Modeling Observability in Adaptive Systems to Defend Against Advanced Persistent Threats

Cody Kinneer
School of Computer Science
Carnegie Mellon University
ckinneer@cs.cmu.edu

Ryan Wagner
School of Computer Science
Carnegie Mellon University
rrwagner@cs.cmu.edu

Fei Fang
School of Computer Science
Carnegie Mellon University
feif@cs.cmu.edu

Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

David Garlan
School of Computer Science
Carnegie Mellon University
garlan@cs.cmu.edu

ABSTRACT

Advanced persistent threats (APTs) are a particularly troubling challenge for software systems. The adversarial nature of the security domain, and APTs in particular, poses unresolved challenges to the design of self-* systems, such as how to defend against multiple types of attackers with different goals and capabilities. In this interaction, the observability of each side is an important and under-investigated issue in the self-* domain. We propose a model of APT defense that elevates observability as a first-class concern. We evaluate this model by showing how an informed approach that uses observability improves the defender's utility compared to a uniform random strategy, can enable robust planning through sensitivity analysis, and can inform observability-related architectural design decisions.

CCS CONCEPTS

• **Security and privacy** → **Formal security models**; *Systems security*; • **Computing methodologies** → *Artificial intelligence*; *Model development and analysis.*

KEYWORDS

Advanced Persistent Threats, Game Theory, Observability, Adaptive Systems

ACM Reference Format:
Cody Kinneer, Ryan Wagner, Fei Fang, Claire Le Goues, and David Garlan. 2019. Modeling Observability in Adaptive Systems to Defend Against Advanced Persistent Threats. In *17th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE '19), October 9–11, 2019, La Jolla, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3359986.3361208

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MEMOCODE '19, October 9–11, 2019, La Jolla, CA, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6997-8/19/10…$15.00
https://doi.org/10.1145/3359986.3361208

1 INTRODUCTION

As society has become more dependent on software, the nature of cyber threats has evolved. Of particular concern is the increase in attacks known as advanced persistent threats (APTs). Rather than one-off incidents, these attacks are marked by a highly motivated and capable adversary willing to devote significant time to conducting an attack. APTs are characterized by an extended interaction between the attacker and defender, with the attacker often conducting extensive surveillance against the defender's system to facilitate their malicious goals. The 2013 attack against the retail company Target, during which an attacker stole 40 million consumer credit cards, is an example of such an attack [3]. Organized crime, terrorists, and nation-states can all be APT attackers.

Self-adaptive, self-protecting, and self-healing (self-*) software systems automatically respond to changes in their environments to continue satisfying their quality objectives. While self-* systems have successfully been applied to autonomously adapt to uncertainty in the environment, the security domain presents unique challenges to their design [7, 8]. First, the environment is adversarial: it is attempting to cause harm and can respond to defensive measures by changing the attack approach. This is in contrast to other adaptation drivers (e.g., system performance, deployment cost, internal faults, and system availability) for which average case analysis is often sufficient, and which occur largely independently of adaptive strategies used to improve them.

Second, security attacks are associated with a high degree of uncertainty: the defender may know little about the identity or techniques of the attacker. This uncertainty is complicated by the potential for many kinds of adversaries each with different goals and tactics, techniques, and procedures (TTPs). Moreover, attackers may switch their tactics when they believe they have been detected.

Third, complex attacks such as APTs can take place over a long period of time, and it is therefore difficult to spot the existence and extent of an attack. Attack actions are rare, relative to normal system behavior, and easily camouflaged. This lack of knowledge directly impacts the ability of a defender to react effectively, since it is hard to predict the likely sources of compromise and attackers' future courses of action.

Hence, when a system is under attack, its human defenders must balance attempting to evict the attacker from the system — which may tip the defender's hand and result in the attacker digging in — versus waiting to gather more information about the attacker

for a more likely eviction at a later time. In this interaction, both sides have a limited ability to observe the other, and both benefit from concealing themselves while trying to learn more about their opponent. To autonomously defend against these attackers, self-* systems need to reason about the observability of both sides in the interaction to effect an optimal defense.

