CERIAS Tech Report 2016-5

by Christopher Gutierrez, Mohammed Almeshekah, Eugene Spafford, Saurabh Bagchi, Jeff Avery, Paul Wood
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Modeling Deception In Information Security As A Hypergame – A Primer

Christopher N. Gutierrez
Purdue University
Dept. of Computer Science
305 N. University St.
West Lafayette, IN
gutier20@purdue.edu

Mohammed H. Almeshekah
King Saud University
Computer Science Dept.
Riyadh, Saudi Arabia
meshekah@ksu.edu.sa

Eugene H. Spafford
Purdue University
Dept. of Computer Science
305 N. University St.
West Lafayette, IN
spaf@purdue.edu

Saurabh Bagchi
Purdue University
School of Electrical and
Computer Engineering
West Lafayette, IN
sbagchi@purdue.edu

Jeff Avery
Dept. of Computer Science
West Lafayette, IN
avery0@purdue.edu

Paul Wood
Purdue University
School of Electrical and
Computer Engineering
West Lafayette, IN
pwood@purdue.edu

ABSTRACT

Hypergames are a branch of game theory used to model and analyze game theoretic conflicts between multiple players who may have misconceptions of the other players' actions or preferences. They have been used to model military conflicts such as the Allied invasion of Normandy in 1945 [19], the fall of France in WWII [5], and the Cuban missile crisis [7]. Unlike traditional game theory models, hypergames give us the ability to model misperceptions that result from the use of deception, mimicry, and misinformation. In the security world, there is little work that shows how to use deception in a principled manner as a strategic defensive mechanism in computing systems. In this paper, we present how hypergames model deception in computer security conflicts. We discuss how hypergames can be used to model the interaction between adversaries and system defenders. We discuss a specific example of modeling a system where an insider adversary wishes to steal some confidential data from an enterprise and a security administrator is protecting the system. We show the advantages of incorporating deception as a defense mechanism.

1. INTRODUCTION

Information security is a balancing act of allocating fixed resources to defend against threats. As an example, security administrators are required to ensure the security of a set of workstations, networks, and data within a company with a fixed dollar budget and fixed number of staff members. At a high level, priorities must be placed on certain technologies, policies, and general security practices to minimize breach of integrity, confidentiality, or availability of critical assets. Game theory is a technique that can optimize this balance. However, when information about adversaries, resources, and attack strategies are not known, it is challenging to determine an optimal defense strategy.

In classic game theory models, each player in the game has a set of actions and a list of preferred outcomes. Hypergames extend the classic game theory model by incorporating the *perception* of each player in the game analysis. In a hypergame, each player is operating within their own *perceived* game based on their present understanding of other players' actions and preferences. Modeling the perception of players enables hypergames to express conflicts where players attempt to deceive other players and influence their perception, and thus subsequent actions. Further, hierarchical hypergames model the perception of players that are aware of deceptive strategies of other players. For example, Takahashi *et al.*'s work in [19] used hierarchical hypergames to analyze the Allied invasion of Normandy where the Allies relied on the use of deception despite the fact that Germany perceived the use of deception by the allies.

The use of deception for cyber defense can be employed to supplement existing defense infrastructures. Deception can be used to *deny* attackers access to valuable resources, *misdirect* them away from critical assets, or *confuse* them by presenting plausible yet deceiving information, protocols, or applications [2]. Honeypots are used to lure attackers into interacting with a system configured to gather information or detect the presence of an attacker. Some systems use decoys to protect stored authentication credentials [11], [1]. On the other hand, deception has seen widespread use in malicious applications. Trojan horses, polymorphic malware [17], and rootkits [13, 22] (just to name a few) utilize deception to infiltrate computing systems or to remain hidden. We posit that hypergames form a logical game theoretic model to analyze when and how to use deception in protecting computing systems.

The use of deception-based defenses could be modeled as a hypergame as defenders attempt to deceive an attacker regarding the state or functionality of the targeted system. While prior work has explored the general use of hypergames to model information security defenses where information asymmetry exists between attackers and defenders [9, 10], little work has focused on how to model cyber conflicts that incorporate active deception with hypergames. Such analysis shows the advantages of deploying deceptive defenses to enhance the security of computer systems. In addition, hypergames provide insights on the level of effort needed to defend a system successfully through the use of deception.

Our work described here sheds some light on how an attacker's perception of security defenses can be modeled and

how they influence the attack strategy[1]. We then take the next logical step to discuss how the adversary's perception is shaped, through the strategic use of deception, to the benefit of the system. We use hypergames as the foundational principle for our analysis and apply results from that realm. The work presented is designed as a primer to model cyber defense systems that incorporate deception. We describe deceptive "kernels" specifically in the domain of cyber security to help the reader model deceptive defense strategies. We apply these deceptive "kernels" to cyber defense scenarios where an insider within an enterprise environment is attempting to exfiltrate critical assets. Three deceptive strategies are explored from the defender's perspective to show how hypergames can be used to model such situations and provide insights on defense strategies.

2. HYPERGAMES

In this section, a definition of Game and Hypergame is provided. For simplicity, we first describe the case of a two-player game and a two-player hypergame. We conclude this section with a formal definition of hypergames.

As described in [4], Hypergames are an extension of Game Theory. Game theory analytics assumes that each player in the game has a *common perception* "of the game being played." Hypergames extends this notion by allowing each player to play a game that reflects their perception of the world. Thus, hypergames consist of a set of perceived games that reflect each player's belief of what is happening [4]. Hypergames model conflicts where complete information is not available to a subset of players at a particular point in time [7]. For example, military conflicts such as the D-Day invasion where the allies employed the use of deception to trick Germany into believing that allied invasion would occur in Calais rather than Normandy [19]. Further, hypergames can be used to model conflicts that employ the use of Deception and Mimicry [14].

2.1 Two-Player Game Definition

A two-player game consists of a set of players $\mathbf{Players} = \{A, B\}$ and a non-empty finite set of actions for each player. A player represents an entity, individual, or party that is motivated to maximize some preferred outcome in the game.

Let $\mathbb{A}_A$ represent the set of *actions* that *player-A* can take and let $\mathbb{A}_B$ represent the set of *actions* that *player-B*. Actions are moves that players take to achieve their goal.

$$\mathbb{A}_A = \{a_1, a_2, \cdots, a_n\}$$

$$\mathbb{A}_B = \{b_1, b_2, \cdots, b_m\}$$

Note that the number of actions each player may or may not be the same. An outcome of a game consists of an action selected by *player-A* and an action selected by *player-B*. Thus, the set of possible *outcomes* are

$$\mathbb{O} = \mathbb{A}_A \times \mathbb{A}_B = \{(a_1, b_1), (a_1, b_2), \cdots (a_n, b_m)\}$$

Each player has an ordered list of preferred outcomes called a *preference list*. Let the preference list for *player-A* and *player-B* be:

$$\mathbf{Pref}_A = \langle o_{A1}, o_{A2}, \cdots o_{An\cdot m}\rangle,$$

$$\mathbf{Pref}_B = \langle o_{B1}, o_{B2}, \cdots o_{Bn\cdot m}\rangle,$$

where each element in $\mathbf{Pref}_A$ (or $\mathbf{Pref}_B$) is also in $\mathbb{O}$. Further the elements within the preference vector are ordered from most preferred to least preferred: $\forall o_i, o_{i+1} \in \mathbf{Pref}$, o_i is more preferred than o_{i+1}.
In summary, a two-player game G consists of the following:

Players: Player A and player B.

Actions: $\mathbb{A}_A$ and $\mathbb{A}_B$.

Preferences: $\mathbf{Pref}_A$ and $\mathbf{Pref}_B$

We use the following notation to represent a game:

$$G_{A,B} = (\underbrace{[A,B]}_{\text{Players}}, \underbrace{[\mathbb{A}_A, \mathbb{A}_B]}_{\substack{\text{A and B's}\\ \text{action set}}}, \underbrace{[\mathbf{Pref}_A, \mathbf{Pref}_B]}_{\substack{\text{A and B's}\\ \text{preference list}}})$$

2.2 Two-Player Hypergame Definition

A two-player hypergame consists of two games, one for each player, based on their perception of the conflict at hand.

$$H(\underbrace{A,B}_{\text{Players}}) = \left\{ \underbrace{p(A, G_{A,B})}_{\substack{\text{Game perceived by}\\ \text{Player A}}} \quad , \quad \underbrace{p(B, G_{A,B})}_{\substack{\text{Game perceived by}\\ \text{Player B}}} \right\}$$

In the definition above, the function p denotes the perception from an individual player. For instance $p(A, G_{A,B})$ is a game as perceived by player A. Likewise, $p(A, \mathbb{A}_B)$ denotes the set of actions for player B as perceived by player A. Further, the first parameter in the function p may contain multiple players. For example $p(AB, G_{A,B})$ is player A's perception of B's perceived game [2]. Each player in the hypergame has a perception of the other player's actions and preferred outcomes. For example, player A's perceived actions and perceived preferences of player B may not be the true actions and preferences of player B. More formally,

$$\underbrace{p(A, \mathbb{A}_B)}_{\substack{\text{Player A's perceived}\\ \text{actions for B in A's}\\ \text{perceived game}}} \stackrel{?}{=} \underbrace{p(B, \mathbb{A}_B)}_{\substack{\text{Player B's}\\ \text{actions in B's}\\ \text{perceived game}}} ,$$

$$\underbrace{p(A, \mathbf{Pref}_B)}_{\substack{\text{Player A's perceived}\\ \text{preference list for B}\\ \text{in A's perceived game}}} \stackrel{?}{=} \underbrace{p(B, \mathbf{Pref}_B)}_{\substack{\text{Player B's}\\ \text{preference list in}\\ \text{B's perceived game}}} .$$

Note that player A may have an inaccurate perception of player B, or vice versa. A player's misperception may be due to a lack of available information about other players in the game or it may be strategically placed *misinformation* designed to deceive a player.

