishek N. Kulkarni, Jie Fuand Huan Luo, Charles A. Kamhoua, and Nandi N. Leslie. Decoy allocation games on graphs with temporal logic objectives. In *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2020.

Abhishek N Kulkarni, Huan Luo, Nandi O Leslie, Charles A Kamhoua, and Jie Fu. Deceptive labeling: Hypergames on graphs for stealthy deception. *IEEE Control Systems Letters*, July, 2021.

Q. D. La, T. Q. Quek, J. Lee, S. Jin, and H. Zhu. Deceptive attack and defense game in honeypot-enabled networks for the Internet-of-Things. *IEEE Internet of Things Journal*, 3(6):1025–1035, 2016.

K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: Social honeypots + machine learning. In *Proc. 33rd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 435–442, 2010.

K. Lee, B. D. Eoff, and J. Caverlee. Seven months with the devils: A long-term study of content polluters on Twitter. In *Proc. 5th Int'l AAAI Conf. on Weblogs and Social Media*, pages 185–192, 2011.

Y. Liu, H. Wang, M. Peng, J. Guan, J. Xu, and Y. Wang. DeePGA: A privacy-preserving data aggregation game in crowdsensing via deep reinforcement learning. *IEEE Internet of Things Journal*, 7(5): 4113–4127, 2020.

Z. Lu, C. Wang, and S. Zhao. Cyber deception for computer and network security: Survey and challenges. *arXiv preprint arXiv:2007.14497*, 2020.

G. Lyon. Nmap website. https://nmap.org/. Accessed: 07-18-2021.

D. C. MacFarland and C. A. Shue. The SDN shuffle: Creating a moving-target defense using host-based software-defined networking. In *Proc. ACM Workshop on Moving Target Defense*, pages 37–41, 2015.

D. Mao, S. Zhang, L. Zhang, and Y. Feng. Game theory based dynamic defense mechanism for SDN. In *Proc. Int'l Conf. on Machine Learning for Cyber Security*, pages 290–303. Springer, 2019.

Mirco Marchetti, Fabio Pierazzi, Alessandro Guido, and Michele Colajanni. Countering Advanced Persistent Threats through security intelligence and big data analytics. In *2016 8th International Conference on Cyber Conflict (CyCon)*, pages 243–261. IEEE, 2016.

A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic theory*, volume 1. Oxford University Press New York, 1995.

Mohammad Miah, Nazia Sharmin, Iffat Anjum, Mu Zhu, Christopher Kiekintveld, William H. Enck, and Munindar P. Singh. Optimizing vulnerability-driven honey traffic using game theory. In *Proc. Workshop on Artificial Intelligence for Cyber Security (AICS)*, pages 1–7, New York, Feb. 2020.

Stephanie Milani, Weiran Shen, Kevin S. Chan, Sridhar Venkatesan, Nandi O. Leslie, Charles A. Kamhoua, and Fei Fang. Harnessing the power of deception in attack graph games. In *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2020.

A. Mohammadi, M. H. Manshaei, M. M. Moghaddam, and Q. Zhu. A game-theoretic analysis of deception over social networks using fake avatars. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 382–394. Springer, 2016.

D. Montigny-Leboeuf and F. Massicotte. Passive network discovery for real time situation awareness. Technical report, Communications Research Centre Ottawa (Ontario), 2004.

Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

S. Nan, S. Brahma, C. A. Kamhoua, and N. Leslie. Behavioral cyber deception: A game and prospect theoretic approach. In *Proc. 2019 IEEE Global Communications Conf. (GLOBECOM)*, pages 1–6. IEEE, 2019.

S. Nan, S. Brahma, C. A. Kamhoua, and L. L. Njilla. *On Development of a Game-Theoretic Model for Deception-Based Security*, pages 123–140. Wiley Online Library, 2020a.

Satyaki Nan, Swastik Brahma, Charles A Kamhoua, and Nandi O Leslie. Mitigation of jamming attacks via deception. In *Proc. 2020 IEEE 31st Annual Int'l Symp. on Personal, Indoor and Mobile Radio Communications*, pages 1–6. IEEE, 2020b.