One approach explored by the self-* research community is to use game-theoretic models for analysis and planning to mitigate security threats. Games provide an appropriate context for analyzing security because they can model the adversarial nature of the setting together with uncertainty in predictions of attacker behavior, and they have been shown to be effective in responding to attacks such as a Distributed Denial of Service attacks, as well as several APT scenarios [9, 10, 22–25, 29, 31].

However, prior games theory based approaches to self-securing systems have not addressed the specific challenges of designing systems robust to APT-like attacks, which include (a) the value of increased information over time in both attacking and defending, (b) the existence of specific tactics to proactively increase information (on both sides), and, most importantly, (c) the need to reason about the effects of the visibility of actions on the behavior of the other player.

In this paper, we show how to address these challenges using a game-theoretic approach in which observability is modeled as a first-class concern to facilitate effective reasoning about security threats during system design. Specifically, we introduce a model treating the defender deployed system as a game in which the defender can decide to attempt an eviction of an attacker based on the knowledge at hand, wait, or even take active measures to gather more information for use in an eviction attempt on a subsequent turn. For example, a defender might throttle the download speed of an attacker attempting to exfiltrate data, allowing the defender more time to observe the attacker. In contrast, an attacker can, with some cost, enhance their persistence in the system to evade future eviction following a failed eviction attempt. Significantly, the actions taken by a defender or attacker are associated with a measure of observability that may inform the future actions of the other player. As we will show, this model can be used to make robust decisions despite uncertainty in the true probability distribution of attacker types. Additionally, we show how the model can inform architectural decisions at design time, and we demonstrate this in a scenario where system designers can pay a cost to build decoys into the system. We evaluate the model by showing, through an example quantification, the improvement in the defender's utility by taking into account observability information compared to using an uninformed approach, as well as demonstrating that the model can scale to a practically useful time horizon. The key contributions are therefore:

- A model of self-* APT defense: The Observable Eviction Game (OEG) with observability as a first-class concern.
- An evaluation of the model compared to an uninformed approach, and a demonstration that the model scales to practically useful time horizons.
- A sensitivity analysis showing the ability to provide robust solutions in uncertain security environments.
- An exemplar scenario showing an improvement using the model to make architectural decisions at design time.

Together, these contributions will aid software engineers to design self-* systems that are robust to APTs. The remainder of the paper is organized as follows. Section 2 provides the necessary background, including an explanation of APTs and the basics of game theory. Section 3 describes our modeling approach. Section 4 evaluates the approach. Section 5 positions our contribution with respect to the related work. Lastly, section 6 concludes.

2 BACKGROUND

This section provides background content for the rest of the paper, including self-* systems, APTs, and game theory.

2.1 Self-* Systems

Many self-* systems are based on a MAPE-K loop [14]: they monitor, analyze, plan, and execute based on a shared set of knowledge that is shared throughout the loop. These systems are able to detect system and environmental state, make informed decisions based on that information, and then adapt in response to the information gathered to improve system utility. These types of systems can be used to adapt to varying system loads, faults, and other conditions [5]. An emerging use of self-* systems is real-time adaptation in reaction to security threats. Current security systems, like intrusion prevention systems, are rudimentary, with the primary adaptation of blocking suspicious traffic flows. However, more sophisticated systems could incorporate a richer set of defensive tactics [25].

2.2 Security and Advanced Persistent Threats

The security landscape has evolved to include sophisticated actors known as advanced persistent threats (APTs). The US National Institute of Standards and Technology (NIST) defines an APT as "An adversary that possesses sophisticated levels of expertise and significant resources.... The advanced persistent threat: (i) pursues its objectives repeatedly over an extended period of time; (ii) adapts to defenders' efforts to resist it; and (iii) is determined to maintain the level of interaction needed to execute its objectives" [16].

