The Credential is Not Enough: Deception With Honeypots and Fake Credentials

[1]Sonia Cromp$^{[0009-0002-6556-3317]}$, [2]Mark Bilinski$^{[0000-0003-3635-2027]}$, [2]Ryan Gabrys$^{[0000-0002-9197-3371]}$, and [1]Frederic Sala$^{[0000-0003-0379-2827]}$

[1] University of Wisconsin-Madison, Madison WI 53706, USA
[2] Naval Information Warfare Center Pacific, San Diego, CA, USA
cromp@wisc.edu, mark.bilinski.civ@us.navy.mil,
ryan.c.gabrys.civ@us.navy.mil, fredsala@cs.wisc.edu

Abstract. Honeypots are a classic cyber-deceptive technique that allows a defender to add false information into the system in an effort to deter/delay/distract potential attackers. However, the effectiveness of honeypots is dependent on their design along with the environment into which they are deployed. In this work, we consider the scenario where there is a collection of honeypots along with a set of fake credentials. In the first part of the paper, we uncover fundamental bounds that relate to how long these deceptive elements remain effective. In the second part of the paper, we take our results one step further and analyze a two-person game where the attacker attempts to access desired resources within a system according to a preference model and the defender attempts to design honeypots that slow attacker progress. While prior work has demonstrated the defender's ability to learn attacker preferences by observing the attacker's actions, we enrich both parties' action spaces by allowing the attacker to query whether a server is real or honeypot and by allowing the defender to choose between honeypots that better reveal attacker behavior, or honeypots that exploit current knowledge of attacker behavior. In this setting, we provide and analyze optimal strategies for the attacker, along with a learning bound for and simulation of defender strategies.

Keywords: Adversarial game · Cyber-deception · Active learning

1 Introduction

Cybersecurity is an area of great importance for any organization, whether in industry, government, military, or other settings [4, 13]. Despite increased focus, systems meant to provide security face two challenges. First, many techniques offer a plausible approach to defense, but lack provable security guarantees, or offer them in highly narrow settings. Second, the space is dynamic, with frequent appearances of new cyberattacks and defenses, as well as combinations of extant techniques. These obstacles must be overcome for organizations to have confidence in their cybersecurity. As a result, we are interested in studying settings that have rich attack models and defense strategies with provable guarantees.

Among the most important such settings involve the use of *deception* for providing means of defense [1]. For defenders, such deception can be expressed through *honeypots*—false information or devices that are added within an ecosystem to slow down or block attackers [6, 3]. Honeypot-based systems naturally admit a game-theoretical formulation [12], but there is a wide variety of such settings. Many of the most realistic settings have not yet been addressed.

We focus on cyberattack settings where the attacker's interests can be described via a preference model, as in [6, 7]. In contrast to earlier work, the attacker may choose between two attack vectors. In one approach, the attacker attempts to use accessible credentials to gain access to a system, allowing the attacker to access resources within the system of the most interest to the attacker. In the other, the attacker performs random attacks, which may be faster than relying on credentials but is less targeted towards the specific resources which the attacker would like to access. To prevent these attacks, the defender has the ability to use *deception*—setting up honeypot systems and creating false sets of credentials in order to slow down or prevent the attacker from breaking in. This work studies the fundamentals of such multiple-attack scenarios.

2 Related Work

A rich line of literature has studied deception in the context of cybersecurity. In such work, a typical scenario involves a two-player game between an *attacker* and a *defender*. The goal of the attacker is to access some system resources. The defender can prevent this from happening (or slow it down) by presenting honeypots—fake resources—to the user. Such games have been extensively studied [5, 11, 2], usually in the form of zero-sum games.

A closely related line of work involves learning from preferences. For example, attackers may have particular interests in accessing certain resources. Defenders therefore seek to learn these preferences. Doing so enables them to potentially deploy honeypots and other means of deception. Such work, including [6], obtain learning bounds on the number of *interactions* required for defenders to learn a sufficiently accurate estimate of the attacker model.

This work studies richer scenarios with multiple attack vectors. The first of these is inspired by [15]; here the attacker seeks to obtain credentials via querying servers, but must deal with honeypot servers that do not tell the truth (like the *spies* in [15]). The second involves a preferential model as in [6]. In contrast to this work, we tackle two additional factors. The first is that defenders must handle the two attack factors. The second is that the environment is changed by deployment of honeypots—complicating learning attacker preferences.

3 Setup and Structure

In this work, we consider a system of s servers, of which ℓ are honeypots and μ are real, and c credentials, of which ρ are fake. Let $\mathcal{S}$ denote the set of indices of all servers, $\mathcal{S}_h$ the set of honeypots and $\mathcal{S}_r$ the set of real servers. While the defender knows the identities of all servers and credentials, the attacker discovers them over the course of the game. Instead, the attacker only knows that there are *at most* t_H honeypot servers and t_F false credentials.

The i-th server is described by a feature vector $x_i \in \mathbb{R}_+^d$, which intuitively may be thought of as the embedding of various features which describe the server, such as name and location. The attacker has a vector of preferences within the same embedding space, denoted $w \in \mathbb{R}_+^d$. For instance, the attacker may be interested in servers located in a particular region. Both attacker and defender observe x_i for each server i, while only the attacker knows their own preference vector w. We say that the attacker's relative "interest" in a server i is proportional to $w^T x_i$. We refer to the α real servers with highest attacker interest as the attacker's *desired servers* $\mathcal{S}_d$. The game ends when the attacker performs β accesses to each of the α desired servers.

The attacker may choose one of two strategies: performing queries to learn the identities of each server and credential, then accessing the desired servers over the following $\alpha\beta$ timesteps, or performing random attacks. If the attacker elects to perform identity queries, the attacker selects a server $i \in [s]$ and credential $j \in [c]$, then poses a question of the form "Server i, is credential j real (R) or fake (F)?" in each timestep until learning all identities. If instead they perform random attacks, they access the i-th server with probability

$$p_i = \frac{\exp(w^T x_i)}{\sum_{j \in \mathcal{S}} \exp(w^T x_j)}.$$

One attack is carried out in each timestep until all $\alpha\beta$ goal accesses are performed. When attacking server i, the attacker gains access to a portion of its private data if i is real; otherwise, the attacker discovers that i is a honeypot.

The attacker gains 1 point each time they complete one of the $\alpha\beta$ desired accesses and loses 1 point each time they attack a honeypot – upon accessing or querying server i for the j-th time in timestep t, the attacker reward is

$$R^t(i,j) = \begin{cases} -1 & \text{if attack and } i \in \mathcal{S}_h \\ 1 & \text{if attack, } i \in \mathcal{S}_d \text{ and } j < \beta \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Rewards are also multiplied by a discount factor Δ^t in timestep t, for $0 < \Delta \leq 1$. While the identity query strategy avoids the random access strategy's risk of penalization from attacking honeypots, this strategy's accesses may be performed later than random strategy accesses, and thus encounter lower rewards due to the discount Δ.

Meanwhile, the defender chooses the x-vectors of their ℓ honeypots, selecting between honeypot placements that allow the defender to learn a better estimate $\tilde{W}$ of w and placements that maximize honeypot attack probability. The defender receives reward $-R^t(i,j)$ in timestep t, forming a zero-sum game.

The remainder of this work is structured as follows. First, we consider the attacker's two actions in Section 4. Sections 4.1 and 4.2 discuss the amount of queries needed for the attacker to learn servers' and credentials' identities prior to performing the $\alpha\beta$ desired accesses, while Section 4.3 analyzes the random attack strategy, determining the the expected number of random attacks needed to perform the $\alpha\beta$ desired accesses. We then discuss the game from the defender's point of view in Section 5. This section compares honeypot placement strategies, establishing a learning bound on the defender's ability to learn w (and thereby more effectively target honeypots that lure the attacker) and demonstrating the defender's strategies in a simulated game environment.

4 Attacker Strategy

In this section, the defender is fixed such that the x-vectors of honeypots are held constant. We first study the attacker's identity query strategy, where the attacker asks a query of the form, "Server i, is credential j real (R) or fake (F)?" in each timestep until learning all identities and performing the desired accesses.

We assume that our ℓ honeypots are low-interaction, which implies that they will always respond with the answer F, whereas the remaining μ real servers will respond truthfully with either R or F, depending on whether the credential is real or fake, respectively. As such, a response of R only occurs when i and j are both real; otherwise the response is F. Only once that the attacker has learned the identities of all s servers and c credentials can the attacker perform their desired $\alpha\beta$ accesses. Under this setup, Section 4.1 determines the worst-case maximum number of identity queries until the attacker can perform their desired accesses, while Section 4.2 finds the average case number of queries. A greater the number of timesteps prior to the desired accesses will result in a greater time penalty on the attacker's positive rewards during the desired accesses in (1).