[1]For simplicity of exposition and then tractability of the analysis, we assume a single adversary throughout the paper. If there are indeed multiple adversaries in the system, then our model can be taken to be the pessimistic case where all adversaries collude perfectly.

[2]Players perceiving other player's perception are modeled as multi-level hypergames.

3. KERNELS TO MODEL PLAYER MISPERCEPTION IN CYBER CONFLICTS

A player's perception of a conflict does not necessarily reflect the truth due to a player's bias, misinformation, misinterpretation, and ignorance. We model this notion with the perception function p, which can be interpreted as a function that maps the truth of a conflict to how a given player views the conflict. Wang, Hipel, and Fraser in [21] describe that a player's perception maps the sets **Players**, $\mathbb{A}$, or **Pref**, to some alternative sets that reflect a given player's understanding of the conflict. Perception mapping is roughly broken into three misperception kernels as inspired by the mappings discussed in [21]: misinterpretation, over-perception, and under-perception. The null case, where a player accurately perceives components of a conflict, is called *preservation*. For each kernel, the perceptual mapping may occur on actions, preference lists, or players in the game. In this case, the player's perception of the conflict matches the reality of the conflict. On the other hand, *misinterpretation*, *over-perception*, *under-perception* are specific types of misperceptions. Confusion may also be considered as a type of misperception. However, confusion can be accurately modeled as a player who under-perceives components of a conflict. Further, player uncertainty can be constructed as a hierarchical hypergame by considering alternative perceptions of the conflict.

The misperception kernels are the building blocks for modeling deception in a cyber conflict. For each kernel, we describe how the perceptual mapping may be used to deceive a player in a cyber security conflict. Note that each kernel is *not mutually exclusive*. For example, a given player A may under-perceive certain actions for a player B while also under-perceiving the existence of player C.

It is also important to note the general challenges of modeling cyber conflicts. Prior work highlights examples of hyphergames applied to *physical* conflicts [19, 5, 7]. However, cyber conflicts introduce complexities that do not have physical analogies. For example, an attacker may utilize some zero-day exploit and remain undetected and deeply penetrated within an enterprise environment. Security software produces false-positive alerts that potentially cut into administrator resources which further complicates the detectability of an attacker. Attack attribution in cyber conflicts are difficult and are usually conducted long after an incident. The notion of a user also introduces complexities. A user's workstation could be used as a proxy for launching an attack. Poor user choices may unintentionally subvert security policies, software, and etc. Attackers and defenders can utilize deception even further to complicate cyber conflicts. While not exhaustive, we provide several examples that touch on these issues to demonstrate the applicability of hypergames on cyber conflicts that utilize some deceptive components.

3.1 Misinterpretation

Misinterpretation is the situation when a player does not correctly interpret an action, player, or preference vector. Formally, a player may misinterpret the true nature of players in a conflict. For a set of players in A's game, $\mathbf{Players}_A = \{A, B\}$ where $\mathbf{Players}_A \in p(A, G_{A,B})$, a misinterpretation of the players in the game from B's perspective can be expressed as

$$p(B, \mathbf{Players}_A) = \{A', B\}, \text{where } A' \notin \mathbf{Players}_A, A' \neq A$$

For clarity, B's perception of player A is A' where A' does not accurately encompass the true nature of A. The misinterpretation of players can accurately model social engineering attacks. For example, from player B's perspective, she is helping remote A' (a colleague) by providing critical source code via FTP but in reality, A' is A — an attacker conducting corporate espionage for a competitor.

The above example models a player's perception of a conflict that does not match the reality of the conflict. The specific details of the misinterpretation are described by the differences of perceived player actions or preferences vs the actual player actions or preferences. For an action set $\mathbb{A}_A = \{a_1, a_2, \cdots, a_n\}$ for player A, a misinterpretation for a single action from B's perspective can be expressed

$$p(B, \mathbb{A}_A) = \{a_1', a_2, \cdots, a_n\}, \text{where } a_1' \notin \mathbb{A}_A, a_1' \neq a_1$$

Communication through a hidden channel can be modeled as misinterpretation where the observer of the cover media is not aware of the covert channel. Say Player A has an action a_1 that Player B observes as a_1'. Player B interprets a_1' as "Player B posts vacation photos on a personal website." However, Player A's true action a_1 is "Post vacation photos that contain hidden trade secrets that player C will decode and sell to a competitor."

Finally, a player may misinterpret another player's preference list. A preference list for player A $\mathbf{Pref}_A = \{o_1, o_2, \cdots o_n\}$ may be misinterpreted as

$$p(B, \mathbf{Pref}_A) = \{o_i, o_j, \cdots, o_k\},$$

$$\text{where } i, j, \cdots, k \in 1 \cdots n \text{ and } i \neq j \neq, \cdots, \neq k$$

Note that B's perception of A's preference list is permutation of A's true preference list. From a security perspective, misinterpretation could be port scanning for program interoperability (e.g. service discovery) or port scanning for vulnerability assessments. The action itself remains unchanged, but the hypergame allows the exploration of interpretation possibilities.

3.2 Over-Perception

As the name implies, a player may over-perceive a given conflict by believing that there are additional players or player actions which do not exist. More formally, for a given action set, $\mathbb{A}_A = \{a_1, a_2, \cdots, a_n\}$ player B may over-perceive the action of player A:

$$p(B, \mathbb{A}_A) = \{a_1, a_2, \cdots, a_n, a_{n+1}\},$$

$$\text{where some } a_i \notin \mathbb{A}_A$$

Broadly speaking, security software that produces false alarms could lead to players over-perceiving a situation. For example, say anti-malware software flags an application as malicious. A system administrator (Player A) can interpret the situation as a coworker (Player B) attempting to compromise systems (action $a_i \in p(B, \mathbb{A}_A)$) but in reality, the software is benign ($a_i \notin \mathbb{A}_A$).

A player may over-perceive a conflict to include players that are not present within a conflict. A player B may perceive some player C that may not exist in the conflict. Let

$\mathbf{Players}_A = \{A, B\}$ be the player in A's game. Player B' over-perceives the players by

$$p(B, \mathbf{Players}_A) = \{A, B, C\}, \text{where } C \notin \mathbf{Players}_A$$

Building on the previous example, the system administrator believes that someone within the company is installing malicious software. However, this person does not exist.

For example, Player A believes that Player B's workstation is a zombie controlled by player C. In reality, some benign software or behavior triggered a false alarm, so player C does not exist.

If a player over-perceives a conflict, outcomes will be considered that do not reflect the actuality of the situation. These additional outcomes will impact the ordering of the perceived preference vectors. A preference list for player A, $\mathbf{Pref}_A = \{o_1, o_2, \cdots o_n\}$, may be over-perceived as

$$p(B, \mathbf{Pref}_A) = \{o_1, o_2, \cdots, o_n, o_{n+1}\}, \text{where } o_{n+1} \notin \mathbf{Pref}_A$$

From the previous example, Player A may perceive outcomes such as conducting incident response and forensic examination on B's workstation. Conducting actions that lead to those outcomes would waste resources and time because in actually, Player C does not exist and Player B's workstation is not compromised.

3.3 Under-Perception

A player in a conflict may also under-perceive the presence of other players or the types of actions that a player may execute. Given a set of actions $\mathbb{A}_A = \{a_1, a_2, \cdots, a_n\}$ a player B may under-perceive a conflict as

$$p(B, \mathbb{A}_A) = \{a_1, a_2, \cdots, a_{n-1}\}, \text{ such that,}$$

$$a_i \in \mathbb{A}_A \text{ but } a_i \notin p(B, \mathbb{A}_A).$$

Examples include situations where a player under-estimates the resources that another player has. For example, a user (Player A) may under-perceive the actions of an application (Player B) that she is installing on her workstation. Player A believes that the application's actions deal with photo editing but she under-perceives a hidden action that periodically uploads her keystrokes to a remote server.

Players in a conflict may also be under-perceived. For a set of players in A's game, $\mathbf{Players}_A = \{A, B, C\}$, player B my under under-perceive the conflict as

$$p(B, \mathbf{Players}_A) = \{A, B\}.$$

Note that player C is not perceived by player B. For example, a system administrator (Player B) is aware of an external attacker (Player A) but is unaware that the external attacker is coordinating an attack with an insider (Player C).

In cases where players or actions are under-perceived, the outcome preference list is reduced in size and may be of a different order. A preference list $\mathbf{Pref}_A = \{o_1, o_2, \cdots o_n\}$ for player A, may be under perceived by player B as

$$p(B, \mathbf{Pref}_A) = \{o_1, o_2, \cdots, o_{n-k}\}, \text{ such that } k < n, \text{ and}$$

$$o_{k+1} \cdots o_n \in \mathbf{Pref}_A \text{ but } o_{k+1} \cdots o_n \notin p(B, \mathbf{Pref}_A)$$

Since player B under-perceives the presence of other players or certain actions, the perceived preference vectors of players in the conflict may not contain certain true outcomes. Likewise, the ordering of the preference vectors may also change. For example, a malware analyst (Player A) may run some potentially malicious application (Player B) within a sandbox to understand the specific actions of the malware. However, the malware may withhold executing certain actions (e.g. zero-day exploit to move laterally undetected) because it is aware that it is running within a sandbox [3]. The outcomes from Player A's perspective underestimates the severity of the malware.