S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang. Predicting network attack patterns in SDN using machine learning approach. In *Proc. IEEE Conf. on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 167–172. IEEE, 2016.

Nextgate. Research report 2013 state of social media spam, 2019. URL https://www.slideshare.net/prayukth1/2013-state-of-social-media-spam-research-report. Accessed: 08-05-2019.

H. Okhravi, M. A. Rabe, W. G. Leonard, T. R. Hobson, D. Bigelow, and W. W. Streilein. Survey of cyber moving targets. TR 1166, Lexington Lincoln Lab, MIT, 2013.

J. Pawlick and Q. Zhu. Deception by design: Evidence-based signaling games for network defense. *arXiv preprint arXiv:1503.05458*, 2015.

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*, pages 289–308. Springer, 2015.

J. Pawlick, E. Colbert, and Q. Zhu. Modeling and analysis of leaky deception using signaling games with evidence. *IEEE Transactions on Information Forensics and Security*, 14(7):1871–1886, 2018.

J. Pawlick, E. Colbert, and Q. Zhu. A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. *ACM Computing Surveys (CSUR)*, 52(4):1–28, 2019.

T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys (CSUR)*, 39(1):3–es, 2007.

Luan Huy Pham, Massimiliano Albanese, Ritu Chadha, Cho-Yu J Chiang, Sridhar Venkatesan, Charles Kamhoua, and Nandi Leslie. A quantitative framework to model reconnaissance by stealthy attackers and support deception-based defenses. In *Proc. 2020 IEEE Conf. on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2020.

L. Phlips. *The economics of imperfect information.* Cambridge University Press, 1988.

R. Píbil, V. Lisỳ, C. Kiekintveld, B. Bošanskỳ, and M. Pěchouček. Game theoretic model of strategic honeypot selection in computer networks. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 201–220. Springer, 2012.

S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares. Iioteed: An enhanced, trusted execution environment for industrial iot edge devices. *IEEE Internet Computing*, 21(1):40–47, 2017.

R. Poovendran. Cyber-Physical Systems: Close encounters between two parallel worlds. *Proc. IEEE*, 98 (8), Aug. 2010.

J Ross Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.

Mohammad Ashiqur Rahman, Mohammad Hossein Manshaei, and Ehab Al-Shaer. A game-theoretic approach for deceiving remote operating system fingerprinting. In *Proc. IEEE Conf. on Communications and Network Security (CNS)*, pages 73–81. IEEE, 2013.

Shailendra Rathore, Pradip Kumar Sharma, Vincenzo Loia, Young-Sik Jeong, and Jong Hyuk Park. Social network security: Issues, challenges, threats, and solutions. *Information Sciences*, 421:43–69, 2017.

Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. Flow-based network traffic generation using generative adversarial networks. *Computers & Security*, 82:156–172, 2019. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2018.12.012.

J. J. P. C. Rodrigues, D. B. De Rezende Segundo, H. A. Junqueira, M. H. Sabino, R. M. Prince, J. Al-Muhtadi, and V. H. C. De Albuquerque. Enabling technologies for the Internet of Health Things. *IEEE Access*, 6:13129–13141, 2018.

Kohavi Ron and Provost Foster. Special issue on applications of machine learning and the knowledge discovery process. *Journal of Machine Learning*, 30:271–274, 1998.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *NIPS*, pages 1225–1232, 2007.

Neil C Rowe. Finding logically consistent resource-deception plans for defense in cyberspace. In *Proc. 21st Int'l Conf. on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 1, pages 563–568. IEEE, 2007.

Neil C. Rowe and Julian Rrushi. *Introduction to Cyberdeception.* Springer, 2016.

Neil C Rowe, Binh T Duong, and E John Custy. Fake honeypots: A defensive tactic for cyberspace. In *Proc. IEEE Workshop on Information Assurance*, pages 223–230, 2006.

S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

Malek Ben Salem and Salvatore J Stolfo. Decoy document deployment for effective masquerade attack detection. In *Proc. Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 35–54. Springer, 2011.