Lastly, Section 4.3 considers the attacker's random attack strategy.

4.1 Determining Server and Credential Identities - Worst Case

We refer to the property of a credential being real or fake as the "type" of the credential. Analogously, we refer to the property of a server being a real server or a honeypot as the identity of the server. We will study the following problems:

1. How many questions are necessary and sufficient to determine the type of each of the credentials?
2. How many questions are necessary and sufficient to determine the identity of each of the servers?

We note that as a result of the symmetry of the game setup, the solutions to both of these problems is the same, and consequently we will focus on the first problem. For shorthand, we will refer to the solution to 1) above as $Q^*(s, c, t_H, t_F)$ so that after $Q^*(s, c, t_H, t_F)$ questions, the attacker will always be able to determine the type of each of the c credentials. For notational convenience, when the parameters s, c, t_H, t_F are understood, we will abbreviate $Q^*(s, c, t_H, t_F)$ as Q^*.

Our main result is to show that, for the case where $t_H \geq t_F - 1$, at most $Q^* = t_H + t_F - 1 + c$ questions (or queries) are necessary and sufficient. This implies that, under the scenario where there are roughly as many fake credentials as honeypots, both types of deception appear to impact the attacker nearly equally. However, when $t_H << t_F - 1$, this trend does not hold. We show in Section 4.1 that at most only $c + \mathcal{O}(\sqrt{t_F})$ queries are needed for the case where t_H is a constant. From a defender point of view, this implies that increasing the number of fake credentials (typically easier) has a similar effect as increasing the number of honeypots, provided they are roughly comparable in number. In this setting, introducing either one additional fake honeypot or one additional fake credential requires the attacker ask one additional query. When $t_H << t_F - 1$, introducing a honeypot appears to be significantly more impactful than a fake credential. For this setting, introducing one honeypot requires the attacker ask at most $\sqrt{t_F}$ additional queries whereas increasing the number of fake credentials in certain cases only increases the overall number of queries by at most a constant.

Upper Bound on Q^* In the following, we show that $Q^* \leq t_H + t_F - 1 + c$. This result is stated more formally in Lemma 1. Afterwards, for the setting where $t_H < \sqrt{t_F}$, we show that this quantity is at most only $\sqrt{t_F}\,(2t_H + 1) + 1 + c$. We begin with the following observation, which we state as a claim for clarity.

Claim 1. *If server i responds R when queried about credential j, then credential j is real and server i is a real machine.*

The basic idea behind our first approach, which shows $Q^* \leq t_H + t_F - 1 + c$, is to ask as few questions as possible in order to produce an answer of R to one of our questions. Afterwards, we will query this server about the identity of each of the other $c - 1$ credentials. The output of the following procedure will be the set $\mathcal{C}_R \subseteq [c]$, which we will later show, contains the identity of the real credentials.

Initialize $\mathcal{C}_R = \emptyset$. We proceed as follows:

- Step 1: Generate $t_H + t_F + 1$ pairs of elements say (i_1, j_1), (i_2, j_2), $\ldots$, $(i_{t_H+t_F+1}, j_{t_H+t_F+1})$ where $|\{i_1, \ldots, i_{t_H+t_F+1}\}| = |\{j_1, \ldots, j_{t_H+t_F+1}\}| = t_H + t_F + 1$. (i_k, j_k) represents the question: "Server i_k, is credential j_k R or F?"
- Step 2: Starting with question (i_1, j_1), the attacker asks each of the questions in the list generated at step 1. Suppose that k^* is the first question that generates the response R. Claim 1 implies that server i_{k^*} is a real machine and also that credential j_{k^*} is a real credential. Add j_{k^*} to the set $\mathcal{C}_R$.
- Step 3: If $k^* = t_H + t_R + 1$, then add $[c] \setminus [k^*]$ to $\mathcal{C}_R$. Otherwise, ask server i_{k^*} about the credentials $\{j_{k^*+1}, \ldots, j_c\}$. For any $k \in [c] \setminus [k^*]$, if server i_{k^*} responds R, then add j_k to the set $\mathcal{C}_R$.
- Step 4: For $k \in \{1, 2, \ldots, k^* - 1\}$ ask server i_{k^*} about credential j_k. For any $k \in [k^* - 1]$, if server i_{k^*} responds R, then add j_k to the set $\mathcal{C}_R$.

We begin with the following observation.

Claim 2. *In step 2, $k^* \leq t_H + t_F + 1$. Furthermore, if $k^* = t_H + t_F + 1$, then the credentials $\{j_{t_H+t_F+1}, j_{t_H+t_F+2}, \ldots, j_c\}$ are real.*

Note that according to Claim 1, server i_k^* has correctly identified $c - k^* + 1$ of the m credentials by the end of step 3, which implies that we have not asked server i_k^* about the identity of $k^* - 1$ credentials and in particular about the credentials $\{1, 2, \ldots, k^* - 1\}$. We now arrive at the main result of this section.

Lemma 1. *For any $j \in [c]$, $j \in \mathcal{C}_R$ if and only if credential j is real. The number of questions asked in steps 1-4 is at most $t_H + t_F - 1 + c$ provided $c, s \geq t_H + t_F + 2$.*

Proof. The fact that the set $\mathcal{C}_R$ contains the identities of each of the real credentials follows immediately from the previous discussion. The number of questions asked at steps 1) and 4) are k^* and $k^* - 1$, respectively. Letting q_3 denote the number of questions asked at step 3, the total number of questions is:

$$2k^* - 1 + q_3. \tag{2}$$

Here, there are two cases to consider. If $k^* < t_H + t_F + 1$, then $q_3 \leq c - k^*$ and (2) is at most $c + t_H + t_F - 1$ as desired. Otherwise, if $k^* = t_H + t_F + 1$, then $q_3 = 0$ and (2) is $2(t_H + t_F + 1) - 1$.

Although the previous approach has the advantage of working for a wide range of parameter choices for t_H, t_F, it is far from optimal in many cases. In particular, for the setting where t_H is a constant with respect to t_F, it turns out that a better strategy exists which requires $c + \mathcal{O}(\sqrt{t_F})$.

For simplicity, we assume that t_F is a square, although it is straightforward to extend to a more general setting. The attacker first chooses a set of t_F credentials, denoted $j_1, \ldots, j_{t_F}$ and partitions this set of credentials into $\sqrt{t_F}$ groups denoted:

$$\begin{aligned}
\mathcal{J}_1 &= \{j_1, \ldots, j_{\sqrt{t_F}}\}, \\
\mathcal{J}_2 &= \{j_{\sqrt{t_F}+1}, \ldots, j_{2\sqrt{t_F}}\}, \\
&\vdots \\
\mathcal{J}_{\sqrt{t_F}} &= \{j_{t_F - \sqrt{t_F}+1}, \ldots, j_{t_F}\}.
\end{aligned}$$

Initialize $\mathcal{C}_R = \emptyset$. We proceed as follows.

- Step 1: Generate t_F pairs of questions (i_1, j_1), $(i_1, j_2), \ldots, (i_1, j_{\sqrt{t_F}})$, $(i_2, j_{\sqrt{t_F}+1})$, $(i_2, j_{\sqrt{t_F}+2})$, $\ldots, (i_2, j_{2\sqrt{t_F}})$, $\ldots$, $(i_{\sqrt{t_F}}, j_{t_F})$ where $i_1, \ldots, i_{\sqrt{t_F}}$ are $\sqrt{t_F}$ distinct hosts. Starting with (i_1, j_1) ask each of these t_F queries.
- Step 2: Generate an additional (at most) $t_H\sqrt{t_F} + 1$ queries of the form (i, j) such that for each such query the following holds: (i) Host i has not been queried previously and (ii) Credential j has not appeared in any previous queries. If any any point we receive the response T, we proceed to Step 3).
- Step 3: At this point, we have received the response T. Suppose the query which receives this response is (i_k, j_k). Insert j_k into $\mathcal{C}_R$. We next ask the following $\sqrt{t_F}$ questions: $(i_1, j_k), (i_2, j_k), \ldots, (i_{\sqrt{t_F}}, j_k)$. Let $\mathcal{I}_F$ denote the set of servers whose response is F. Go to Step 4).

– Step 4: For each $v \in [\sqrt{t_F}]$, if $i_v \in \mathcal{I}_F$, then for each $j \in \mathcal{J}_v$, we perform the query (i_k, j) and if the response is T, then we add credential j to $\mathcal{C}_R$.
– Step 5: For each credential j outside the set $j_1, j_2, \ldots, j_{t_F}$, we perform the query (i_k, j) and if the response if T, we add j to the set $\mathcal{C}_R$.