3.4 Preservation

Preservation, as the name implies, is where a player can preserve a conflict exactly as another player. Given a perceived game from player A's perception, player B perceives the same game,

$$\begin{aligned} p(B, \mathbf{Players}_A) &= \mathbf{Players}_A, \\ p(B, \mathbb{A}_A) &= \mathbb{A}_A, \\ p(B, \mathbf{Pref}_A) &= \mathbf{Pref}_A. \end{aligned}$$

Situations where a player can perceive a conflict exactly as another player may be due to some hidden entity that leaks information to another player. Conflicts include situations where moles within a secret government agency who leak information to adversaries. Note that preservation of perception is not both ways. For example, player A may know the actions and preferences of player B but player B may not know the actions or preferences of player A.

3.5 Stability Analysis

An important aspect of Game Theory analysis is the Nash equilibrium. This notion is extended in hypergames such that an equilibrium is an outcome that is stable for all players in the hypergame. For a hypergame, an equilibrium is determined in a two-part analysis. The first step is to identify each player's *perceived optimal action* which is derived from the Nash equilibrium for each player's perceptual game. Note that a player's *perceived optimal action* is calculated based on a player's accurate perception or misperception of the conflict. Next, an overall stability analysis is conducted based on the perceived rational action for each player in the conflict. The output of the overall stability analysis is a set of *Rational Outcomes* for the conflict. Based on the formal definition described in [21], we define the equilibrium of two-player hypergames as

Given a two-player hypergame,

$$H(A, B) = \{p(A, G_{A,B}),\ p(B, G_{A,B})\},$$

determine the Nash equilibrium for each individual game $p(A, G_{A,B})$ and $p(B, G_{A,B})$. Let the equilibrium outcome of player A's perceived game be

$$p(A, E) = p(A, \{(a_{e1}, b_{e1}), (a_{e2}, b_{e2}), \cdots, (a_{ek}, b_{ek})\}),$$

where $a_{ei} \in p(A, \mathbb{A}_A)$ and $b_{ei} \in p(A, \mathbb{A}_B)$ for all $i = 1 \cdots k$. Note that $p(A, \{(a_{ei}, b_{ei}) \mid \forall\ i\})$ are the outcomes that represent player A's perceived Nash equilibrium. Likewise, let the equilibrium for player B's perceived game be

$$p(B, E) = p(B, \{(a_{e1}, b_{e1}), (a_{e2}, b_{e2}), \cdots, (a_{el}, b_{el})\}),$$

[3]Malware that exhibits sandbox detection is report in [17]

where $a_{ei} \in p(B, \mathbb{A}_A)$ and $b_{ei} \in p(B, \mathbb{A}_B)$ for all $i = 1 \cdots l$. Note that $p(A, \{(a_{ei}, b_{ei}) \mid \forall i\})$ are the outcomes that represent player B's perceived Nash equilibrium. Let

$$p(A, S) = p(A, \{a_{e1}, a_{e2}, \cdots a_{em}\}),$$

where $a_{ei} \in p(A, \mathbb{A}_A)$ and $a_{ei} \in p(A, o)$ for all $i \in 1 \cdots m$, and $p(A, o) \in p(A, E)$. Note that $p(A, S)$ represents the set of all rational actions for player A with respect to player A's perception of the conflict. Likewise let

$$p(B, S) = p(A, \{b_{e1}, b_{e2}, \cdots b_{en}\}),$$

where $b_{ei} \in p(B, \mathbb{A}_B)$ and $b_{ei} \in p(B, o)$ for all $i \in 1 \cdots n$, and $p(B, o) \in p(B, E)$.

Note that the $p(A, S)$ and $p(B, S)$ represents the set of *perceived optimal actions* for each individual player's *perceived game*.

The second step in hypergame stability analysis considers each player's *perceived optimal actions* to determine the *rational outcome*. The set of rational outcomes is the Cartesian product of the player's *perceived optimal actions*. The set of *rational outcomes* for a hypergame may not necessarily be stable for all players if subsequent rounds of the conflict are played [21].

For example, say that the rational outcome of a military conflict is for player A to betray player B via a surprise attack. After the actions are executed, if player B is aware of the betrayal, B's preference vectors will shift from trusting A to counter-attacking A. A stability analysis at this point of the conflict will be different compared to the stability analysis before the betrayal. Wang et al. in [21] describe *unstable equilibrium* as rational outcomes that change over the course of a conflict. An unstable equilibrium exists if there are players that can improve their outcome of a conflict by unilaterally changing actions, given that the other player(s) does not alter their action. Such instances are called *hypergame-destroying equilibrium*. If there exists an outcome that all players perceived as a Nash equilibrium, then the outcome is a *hypergame-preserving equilibrium*. Alternatively, it is possible for a conflict to consist of a single round. That is, each player selects a perceived optimal action and the *rational outcome* is the "equilibrium" of the conflict. Wang et al. dub this a *snap-shot equilibrium* since each player selects an action based on their individual perception and the conflict immediately ends. In other words, the players do not have an opportunity to observe the actions of the other players to change their strategies.

3.6 Deception, Information Security, and Stability Analysis

When analyzing conflicts, it is crucial to consider the type of equilibrium that is appropriate. A *hypergame-destroying equilibrium* may require multiple rounds of action, observation, and strategy adjustments until a *hypergame-preserving equilibrium* is reached. On the other hand, some scenarios may be modeled as a snap-shot decision conflict [21] where each player makes a single rational decision, which determines the conflict equilibrium.

Information security conflicts do not stabilize to a steady state. Adversaries find new exploits to compromise systems as new patches, policies, and defensive security software is deployed. Modeling a cyber conflict as a hypergame and conducting a stability analysis may not produce a final resolution. We emphasize that the process of stability analysis produce meaningful results even if a global conflict resolution does not stabilize. In the following sections, we model cyber conflicts that utilize deception and conduct a stability analysis to find the *rational outcomes* of the conflict. The *rational outcomes* for a given conflict will indicate how an adversary may react to some deceptive component or show the inefficiencies in deploying a deceptive system.

We define deceptive defense systems as a collection of software, personnel, data, or policy that is used to deceive an attacker which negatively impacts their ability to achieve their mischievous goals. Examples include honey(pots[18], files[23], words[11]), ErsatzPasswords [1], Tripwire [12], and etc.

We focus on conflicts where deceptive defense systems are deployed in conjunction with conventional information security systems. We limit the scope of the conflict analysis to determine the effectiveness of the deceptive components. This is determined by outcomes where the attacker falls for the deception thus exposing the presence of an attacker to the system administrators. Three plausible situations are explored in this paper. First is the case where the presence of deceptive components is completely unknown to adversaries. Second, we model the case that an adversary is aware that some deception is deployed as a defense mechanism, but the adversary does not know the details of the deception. The third is the case where the system defenders deploy evidence of a deceptive component to convince the adversary that some deception is in place. However, the system defender does not deploy the deceptive component. The stability analysis produces *rational outcomes*, but they are clearly not a global resolution of the conflict. For example, if an adversary falls for a deceptive component, future adversaries will be aware of the type of deception and adjust their strategies accordingly.

4. MODELING AN INSIDER THREAT AS A GAME

In this section, we show how a game can be used to model and analyze the interaction between security administrators and attackers. We explore the scenario where an insider within a company wishes to expose some confidential data unbeknown to the company's security team. We start by describing each player in the game then briefly discuss their goals and motivations. Next, we describe the actions that each player can take within our model. In the following section, we build on the game and introduce a hypergame to model deceptive defensive strategies.

4.1 Players

The Insider Threat game consists of two players who work in an enterprise environment: An *Insider* and a *Security Administrator*. The *Insider* is an employee who is determined to exfiltrate some confidential data for personal gain at the detriment to the company. As discussed in the 2014 DBIR [20], an insider may be bribed by criminals to steal data for fraudulent schemes or by corporate competitors to gain a competitive business advantage. Further, insider bribery from criminals or competitors comprises of over half the instances reported in the 2014 DBIR [20]. Despite traditional measures to mitigate these kinds of attacks, via strict laws, background checks, and audits, the threats still exist.

In regards to actions within the Insider Threat Game, *In-*

sider can behave in an inconspicuous manner, which consists of attending to their assigned tasks as instructed by their superiors. They may also choose to *Probe* the network for confidential information. We assume that behaving inconspicuous does not raise any suspicion, and thus the insider remains hidden. However, this action does not take them any closer to achieving their ultimate goal of obtaining confidential data. An *insider* may either *probe* for information or behave in an *inconspicuous* manner. In other words, the actions are mutually exclusive. We assume that probing the network may potentially leave digital evidence of malice on the insider's workstation or systems he/she is probing.

On the other hand, *Security Administrators* are responsible for preventing and mitigating any malice within the corporate computing infrastructure. If the security administrator suspects malice, then they focus their attention on the potential breach at hand – we refer to this action as *Incident Response*. An *Incident Response* requires a security administrator to investigate the incident closely. Performing *Incident Response* is time-consuming, and security administrators prefer to investigate a potential insider's machine only if there is evidence of malice. If the security administrator is not responding to an incident, the security administrator focuses on system maintenance.

4.2 Actions

Table 1 shows the possible actions that can be taken by a security administrator. As discussed above, a player can only choose a single action at a time. Each action is modeled as a binary decision. An insider may choose to *probe* the system or behave normally. The security administrator may analyze logs and IDSs and respond by investigating a potential insider's machine.

$\mathbb{A}_{Insider}$	$\mathbb{A}_{SecurityAdministrator(S.Admin.)}$
$\{Probe\}$	$\{Incident\ Response\ (Inc.\ Rsp.)\}$

Table 1: Actions for **Insider** and **Security Administrator** for the Insider Threat Game.

4.3 Outcomes

Table 2 shows all the possible outcomes in the game discussed above. An administrator *ideal* or an insider *no progress* outcome is the case where the potential insider does not attempt to disclose confidential data. An administrator *breach* or insider *success* outcome is where an insider exposes confidential data and the security administrator is unaware. The *false accusation* outcome occurs when an administrator suspects that an insider has disclosed some confidential data but the suspected insider is only performing their normal tasks. An administrator *data disclosure* or insider *caught afterward* outcome occurs when the security administrator launches an incident response act when the insider disclosed some confidential data.