Malek Ben Salem, Shlomo Hershkop, and Salvatore J Stolfo. A survey of insider attack detection research. In *Insider Attack and Cyber Security*, pages 69–90. Springer, 2008.

Muhammed O. Sayin and Tamer Başar. Deception-as-defense framework for cyber-physical systems. *arXiv preprint arXiv:1902.01364*, 2019.

D. Serpanos. The cyber-physical systems revolution. *Computer*, 51(3):70–73, 2018.

Jia Shan-Shan and Xu Ya-Bin. The APT detection method based on attack tree for SDN. In *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, pages 116–121, 2018.

Lloyd S. Shapley. Stochastic games. *Proc. National Academy of Sciences*, 39(10):1095–1100, 1953.

Walter L. Sharp. Military deception. Technical Report 3-13.4, Joint War-Fighting Center, Doctrine and Education Group, 2006.

Zheyuan R. Shi, Ariel D. Procaccia, Kevin S. Chan, Sridhar Venkatesan, Noam Ben-Asher, Nandi O. Leslie, Charles Kamhoua, and Fei Fang. Learning and planning in feature deception games. In *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2020.

Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):299–315, 2015.

Chris Snijders, Uwe Matzat, and Ulf-Dietrich Reips. "big data": Big gaps of knowledge in the field of Internet science. *Int'l Journal of Internet Science*, 7(1):1–5, 2012.

Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, and Koji Nakao. Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. In *Proc. First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 29–36, 2011.

Lance Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23, 2003.

Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proc. 26th Annual Computer Security Applications Conf.*, pages 1–9. ACM, 2010.

Steven Tadelis. *Game Theory: An Introduction*. Princeton University Press, 2013.

Hassan Takabi and J Haadi Jafarian. Insider threat mitigation using moving target defense and deception. In *Proc. 2017 Int'l Workshop on Managing Insider Security Threats*, pages 93–96, 2017.

Colin Tankard. Advanced Persistent Threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

Omkar Thakoor, Milind Tambe, Phebe Vayanos, Haifeng Xu, Christopher Kiekintveld, and Fei Fang. Cyber camouflage games for strategic deception. In *Proc. Int'l Conf. on Decision and Game Theory for Security*, pages 525–541. Springer, 2019.

Olivier Tsemogne, Yezekael Hayel, Charles A. Kamhoua, and Gabriel Deugoue. Partially observable stochastic games for cyber deception against epidemic process. In *Proc. Int'l Conf. on Decision and Game Theory for Security*. Springer, 2020.

Maria Vergelis, Tatyana Shcherbakova, and Tatyana Sidorina. Spam and phishing in Q1 2019, 2019. URL https://securelist.com/spam-and-phishing-in-q1-2019/90795/. Accessed: 05-19-2019.

Nikos Virvilis, Bart Vanautgaerden, and Oscar Serrano Serrano. Changing the game: The art of deceiving sophisticated attackers. In *Proc. 6th Int'l Conf. On Cyber Conflict (CyCon)*, pages 87–97. IEEE, 2014.

Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

Jonathan Voris, Jill Jermyn, Nathaniel Boggs, and Salvatore Stolfo. Fox in the trap: Thwarting masqueraders via automated decoy document deployment. In *Proceedings of the Eighth European Workshop on System Security*, pages 1–7, 2015.

Gérard Wagener, Alexandre Dulaunoy, Thomas Engel, et al. Self adaptive high interaction honeypots driven by game theory. In *Symp. on Self-Stabilizing Systems*, pages 741–755. Springer, 2009.

Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. Serf and turf: Crowdturfing for fun and profit. In *Proc. 21st Int'l Conf. on World Wide Web*, pages 679–688. ACM, 2012.

Shuo Wang, Qingqi Pei, Jianhua Wang, Guangming Tang, Yuchen Zhang, and Xiaohu Liu. An intelligent deployment policy for deception resources based on reinforcement learning. *IEEE Access*, 8:35792–35804, 2020.

Bryan C Ward, Steven R Gomez, Richard Skowyra, David Bigelow, Jason Martin, James Landry, and Hamed Okhravi. Survey of cyber moving targets second edition. Technical report, MIT Lincoln Laboratory Lexington United States, 2018.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Ben Whitham. Minimising paradoxes when employing honeyfiles to combat data theft in military networks. In *Proc. Military Communications and Information Systems Conf. (MilCIS)*, pages 1–6. IEEE, 2016.