Lemma 2 states that the maximum number of queries necessary to determine the type of each of the c credentials. First, we present an illustrative example.

Example 1. Suppose $t_H = 1$, and $t_F = 16$. We assume in the following that i_1 is a honeypot and $j_5, j_6, \ldots, j_{16}, j_{17}, j_{18}, j_{19}$ are fake credentials. According to the previous procedure, in step 1 suppose we formulate $t = 16$ queries:

$$\begin{bmatrix} (i_1, j_1) & (i_2, j_5) & (i_3, j_9) & (i_4, j_{13}) \\ (i_1, j_2) & (i_2, j_6) & (i_3, j_{10}) & (i_4, j_{14}) \\ (i_1, j_3) & (i_2, j_7) & (i_3, j_{11}) & (i_4, j_{15}) \\ (i_1, j_4) & (i_2, j_8) & (i_3, j_{12}) & (i_4, j_{16}) \end{bmatrix}. \tag{3}$$

More specifically the attacker will first ask the query (i_1, j_1) followed by (i_1, j_2) and so on until we have asked all 16 queries. Since i_1 is a honeypot and $j_5, \ldots, j_{20}$ are fake, it follows that the response to each of these queries is F.

For step 2, assume the next 5 queries are $(i_5, j_{17}), (i_6, j_{18}), (i_7, j_{19}), (i_8, j_{20}), (i_9, j_{21})$. We will receive the response T on the second to last query since i_8 is not a honeypot and j_{20} is a real credential. At this point we add j_{20} to our list of real credentials and after the query (i_8, j_{20}) we will proceed to the third step.

At step 3), we ask $(i_1, j_{20}), (i_2, j_{20}), (i_3, j_{20}), (i_4, j_{20})$. (i_1, j_{20}) returns F and the others return T. Because of the T responses, each of the credentials in the last 3 columns of (3) are of type fake. Next we proceed to step 4).

At step 4), in order to determine the identity of the credentials in the set $j_1, \ldots, j_{20}$ it suffices to query the host i_8 (which we know is not a honeypot) about each of the credentials in the first column of (3). Since each of these credentials are by assumption real, it follows that $j_1, \ldots, j_4$ will be added to $\mathcal{C}_R$.

Finally, in step 5) we add each credential outside $\{j_5, j_6, \ldots, j_{19}\}$ to $\mathcal{C}_R$. Note that this step requires $c - 16$ additional queries to i_8. In all, for each subsequent step we have performed respectively 16, 4, 4, 4, and $c - 16$ queries. In total $c + 12$ queries which, for this choice of parameters, is less than or equal to $c + \sqrt{t_F}(2t_H + 1) + 1 = c + 4 \cdot (2 + 1) + 1 = c + 13$ as claimed.

Lemma 2. *For the setup where $t_H < \sqrt{t_F}$ and where $c, s > t_H + t_F$, the number of queries to determine the type of each credential is at most $Q^* \leq \sqrt{t_F}\,(2t_H + 1) + 1 + c$.*

Lower Bound on Q^* We next turn to the question of optimality and we will show that for the case where $t_H \geq t_F - 1$, at least $t_H + t_F - 1 + c$ questions are also necessary. We can consider our setup as a game, that is being played between an attacker and Mother Nature (MN) where the attacker is allowed to ask any questions of the same form as described earlier and Mother Nature is allowed to fix the identities and types of each of the servers and credentials.

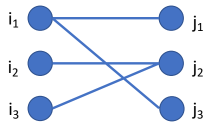
The main challenge, which we focus on now, is to establish the result for the case where $t_H = t_F - 1$. Our key technical result is described in the next lemma.

Lemma 3. *Suppose the attacker asks $t_H + t_F - 1$ queries and, among these queries, there are exactly c_0 credentials contained across the $t_H + t_F - 1$ queries. Then, in order to determine the identity of these c_0 credentials, there exists an assignment of identities to credentials and servers by MN such that c_0 additional queries are necessary.*

Under our setup, we assume that MN always assigns at most t_H honeypots and at $t_F - 1$ fake credentials to ensure that the first $t_H + t_F - 1$ queries receive the response F where during these first $t_H + t_F - 1$ queries at least $t_H + 1$ servers are queried and at least $t_H + 1$ credentials are also queried. Note that this scenario is always indeed possible since it can be the case that the first t_H hosts queried are honeypots and the following $t_F - 1 = t_H$ credentials queried are of type fake. Using the result stated in the previous lemma, we will show in Theorem 1 that an additional c_0 queries are necessary to determine the first c_0 credentials that were queried along with an additional $c - c_0$ queries (which each pertain to credentials outside the first c_0 queried), implying a total of $t_H + t_F - 1 + c_0 + (c - c_0) = t_H + t_F - 1 + c$ queries are necessary.

In order to tackle this problem, we will visualize the first $t_H + t_F - 1$ queries by means of edges in a bi-partite graph where the vertices on the left side of this graph, which we denote as $V_{\mathcal{S}}$, represent each of the servers queried in the first $t_H + t_F - 1$ queries and the vertices on the right hand side of the graph, which we denote as $V_{\mathcal{C}}$, represent the credentials queried in the first $t_H + t_F - 1$ queries. There exists an edge between vertex i on the left side of the graph and vertex j on the right hand side of the graph if the attacker asks the question (i, j). We illustrate this setup by means of the next example. For shorthand, we refer to this graph as the ***question graph*** $\mathcal{G}$ for the game.

Example 2. Suppose the attacker asks $(i_1, j_1), (i_2, j_2), (i_1, j_3), (i_3, j_2)$. Then, the question graph $\mathcal{G}$ that represents this sequence of questions is shown below:



For our analysis, we assume that the first $t_H + t_F - 1$ queries, which by assumption all have received the response F, are fixed before the start of the game. We say that the question sequence $\mathcal{Q}$ *belongs to* $\mathcal{G}$ if for every query in $\mathcal{Q}$ either the server queried or the credential queried about (or both) are contained in $\mathcal{G}$ and we represent this as $\mathcal{Q} \in \mathcal{G}$. As will be discussed in Claim 6, we assume that when a query contains either a host or credential *not* represented in $\mathcal{G}$, then their identity is not labeled F. Conceptually, the questions in $\mathcal{Q}$ are questions that are asked by an attacker (after the initial $t_H + t_F - 1$ queries) that can be used to recover the identity of each credential in $\mathcal{G}$.

The graph $\mathcal{G}$ will always have $t_H + t_F - 1$ edges. Given the graph $\mathcal{G}$, we can recast our problem as a labeling game for MN where she can label at most t_H vertices in $V_\mathcal{S}$ to be F (which means they're honeypots) and at most t_H vertices in $V_\mathcal{C}$ to be F (which means the corresponding credential is of type fake). In order to make this a binary labeling, we will assume the other vertices that are not labeled F are labeled R. The goal will be to show that for any given $\mathcal{G}$ and $\mathcal{Q}$, there exists a *labeling procedure*, denoted by the function L, which takes as input $\mathcal{G}$ and $\mathcal{Q}$ and outputs a labeling $L(\mathcal{G}, \mathcal{Q})$ such that:

1. The labeling $L(\mathcal{G}, \mathcal{Q})$ is ***consistent*** - This means that for any edge in the question graph $\mathcal{G}$ at least one vertex in that edge is labeled F under $L(\mathcal{G}, \mathcal{Q})$.
2. The labeling procedure $L(\mathcal{G}, \mathcal{Q})$ is ***robust*** - If $|\mathcal{Q}| < |V_\mathcal{C}|$ it is not possible for the attacker to ascertain the identity of each credential in $\mathcal{G}$ by asking the queries in $\mathcal{Q}$ and $\mathcal{G}$.

Note that in order to satisfy the consistency constraint, for any edge $(i, j) \in E_\mathcal{G}$ (where $E_\mathcal{G}$ represents the edge set for the question graph $\mathcal{G}$), either i is labeled F or j is labeled F or both. Note also that if i is labeled R and $(i, j) \in E_\mathcal{G}$, then it follows under our setup that j must be labeled F and vice versa.