4.4 Preference Vectors

The final component in the insider threat game is the preference vector of each player. Table 3 shows the ordered preference vector of the security administrator. The security administrator prefers that fellow employees behave normally, which allows the security administrator to maintain conventional defense systems to prevent a potential insider from leaking confidential data. From the security administrator's perspective, this is the *ideal* outcome of the conflict. If all the conventional security systems fail to detect the insider, then the security administrator must conduct an incident response to assess the damages and conduct attribution analysis. This is the *data disclosure* outcome from the security administrator's perspective since confidential data is known to be exposed. The next outcome in the system administrator's preference list is falsely accusing the user of data exfiltration. The *false accusation* outcome is undesirable because it would decrease the work morale of the company, which has been noted as a challenge in mitigating insider threats [16]. Finally, the least preferred outcome is a *breach* of data where the insider successfully leaks information to a third party without the administrator's knowledge.

The preference vector of the insider is shown in Figure 4. The insider's most preferred outcome, *success*, is to successfully exfiltrate confidential data without the security administrator noticing. The second most preferred outcome, *Caught Afterward*, is to probe the network for confidential information and have the administrator respond to an incident. There are two reasons that we decided to place *Caught Afterward* as the second most preferred outcome for the insider. First, it explores an aggressive insider threat who is willing to risk getting caught rather than leaving empty handed. There is also the possibility that the insider may steal the data and immediately resign from the company. The rationale is discussed in the 2014 DBIR [20]:

> The CERT Insider Threat Center (another partner of ours) focuses research on insider breaches, and it determined that in more than 70% of the IP theft cases, insiders stole the information within 30 days of announcing their resignation.

The insider next prefers the *No Progress* outcome where the insider engages in tasks assigned by company management. As the name indicates, the *No Progress* action does not achieve the goal of exfiltrating the data but is also not suspected to be an insider. The insider's next most preferred outcome is a *false accusation*. Although such an incident may put the insider in an uncomfortable position, the insider does not run the risk of discovery.

4.5 Equilibrium Analysis

Table 5 shows the results from running the equilibrium analysis with HYPANT [4] - a hypergame analysis tool [6]. HYPANT produces a stability table which describes various strategies for a player compared to other players as well as outcomes that are suitable for all players [6]. The two rightmost columns in Table 5 (a) and (b) describe the stability analysis from each player's point of view. The Unilateral Improvements (UIs) column indicates that for a given outcome, the player could change their strategy to one more preferred in their preference vector assuming that the other player does not change strategies [6]. In *stability* column indicates the type of stability from a player's point of view. A *Rational* (r) outcome indicates that the player does not have a unilateral improvement available. A *sequentially sanctioned* (s) outcome indicates that there are alternative actions available to the given player that are more preferred but allows the opponent to change their strategy to a less preferred

[4] `www.csse.monash.edu.au/hons/se-projects/2003/Brumley/`

Outcomes		S. Admin.	Insider	
Sec. Admin.	**Insider**	*Inc. Rsp.*	*Probe*	**Description**
Ideal	No progress	0	0	Sec. Admin. maintenance. Inconspicuous Insider.
Breach	Success	0	1	Sec. Admin. maintenance. Insider exfiltrates.
False Accusation	False Accusation	1	0	Sec. Admin. falsely accuses insider.
Data Disclosure	Caught Afterwards	1	1	Sec. Admin. responds to breach.

Table 2: All possible outcomes with the actions described in Table 1. Each outcome has an designated name in the first and second column from the perspective of the administrator and the insider respectively.

		S. Admin.	Insider
Rank	**Outcome**	*Inc. Rsp.*	*Probe*
1	Ideal	0	0
2	False Accusation	1	0
3	Data Disclosure	1	1
4	Breach	0	1

Table 3: The security administrator's preference vector. The table is sorted from most preferred outcome (first row) to least preferred outcome (last row).

		S. Admin.	Insider
Rank	**Outcome**	*Inc. Rsp.*	*Probe*
1	Success	0	1
2	Caught Afterwards	1	1
3	No Progress	0	0
4	False Accusation	1	0

Table 4: The insider's preference vector. The table is sorted from most preferred outcome (first row) to least preferred outcome (last row).

outcome [6]. *Unstable* (u) indicates that a player has a UI where the opposing player cannot select an action that produces an outcome that is less preferred for the given player [7, 7].

From the stability and UI columns, we can easily determine the equilibrium. If we eliminate all the outcomes that are unstable for at least one player, we are only left with *Data Disclosure/Caught Afterward* outcome. Note that the outcome is advantageous for the Insider since they can successfully achieve their goal of exfiltrating confidential information but at the risk of getting caught. The insider may choose to leave the jurisdiction where the crime was committed since the incident is not discovered immediately.

Note that in a game, the perception of each player is not considered. Each player is aware of the other player's actions and preferences. In the following sections, we model the insider threats as a hypergame where the perceptions each players' understanding of the conflict are modeled.

5. MODELING MISPERCEPTION IN INSIDER THREAT CONFLICTS

Building from the players and actions discussed in the previous sections, we introduce a new action that a security administrator may utilize to expose potential insiders. As an additional action, a security administrator may use a *stratagem* to deceive an Insider into revealing themselves.

The Merriam-Webster definition of stratagem is "a trick or plan for deceiving an enemy" [5]. The use of a stratagem in defending computer systems has been around for two decades [3] with notable examples including Tripwire [12], Honeypots [18], Honeyfiles [23], and honeywords [11]. Note that we use the term *stratagem* abstractly to mean any of the previously mentioned deceptive mechanisms. The scenarios we explore below focuses on the use of baiting an insider to reveal themselves. The subtle yet critical details that each deceptive mechanism offers will be explored in future work.

We explore three different insider threat scenarios by modeling and analyzing the conflict as a hypergame. The first hypergame, "Deceiving a naïve insider," describes a conflict where the insider is not aware that the security administrator is using a stratagem. The insider *under-perceives* the arsenal that the security administrator has in its repertoire. Hypergame 1 illustrates the effectiveness of using a stratagem as a defensive strategy, given that the Insider is completely unaware of the stratagem. The second hypergame,"Informed Insider," modifies the first hypergame to reflect a more realistic scenario. The insider perceives that the security administrator uses a stratagem and the security administrator *under-perceives* the insider by believing that insider is unaware of the use of the stratagem. The third hypergame, "Bluffing the insider" explores the conflict where the insider *over-perceives* the security administrator and believes that the use of a stratagem is possible. The security administrator, on the other hand, plants evidence that points to the use of a stratagem but an actual stratagem is not deployed. Further, the security administrator accurately perceives the insider's under-perception.

5.1 Actions

For a stratagem to be effective, careful planning and deployment are necessary. For example, a security administrator may decide to deploy Honeyfiles [23] that trigger an alarm and logs information for attribution upon access.

The location, name, type, size, permission, and context must all be plausible to entice an insider to probe the file without raising suspicion. For the following scenarios, we assume that security administrators are overworked, and thus consider the act of *Incident Response* and *Stratagem* to be mutually exclusive.

Alternatively, a security administrator may plant evidence that indicates the use of a stratagem but without actually deploying a stratagem. Rather than deploying a stratagem, the security administrator engages in a *false stratagem*. For example, *false stratagem* action may involve generating fake invoices that show the hiring of personnel or the purchase

[5] `http://www.merriam-webster.com/dictionary/stratagem`

		Administrator	Insider	Eq. Analysis	
Rank	**Outcome**	*Inc. Rsp.*	*Probe*	Stability	UIs
1	Ideal	0	0	r	-
2	False Accusation	1	0	s	Ideal
3	Data Disclosure	1	1	r	-
4	Breach	0	1	u	Data Disclosure

(a) Security Administrator's Stability Table

		Administrator	Insider	Eq. Analysis	
Rank	**Outcome**	*Inc. Rsp.*	*Probe*	Stability	UIs
1	Success	0	1	r	-
2	Caught Afterwards	1	1	r	-
3	No Progress	0	0	u	Success
4	False Accusation	1	0	u	Caught Afterwards

(b) Insider's Stability Table

	Action	**Outcome**
S. Admin	Incident Response	Data Disclosure
Insider	Probe	Caught Afterwards

(c) Equilibrium for Insider Threat Game

Table 5: Equilibrium Analysis for insider threat game.

of software to deploy a stratagem. Prior work that utilizes *false stratagem* includes "fake Honeypots" [15], where a real user's computer is configured to look like a honeypot to scare away attackers that are trying to penetrate the system. We assume that creating false documents, emails, invoices, or a fake honeypot within a computing infrastructure is more cost effective than crafting a real stratagem. False stratagems are explored in Hypergame 3 to illustrate that altering the perception of an adversary may be a viable defense strategy. Table 6 contains all the actions considered in the three hypergames presented.

When the security administrator uses a *stratagem* the following outcomes of a conflict are possible. The security administrator's *precaution* or the insider's *suspect* outcome occurs when the administrator deploys a *stratagem* but the insider does not probe the network for confidential data. The security administrator's *Deceive* or *Tricked* occurs when the insider falls for the stratagem and the thus a data breach is prevented.

Similarly, there are two possible outcomes if the security administrator uses the *false stratagem* action. If the system administrator deploys false stratagem and the insider does not probe, then the outcome is *Deceptive Precaution*. That is, the insider may believe that the false stratagem is simply a stratagem deployed by the security administrator. Table 7 lists all the possible outcomes for the presented hypergames. Note we explore the use of *false stratagem* only in hypergame 3.