Ben Whitham. Automating the generation of enticing text content for high-interaction honeyfiles. In *Proc. 50th Hawaii Int'l Conf. on System Sciences*, 2017.

B. Xi and C. A. Kamhoua. *A Hypergame-Based Defense Strategy Toward Cyber Deception in Internet of Battlefield Things (IoBT)*, pages 59–77. Wiley Online Library, 2020.

Jie Xu and Jun Zhuang. Modeling costly learning and counter-learning in a defender-attacker game with private defender information. *Annals of Operations Research*, 236(1):271–289, 2016.

Q. Xu, Z. Su, and R. Lu. Game theory and reinforcement learning based secure edge caching in mobile social networks. *IEEE Transactions on Information Forensics and Security*, 15:3415–3429, 2020.

W. Xu, F. Zhang, and S. Zhu. The power of obfuscation techniques in malicious JavaScript code: A measurement study. In *Proc. 7th Int'l Conf. on Malicious and Unwanted Software*, pages 9–16, 2012.

Chao Yang, Jialong Zhang, and Guofei Gu. A taste of tweets: Reverse engineering Twitter spammers. In *Proc. 30th Annual Computer Security Applications Conf.*, pages 86–95. ACM, 2014.

Yue Yin, Bo An, Yevgeniy Vorobeychik, and Jun Zhuang. Optimal deceptive strategies in security games: A preliminary study. In *Proc. AAAI Conf. on Artificial Intelligence*, 2013.

I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *Proc. Int'l Conf. on Broadband, Wireless Computing, Communication and Applications*, pages 297–300, 2010.

J. J. Yuill. *Defensive Computer-Security Deception Operations: Processes, Principles and Techniques*. PhD thesis, North Carolina State University, 2006.

Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer. Honeyfiles: Deceptive files for intrusion detection. In *Proc. 5th Annual IEEE Information Assurance Workshop*, pages 116–122, 2004.

Jim Yuill, Dorothy Denning, and Fred Feer. Using deception to hide things from hackers: Processes, principles, and techniques. *Journal of Information Warfare*, 5(3):26–40, 2006.

M. Zalewski. p0f v3, 2014. URL https://lcamtuf.coredump.cx/p0f3/. Accessed: 07-18-2021.

Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials*, 15 (4):2046–2069, 2013.

Xiaoxiong Zhang, Keith W. Hipel, Bingfeng Ge, and Yuejin Tan. A game-theoretic model for resource allocation with deception and defense efforts. *Systems Engineering*, 22(3):282–291, 2019.

Yan Zhou, Murat Kantarcioglu, and Bowei Xi. A survey of game theoretic approach for adversarial machine learning. *WIREs Data Mining and Knowledge Discovery*, 9(3):e1259, 2019.

Hangcheng Zhu. *Fighting Against Social Spammers on Twitter by Using Active Honeypots*. PhD thesis, McGill University Libraries, 2015.

Mu Zhu and Munindar P. Singh. Mee: Adaptive honeyfile system for insider attacker detection. In *2021 IEEE Conference on Communications and Network Security (CNS) (IEEE CNS 2021)*, Oct. .

Mu Zhu, Ahmed H Anwar, Zelin Wan, Jin-Hee Cho, Charles Kamhoua, and Munindar P Singh. Game-theoretic and machine learning-based approaches for defensive deception: A survey. *arXiv preprint arXiv:2101.10121*, 2021.

Quanyan Zhu, Andrew Clark, Radha Poovendran, and Tamer Başar. Deceptive routing games. In *Proc. 51st IEEE Conf. on Decision and Control (CDC)*, pages 2704–2711. IEEE, 2012.

Quanyan Zhu, Andrew Clark, Radha Poovendran, and Tamer Başar. Deployment and exploitation of deceptive honeybots in social networks. In *Proc. 52nd IEEE Conf. on Decision and Control*, pages 212–219. IEEE, 2013.