Next, we introduce notation addressing robustness. Let $\mathcal{D}$ be a decoding rule such that given $\mathcal{Q}$ along with a sequence of responses F_R to each of the queries in $\mathcal{Q}$, the output of $\mathcal{D}$ is the set of credentials whose identities are known. Given an assignment of identities to the vertices in $\mathcal{G}$, the response from each of the servers is deterministic. We capture this relationship by letting F_R be a function which takes as input the assignment of identities to servers and credentials. The robustness property of L requires that for any sequence of queries $\mathcal{Q} \in \mathcal{G}$ of cardinality less than $|V_\mathcal{C}|$, there exists a labeling $L(\mathcal{G}, \mathcal{Q})$ such that

$$|\mathcal{D}\left(\mathcal{Q}, F_R\left(L(G, \mathcal{Q})\right)\right)| < |V_\mathcal{C}|. \tag{4}$$

Thus, the goal will be to show that there exists a labeling procedure L, which is consistent and robust. With an abuse of notation, we will also say that a labeling for a particular graph $\mathcal{G}$ and a particular set of queries $\mathcal{Q}$ is consistent with respect to $\mathcal{G}$ if each edge in $\mathcal{G}$ has a vertex labeled F. Furthermore, for a specific question sequence $\mathcal{Q}$, we will say that the labeling is robust with respect to $\mathcal{G}, \mathcal{Q}$ if (4) holds provided $|\mathcal{Q}| < |\mathcal{V}_\mathcal{C}|$.

The next three claims, whose proofs are deferred for the extended version of the paper, will be useful in our subsequent derivations and in particular will be invoked in the proof of Lemma 4. Let $L_{t_H, t_F - 1}(\mathcal{G}, \mathcal{Q})$ be a labeling procedure that assigns at most t_H F labels to vertices in $V_\mathcal{S}$ and $t_F - 1$ F labels to vertices to vertices in $V_\mathcal{C}$. When it is clear from the context, the parameters $t_H, t_F - 1$ may be omitted from the notation for the labeling procedure L.

Claim 3. *Let $\mathcal{G}'$ be a question graph and $\mathcal{Q} \in \mathcal{G}'$ be a question sequence where $\mathcal{G}'$ has vertex set $V_\mathcal{S} \cup V_\mathcal{C}$ and $\mathcal{G}'$ has edge set $E_{\mathcal{G}'}$. Let $\mathcal{G} = \mathcal{G}' + e_1, e_2$ where $e_1 = (i_1, j_1), e_2 = (i_2, j_2)$ and where the degree of any vertex in $\mathcal{G}'$ is at most t_H and $j_1 \neq j_2$. If, for any sequence $\mathcal{Q}$ there exists a labeling $L_{t_H - 1, t_F - 2}(\mathcal{G}', Q)$,* *which is consistent and robust with respect to $\mathcal{G}', \mathcal{Q}$, then there exists a labeling $L_{t_H,t_F-1}(\mathcal{G}, \mathcal{Q})$ that is consistent and robust with respect to $\mathcal{G}, \mathcal{Q}$.*

Claim 4. *Suppose $t_H = t_F - 1$ and $\mathcal{G}$ is a question graph after $t_H + t_F - 1$ queries by the attacker where $|V_{\mathcal{C}}| \geq t_H + 1$, $|V_{\mathcal{S}}| \geq t_H + 1$, and at most one vertex in $V_{\mathcal{C}}$ has degree at least 2 and the remaining vertices in $V_{\mathcal{C}}$ have degree one. For any question sequence $\mathcal{Q} \in \mathcal{G}$, there exists a labeling that is consistent and robust with respect to $\mathcal{G}, \mathcal{Q}$.*

Claim 5. *Let $\mathcal{G}$ be a question graph where there exists $v^* \in V_{\mathcal{S}}$ with degree t_H and where each neighbor of v^* has degree exactly one. For any question sequence $\mathcal{Q} \in \mathcal{G}$, there exists a labeling that is consistent and robust with respect to $\mathcal{G}, \mathcal{Q}$.*

We now aim to prove Lemma 3 by induction on $t_H = t_F - 1$ and the next claim considers the base case. Recall that for now, we assume the degree of each vertex in $V_{\mathcal{C}}$ is less than $t_H + 1$ after the initial $t_H + t_F - 1$ queries. We will show later that when this restriction is removed the result still holds afterwards.

Claim 6. *Suppose $t_H = t_F - 1 = 1$, $\mathcal{G}$ is the question graph after $t_H + t_F - 1 = 2$ queries by the attacker where $|V_{\mathcal{C}}| \geq t_H + 1 = 2$ and $|V_{\mathcal{S}}| \geq t_H + 1 = 2$. Then, there exists a labeling procedure that is consistent and robust.*

Using the previous claim, we have the following lemma.

Lemma 4. *Suppose $t_H = t_F - 1 > 1$ and $\mathcal{G}$ is a question graph after $t_H + t_F - 1$ queries by the attacker where $|V_{\mathcal{C}}| \geq t_H + 1$, $|V_{\mathcal{S}}| \geq t_H + 1$, and the degree of any vertex in $\mathcal{G}$ is at most t_H. Then, for any question sequence $\mathcal{Q}$, there exists a labeling procedure that is consistent and robust.*

Proof. The proof will be by induction on t_H (and $t_F - 1$) where the base case was proven in Claim 6. Suppose the result holds for all $t_H, t_F - 1 \leq L$ and consider the case where $t_H = t_F - 1 = L + 1$. Let $\mathcal{G}'$ denote the question graph if we remove two edges (or queries) from $\mathcal{G}$ where $j_1 \neq j_2$. By the induction hypothesis, $\mathcal{G}'$ has a labeling procedure that is consistent and robust. If $\mathcal{G}'$ has any unconnected vertices, we remove those from the graph as well. For the case where there exists a vertex $v \in V_{\mathcal{C}}$ or $v \in V_{\mathcal{S}}$ with degree t_H then one of the edges removed from $\mathcal{G}$ must be adjacent to v. Because $|V_{\mathcal{C}}|, |V_{\mathcal{S}}| \geq t_H + 1$, there can be at most one vertex in $V_{\mathcal{C}}$ with a degree t_H and at most one vertex in $V_{\mathcal{S}}$ with degree t_H. Let $\mathcal{E} = \{(i_{r_1}, j_{r_1}), (i_{r_2}, j_{r_2})\}$ denote the set of two vertices that were removed from $\mathcal{G}$ to obtain $\mathcal{G}'$ where we require that $j_{r_1} \neq j_{r_2}$.

Next we consider the choice of the two edges in $\mathcal{E}$. If there is a choice of edges such that no vertices are isolated by their removal, then the vertex sets of $\mathcal{G}$ and $\mathcal{G}'$ are the same and the result follows from Claim 3. Next, we consider the case where at least one vertex in V_C appears in $\mathcal{G}$ but not $\mathcal{G}'$. Note that if this scenario occurs, then one of the following holds:

1) All the vertices in $V_{\mathcal{C}}$ have degree one,
2) There is exactly one vertex in $V_{\mathcal{C}}$ that has degree at least two and the remaining have degree one, or

3) $\mathcal{G}$ contains a vertex $v \in V_{\mathcal{S}}$ that has degree t_H and each neighbor of v has degree exactly one.

1) and 2) fall under the conditions of Claim 4. 3) is handled by Claim 5.

Next, we consider scenarios where there appears at least one vertex from V_S that appears in $\mathcal{G}$ but not $\mathcal{G}'$ as a result of removing edges $(i_{r_1}, j_{r_1}), (i_{j_2}, j_{r_2})$ where $j_{r_1} \neq j_{r_2}$. This result can be proven using similar logic to Claim 3. Suppose that $\mathcal{Q} \in \mathcal{G}$ is any valid question sequence $\mathcal{Q} \in \mathcal{G}$. Let $\mathcal{Q}' \subseteq \mathcal{Q}$ be such that $\mathcal{Q}' \in \mathcal{G}'$. Note that $\mathcal{Q}'$ is simply the result of removing at most two queries that involve either i_{r_1}, i_{r_2} and some credential outside $\mathcal{G}$. By the inductive assumption, there exists a labeling which is consistent and robust with respect to $\mathcal{G}'$, $\mathcal{Q}'$. Suppose $v \in \mathcal{G}'$ ($v \in \mathcal{G}$ as well) represents a credential which is unknown given the queries $\mathcal{G}'$, $\mathcal{Q}'$. If v is not adjacent to i_{r_1}, i_{r_2} and it is not equal to j_1, j_2, then setting i_{r_1}, j_{r_2} to be F results in a labeling which is robust and consistent with respect to $\mathcal{G}, \mathcal{Q}$. If v is adjacent to i_{r_1}, then setting i_{r_1} to be F and j_{r_2} to be F results in a robust and consistent labeling. Otherwise, if v is adjacent to i_{r_2} setting i_{r_2} to be F and j_{r_1} to be F results in a robust and consistent labeling.