For each player of the hypergame, we consider the following preference vectors. A security administrator may decide not to use a stratagem as illustrated in the game in the previous section. We use this preference vector to model instances where the insider perceives that the security administrator does not use a stratagem. The outcomes are shown in preference list (A1) prioritizes outcomes where no confidential data is compromised.

Alternatively, a security administrator may decide to deploy a stratagem, as indicated in preference list (A2). The most preferred outcome for the security administrator is the *ideal* outcome where no data is exfiltrated and the security

Action	**Description**	**Comment**
(No Action)	Focus on maintaining systems and policies	Action available in all hypergames presented.
Stratagem	Utilize deception as a defense	Action Available in hypergame 1 and 2.
Inc. Rsp.	Respond to a perceived threat	Action available in all hypergames presented.
False Stratagem	Plant evidence of a stratagem	Used only in hypergame 3.

(a) All possible actions for the **Security Administrator**. We assume that Security Administrator can deploy a stratagem or conduct an incident response but not both at the same time. *no action* is also mutually exclusive with other actions.

Insider	**Description**	**Comment**
(No Action)	Insider behaves inconspicuously.	Action available in all hypergames presented.
Probe	Probe for confidential data to exfiltrate.	Action available in all hypergames presented.

(b) All possible actions for the **Insider**. As the name suggests, *no action* is mutually exclusive with the *probe* action

Table 6: Possible actions for each player for all hypergames presented. The **Comment** column indicates which actions are available for each hypergame.

Perceived Outcome Name		S. Admin		Insider		
S. Admin.	**Insider**	*Stratagem*	*Inc. Rsp.*	*Probe*	**Description**	
Ideal	No progress	0	0	0	Admin maintains and insider behaves inconspicuously	
Breach	Success	0	0	1	Admin maintains and insider steals confidential info	Possible outcomes for hypergames 1, 2, and 3
False Accusation	False Accusation	0	1	0	Incident response admin with inconspicuous insider	
Data Disclosure	Caught Afterwards	0	1	1	Admin Incident response after insider steals info	
Precaution	Suspect	1	0	0	Admin sets bait while insider behaves inconspicuously	Possible Outcomes for hypergames 1, 2
Deceive	Tricked	1	0	1	Admin catches the insider stealing info with bait	

Perceived Outcome Name		S. Admin		Insider		
S. Admin.	**Insider**	*F. Stratagem*	*Inc. Rsp.*	*Probe*	**Description**	
False Precaution	Suspect	1	0	0	Insider behaves inconspicuously does not act on false stratagem	Possible Outcomes for only hypergame 3
Failed Deception	Success	1	0	1	Insider does not fall for the false stratagem	

Table 7: All possible outcomes with the actions described in Table 6. Each outcome has an designated name in the first and second column from the perspective of the administrator and the insider respectively.

		S. Admin.	Insider
	Outcome Name	*Inc. Rsp.*	*Probe*
(Most Preferred)	Ideal	0	0
	False Accusation	1	0
	Data Disclosure	1	1
(Least Preferred)	Breach	0	1

(A1) Preference vector where a Security Administrator does not deploy a *stratagem*

		S. Admin		Insider
	Outcome Name	*Stratagem*	*Inc. Rsp.*	*Probe*
(Most Preferred)	Ideal	0	0	0
	Deceive	1	0	1
	Precaution	1	0	0
	False Accusation	0	1	0
	Data Disclosure	0	1	1
(Least Preferred)	Breach	0	0	1

(A2) A preference vector for a Security Administrator that uses a *stratagem* to deceive an insider

		S. Admin.		Insider
	Outcome Name	*False Stratagem*	*Inc. Rsp.*	*Probe*
(Most Preferred)	Deceptive Precaution	1	0	0
	Ideal	0	0	0
	False Accusation	0	1	0
	Data Disclosure	0	1	1
	Breach	0	0	1
(Least Preferred)	Failed Deception	1	0	1

(A3) A preference vector for a Security Administrator who plants *evidence of stratagem* rather deploying the stratagem

Table 8: Preference vectors for Security Administrator

administrator focuses on system maintenance. The following two preferred outcomes in the administrator's preference vector include situations that deploy the use of a stratagem. The security administrator prefers to deploy a stratagem if there is an active insider who is probing the network for confidential data. We call this the *deceive* outcome since we assume that the stratagem is effective in deceiving a probing insider. Next, the security administrator prefers the *precaution* outcome where a stratagem is planted within the system, but the insider decides not to seek out confidential data. The remaining outcomes deal with adverse consequences for the security administrator. The next outcome in the administrator's preference list is *false accusation* where the security administrator suspects that an insider is present in the system and decides to conduct a deep investigation. However, there is no compelling evidence that the accused has done anything malicious. The outcome wastes the time of both the security administrator and the falsely accused "insider." Further, the *false accusation* outcome reflects poorly on the security administrator's ability to analyze a situation accurately and wastes company resources. The final two least preferred outcomes for the security administrator deal with unauthorized data disclosure. The *Data Disclosure* outcome detects a breach after the data has already been disclosed to an unauthorized third party. The insider at this point may no longer be employed by the company. At the bottom of the list, the *Breach* outcome is least preferred because the insider successfully exfiltrates confidential data without the security administrator's knowledge.

Finally, preference list (A3) explores the use of *false stratagem*. At the top of the preference list is the administrator's *deceptive precaution* outcome. The rationale is twofold. First, from the security administrator's perspective, an insider is present in the corporate infrastructure. Secondly, the cost of crafting *false stratagem* is relatively small. Next, the *Failed Deception* outcome is placed last in the security administrator's preference list. The false stratagem failed to deceive the insider and security administrator wasted time

	S. Admin.	Insider
Outcome Name	*Inc. Rsp.*	*Probe*
(Most Preferred) Success	0	1
Caught Afterwards	1	1
False Accusation	1	0
(Least Preferred) No Progress	0	0

(I1) A preference vector for an Insider that does not consider the security administrator's usage of a *stratagem*.

	S. Admin.		Insider
Outcome Name	*Stratagem*	*Inc. Rsp.*	*Probe*
(Most Preferred) Success	0	0	1
False Accusation	0	1	0
Suspect	1	0	0
No progress	0	0	0
Caught Afterwards	0	1	1
(Least Preferred) Tricked	1	0	1

(I2) A preference vector for an Insider that is aware that the security administrator deploys a *stratagem*.

Table 9: Possible preference vectors for insiders.

and effort to deploy the false stratagem. The insider also stole confidential data without the security administrator's knowledge.

There are two preference vectors that we consider for the insider. Preference vector (I1) models an insider that underperceives the security administrator's use of a stratagem. The two most preferred preferences include outcomes where the insider successfully probes and exfiltrates confidential data. The most preferred outcome is *Success* where the security administrator is unaware that confidential information has been leaked. The *Caught Afterward* outcome is placed in the second preferred position. We assume that the insider will steal confidential data and then immediately quit the company. The latency for detecting the data disclosure gives the insider enough time to leave the jurisdiction where they committed the crime. There exist other rational preference vectors that an insider may consider such as placing the *Caught Afterward* lower in the preference list.

The second preference vector is shown in (I2), which models an insider that is aware of the use of a stratagem. Since the insider is aware of the use of a stratagem, they are cautious and prefer not to get caught, so *Data Disclosure* and *Tricked* are the least preferred outcomes. An alternative preference vector could rationally place the *data disclosure* outcome higher on the list. However, such an ordering does not impact the final rational outcome of the hypergame. The notable difference in (I2) compared to (I1) is the placement of *Caught Afterward* in the second to least preferred outcome. The rationale is since the insider is aware of the stratagem, they are more cautious in probing the network for confidential data. Also, note that the placement of *False Accusation* and *Suspect* in the second and third most preferred outcomes. The reason is that insider would rather waste the security administrator's resources than not make progress in exfiltrating confidential data.

5.2 Hypergame 1 - Deceiving a Naïve Insider

The motivation for Hypergame 1 is to explore an ideal situation for security administrators who use a stratagem to help defend critical assets. We assume that the insider believes that the security administrator is unwilling to deploy

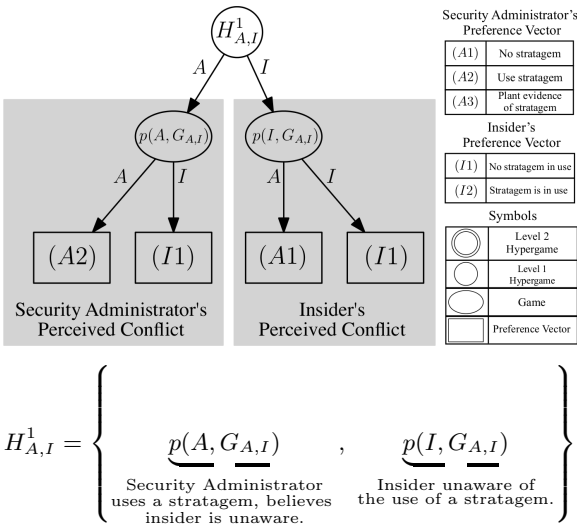


Figure 1: Hypergame 1 models the case where security administrator employees the use of a stratagem without the insider's knowledge.

deceptive mechanisms in favor for more conventional defensive strategies. We also assume that the security administrator can covertly deploy a stratagem without the insider's knowledge.

As discussed in Section 3.6, if a given player *under-perceives* a conflict, then an opposing player may have an opportunity to call on a strategy that is unexpected by the given player. In this hypergame, the security administrator has a "strategic surprise" because the use of a stratagem is not known to the insider.

