A Hypergame Analysis for ErsatzPasswords

Christopher N. Gutierrez[1](✉), Mohammed H. Almeshekah[2], Saurabh Bagchi[1], and Eugene H. Spafford[1]

[1] Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University, 656 Oval Dr, West Lafayette, IN 47907, USA
{gutier20,sbagchi,spaf}@purdue.edu

[2] Department of Computer Science, King Saud University, King Khalid Rd, King Saud University, Riyadh, Saudi Arabia
meshekah@ksu.edu.sa

Abstract. A hypergame is a game theoretic model capturing the decisions of rational players in a conflict where misperceptions, from deception or information asymmetry, are present. We demonstrate how hypergames can model an actual security mechanism: ErsatzPassword, a defense mechanism to protect password hashes from offline brute-force attacks. Two ErsatzPassword defensive strategies are considered: to block the attacker and trigger an alarm, or to redirect the attacker into a honeynet for attack analysis. We consider the scenario where there is information asymmetry in the system and one side under-estimates or over-estimates the risk tolerance of the other side. We analyze plausible strategies for both attacker and defender and then solve 57,600 hypergame configurations to determine the optimal 1st line defense strategies under various levels of risk tolerance and misperceptions.

Keywords: Computer security · Deception · Game theory

1 Introduction

Information security is a balancing act of allocating resources to defend against threats. As an example, security administrators are required to ensure the security of digital assets with a fixed budget. Priorities are placed on technologies, policies, and practices to minimize breaches. Game theory is a technique that to determine an appropriate strategy once the costs, benefits, and strategy space are rigorously defined [10]. However, perfect information is not always available.

Hypergames extend the classical game theory model by incorporating the *perception* of each player in the game analysis [4]. In a hypergame, each player is operating within a *perceived* game based on her present understanding of

© IFIP International Federation for Information Processing 2018
Published by Springer Nature Switzerland AG 2018. All Rights Reserved
L. J. Janczewski and M. Kutyłowski (Eds.): SEC 2018, IFIP AICT 529, pp. 47–61, 2018.
https://doi.org/10.1007/978-3-319-99828-2_4

other players' actions and preferences. Modeling the perception of players enables hypergames to express conflicts where players attempt to deceive other players, thus influencing their perceptions and actions.

The use of deception-based defenses can be modeled as a hypergame as defenders attempt to deceive an attacker regarding the behavior of the targeted system. Prior work highlights hypergames applied to physical conflicts [3,6,13]. Far less studied is the application of hypergames when defenders can use deception in cyber conflicts, to *deny* attackers access to valuable resources, *misdirect* them away from critical assets, or *confuse* them by presenting plausible yet deceiving information [2]. Further, cyber conflicts introduce complexities that do not have physical analogs. For example, an attacker may utilize some zero-day exploit and remain undetected within an enterprise environment, or security software produces false-positive alerts that consume defensive resources.

We show that hypergames can be a valuable game theoretic model to analyze how to use deception. We demonstrate this by applying hypergames to Ersatz-Password [2]*, a deceptive scheme that protects passwords from offline dictionary attacks against password hashes.*

The ErsatzPassword scheme [2] supplements common hashed passwords by producing fake passwords under brute-force attacks. We apply hypergame theory to model the attacker and the defender's actions, such as, for the defender, deploying the ErsatzPassword scheme, and for the adversary, use a cracked password. We solve the hypergame models to determine the equilibrium conditions for each point in the search space, which are stated concerning the course of action taken by the defender and the adversary. The equilibrium condition depends on the degree of misperception on the part of each party. Our hypergame models consider two defense configurations: detect and block an attacker or direct the attacker to a honeynet system upon detecting a fake password in use. Our overall results show that ErsatzPassword is a useful tool to counter brute-force attacks on password hashes. Under all hypergames considered, the attacker is not successful in breaching the system and is forced to look elsewhere for entry.

2 Mathematical Formulation of Hypergames

Game theory analysis assumes that each "player" in the "game" has a *common perception* "of the game being played." Hypergames is an extension allowing each player to see 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 one or more stages of the game [6].

We consider single-stage games where the rational players make a single move based on their perception of the situation. Multi-stage games evolve by adjusting the player perception based on the information gained or obfuscated in prior stages but are outside the scope of this paper.

2.1 Two-Player Game and Hypergames Definitions

A two-player game consists of a set of players **Players** $= \{A, B\}$ and a non-empty finite set of actions for each player. A player represents an entity 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* for *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 for 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: $\mathbf{Pref}_A = \langle o_{A_1}, \cdots, o_{A_{n \cdot m}} \rangle$, $\mathbf{Pref}_B = \langle o_{B_1}, \cdots, o_{B_{n \cdot m}} \rangle$, where each element in $\mathbf{Pref}_A$ (or $\mathbf{Pref}_B$) is also in $\mathbb{O}$. The elements within the preference list 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}. We use the following notation to represent a game:

$$G_{A,B} = (\underbrace{[A, B]}_{\text{Players}}, \underbrace{[\mathbb{A}_A, \mathbb{A}_B]}_{\text{A/B's actions}}, \underbrace{[\mathbf{Pref}_A, \mathbf{Pref}_B]}_{\text{A \& B's preferences}})$$