Theorem 1. *In order to identify the identity of all c credentials, there exists a strategy for MN that always requires the attacker to ask at least $t_H + t_F - 1 + c$ questions when $t_H = t_F - 1$.*

Proof. Suppose first that there exists a vertex $v \in V_{\mathcal{S}}$ that is queried $t_H + 1$ times. Then in this case, we can assume MN labels v to be F. Furthermore, if MN labels the next $t_H - 1$ servers to be F, then it follows that at least $t_H + 1 + (t_H - 1) + c = c + t_H + t_F - 1$ queries are necessary. Similarly, if there exists a vertex $v \in V_{\mathcal{C}}$ that is queried $t_H + 1$ times then we can also assume v is labeled F and so given $t_H + 1$ queries the attacker will have identified exactly one credential. We can assume that MN labels the next t_H honeypots queried to be F, which implies in this case that an additional t_H queries are required. Finally, since at this point, the attacker has recovered the identity of only a single credential, the attacker needs to produce an additional $c - 1$ queries implying a total of $t_H + 1 + t_H + (c - 1) = c + t_H + t_F - 1$ queries are necessary in this case as well.

As a result of the logic in the previous paragraph, we can assume that credential is queried at most t_H times and each server is also queried at most t_H times, which means we can invoke Lemma 3. Suppose that given this setup, the first $t_H + t_F - 1$ queries each receive F from the attacker and the following query receives T. Then according to Lemmas 3 and 4, in order to determine the set of c_0 credentials asked about during the first $t_H + t_F - 1$ queries at least another c_0 queries are necessary. Since among these c_0 credentials there are at most $t_F - 1$ fake credentials, the attacker must also query each of the remaining $c - 1 - c_0$ credentials to determine their identity implying that a total of $t_H + t_F - 1 + 1 + c_0 + (c - 1 - c_0) = c + t_H + t_F - 1$ queries are necessary.

The next result follows by induction on t_H with base case in Theorem 1.

Corollary 1. *For $t_H \geq t_F - 1$, there exists a strategy for MN that always requires the attacker to ask at least $t_H + t_F - 1 + c$ questions.*

4.2 Determining Server and Credential Identities - Average Case

In this section, we derive an explicit expression for the expected number of queries that are sufficient to determine the identity of each of the fake credentials provided the strategy outlined in Lemma 1. Recall we have s servers, of which ℓ are honeypots along with c credentials among which there are ρ that are fake.

First, we compute the probability that we can use exactly $k+1$ questions to obtain the first T response. This means that the first k questions in our strategy either have a fake credential, a honeypot, or both. Suppose that of these, j of the k questions involve a fake credential, and the remaining $k-j$ involve a true credential. This constrains these last $k-j$ to use a honeypot, while the j questions involving a fake credential can have a true or honeypot server.

Next, we count the ways we can obtain the credentials. This is just $\binom{\rho}{j}\binom{c-\rho}{k-j}$, where the two coefficients select from the false and then true credentials. Next, we must allocate the servers. Recalling our constraint, the $k-j$ questions with a true credential must be allocated to $k-j$ of ℓ honeypots, while the remaining j servers can be either real or honeypots. Suppose u of these j servers are chosen as honeypots. This gives $\binom{\ell}{k-j+u}\binom{s-\ell}{j-u}$. Further, there are $\binom{j}{u}$ ways of ordering the two types of servers paired with fake credentials relative to each other.

Our focus thus far has been aimed at getting an F response for the first k questions. We need to now obtain T for the $k+1$st question. This means using remaining true credentials and true servers, of which we now have $(c-\rho-(k-j))\times(s-\ell-(j-u))$. Next we must sum over the possibilities j and u, so that our overall number of ways to select the questions is given by $B_k =$

$$\sum_{j=0}^{k}\sum_{u=0}^{j}\binom{\rho}{j}\binom{c-\rho}{k-j}\binom{\ell}{k-j+u}\binom{s-\ell}{j-u}\binom{j}{u}(c-\rho-(k-j))\times(s-\ell-(j-u)),$$

noting, of course, that there are cases where these coefficients reduce to zero simply because there are insufficient credentials or questions to allocate.

The probability that we obtain the first T response on the $(k+1)$-th question is simply the number of possible sequences of k F responses followed by one T response divided by the number of sequences of F responses of any length followed by one T response, i.e. $P_k = \sum_{i=0}^{t+\ell} \frac{B_k}{B_i}$

Further, after the first "true" answer at the $(k+1)$-th query, we must perform $c-1$ more queries to identify the remaining credentials. As such, $c+k$ questions are required in total which yields an expected number of questions

$$\mathbb{E}[Q] = \sum_{k=0}^{\rho+\ell}(c+k)P_k. \tag{5}$$

4.3 Attacker Random Access Strategy

We next analyze the game scenario in which the attacker repeatedly performs random accesses. Holding all honeypots constant, we bound the expected number of timesteps for the attacker to complete their $\alpha\beta$ desired accesses. Let $T_{\alpha,\beta}$

denote this quantity of timesteps. Then, the attacker can select between pursuing this all-access strategy (incurring $\mathbb{E}[T_{\alpha,\beta}]$ accesses on average) or pursuing Section 4.1's all-credential querying strategy (incurring $\mathbb{E}[Q]$as defined in Equation (5), plus $\alpha\beta$). Below, we provide bounds on $\mathbb{E}[T_{\alpha,\beta}]$.

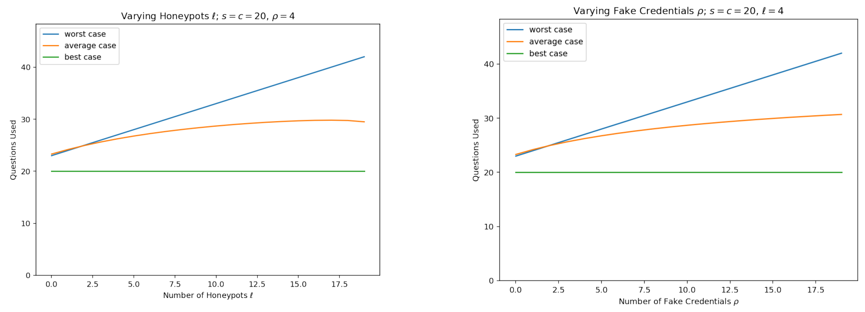


Fig. 1: # of questions used to learn identities of all credentials given average, worst, or best case, varying quantity of honeypots or fake credentials respectively.

Let p_i be the probability the real server with the i-th highest value of wx_i is accessed. Further, let p_0 equal the probability a honeypot or real server outside the top α is accessed. Clearly, $\sum_{i=0}^{\alpha} p_i = 1$, while the probability of a desired $\alpha\beta$ access at the first timestep is no less than αp_α and no greater than αp_1.

Theorem 2 (Necessary and sufficient total number of accesses). *Let $h(\alpha,\beta) = \log\alpha + (\beta-1)\log\log\alpha - \log\log(\beta-1)! + C$. Then, for a constant C,*

$$\frac{1}{p_1}h(\alpha,\beta) \le \mathbb{E}[T_{\alpha,\beta}] \le \frac{1}{p_\alpha}h(\alpha,\beta).$$

Proof. Let $\nu_{\alpha,\beta}$ be the total number of accesses to the α desired servers in order for each server to have β accesses, and let $P(p_0 = \eta - \pi, \nu_{\alpha,\beta} = \pi)$ be the probability that out of η accesses, $\eta - \pi$ are to p_0 and the remaining π are enough to complete the desired accesses. Where line 2 follows by the binomial theorem and using p_1 as an upper bound for the probability that any one desired server is accessed (i.e. $p_1 = \max_{i\in[\alpha]p_i}$), we have that $P(T_{\alpha,\beta} = \eta) =$

$$\sum_{\pi=\alpha\beta}^{\eta} P(p_0 = \eta - \pi, \nu_{\alpha,\beta} = \pi) \le \sum_{\pi=\alpha\beta}^{\eta} \binom{\eta-1}{\eta-\pi} (1-\alpha p_1)^{\eta-\pi}(\alpha p_1)^{\pi} P(\nu_{\alpha,\beta} = \pi)$$

$$= \sum_{\pi=\alpha\beta}^{\eta} \binom{\eta-1}{\eta-\pi} (1-\alpha p_1)^{\eta-\pi}(\alpha p_1)^{\pi} \times$$

$$\left[\exp\left(-\frac{e^{-(\pi - r_{\alpha,\beta})/\alpha}}{\beta!}\right) - \exp\left(-\frac{e^{-(\pi - r_{\alpha,\beta}-1)/\alpha}}{\beta!}\right)\right],$$

for $r_{\alpha,\beta} = \alpha \log \alpha + \alpha\beta \log\log \alpha$. Line 3 follows by [9], who also show that $\mathbb{E}[\nu_{\alpha,\beta}] = \alpha \log \alpha + \alpha(\beta - 1) \log\log \alpha - \alpha \log\log{(\beta-1)!} + \alpha C$. Therefore, $\mathbb{E}[T_{\alpha,\beta}] \leq \frac{1}{\alpha p_1}(\alpha \log \alpha + \alpha(\beta-1)\log\log\alpha - \alpha\log\log{(\beta-1)!} + \alpha C)$.