Recall that a hypergame consists of individual games as perceived by each player. As shown in Figure 1,the hypergame consists of two perceived games - one from the security administrator's perspective $p(A, G_{A,I})$ and the other from the Insider's perspective $p(I, G_{A,I})$. The edges of the graph shown in Figure 1 indicates the dependency of the hypergame. The root of the binary tree is the hypergame, which depends on the individual perceived games by each player. Each perceived game is dependent on the player's preference vector.

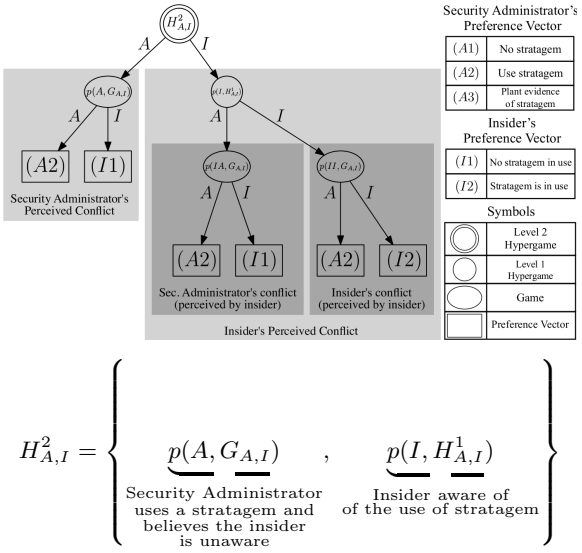
The actions of the security administrator and the Insider are shown in Table 6a, excluding the *false stratagem* action. The security administrator can conduct an *incident response* or utilize a *stratagem* within the system. If neither action is chosen, then the security administrator maintains the software, hardware, and the security policies of the enterprise infrastructure. The insider can choose to *probe* for confidential data or conduct assigned tasks. If an insider chooses the *probe* action and if the security administrator chooses *stratagem*, then the insider is caught.

Consider the perceived conflict from the security administrator's perspective. The administrator's preference vector, $p(A, \mathbf{Pref}_A)$, is illustrated in (A1). The insider's preference list as perceived by the security administrator, $p(A, \mathbf{Pref}_I)$, is shown in (I1). The key difference between the security administrator's preference vector and insider's preference vector is the lack of the *stratagem* action the insider's preference vector. The security administrator accurately perceives the insider's true preference vector, in other words, $p(A, \mathbf{Pref}_I) = p(I, \mathbf{Pref}_I)$.

Next, consider the perceived conflict from the perspective of the insider. (A1) contains the security administrator's preference vector as perceived by the insider, $p(I, \mathbf{Pref}_A)$. The insider perceives that the security administrator prefers outcomes where no confidential data is exposed while the data disclosure and breach are the least preferred outcomes. From the insider's perspective, the security administrator will attempt to detect a data breach and respond to the incident. As previously mentioned, the insider's true preference vector, $p(I, \mathbf{Pref}_I)$, is shown in (I1).

The rational outcome for Hypergame 1 shows that the security administrator should deploy a stratagem when the insider is unaware of the use of a stratagem. Table 10 shows the perceived equilibrium for each player in the game. Recall that the security administrator perceives the correct strategy of the insider. The security administrator's perceived equilibrium is to deploy a stratagem. The insider under the security administrator's perceived game will either probe or not. The equilibrium under the insider's perceived game is to probe at the risk of being caught afterward by the security administrator's incident response.

Note that the perceived equilibrium for each player is different, and thus they are considered *hypergame-destroying*



$$H^2_{A,I} = \left\{ \underbrace{p(A, G_{A,I})}_{\substack{\text{Security Administrator uses a stratagem and believes the insider is unaware}}} , \underbrace{p(I, H^1_{A,I})}_{\substack{\text{Insider aware of of the use of stratagem}}} \right\}$$

Figure 2: Hypergame 2 models the case where the security administrator uses a stratagem and the insider is aware. The insider perceives the conflict as a hypergame. Note that the insider accurately perceives that the security administrator believes that the insider is unaware of the use of a stratagem.

equilibrium. In other words, each perceived equilibrium will not hold if the conflict consists of multiple rounds. However, since we model the conflict as a *snap-shot decision*, we consider the rational actions from each player's perspective. The security administrator selects the *stratagem* action as it is the only rational action available for each equilibrium. Likewise, the insider selects the *probe* action. Therefore, we conclude that the conflict equilibrium for hypergame 1 is the *Deceive/Tricked* outcome as shown in the bottom of Table 10.

5.3 Hypergame 2 - Informed Insider

Hypergame 2 explores the case where the security administrator uses a stratagem and believes that the potential insider is unaware. However, the insider is *aware* that the administrator is utilizing a stratagem. We construct Hypergame 2 as a second-level hierarchical hypergame since the insider is aware of the administrator's misperception of the conflict. As explained in [7], to model a player who perceives another player's misperception, a second-level hypergame is needed. For each player in a given conflict who is aware of some other player's misperception, the player views the conflict as a hypergame as oppose to a game.

From the security administrator's perception, he/she views the conflict as $p(A, G_{A,I})$, which is equivalent to the security administrator's perceived game presented in Hypergame 1. On the other hand, the insider views the conflict as a hypergame. The hypergame consists of $p(IA, G_{A,I})$, the security administrator's perceived game as perceived by the insider, and $p(II, G_{A,I})$, the insider's perceived game. The preference vectors are shown in (A2) for the security administrator and (I1) lists the preference for the insider as perceived by the security administrator.

The conflict as perceived by the insider is a hypergame consisting of perceived games $p(IA, G_{A,I})$ and $p(II, G_{A,I})$.

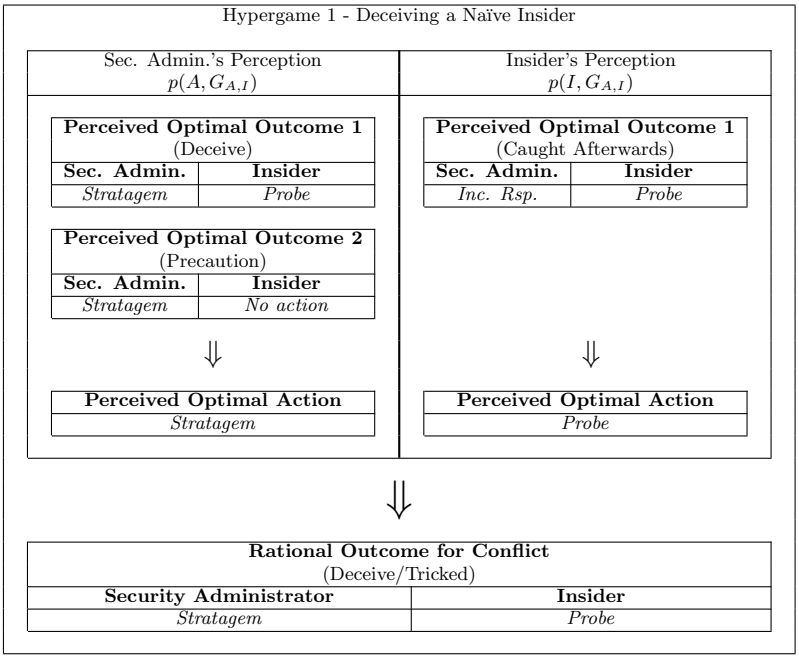


Table 10: Stability analysis for Hypergame 1 - Deceiving a Naïve Insider

The security administrator's game as perceived by the insider, $p(IA, G_{A,I})$, is the same as the security administrator game, $p(A, G_{A,I})$. The preferences vectors for the security administrator's game as perceived by the insider are shown (A2) and (I1). Note that $p(A, G_{A,I}) = p(IA, G_{A,I})$ because the insider correctly perceives the security administrator's preference vector. The insider's perceived game, $p(II, G_{A,I})$, is composed of the preferences (A2) and (I2). The preferences for the security administrator as perceived by the insider, $p(II, \mathbf{Pref}_A)$, incorporates the *stratagem* action. On the other hand, the insider's preference vector, $p(II, \mathbf{Pref}_I)$, reflects that a stratagem is in use. In summary, the preference vectors represented in the insider's perceived conflict models the perception that the security administrator is using a stratagem and believes the insider is unaware.

The stability analysis builds on the findings in Hypergame 1. Since the security administrator's perception in Hypergame 2 is the same as in Hypergame 1, we conclude that the perceived optimal action for the security administrator is to deploy a *stratagem*. Determining the insider's rational action requires stability analysis of each perceived game. The security administrator's game as perceived by the insider follows the same analysis in Hypergame 1. The insider concludes that security administrator will deploy a *stratagem*. The insider's perceived game contains two equilibrium. In no particular order of preference, the first possible equilibrium is the *no progress* outcome, meaning that insider and security administrator do not invoke an action. The second equilibrium is the *suspect* outcome where the security administrator deploys a stratagem, but the insider does not act on it. In both cases, the Insider's rational action is *no action*, which leads to the rational outcome of the conflict as *Precaution/Suspect*.

Alternatively, an insider may decide to be more aggressive in pursuing confidential data to exfiltrate. For example, the Insider decides to place the *caught afterward* outcome in the second position in the preference list. This modification does not impact the rational outcome of the conflict.

A follow-up question is whether or not the Security Administrator can make the Insider believe that they are using a stratagem but not dedicate the resources into deploying some stratagem. In particular, it may be advantageous, from an economic perspective, to dedicate resources into fooling an insider into believing that a stratagem is deployed rather than deploying the stratagem.

5.4 Hypergame 3 - Bluffing the Insider

Hypergame 3 explores the security administrator's strategy in deceiving the insider into believing that a stratagem is in place within the corporate infrastructure, but without actually dedicating resources to plant real stratagem in the system. The scenario is analogous to a homeowner that places a sign outside their home that says "beware of vicious dog" without actually owning a dog. The administrator must *mask* the reality that no stratagem is in place while *mimicking* or *inventing* evidence to fool the insider into believing that a stratagem is deployed. In this hypergame, the security administrator uses the *false stratagem* action rather than the *stratagem* action. Further, the security administrator correctly perceives the insider's preference vector,$p(A, \mathbf{Pref}_I) = p(I, \mathbf{Pref}_I)$ and thus believes that the insider is aware of the use of a stratagem.