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

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

In the definition above, the function p denotes the perception of an individual player, For instance, $p(A, G_{A,B})$ is a game 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. Each player in the hypergame has a perception of the other player's actions and preferred outcomes; however, player A's perceived actions and perceived preferences of player B may not be the true actions and preferences of player B. More formally,

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

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

A player's misperception may be from a lack of information about other players in the game or strategically placed *misinformation* designed to deceive a player; e.g., a defender may let it be known that a host-based intrusion detection system monitors all system calls on a computing system. This would lead to an over-perception of the security of the system by other actors in the system.

3 Misperception in Cyber Conflicts

A player's perception of a conflict may not reflect the truth because of a player's bias, misinformation, misinterpretation, or ignorance. Wang et al. describe in [14] that a player's misperception maps the sets mentioned above in a hypergame—**Players**, $\mathbb{A}$, or **Pref**—to other sets that reflect a given player's possibly mistaken understanding of the conflict. The perception function p introduced in Sect. 2 maps the truth of a conflict to how a given player views the conflict. As inspired by Wang et al. [14], perception mapping is broken into three misperception kernels: *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. Next, we shape the three misperception kernels to be applicable to cyber conflicts and provide practical examples in that domain; The next section provides an example with more detail.

Misperception kernels are the building blocks for modeling deception in a cyber conflict. We next describe how the perceptual mapping function p can be used to model misinterpretation. Over-perception and under-perception are briefly described but the formal treatment is omitted as they were not part of our experimental evaluation and because of space constraints.

3.1 Misinterpretation

Misinterpretation is when a player does not correctly interpret an action, player, or preference vector. 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 $p(B, \mathbf{Players}_A) = \{A', B\}$, where $A' \neq A$. For clarity, B's perception of player A is A' where A' is not identical to the true nature of the actual player 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.

Misinterpretation can manifest as differences of perceived player actions or preferences from the actual player. 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 as $p(B, \mathbb{A}_A) = \{a'_1, \cdots, a_n\}$, where $a'_1 \notin p(A, \mathbb{A}_A), a'_1 \neq a_1$.

As an example, communication through a hidden channel can be modeled as misinterpretation where the observer of the medium 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 A posts vacation photos on a personal website." However, Player A's correct action a_1 is "Post vacation photos that contain hidden trade secrets through the use of steganography."

Finally, a player may misinterpret another player's preference list. A's preference list, $\mathbf{Pref}_A = \{o_1, o_2, \cdots o_n\}$, may be misinterpreted as $p(B, \mathbf{Pref}_A) = \{o_i, o_j, \cdots, o_k\}$, where $i, j, \cdots, k \in 1 \cdots n$. For instance, let Player A represent a system administrator, and Player B be a valid user. B's perception of A's preference list is a permutation of A's true preference list. From a security perspective, misinterpretation could be port scanning for program interoperability (e.g., service discovery). In this scenario, B, the valid user, assumes that service discovery on a corporate network is a valid action, but in reality, A, the system administrator, places service discovery low on her preference list as it may create false alarms in security monitoring tools. The action remains unchanged, but the hypergame allows the exploration of different possibilities of misinterpretation.

3.2 Over-Perception

A player may over-perceive a conflict by believing that there are additional players or actions that do not exist. Security software that produces false alarms could lead to over-perceiving a situation, e.g., anti-malware software flagging the installation of a benign application as malicious. A security administrator (A) may interpret a coworker's (B) action as installing malware to compromise the system but in reality, the application is benign, and B is non-adversarial.

A player may over-perceive a conflict to include players that are not present in a conflict. Building on the previous example, the security administrator believes that someone within the company is installing malicious software. However, such an actor does not exist.

If a player over-perceives a conflict, outcomes will be considered that do not reflect reality, impacting the actions taken under equilibrium conditions. From the previous example, an administrator may over-perceive and thus take unnecessary and intrusive actions, such as incident response and forensic examination on Player B's workstation, interfering with the ability to work.

3.3 Under-Perception

A player in a conflict may also under-perceive the presence of other players, the types of actions that a player may execute, or the resources that another player has. For example, a user A may under-perceive the actions of an application that she installs on her system. A believes that the application is a photo editor but she under-perceives a hidden action that uploads her files to a remote server.

Players in a conflict may also be under-perceived, such that some players may be unaware of the existence of other players. For example, a security administrator (Player B) is aware of an external attacker (Player A) but is unaware that the external attacker is colluding 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. For example, a malware analyst (A) may run a suspicious application (B) within a sandbox to determine if the application is malicious. However, the application may withhold malicious actions when running within a sandbox [11]. A concludes that the application is benign, and thus, under-perceives the nature of the test application.