$$\begin{aligned}\mathbb{E}[T_{\alpha,\beta}] &\leq \frac{1}{\alpha p_1}(\alpha \log \alpha + \alpha(\beta-1)\log\log\alpha - \alpha\log\log{(\beta-1)!} + \alpha C)\\ &= \frac{1}{p_1}(\log \alpha + (\beta-1)\log\log\alpha - \log\log{(\beta-1)!} + C) = \frac{1}{p_1}h(\alpha,\beta).\end{aligned}$$

The same process, with p_α to lower bound any one desired server's access probability, gives that $P(T_{\alpha,\beta} = \eta) \geq \sum_{\pi=\alpha\beta}^{\eta} \binom{\eta-1}{\eta-\pi}(1-\alpha p_\alpha)^{\eta-\pi}(\alpha p_\alpha)^{\pi} * \left[\exp\left(-\frac{e^{-(\pi - r_{\alpha,\beta})/\alpha}}{\beta!}\right) - \exp\left(-\frac{e^{-(\pi - r_{\alpha,\beta-1})/\alpha}}{\beta!}\right)\right]$, so $\mathbb{E}[T_{\alpha,\beta}] \geq \frac{1}{p_\alpha}h(\alpha,\beta)$.

5 Defender Strategy

We next analyze the game from the defender's perspective. While real servers are held constant, the defender will design honeypots to "mimic" the real servers: dynamically setting $x_i = x_j$ for honeypot i and real server j to confuse and distract the attacker. For simplicity, the attacker is fixed to perform random accesses; we save the attacker's identity query strategy for future work.

We will demonstrate that specific honeypot placements are most effective for exploiting knowledge of the attacker's preferences, while other honeypot placements are most effective for gaining knowledge of the attacker's preferences. Specifically, setting honeypot server i to mimic real server j with maximum attack probability increases the chance of the honeypot being attacked. This is an "exploit" action. However, the defender must form an estimate $\tilde{W}$ of the attacker's preferences w to determine which server is max. As we shall see, mimicking servers with low attack probabilities enables faster rates of learning. This is an "explore" action, where the reward is not maximized in the short term.

We begin by providing theoretical results on the defender's ability to learn the attacker preferences w in Section 5.1. Then, Section 5.2 validates our theoretical results using a simulation, demonstrating that the defender stalls the attacker from finishing their desired accesses by an average of 5.8% timesteps and scores an average of 32% points more than a defender baseline with constant honeypots.

5.1 Theoretical Results

We bound the expected preference estimation error $\mathbb{E}[||\tilde{W} - w||]$ in terms of parameters such as the number of timesteps and servers. For notational simplicity, we shall make the following assumption: the defender may change one honeypot's x-vector every τ interactions between the attacker and defender. Let T denote the total number of periods with constant honeypot values. Though the final time period may be shorter than τ timesteps if the attacker completes their $\alpha\beta$ desired accesses prior to the period's τ-th timestep, we suppose for simplicity that all periods contain τ steps. However, all results are easily modified to accommodate a shorter final time period.

Estimating Attacker Preferences First, we outline a procedure for the defender to form their estimate $\tilde{W}$. Concretely, suppose that each honeypot takes on the x-value of a real server. While there are consequences on the reward when the attacker accesses real server i versus a honeypot server with vector x_i (i.e., if the honeypot is sampled by the attacker, the defender has a positive reward), there is *no difference* from the point of view of learning w. For this reason, we can restrict the learning problem to operate on $x_1, \ldots, x_\mu$, where $\mu = |\mathcal{S}_r|$, the number of real servers. Let b_i^k be the total number of servers, real or honeypot, that have value x_i in time period $1 \leq k \leq T$, so that $\sum_{j \in [\mu]} b_j^k = \mu + \ell = s$ and that $1 \leq b_j^k \leq 1 + \ell$ for all j. Then, the distribution p^k in time period k is

$$p_i^k = \frac{b_i^k \exp(w^T x_i)}{\sum_{j \in \mathcal{S}} b_j^k \exp(w^T x_j)}. \tag{6}$$

The defender will run the following algorithm: for the k-th time period, let a_i^k be the number of observed attacks on any of the b_i^k servers with x-vector equal to that of the i-th real server. The defender will estimate the attack distribution as $\tilde{p}_i^k = a_i^k/\tau$ for all $1 \leq i \leq \mu$, then apply smoothing with a small constant ω so that $\tilde{p}_i^k > 0$. Smoothing avoids dividing by 0 in the learning process. The defender will then form and solve a system as in [8], i.e., $A\tilde{w}^k = \tilde{Y}^k$ for

$$A = \begin{bmatrix} x_1 - x_2 \\ x_2 - x_3 \\ \cdots \\ x_{\mu-1} - x_\mu \end{bmatrix} \text{ and } \tilde{Y}^k = \begin{bmatrix} \log(\tilde{p}_1^k/\tilde{p}_2^k) \\ \log(\tilde{p}_2^k/\tilde{p}_3^k) \\ \cdots \\ \log(\tilde{p}_{\mu-1}^k/\tilde{p}_\mu^k) \end{bmatrix} - \begin{bmatrix} \log(b_1^k/b_2^k) \\ \log(b_2^k/b_3^k) \\ \cdots \\ \log(b_{\mu-1}^k/b_\mu^k) \end{bmatrix}.$$

In order to perform linear least squares, we assume $\mu - 1 \geq d$. This will produce an estimate $\tilde{w}^k$ for each period. The final estimate is then just $\tilde{W} = 1/T \sum_{k=1}^T \tilde{w}^k$.

Preference Learning Bound At first glance, the learning bound in [8] appears to have a similar flavor, where we could use the τT observed interactions to obtain a bound on $\|\tilde{W} - w\|$. Critically, this result does not apply to our setting, because the distributions of our samples are different in each of the T periods due to the varying b^k. Instead, we find

Theorem 3. *Let $B = (A^T A)^{-1} A^T$. Further, let $g(p^k, \delta) = c_1^2 ||B||^2 \log^2\left(\frac{2T\mu}{\delta}\right) \sum_{j=1}^{\mu} \frac{1}{(p_j^k)^4}$ for constant c_1 and $k_{\max} = \arg\max_k g(p^k, \delta)$. Then, the defender can learn an estimate $\tilde{W}$ that with probability at least $1 - \delta$ satisfies $\mathbb{E}[||\tilde{W} - w||] \leq$*

$$\frac{1}{\tau T}\sqrt{\log(d+1)\sum_{k=1}^{T} g(p^k, \delta)} + \frac{2\log(d+1)}{3\tau T}\sqrt{\frac{1}{8} g(p^{k_{\max}}, \delta)} + \frac{1}{\tau T}\sum_{k=1}^{T}\sqrt{\frac{1}{8} g(p^k, \delta)}$$

Interpreting the Bound. Before tackling the proof, we make some remarks. First, there are three summands in the bound. The scaling with respect to game length for the first two summands is given by $O(\frac{1}{\tau\sqrt{T}})$, for the second by $O(\frac{1}{\tau T})$ and for the third by $O(1/\tau)$. To see this, note that we are summing over T terms in the first and third summand. In the first, this is inside the square root, while in the third, there is no square root, canceling the $1/T$ in front. Therefore, the third term is dominant. However, when $\tau > T$, we have that $\frac{1}{\tau} < \frac{1}{\sqrt{\tau T}}$ so that we obtain the same overall learning rate as if all the distributions were identical.

Next, consider the term $g(p^k, \delta)$. We have that $\mathbb{E}[\|\tilde{W} - w\|]$ scales with μT (the product of the number of periods with the number of genuine servers), though the factor is only squared-logarithmic. A more interesting term is $\sum_{j=1}^{\mu} \frac{1}{(p_j^k)^4}$. The learning rate is worse when particular probabilities p_j^k are small. In fact, to obtain the best learning rate, *the defender would want a uniform distribution.* To this end, the defender may design their honeypots so as to cause p^k to most closely resemble the uniform distribution.

The last observation translates into a simple "explore" strategy for the defender. Given ℓ total honeypots, and that $\sum_j b_j^k = \mu + \ell$, the defender should allocate b_i^k's so the resulting model in (6) is as close to uniform as possible.

We now prove the result. The following lemma will prove useful.