On other words, the Insider perceives that some stratagem is in use by misinterpreting the *false stratagem* action as a *stratagem*.

The perceived equilibrium analysis and rational outcome for Hypergame 3 are shown in Table 12. From the security administrator's perspective, the rational action is to plant *false stratagem* while the insider does *no action*. On the other hand, as in hypergame 2, the security administrator

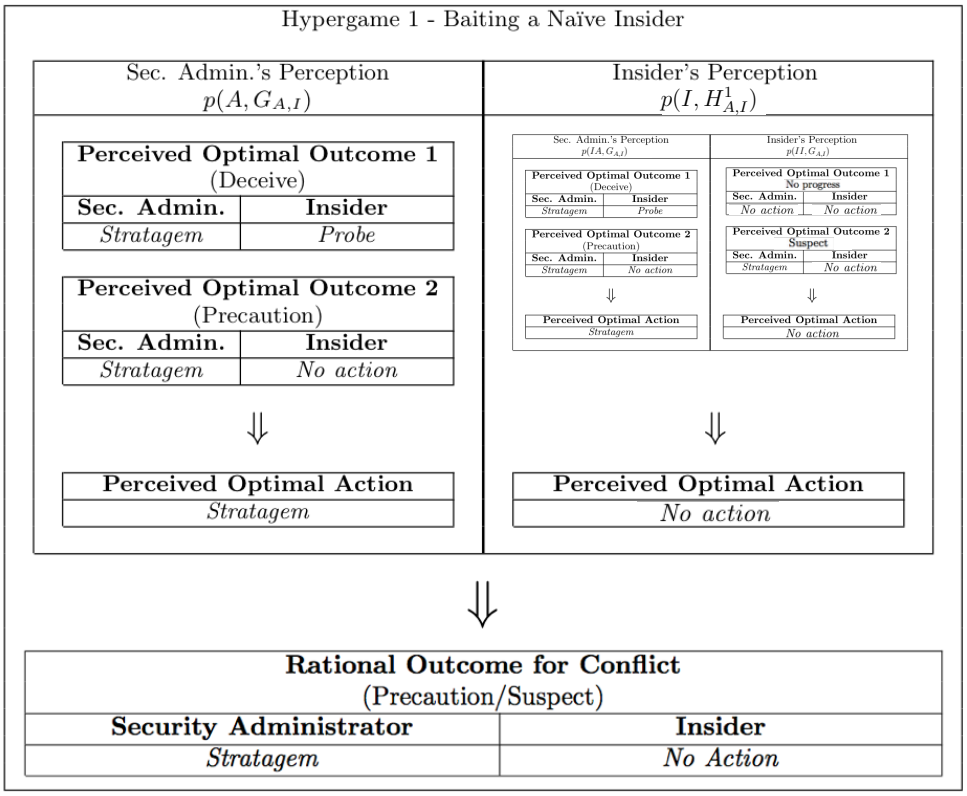


Table 11: Stability Analysis for hypergame 2 - Informed Insider

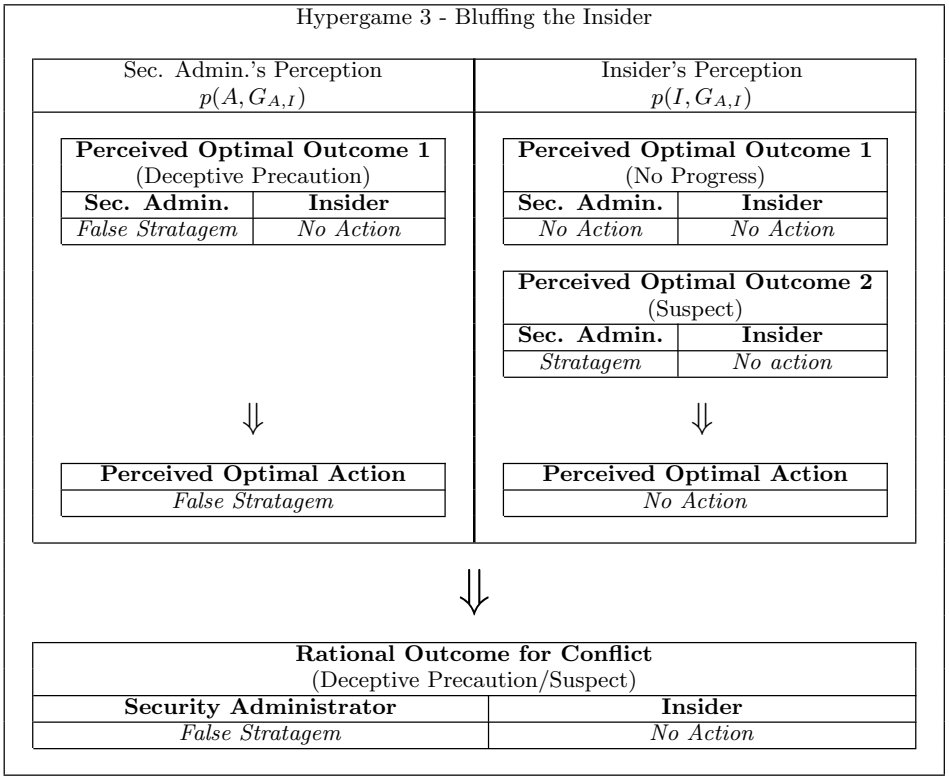
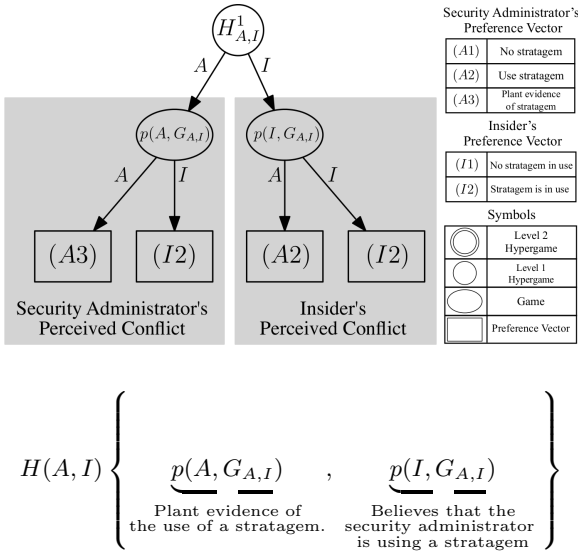


Table 12: Stability analysis for Hypergame 3 - Bluffing the Insider



$$H(A,I)\left\{\begin{array}{ccc} \underbrace{p(A,\underbrace{G_{A,I}}_{\text{Plant evidence of the use of a stratagem.}})}_{} & , & \underbrace{p(I,\underbrace{G_{A,I}}_{\text{Believes that the security administrator is using a stratagem}})}_{} \end{array}\right\}$$

Figure 3: Hypergame 3 insider believes that the administrator is using bait but in reality is not

perceived rational action is *no action* while the security administrator deploys a *stratagem*. Finally, the rational outcome of hypergame 3 is *Deceptive Precaution*.

Hypergame 3 indicates that bluffing may provide a defense advantage. In the best case, the insider may decide that the risk of exfiltrating confidential data is too high. Alternatively, the insider may choose to search for clues about the true nature of the administrator's action in subsequent rounds of the conflict. The latter situation is advantageous in the sense that it forces the attacker to slow down and second guess the best course of action.

6. DISCUSSION

6.1 Sensitivity Analysis

For each of the presented hypergames, a description of the player's strategy and assumptions are given. We provide a plausible rationalization for each preference vector, but there are viable alternatives for each player. In Hypergame 2, we explored an alternative preference vector that reflects a more aggressive insider than we previously assumed. However, even with a more aggressive insider, the rational outcome remained the same. Exploring alternative yet plausible preference vectors for players in a hypergame can be used to analyze the sensitivity of a player's strategies to the knowledge of the perceptions that she and others have in the game.

Fraser and Hipel in [7] discuss the importance of *sensitivity analysis*. Sensitivity analysis explores the viability of the results derived from a hypergame, even if the perceived information of the conflict is erroneous. A sensitivity analysis may provide insights on the robustness of particular strategies for players engaged in a conflict. Fraser and Hipel propose a sensitivity analysis by analyzing hypergames with various conceivable preference vectors. Additionally, a coalition analysis explores cases where multiple players have commonly aligned goals or work together in secrecy against another player. A strong defense strategy should be robust against some misperceptions of the conflict.

An extension of the presented hypergame models may include a third player, an outsider, who coordinates in secrecy with an insider to exfiltrate information. An optimal defense strategy for a security administrator should be robust against a given level of collusion among adversaries, an issue that we have sidestepped in this work. Further, a sensitivity analysis should consider insiders and security administrators that do not always choose the most optimal strategy. The models presented in this work assume that players always choose the most rational action. The sensitivity analysis should consider players who strategically decide on a less-than-optimal action due to practical constraints that have not been modeled in detail in the game framework. Simply reordering the preference vectors of a player may not be enough to capture this notion and one should consider alternative actions that may not be the most rational decision for a given player.

6.2 Perception Evolution

Each hypergame explored in this paper consists of a snapshot in time. The players in the game perceive each player's actions and rationalize the most optimal action to execute. However, executing an action may reveal information regarding the strategies or actions for players. Particular actions may reveal, for example, details about a particular stratagem designed to deceive a player. Over time, a player's perception of the conflict changes.