3.4 Stability Analysis

An important aspect of Game Theory analysis is the Nash Equilibrium (NE). This notion is extended in hypergames such that an equilibrium is an outcome that is stable for *all* players in the hypergame. The equilibrium analysis provides us the rational actions that each player chooses. It provides insight into what is to be expected given the misperceptions on players, player actions, or preferences. For a hypergame, an NE is determined in a two-part analysis which is formally defined in [14] and summarized here. The first step is to identify each player's *perceived optimal action*, which is derived from the NE 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. Note that as in game theory, the outcome of a hypergame stability analysis may have multiple equilibria.

If the hypergame consists of multiple rounds, the stability analysis will differ from prior rounds as action, players, and preferences are revealed, clarifying misperceptions of the conflict. Wang et al. in [14] describe *unstable equilibria* as rational outcomes that change over the course of a conflict. An unstable equilibrium exists if there are players that can improve their outcome by changing actions, given that the other players do not alter their actions. Such instances are called *hypergame-destroying equilibria*, and If there exists an outcome that all players perceived as an NE, then the outcome is a *hypergame-preserving equilibrium*. A conflict may also 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* because all players select actions based on their perceptions and the conflict immediately ends; e.g., the players do not have an opportunity to observe the actions of the other players to change their strategies. In this paper, we consider only single stage hypergames, and thus our analysis uses snap-shot equilibria.

3.5 Information Security and Stability Analysis

When analyzing conflicts, it is crucial to consider the type of equilibrium that is appropriate. Some scenarios may require multiple rounds of action, observation, and strategy adjustments to reach equilibrium while other scenarios are modeled as snap-shot conflicts of a single rational decision, as described above.

Cyber conflicts do not stabilize to a steady state. Adversaries find new exploits to compromise systems as new patches, policies, and defenses are 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. We model a cyber conflict that utilizes deception in defense and conducts a stability analysis to find the *rational outcomes* of the conflict. The *rational outcomes* will indicate how an adversary may react to some deceptive component or show the limits/power in deploying a deceptive defense system.

We define deceptive defense systems as software, personnel, data, or policies to deceive attackers and decrease their ability to succeed. Examples include honeypots [12], honeyfiles [15], honeywords [9], and ErsatzPasswords [1].

4 Modeling the ErsatzPassword Scenarios

The cyber conflict scenarios consists of two players: an *Attacker* (abbreviated as "$\mathcal{A}$") and a *Security Administrator* (abbreviated as "$\mathcal{S}$"). The ErsatzPassword (abbreviated as EPW) security control is modeled as several different hypergames. Each hypergame considers two EPW configurations with various levels of misinterpretation and constraints. Each setting of the preference list for $\mathcal{A}$ and $\mathcal{S}$ defines one hypergame, which is solved for an equilibrium condition.

The equilibrium for each hypergame is solved with Hypant [5]. However, we have extended the Hypant tool to allow the user to set different levels of misinterpretation and risk tolerance for each player.

The equilibria solutions provide insight on what EPW configuration $\mathcal{S}$ should use, under what conditions (e.g., degrees of misinterpretation), and how useful it is in blocking, catching, or deterring $\mathcal{A}$.

4.1 Background on Security Control

The EPW scheme [1] uses deception to protect salted password hashes (SHP) from offline brute-force attacks by modifying password hash creation. The term *"ersatz password"* can be interpreted as "fake passwords", *i.e.*, not passwords of legitimate users in the system. For each user/password pair, a fake password and a hardware dependent function (HDF) are used to produce the EPW hash that is indistinguishable from salted password hashes. If $\mathcal{A}$ steals and brute-forces the hashes, fake passwords are found rather than the real ones; $\mathcal{A}$ needs access to the HDF to recover the valid passwords. Using fake passwords triggers alarms. This kind of security control has become increasingly important because passwords are still the predominant form of authentication, and leaks of password files followed by brute-force attacks are an all-too-common occurrence.

We consider two configurations of the EPW scheme. In the first, an alarm triggers when $\mathcal{A}$ attempts to authenticate with a fake password, effectively blocking $\mathcal{A}$ from infiltrating the system. We abbreviate this as *EPW Alarm*. The second configuration directs $\mathcal{A}$ into a network of honeypot systems (honeynet). Once $\mathcal{A}$ enters the honeynet system, $\mathcal{S}$ can observe $\mathcal{A}$'s actions and potentially attribute the attacker or gain knowledge of her actions. We abbreviate this configuration as *EPW Redirect*. In our model, $\mathcal{A}$ has access to the users' hashed passwords, knows how the EPW scheme works, but does not know if the scheme is deployed on the target system, and does not have access to the HDF.

4.2 Players and Actions

We assume that $\mathcal{A}$ has a set of cracked password hashes for user accounts on a system defended by $\mathcal{S}$. After cracking, $\mathcal{A}$ attempts to authenticate with the cracked password. If $\mathcal{S}$ deploys an EPW scheme and $\mathcal{A}$ uses a cracked password, then $\mathcal{A}$ risks exposure by triggering an alarm that alerts $\mathcal{S}$.