Lemma 5. *For any k, with probability at least $1 - \delta$, it holds that*

$$||\mathbb{E}[\tilde{w}^k - w]|| \leq \mathbb{E}[||\tilde{w}^k - w||] \leq \|B\| \frac{c_1}{\tau} \log\left(\frac{2\mu T}{\delta}\right) \sqrt{\frac{1}{8}\sum_{j=1}^{\mu} \frac{1}{(p_j^k)^4}}.$$

Proof (Proof of Lemma 5). Observe that $\tilde{w}^k = B\tilde{Y}^k$ and $w = BY^k$, where Y^k is analogous to $\tilde{Y}^k$ for p^k. Let $e_i^k = \tilde{p}_i^k - p_i^k$. Then,

$$\tilde{w}^k - w = B \begin{bmatrix} \log(\frac{\tilde{p}_1^k}{\tilde{p}_2^k}) \\ \cdots \\ \log(\frac{\tilde{p}_{\mu-1}^k}{\tilde{p}_\mu^k}) \end{bmatrix} - B \begin{bmatrix} \log(\frac{p_1^k}{p_2^k}) \\ \cdots \\ \log(\frac{p_{\mu-1}^k}{p_\mu^k}) \end{bmatrix} = B \begin{bmatrix} \log(1 + \frac{e_1^k}{p_1^k}) - \log(1 + \frac{e_2^k}{p_2^k}) \\ \cdots \\ \log(1 + \frac{e_{\mu-1}^k}{p_{\mu-1}^k}) - \log(1 + \frac{e_\mu^k}{p_\mu^k}) \end{bmatrix}.$$

So, $\dfrac{\mathbb{E}[\|\tilde{w}^k - w\|]}{\|B\|} \leq E \left[\left\| \begin{bmatrix} \log(1 + \frac{e_1^k}{p_1^k}) - \log(1 + \frac{e_2^k}{p_2^k}) \\ \cdots \\ \log(1 + \frac{e_{\mu-1}^k}{p_{\mu-1}^k}) - \log(1 + \frac{e_\mu^k}{p_\mu^k}) \end{bmatrix} \right\| \right]$

$$= \mathbb{E}\left[\sqrt{\sum_{j=1}^{\mu-1}\left(\log\left(1 + \frac{e_j^k}{p_j^k}\right) - \log\left(1 + \frac{e_{j+1}^k}{p_{j+1}^k}\right)\right)^2}\right]$$

$$\leq \mathbb{E}\left[\sqrt{\sum_{j=1}^{\mu-1}\log^2\left(1 + \frac{e_j^k}{p_j^k}\right) + \log^2\left(1 + \frac{e_{j+1}^k}{p_{j+1}^k}\right)}\right] \leq \mathbb{E}\left[\sqrt{2\sum_{j=1}^{\mu}\log^2\left(1 + \frac{e_j^k}{p_j^k}\right)}\right]$$

and by Jensen's inequality, since $\mathbb{E}[\cdot^{1/2}] \leq \mathbb{E}[\cdot]^{1/2}$,

$$\mathbb{E}[\|\tilde{w}^k - w\|] \leq \|B\|\mathbb{E}\left[2\sum_{j=1}^{\mu}\log^2\left(1+\frac{e_j^k}{p_j^k}\right)\right]^{1/2}. \tag{7}$$

We will now derive a bound on $\log^2\left(1+\frac{e_j^k}{p_j^k}\right)$. From Hoeffding's inequality [10], we have $P(|e_j^k| \geq \epsilon) = P(|\tilde{p}_j^k - p_j^k| \geq \epsilon) \leq 2\exp(-2\tau\epsilon^2)$. Suppose we want $P(|e_j^k| \leq \epsilon)$ with probability $1-\delta$, then $\delta = 2\exp(-2\tau\epsilon^2)$ so that $\epsilon = \sqrt{\frac{1}{2\tau}\log\frac{2}{\delta}}$. We next apply union bound to simultaneously control the deviations for all μT probabilities p_j^k. With probability at least $1-\delta$, we have for all $1 \leq j \leq \mu$ and for all $1 \leq k \leq T$ that $|e_j^k| \leq \sqrt{\frac{1}{2\tau}\log\frac{2\mu T}{\delta}}$.

Note that $\mathbb{E}[|e_j|] = 0$, so by the Taylor expansion and for some constant c_1,

$$\mathbb{E}\left[\log\left(1+\frac{e_j^k}{p_j^k}\right)\right] = \mathbb{E}[\sum_{i=1}^{\infty}(-1)^{i+1}\frac{1}{i}\frac{(e_j^k)^i}{(p_j^k)^i}] = \sum_{i=2}^{\infty}(-1)^{i+1}\frac{1}{i}\frac{\mathbb{E}[(e_j^k)^i]}{(p_j^k)^i}$$
$$\leq \sum_{i=2}^{\infty}(-1)^{i}\frac{1}{i}\frac{\mathbb{E}[(e_j^k)^i]}{(p_j^k)^i} = \frac{1}{2(p_j^k)^2}\mathbb{E}[|e_j|]^2 + \sum_{i=3}^{\infty}(-1)^{i}\frac{1}{i}\frac{\mathbb{E}[(e_j^k)^i]}{(p_j^k)^i}$$
$$\leq \frac{c_1}{2(p_j^k)^2}\sqrt{\frac{1}{2\tau}\log\frac{2\mu T}{\delta}}^2 = \frac{c_1}{4\tau(p_j^k)^2}\log\frac{2\mu T}{\delta}.$$

Since $\mathbb{E}[\log^2(\cdot)] = \mathbb{E}[|\log(\cdot)|]^2 = \mathbb{E}[\log(\cdot)]^2$, from (7) and linearity,

$$\mathbb{E}[\|\tilde{w}^k - w\|] \leq \|B\|\left[2\sum_{j=1}^{\mu}\frac{c_1^2}{16\tau^2(p_j^k)^4}\log^2\frac{2\mu T}{\delta}\right]^{1/2}$$
$$= \|B\|\frac{c_1}{\tau}\log\left(\frac{2\mu T}{\delta}\right)\sqrt{\frac{1}{8}\sum_{j=1}^{\mu}\frac{1}{(p_j^k)^4}}$$

Lastly, $||\mathbb{E}[\tilde{w}^k - w]|| \leq \mathbb{E}[||\tilde{w}^k - w||]$ by Jensen's inequality.

Proof (Proof of Theorem 3). We will bound $\mathbb{E}[||\tilde{W} - w||]$ using the matrix Bernstein inequality [14]. We first define and bound $S^k = (\tilde{w}^k - w) - \mathbb{E}[\tilde{w}^k - w]$. This is a centered version of the error from estimating $\tilde{w}^k$ in one period k. $\mathbb{E}[S^k] = 0$ and $\mathbb{E}[\|S^k\|] \leq \mathbb{E}[\|\tilde{w}^k - w\|] + \|\mathbb{E}[\tilde{w}^k - w]\| \leq 2\mathbb{E}[\|\tilde{w}^k - w\|]$. By Lemma 5,

$$\mathbb{E}[||S^k||] \leq 2\|B\|\frac{c_1}{\tau}\log\left(\frac{2\mu T}{\delta}\right)\sqrt{\frac{1}{8}\sum_{j=1}^{\mu}\frac{1}{(p_j^k)^4}} := L_k.$$

Let $k_{\max} = \arg\max_k L_k$. Note that $\mathbb{E}[||S^k||^2] = \mathbb{E}[||S^k||]^2$. Next, define $Z = \sum_{k=1}^{T}\frac{1}{T}S^k$, so that $\mathbb{E}\left[||Z||^2\right] = \mathbb{E}\left[\left|\left|\sum_{k=1}^{T}\frac{1}{T}S^k\right|\right|^2\right] = \sum_{k=1}^{T}\frac{1}{T^2}\mathbb{E}\left[||S^k||^2\right]$. This follows since the cross terms are products of uncorrelated terms with zero mean.