As a more detailed example, say that we start a conflict as Hypergame 1 where the insider is unaware of the use of a stratagem. Say that the insider falls for the stratagem, and the news of the incident becomes publicly known. Besides catching the Insider, there are two secondary outcomes. First, Insiders in the future will believe that the security administrator may use a stratagem and second, the insider will have an idea of how the stratagem will work. The security administrator, on the other hand, now must consider the bias a new potential insider will have. One possible strategy for the security administrator is to place false evidence that points to a stratagem being used, when in reality it is not. For example, the administrator gives the impression that a honeypot is present within a system when in fact it is not, thus saving on defense costs while gaining some of the advantages of that defense mechanism.

Note that the result of the first hypergame impacts the perception of the conflict over time. Future work will investigate various deceptive defense strategies and analyze the evolution of players' strategy over time. Thus, we will move from a single snapshot game to a game that evolves either in discrete or continuous time over a given time horizon.

6.3 Related Work

Early work in hypergames in the late 1970s and 1980s focused on military conflicts as explored in [5, 19, 7]. In particular, two conflicts illustrate the strategic advantage of exploiting an opponent's misperception. Bennett and Dando in [5] and Fraser and Hipel in [7] analyze the fall of France in WWII as a hypergame where France's *under-perception* of Germany's strategy to invade through the Ardennes led to a "disastrous defeat." Takahashi, Fraser, and Hipel in [19] applied hypergame analysis to the Allied invasion of Normandy where deception was heavily used to draw the German military presence away from Normandy to Calais.

The Fall of France conflict demonstrates the advantage of exploiting an opponent's misperception, and the analysis of the Allied invasion of France shows the advantage of using deception, even if all players in the conflict are aware of the use of deception.

Some prior work also explores hypergames applied to cyber conflicts. Poisel in [14] discusses the applicability of hypergames in the context of Information Warfare. In particular, Poisel notes that the dynamics of hypergames fit well with the observe, orient, decide, and act (OODA) loop, where a player's perception of the conflict occurs in the Observe-Orient phase and a player's action occurs during the Decision-Action phase. At the end of a hypergame, a player's perception of the conflict can be adjusted based on new observations. Furthermore, Poisel notes that Information Warfare strategies such as Denial of Information, Disruption and Destruction, Deception and Mimicry, and Subversion map onto hypergames. The work presented here extends [14] by providing a tutorial of how to apply hypergames models to conflicts that use deception as a defense strategy. As discussed in Section 6.2, to successfully use deception, one must understand the adversary's perception of the conflict, how it evolves as strategies, techniques, policies are revealed, and whether or not actions are resilient to noise. Each hypergame presented builds off of each other in a sequence. The first hypergame assumes that the insider does not know about the stratagem, the second explores an insider that is aware of the stratagem, and the third explores the case where the administrators know that the potential insider is aware of the stratagem. Between the first and second hypergames, the insider learns the use of stratagem and between the second and third, the administrator learns that future insider threats may know the use of certain stratagem. The iterations between each hypergame follow an implicit OODA loop. Future work will expand the hypergames presented here and explore an exhaustive iterative process with evolving perception similar to the work presented in [8].

[10] considers the use of hypergames to model information security conflicts. The work presents an abstract two-level hypergame with a single attacker and defender. The hypergame only considers a conflict where the attacker is not aware of one of the actions available to the defender (e.g. a strategic surprise). In other words, the attacker *under-perceived* a possible strategy of the defender. Our work explores three different hypergames that explore additional plausible scenarios: attacker *under-perceives* the actions of the defender (Hypergame 1)[6], defender *under-perceives* the attacker (Hypergame 2), and where the attacker *misinterprets* the actions of the defender. We also explain how the presented hypergames reflect plausible real life scenarios.

Gibson's thesis [8] explores the application of hypergames in defending computer networks. Gibson models the scenario using Hypergame Normal Form (HNF) that incorporates an updating belief context for the defender. The administrator must select an appropriate response depending on the level of threat. The administrator may choose to shutdown the system (at a high cost) if the adversary is a significant threat rather than just a nuisance. Alternatively, the defender may deploy a ruse, such as a honeypot, as a defense strategy. The administrator's belief context evolves as iterations of the conflict progresses over time. We believe that our contributions will serve as a tutorial to explore other cyber conflicts that incorporate deceptive strategies such as [11] and [1]. Future work will explore the HNFs with evolving believe context for attacker and defenders.

[6]Hypergame 1 is similar to the hypergame presented in [10] only in the sense that the attackers *under-perceives* an action. The preference vectors for each player and hypergame structure differ significantly.

7. CONCLUSION

This paper presents a primer on modeling and analyzing information security conflicts that incorporate deceptive defense mechanisms, using hypergames. We discuss the kernels necessary to model player misperception in deceptive defense strategies in the domain of information security. We demonstrate the use of these kernels by modeling a cyber conflict where an insider wishes to exfiltrate confidential data and where the security administrator combines conventional security practices along with a deceptive strategy to deceive an attacker into exposing themselves and thus preventing a data breach. We model the conflict as three different hypergames that explore plausible misperceptions for each player. The analysis provides insights on the effectiveness of incorporating deceptive mechanisms into information security defense. In particular, a successful deception may catch insiders in the act of exfiltrating confidential information or waste the insider's resources who believe some deceptive strategy is in place. Furthermore, a security administrator may decide to plant evidence of a deceptive strategy without actually deploying it. Such a strategy may have the same effect on the Insider as deploying a real defensive component, but with an economic advantage for the security administrator.

8. HYPERGAME MODELS

The Hypergame Modelling Language source code for each hypergame model presented in the paper is available at: `https://github.com/cngutierr/insider-hypergames`.

9. ACKNOWLEDGMENTS

This work was supported, in part, by a grant from the Northrop Grumman Corporation. The National Science Foundation supported this research under award 1548114. The authors would also like to thank The Center for Education and Research in Information Assurance and Security (CERIAS) for their support and valuable feedback on ideas presented in this paper.

10. REFERENCES

[1] M. H. Almeshekah, C. N. Gutierrez, M. J. Atallah, and E. H. Spafford. Ersatzpasswords: Ending password cracking and detecting password leakage. In *Proceedings of the 31st Annual Computer Security Applications Conference*, ACSAC 2015, pages 311–320, New York, NY, USA, 2015. ACM.

[2] M. H. Almeshekah and E. H. Spafford. Planning and Integrating Deception into Computer. *Proceedings of the New Security Paradigms Workshop (NSPW)*, 2014.

[3] M. H. Almeshekah and E. H. Spafford. The Case of Using Negative (Deceiving) Information in Data Protection. In *9th International Conference on Cyber Warfare & Security: ICCWS 2014*, International Conference on Cyber Warfare and Security, pages 237–246. Academic Conferences and Publishing Limited, 2014.

[4] P. G. Bennett. Toward A Theory of Hypergames. *Omega*, 5(6):749–751, 1977.

[5] P. G. Bennett and M. R. Dando. Complex Strategic Analysis: A Hypergame Study of the Fall of France. *Journal of the Operational Research Society*, 30(1):23–32, 1979.

[6] L. Brumley. *HYPANT : A Hypergame Analysis Tool Monash University*. PhD thesis, Monash University, 2003.

[7] N. Fraser and K. Hipel. *Conflict analysis: Models and Resolutions*. North-Holland series in system science and engineering. North-Holland, 1984.

[8] A. Gibson. *Applied Hypergame Theory for Network Defense*. PhD thesis, Air Force Institue of Technology, 2013.

[9] J. T. House and G. Cybenko. Hypergame Theory Applied to Cyber Attack and Defense. In *SPIE Defense, Security, and Sensing*, pages 766604–766604. International Society for Optics and Photonics, 2010.

[10] Y. Imamverdiyev. A Hypergame Model for Information Security. *International Journal of Information Security Science*, 3(1):148–155, 2014.

[11] A. Juels and R. L. Rivest. Honeywords: Making Password-cracking Detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &; Communications Security*, CCS '13, pages 145–160, New York, NY, USA, 2013. ACM.

[12] G. H. Kim and E. H. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. *Proceedings of the 2nd ACM Conference on Computer and Communications Security – CCS'94*, pages 18–29, 1994.

[13] L. Milkovic. Defeating Windows Memory Forensics. 29th Chaos Communication Congress, 2012. Presentation.

[14] R. Poisel. *Information Warfare and Electronic Warfare Systems:*. Artech House Intelligence and Information Operations. Artech House, Incorporated, 2013.

[15] N. C. Rowe, E. J. Custy, and B. T. Duong. Defending Cyberspace with Fake Honeypots. *Journal of Computers*, 2(2):25–36, 2007.

[16] G. Silowash, T. J. Shimeall, D. Cappelli, A. Moore, L. Flynn, and R. Trzeciak. Common Sense Guide to Mitigating Threats. 4th Editio(December), 2012.

[17] A. Singh and Z. Bu. Hot Knives Through Butter: Evading File-based Sandboxes. *FireEye*, 2014.

[18] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[19] M. A. Takahashi, N. M. Fraser, and K. W. Hipel. A Procedure for Analyzing Hypergames. *European Journal of Operational Research*, 18:111–122, 1984.

[20] Verizon Business. 2014 Data Breach Investigations Report. *Verizon Business Journal*, 2014:1–60, 2014.

[21] M. Wang, K. W. Hipel, and N. M. Fraser. Modeling Misperceptions in Games. *Behavioral Science*, 33(3):207–223, 1988.

[22] J. Williams and A. Torres. ADD - Complicating Memory Forensics Through Memory Disarray. ShmooCon, Jan. 2014.

[23] J. Yuill, M. Zappe, D. Denning, and F. Feer. Honeyfiles: deceptive files for intrusion detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 116–122, June 2004.