$\mathcal{A}$ has two mutually exclusive actions:[1] *use passwords* (*Use PW*) to infiltrate the system or *Find an Alternative Attack* (*Find Alt.*). If $\mathcal{A}$ uses the cracked passwords, the malfeasance is detected only if $\mathcal{S}$ has deployed an EPW scheme, whether EPW Alarm or EPW Redirect. In general, $\mathcal{A}$ prefers to *Find Alt.* if $\mathcal{S}$ has deployed an EPW scheme. However, we also explore cases where $\mathcal{A}$ is risk-tolerant and thus more inclined to use cracked passwords.

$\mathcal{S}$ protects the credentials of users within her system. In addition to standard security practices, she may choose to protect users' passwords on her system with an EPW scheme, rather than the traditional SHP scheme. $\mathcal{S}$ may prefer to use the SHP or the EPW scheme over the more expensive honeynet scheme if the former can protect the system and costs less. $\mathcal{S}$ may also decide to use SHP because $\mathcal{A}$ may avoid using cracked passwords altogether if $\mathcal{A}$ thinks an EPW scheme is in place and the risk of detection is too high. This is what is commonly referred to in economic theory as a "positive externality," *i.e.*, a benefit that accrues to a party without deploying some control.

Table 1. All outcomes for our ErsatzPassword Hypergames. Each outcome has an designated name in the first and second column from the perspective of the security administrator and the attacker respectively.

Outcomes		S. admin. action	Atk action	
S. admin.	Atk	*S. Hash, Alrm, Redir*	*Use PW, Find Alt.*	Description
No Incd	No Prg 1	S. Hash	Find Alt	S. hashes w/o EPW.
PW Prt 1	No Prg 2	EPW Alrm	Find Alt	Prt PWs with Alrm.
PW Prt 2	No Prg 3	EPW Redir	Find Alt	Prt PWs w/Hnet.
Breach	Success	S. Hash	Use PW	Fail to protect PW.
EPW Alrm	Blocked	EPW Alrm	Use PW	Block atkr access.
EPW Redir	Caught	EPW Redir	Use PW	Trap attkr in Hnet.

4.3 Outcomes

Table 1 shows all outcomes in our scenario. Each outcome has a label from the players' perspective. $\mathcal{S}$'s *No Incident* or $\mathcal{A}$'s *No Progress 1* outcome are cases

[1] The action space in our formulation is discrete and each player takes a single action.

where $\mathcal{A}$ does not attempt to use cracked passwords. Instead, $\mathcal{A}$ decides to look for an alternative attack because of the risk of encountering an EPW scheme. The *No Incident* outcome for $\mathcal{S}$ is advantageous because user accounts are protected without deploying EPW. Similarly, $\mathcal{S}$'s *PW Protected 1/2* or $\mathcal{A}$'s *No Progress 2/3* outcomes result when $\mathcal{A}$ seeks an alternative attack for fear of discovery, but in these cases, $\mathcal{S}$ has deployed EPW. The last three outcomes model the case where $\mathcal{A}$ decides to use the cracked passwords. In the $\mathcal{S}$'s *Breach* or $\mathcal{A}$'s *Success* outcomes, $\mathcal{A}$ successfully infiltrates the system by using one of the cracked user passwords. As the EPW scheme is not in use, no alarm is raised. $\mathcal{S}$'s *EPW Alarm* and $\mathcal{A}$'s *Blocked* outcome is the case where $\mathcal{A}$ tries to enter the system using a cracked user password but is blocked because the EPW mechanism triggers an alarm. Finally, $\mathcal{S}$'s *EPW Redirect* and $\mathcal{A}$'s *Caught* outcome is the case where $\mathcal{A}$ believes, erroneously, that she successfully accessed a system with a cracked password. In reality, $\mathcal{A}$ is redirected into a honeynet system.

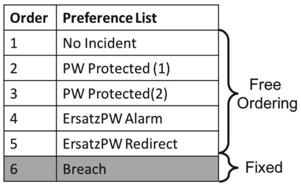
4.4 Preference Lists

A preference list is an ordered list of outcomes for each player, from most preferred outcome to least preferred outcome. Each setting of the preference list for $\mathcal{A}$ and $\mathcal{S}$ defines one hypergame, which is solved for an equilibrium condition. This equilibrium condition defines the actions taken by rational players.

There are a total of four preference lists to consider: The Security Administrator $\mathcal{S}$'s Preference List ($p(\mathcal{S}, \mathbf{Pref}_{\mathcal{S}})$), the Attacker $\mathcal{A}$'s Preference List ($p(\mathcal{A}, \mathbf{Pref}_{\mathcal{A}})$), $\mathcal{S}$'s perception of $\mathcal{A}$'s Preference List ($p(\mathcal{S}, \mathbf{Pref}_{\mathcal{A}})$), and $\mathcal{A}$'s perception of $\mathcal{S}$'s Preference List ($p(\mathcal{A}, \mathbf{Pref}_{\mathcal{S}})$). With four preference lists and six possible outcomes, there are a total of $(6!)^4 \approx 269$ billion scenarios to consider.