By the matrix Bernstein inequality [14] applied to a d-element vector, $\mathbb{E}[||Z||] \leq \sqrt{2\mathbb{E}[||Z||^2]\log(d+1)} + \frac{L_{k_{\max}}}{3T}\log(d+1)$, we obtain

$$
\begin{aligned}
\mathbb{E}[||Z||] \leq & \sqrt{2\log(d+1)\sum_{k=1}^{T}\frac{1}{T^2}\mathbb{E}[||S^k||^2]} \\
& + \frac{2}{3T}\log(d+1)\|B\|\frac{c_1}{\tau}\log\left(\frac{2\mu T}{\delta}\right)\sqrt{\frac{1}{8}\sum_{j=1}^{\mu}\frac{1}{(p_j^{k_{\max}})^4}} \\
\leq & \frac{c_1}{\tau T}\|B\|\log\left(\frac{2\mu T}{\delta}\right)\sqrt{\log(d+1)\sum_{k=1}^{T}\sum_{j=1}^{\mu}\frac{1}{(p_j^k)^4}} \\
& + \frac{2}{3T}\log(d+1)\|B\|\frac{c_1}{\tau}\log\left(\frac{2\mu T}{\delta}\right)\sqrt{\frac{1}{8}\sum_{j=1}^{\mu}\frac{1}{(p_j^{k_{\max}})^4}}.
\end{aligned}
$$

Ultimately, $\mathbb{E}[||\tilde{W} - w||] =$

$$
\begin{aligned}
&\frac{1}{T}\mathbb{E}\left[\left\|\sum_{k=1}^{T}(\tilde{w}^k - w)\right\|\right] = \frac{1}{T}\mathbb{E}\left[\left\|\sum_{k=1}^{T}(\tilde{w}^k - w - \mathbb{E}[\tilde{w}^k - w] + \mathbb{E}[\tilde{w}^k - w])\right\|\right] \\
&\quad = \frac{1}{T}\mathbb{E}\left[\left\|\sum_{k=1}^{T}(S_k + \mathbb{E}[\tilde{w}^k - w])\right\|\right] \quad \leq \frac{1}{T}\mathbb{E}\left[\left\|\sum_{k=1}^{T}S_k\right\| + \left\|\sum_{k=1}^{T}\mathbb{E}[\tilde{w}^k - w]\right\|\right] \\
&\quad = \mathbb{E}\left[\|Z\|\right] + \frac{1}{T}\left\|\sum_{k=1}^{T}\mathbb{E}[\tilde{w}^k - w]\right\| \quad \leq \mathbb{E}\left[\|Z\|\right] + \frac{1}{T}\sum_{k=1}^{T}\left\|\mathbb{E}[\tilde{w}^k - w]\right\|.
\end{aligned}
$$

Substituting for $\mathbb{E}[||Z||]$ and $||\mathbb{E}[\tilde{w}^k - w]||$ (from Lemma 5) gives the result.

5.2 Simulation

We implement the defender strategy with a simple thresholding heuristic: the defender begins by exploring in each time period, then switches to exploiting after the first time period k for which $||\tilde{w}^k - \tilde{w}^{k-1}||$ is less than a hyperparameter ϵ. We use hyperparameters $s = 4, \ell = 1, d = 2, \tau = 100, \epsilon = 0.1, \Delta = 1, \alpha = 2$ and $\beta = 4000$. Further, we use a simplified reward scheme where the defender loses no points when the attacker completes a desired access. We compare our strategy, "mimic explore/exploit" with a constant strategy where the honeypot is set to mimic a random real server at the beginning of the game, as well as a "mimic explore" strategy where the defender never exploits. See Table 1 for a summary; results are averaged across 100 runs.

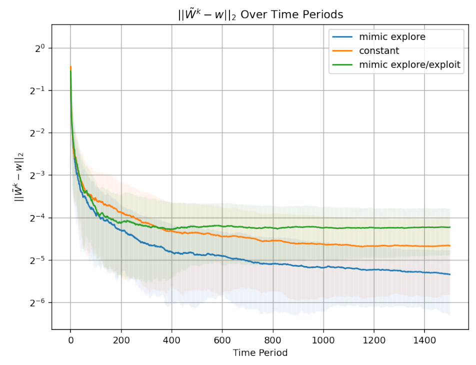


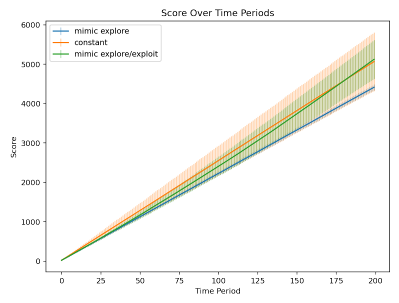
Fig. 2: Preference estimation error $\|\tilde{W}^k - w\|$ across 1500 time periods (τ=100). We find that the explore strategy achieves lowest error, while explore/exploit achieves highest and the constant strategy serves as a middle ground.

For ease of comparison, we first fix each run to last $T = 1500$ time periods. Figure 2 depicts $||\tilde{W}^k - w||$, the error in estimating attacker preferences, across time periods. As expected, the mimic explore strategy consistently achieves the smallest preference estimation error. Mimic explore/exploit begins with comparable performance to mimic explore, but it soon plateaus as it switches to exploiting at time period 741 on average ($\sigma = 446$). Despite that, mimic explore/exploit ultimately does not estimate preferences as well as the constant method, explore/exploit method achieves the highest average score as shown by Figure 3b. Figure 3a demonstrates that, as mimic explore/exploit switches to exploit, it soon overtakes both mimic explore and constant strategies in encouraging the attacker to access the honeypot. Explore/exploit is able to effectively leverage the knowledge gained during explore phase – applying it during exploit phase and scoring an average of 32% more points than constant defender.

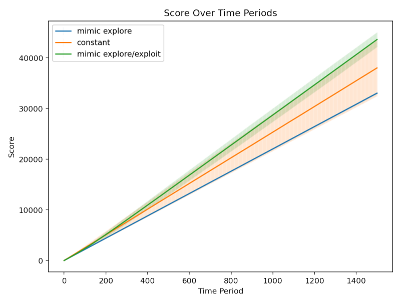
We lastly examine game length, an alternative metric for measuring the effectiveness of honeypot placements in slowing the attacker's progress. Mimic explore/exploit strategy prolongs the attacker from achieving their goal the longest, requiring an average of 5.8% more time periods than the constant method.

	Periods to complete desired accesses	Final preference estimation error	Final score
Mimic explore/exploit	1932.81 (16.18)	0.054 (0.019)	43613.80 (1426.84)
Mimic explore	1756.01 (6.97)	0.025 (0.012)	33038.49 (176.22)
Constant	1826.59 (90.52)	0.040 (0.029)	38015.75 (5544.89)

Table 1: Standard deviation in parenthesis



(a) During first 200 time periods.



(b) During 1500 time periods.

Fig. 3: Defender reward during first 200 or 1500 periods. While explore/exploit performance initially matches the lowest-scoring explore strategy, it surpasses the constant strategy as it switches to exploit, with widening performance gap.

References

[1] Palvi Aggarwal, Cleotilde Gonzalez, and Varun Dutt. "Cyber-security: Role of Deception in Cyber-Attack Detection". In: July 2016.

[2] J. Zhang et al. "Modeling multi-target defender-attacker games with quantal response attack strategies". In: *Reliability Engineering System Safety* (2021).

[3] Ronald M. Campbell et al. "A survey of honeypot research: Trends and opportunities". In: *ICITST* (2015), pp. 208–212.

[4] Y. Wang et al. "A Survey of Game Theoretic Methods for Cyber Security". In: *IEEE Intl. Conf. on Data Science in Cyberspace*. 2016, pp. 631–636.

[5] J. Pawlick et al. "A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy". In: *ACM Computing Surveys* 52.4 (2019).

[6] Mark Bilinski, Ryan Gabrys, and Justin Mauger. "Optimal placement of honeypots for network defense". In: *GameSec*. 2018, pp. 115–126.

[7] Mark Bilinski et al. "Lie another day: Demonstrating bias in a multi-round cyber deception game of questionable veracity". In: *GameSec*. 2020, pp. 80–100.

[8] Mark Bilinski et al. "No Time to Lie: Bounds on the Learning Rate of a Defender for Inferring Attacker Target Preferences". In: *GameSec*. 2021, pp. 138–157.

[9] Paul Erdős and Alfréd Rényi. "On a classical problem of probability theory". In: *Magyar Tud. Akad. Mat. Kutató Int. Közl* (1961), pp. 215–220.

[10] Wassily Hoeffding. "Probability inequalities for sums of bounded random variables". In: *The collected works of W. Hoeffding* (1994), pp. 409–426.

[11] T. Nguyen et al. "Deception in finitely repeated security games". In: *AAAI*. 2019.

[12] Aaron Schlenker et al. "Deceiving Cyber Adversaries: A Game Theoretic Approach". In: *IFAAMAS*. 2018, pp. 892–900.

[13] Nan Sun et al. "Data-Driven Cybersecurity Incident Prediction: A Survey". In: *IEEE Communications Surveys Tutorials* 21.2 (2019), pp. 1744–1772.

[14] Joel A Tropp et al. "An introduction to matrix concentration inequalities". In: *Foundations and Trends® in Machine Learning* (2015), pp. 1–230.

[15] Mark Wildon. "Knights, spies, games and ballot sequences". In: *Discrete Mathematics* 310.21 (2010), pp. 2974–2983.