Certain sub-segments of the preference list are fixed to reduce the preference vectors to be explored. For instance, $\mathcal{A}$ could place the *Success* outcome as the most preferred outcome in her list, but the ordering of the other outcomes, such as *No Progress (1)* and *No Progress (2)*, could be tied in preference. Additionally, the placement of the *Blocked* outcome could be placed higher or lower as $\mathcal{A}$'s preference. A *Risk-Tolerant Attacker* may decide to place the *Blocked* outcome higher on her preference list in comparison to a *Risk-Averse Attacker* who weighs the risk of detection much higher. While both attackers ultimately do not want to be discovered, the second attacker has a greater interest in remaining undiscovered as it allows her to try alternative strategies without raising alarms. We consider both Risk-Averse and Risk-Tolerant Attackers in our evaluation. Each preference list is partially ordered. We make the following assumptions: (i) $\mathcal{S}$'s *Breach* outcome is **always** least preferred; (ii) $\mathcal{A}$'s *Success* outcome is **always** most preferred; (iii) $\mathcal{A}$ knows the exact preference list of $\mathcal{S}$ but $\mathcal{S}$ may have misinterpretation of $\mathcal{A}$'s preference list; (iv) $\mathcal{S}$ and $\mathcal{A}$ are correctly aware of the players in the game ($\mathcal{S}$ and $\mathcal{A}$) and their possible actions (Table 2).

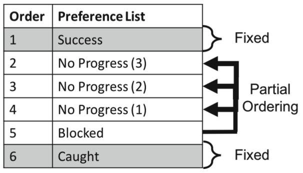
Table 2. Preference vector design space



Order	Preference List
1	No Incident
2	PW Protected (1)
3	PW Protected(2)
4	ErsatzPW Alarm
5	ErsatzPW Redirect
6	Breach

Free Ordering
Fixed

(a) $p(\mathcal{S}, \mathbf{Pref}_{\mathcal{S}})$ template



Order	Preference List
1	Success
2	No Progress (3)
3	No Progress (2)
4	No Progress (1)
5	Blocked
6	Caught

Fixed
Partial Ordering
Fixed

(b) $p(\mathcal{A}, \mathbf{Pref}_{\mathcal{A}})$ template

In our experiments, we make these assumptions to fix some preferences in lists and enumerate through *all* preference lists that are now allowed by the available degrees of freedom. As $\mathcal{S}$'s Breach and $\mathcal{A}$'s Success outcomes are fixed in their position on the preference lists, the degree of freedom in each player's preference list is five slots. There are four possible attacker preference lists (corresponding to where the "Blocked" outcome fits) and 5! possible preference lists that $\mathcal{S}$ perceives for $\mathcal{A}$. Likewise, $\mathcal{S}$ has 5! possible orderings of the preference lists. Thus, we consider $4 \times (5!)^2 = 57,600$ hypergame configurations.

4.5 Overall Equilibrium Analysis

We analyzed all 57,600 hypergames configurations. There are 15 different equilibria that consist of one to three outcomes each. Several hypergames produce multiple equilibria, *i.e.*, players have multiple options in choosing a rational action. From a defensive perspective, the equilibria provide insights into possible outcomes to better prepare for and react to threats. Table 3 lists all equilibria. Note that the *Caught* outcome, the case where $\mathcal{A}$ decides to use the cracked password and $\mathcal{S}$ uses the EPW scheme with the honeynet, is mostly observed. The *Caught* outcome appears in 61.67% of all 57,600 hypergame configurations. As the *Caught* outcome appears most often, a practitioner should take care in preparing the honeynet, train the appropriate personnel, and gather the resources necessary to conduct attribution analysis.

The Blocked outcome, the case where $\mathcal{S}$ uses the EPW scheme with the alarm and $\mathcal{A}$ uses the cracked password, appears in about 29.16% of all hypergames analyzed. Thus, in nearly 90% of the hypergames considered, the most rational choice for $\mathcal{S}$ is to use one of the two EPW configurations.

Equilibria with neither the *Blocked* nor *Caught* outcome account for about 11.67% of all hypergames analyzed. The most common outcomes where the attacker does not use the password is *PW Protected 1*, where $\mathcal{S}$ uses the EPW configuration with an alarm, while $\mathcal{A}$ chooses to find an alternative attack vector. Given our assumption that $\mathcal{A}$ can determine $\mathcal{S}$'s true preference list, note that in the equilibria for all hypergames $\mathcal{A}$ fails to breach the system. The most common outcome from $\mathcal{A}$'s perspective is *Caught.* The second most common are the *No Progress* outcomes, where $\mathcal{A}$ looks elsewhere for its target. Note that in deriving the frequencies of the outcomes if a row has multiple possible equilibrium outcomes we have added that to *all* of the outcomes in that row.

Table 3. Equilibria for all 57,600 hypergames from Sect. 4.4. None of the equilibria contain a successful attacker. (Legend: EPW R = EPW Redirect, EPW A = EPW Alarm, PW P = Password Protected, No Prg = No Progress, No Inc = No Incident, Cght = Caught, Blkd = Blocked)

Obs.	Eq. 1	Eq. 2	Eq. 3
34.16%	EPW R/Cght	PW P2/No Prg 3	
15.42%	EPW R/Cght	PW P2/No Prg 3	
14.38%	EPW A/Blkd	PW P2/No Prg 3	
6.25%	EPW A/Blkd		
5.42%	PW P1/No Prg 2		
5.0%	EPW R/Cght		
3.96%	EPW A/Blkd	No Inc/No Prg 1	PW P2/No Prg 3
3.33%	EPW R/Cght	No Inc/No Prg 1	PW P2/No Prg 3
2.92%	No Inc/No Prg 1	PW P1/No Prg 2	
2.5%	EPW A/Blkd	EPW R/Cght	PW P2/No Prg 3
2.08%	EPW A/Blkd	No Inc/No Prg 1	
1.67%	No Inc/No Prg 1	PW P2/No Prg 3	
1.67%	PW P2/No Prg 3		
1.25%	EPW R/Cght	No Inc/No Prg 1	PW P1/No Prg 2

4.6 Misinterpretation Analysis

There are two scenarios considered in our analysis: (i) $\mathcal{S}$ misinterprets the preference order of the attacker and (ii) scenarios without misinterpretation. For cases where $\mathcal{S}$ and $\mathcal{A}$ do not misinterpret each other, the players, actions, and preferences are perfectly perceived. Under these scenarios, the hypergame reduces to a classical game. Accurate threat intelligence actualizes perfect perception scenarios. However, we later calculate that "Perfect Player Perception" accounts for less than 1% of all hypergame scenarios we consider. This emphasizes the importance of going beyond classical game theory to analyze cyber conflicts where the players do not have perfect information about each other. We posit that this is the common case in practical systems.

The second case is an $\mathcal{A}$ who correctly interprets $\mathcal{S}$'s preferences while $\mathcal{S}$ misinterprets $\mathcal{A}$. Formally, $p(\mathcal{A}, \mathbf{Pref}_{\mathcal{S}}) = p(\mathcal{S}, \mathbf{Pref}_{\mathcal{S}})$ and $p(\mathcal{S}, \mathbf{Pref}_{\mathcal{A}}) \neq p(\mathcal{A}, \mathbf{Pref}_{\mathcal{A}})$. The second case models situations where $\mathcal{A}$ understands $\mathcal{S}$ through careful reconnaissance or through insider information. However, $\mathcal{S}$ fails to understand the preferences of $\mathcal{A}$ because of poor threat intelligence.

Figure 1 shows the results where there is no misinterpretation on either side (left half of the figure) and hypergame configurations where $\mathcal{S}$ misinterprets the order of preferences of $\mathcal{A}$ (right half of the figure). The "Perfect Player Perception" cases account for 0.83% (480/57,600) of all hypergame configurations and misinterpretation accounts for 99.16%. The percentage in each circle represents the percentage of hypergames where the equilibrium contains *Caught*, *Blocked*, or *No Progress*. The results show that misinterpretation has no impact on the overall results because the ratios of Caught/Blocked/No Progress cases are maintained for both the *Perfect Player Perception*, and $\mathcal{S}$ *Misinterprets* $\mathcal{A}$. $\mathcal{A}$ has a perfect perception of the game but is not aware of her accurate perception.

The results also show that the most rational action that $\mathcal{S}$ should take is to deploy the EPW scheme with the honeynet configuration. The Caught outcome accounts for the majority of the hypergame configurations considered and $\mathcal{S}$'s misinterpretation of $\mathcal{A}$ preferences does not impact the ability to discover $\mathcal{A}$. There is one case containing both Caught and Blocked in the equilibrium outcome, which is 2.5% of all hypergames considered (not shown in Fig. 1).

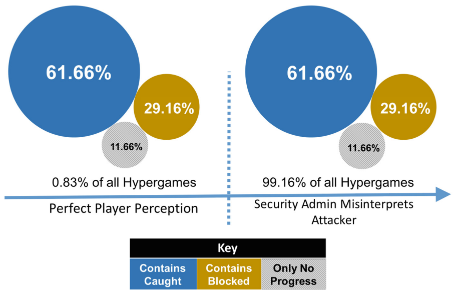


Fig. 1. Relative occurrences of equilibrium: left side, there is perfect perception between the $\mathcal{S}$ and $\mathcal{A}$; right side, $\mathcal{S}$ misinterprets the preference order of $\mathcal{A}$. The most prevalent outcome is "Caught" indicating that $\mathcal{S}$ should deploy EPW Redirect.

4.7 Risk-Tolerant/Averse Attacker Analysis

A class of misinterpretation scenarios that are of practical interest is where $\mathcal{A}$ is more or less tolerant of risk than what $\mathcal{S}$ perceives. We define four $\mathcal{A}$ preferences: two risk-tolerant and two risk-averse (Table 4). The preference lists vary among these levels by where *Blocked* is placed. There are six total outcomes in $\mathcal{A}$'s preference list, of which *Success* is always at the top (position 1) and *Caught* is at the bottom (position 6). We define a risk-tolerant $\mathcal{A}$ as having *Blocked* in positions 2 or 3 in the preference list. Of these two levels, the first column corresponds to the more risk-taking $\mathcal{A}$. We define a risk-averse $\mathcal{A}$ as having *Blocked* in positions 4 or 5 in the preference list. Of these two levels, the fourth column corresponds to the more risk-averse $\mathcal{A}$. We define an $\mathcal{S}$ who *over-estimates* the attacker's willingness to take risks as cases where $\mathcal{S}$ places the perceived $\mathcal{A}$'s *Blocked* preference higher (*i.e.*, more desirable from $\mathcal{A}$'s point of view) than the true $\mathcal{A}$'s *Blocked* preference. Likewise, $\mathcal{S}$ may *under-estimate* the attacker's willingness to take risks by placing the perceived $\mathcal{A}$'s *Blocked* preference higher than the true $\mathcal{A}$'s *Blocked* preference. Both of these are cases of *misinterpretation* by $\mathcal{S}$ of $\mathcal{A}$'s preferences. More formally, a $\mathcal{S}$'s under-estimation of $\mathcal{A}$'s willingness to take risks is defined

Table 4. Definitions of Risk-tolerant attackers (RT) and risk-averse attackers (RA).

RT $\mathcal{A}$ 1	RT $\mathcal{A}$ 2	RA $\mathcal{A}$ 1	RA $\mathcal{A}$ 2	Outcomes	$\mathcal{S}$ Action	$\mathcal{A}$ Action
1	1	1	1	1 Breach	Slt. Hash	Use PW
2	3	4	5	2 Blocked	EPW Alrm	Use PW
3	2	2	2	3 No Prg. (3)	EPW Redirect	Fnd. Alt.
4	4	3	3	4 No Prg. (2)	EPW Alarm	Fnd. Alt.
5	5	5	4	5 No Prg. (1)	Slt. Hash	Fnd. Alt.
6	6	6	6	6 Caught	EPW Redirect	Use PW

as $o_{\mathrm{i}} \in p(\mathcal{S}, \mathbf{Pref}_{\mathcal{A}})$ and $o_{\mathrm{j}} \in p(\mathcal{A}, \mathbf{Pref}_{\mathcal{A}})$, where $o_{\mathrm{i}} = o_{\mathrm{j}}$ and $i < j$. Similarly, an over-estimation is defined as $o_{\mathrm{i}} = o_{\mathrm{j}}$ and $i > j$.

Figure 2 shows the aggregated equilibria for all 57,600 hypergames. We categorize these into five buckets, where, from left to right in the figure, $\mathcal{S}$ respectively under-estimates, correctly estimates, and over-estimates $\mathcal{A}$'s risk tolerance. The percentages are calculated separately for all five buckets. If $\mathcal{S}$ correctly perceives $\mathcal{A}$, the equilibrium most likely contains a *Caught* outcome. If $\mathcal{S}$ over-estimates $\mathcal{A}$'s willingness to take risks, the *Blocked* outcomes are observed most frequently in the equilibria. Alternatively, if $\mathcal{S}$ underestimates $\mathcal{A}$'s willingness to take risks, the equilibria contains either the *Blocked* or *Caught* outcomes. Note that if $\mathcal{S}$ underestimates, equilibria with only *No Progress* outcomes are not observed.

The results show if $\mathcal{S}$ wishes to catch $\mathcal{A}$ (the most preferred outcome for $\mathcal{S}$) then she should invest in threat intelligence. The center bucket shows if $\mathcal{S}$ knows $\mathcal{A}$'s risk tolerance, she can deploy the EPW with the honeynet configuration and potentially catch attackers who use stolen credentials. An over-estimation of $\mathcal{A}$'s willingness to take risks shows that there are instances when both the EPW configurations fail to *Catch* or *Block* $\mathcal{A}$—the attacker prefers to find alternative attack targets. Recall that $\mathcal{A}$ knows the preferences of $\mathcal{S}$, so the results reflect $\mathcal{A}$ is making the most rational decision without misperception.

5 Discussion and Related Work

The hypergames explored consist of a single snapshot in time. The players in each hypergame perceive each player's actions and rationalize the most optimal action to execute. However, executing an action reveals information regarding the strategies of players. Over time, a player's perception of the conflict may change, leading to a different equilibrium action. For instance, $\mathcal{S}$ may learn about $\mathcal{A}$'s motivations after redirecting her into a honeynet. Likewise, $\mathcal{A}$ may learn about the characteristics of the honeynet and potentially detect or escape the honeynet. The new knowledge gained for the players should then be reflected in the hypergames, which we do not consider here. Imamverdiyev [8] considers the use of hypergames to model information security conflicts. The work presents an abstract two-level hypergame with a single attacker and defender.

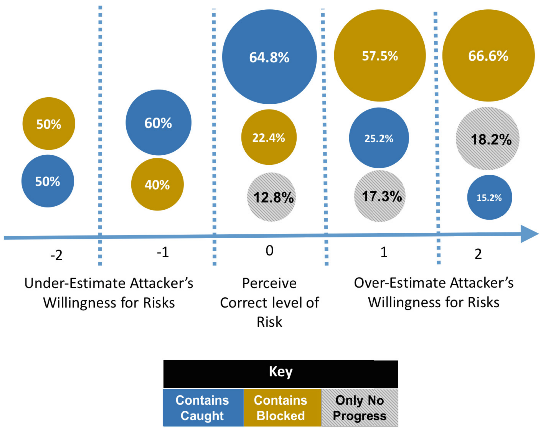


Fig. 2. Equilibriums where the Security Administrator ($\mathcal{S}$)'s perception of the Attacker ($\mathcal{A}$) varies. The chances for $\mathcal{S}$ to block $\mathcal{A}$ is higher under the over-estimation scenarios.

The hypergame only considers a conflict where the attacker is not aware of *one* of the actions available to the defender; our analysis space is thus more substantial. Gibson's thesis [7] 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. HNF is a richer model than ours (*e.g.*, it allows for evolution of beliefs) but it also requires hard-to-quantify information such as utility functions and probabilities for the preferences of each player's choice.

6 Conclusion

This paper presents a technique to analyze cyber conflicts that incorporate deceptive defense mechanisms. The work applies principles from the theory of hypergames, which extends classical game theory by incorporating misperceptions among the players. We discuss the kernels necessary to model player misperception in deceptive defense strategies and then demonstrate their use by modeling the ErsatzPassword scheme, a security control that protects hashed passwords against brute force password-cracking. The hypergames model uses two configurations of the ErsatzPassword scheme, triggering an alarm if an attacker uses a cracked password or redirecting the attacker to a honeynet, for attack analysis and attribution. We analyze 57,600 hypergame configurations under various levels of misperception and attacker strategies. The analysis provides insights on the effectiveness of incorporating deceptive mechanisms to protect stored password credentials. In particular, we found that the scheme works well against both risk-tolerant and risk-adverse attackers. We also showed that the scheme is effective in blocking or catching attackers even under various levels of misperception. We believe that our contributions will serve as a basis to analyze other cyber conflicts that incorporate deceptive defense strategies.

Acknowledgements. The authors acknowledge Jeffery Avery and Paul Wood for their insightful discussion on early versions of this work. Special thanks to the US National Science Foundation for supporting this research through EAGER Grant #1548114.

References

1. Almeshekah, M.H., Gutierrez, C.N., Atallah, M.J., Spafford, E.H.: Ersatzpasswords: ending password cracking and detecting password leakage. In: ACSAC, pp. 311–320 (2015). http://orcid.org/10.1145/2818000.2818015
2. Almeshekah, M.H., Spafford, E.H.: Planning and integrating deception into computer. In: Proceedings of the New Security Paradigms Workshop (NSPW) (2014)
3. Bennett, P.G., Dando, M.R.: Complex strategic analysis: a hypergame study of the fall of France. J. Oper. Res. Soc. **30**(1), 23–32 (1979). https://doi.org/10.1057/jors.1979.3
4. Bennett, P.G.: Toward a theory of hypergames. Omega **5**(6), 749–751 (1977)
5. Brumley, L.: HYPANT: a hypergame analysis tool Monash University. Ph.D. thesis, Monash University (2003)
6. Fraser, N., Hipel, K.: Conflict analysis: models and resolutions. North-Holland series in system science and engineering (1984)
7. Gibson, A.: Applied hypergame theory for network defense. Ph.D. thesis, Air Force Institue of Technology
8. Imamverdiyev, Y.: A hypergame model for information security. Int. J. Inf. Secur. Sci. **3**(1), 148–155 (2014)
9. Juels, A., Rivest, R.L.: Honeywords: making password-cracking detectable. In: Proceedings of Computer and Communications Security, CCS 2013, pp. 145–160 (2013)
10. Píbil, R., Lisý, V., Kiekintveld, C., Bošanský, B., Pěchouček, M.: Game theoretic model of strategic honeypot selection in computer networks. In: Grossklags, J., Walrand, J. (eds.) GameSec 2012. LNCS, vol. 7638, pp. 201–220. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34266-0_12
11. Singh, A., Bu, Z.: Hot Knives Through Butter: Evading File-based Sandboxes. FireEye (2014)
12. Spitzner, L.: Honeypots: Tracking Hackers. Wesley, Boston (2002)
13. Takahashi, M.A., Fraser, N.M., Hipel, K.W.: A procedure for analyzing hypergames. Eur. J. Oper. Res. **18**, 111–122 (1984). https://doi.org/10.1016/0377-2217(84)90268-6
14. Wang, M., Hipel, K.W., Fraser, N.M.: Modeling misperceptions in games. Behav. Sci. **33**(3), 207–223 (1988)
15. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: deceptive files for intrusion detection. In: IEEE SMC Information Assurance Workshop, pp. 116–122, June 2004. http://orcid.org/10.1109/IAW.2004.1437806