Each APT has a set of TTPs that is used to carry out an attack. These TTPs include the tooling and methods used by a group of individuals dedicated to a particular purpose, such as gathering intelligence, stealing merchantable artifacts, or causing disruption. In some cases, a threat actor may have multiple APT groups defined by their distinct TTPs [1]. Because TTPs represent the accumulated knowledge, skills, and abilities of attackers, they can be difficult to change. However, a nation-state with multiple APT groups under its control could reassign responsibility for attacking a target from one APT group to another, or a single APT group could swap out one set of tooling and command and control infrastructure for another if need be. In the most sensitive operations, APT groups will use multiple sets of TTPs, including multiple types of malware, to ensure persistent presence even in the case of detection.

For the defender, knowledge of an attacker's TTPs is often crucial to successful attack mitigation because that knowledge can be used to look for likely places where the system might have been compromised and to predict future courses of action. Such knowledge can be gained in several ways, such as simply waiting to see

what the attacker will do next, or putting in place active detection mechanisms, or *active measures* (e.g., honeynets or camouflage).

However, the desire to know more about an attacker leads to a particularly important challenge in deciding a course of action: the defender can react on incomplete information, and risk an incomplete, or failed, eviction of the attacker; or wait and even take active measures to observe the attack, building a more complete picture of the attacker's TTPs in hopes of later fully evicting the attacker. The problem is further complicated by the fact that there is often an inverse relationship between the effectiveness of actions that a defender can take to observe or mitigate an attack, and the observability of those actions to the attacker.

An example of this quandary was used to describe a recent attack on the US Office of Personnel Management (OPM) that resulted in the loss of personal information for millions of US government employees and contractors:

"At first, the investigators left each piece of malware in place, electing only to throttle its ability to send outbound traffic; if the attackers tried to download any data, they would find themselves confined to dial-up speeds. But on April 21, [OPM senior IT strategist] Mejeur and the US-CERT team began to discuss whether it was time to boot the attackers, who would thus learn that they'd been caught. 'If I miss one remote-access tool, they'll come back in through that variant, they'll reestablish access, and then they'll go dormant for six months to a year at least,' says a US-CERT incident responder…" [2].

Today, deciding whether to observe or act is a manual process, lacking formal foundations or ways to rigorously analyze alternative courses of action. As we elaborate in the remainder of this paper, we propose to address this problem using game-theoretic analyses in which observability of actions is represented as a first-class entity. The game accounts for the fact that an attacker can morph, with some cost, to evade eviction by a defender, and a defender can decide to attempt an eviction of an attacker based on the knowledge at hand or wait to gather more information for use in an eviction attempt on a subsequent turn.

2.3 Game Theory

Unlike other quality attributes that a self-* system may optimize like quality of service, security presents a unique challenge in the form of an attacker. Like the self-* system itself, the attacker can take actions that affect the system, and can themselves gather information and adapt to the behavior of the system to further their own interests. Game theory provides a framework for reasoning mathematically about interactions between multiple agents, or players [20]. A normal-form game is defined by a tuple $(\mathcal{N}, \mathcal{A}, u)$. $\mathcal{N} = \{1...n\}$ is the set of players. $\mathcal{A} = \Pi_{i=1}^{n} \mathcal{A}_i$ is the set of joint actions, where $\mathcal{A}_i$ is the set of actions for player i. $u = (u_1, ...u_n)$ and $u_i : \mathcal{A} \rightarrow \mathbb{R}$ is the payoff or utility function for player i that maps the players' joint action profiles to an outcome value. Each player is seeking to maximize their own individual utility. A player's strategy can be pure (i.e., take a deterministic action) or mixed (i.e., randomly choose an action according to some probability distribution). The Nash equilibrium (NE) of a game is the strategy profile $\sigma = (\sigma_1, ...\sigma_n)$ for all players such that no player can gain from unilaterally changing their strategy. That is, that each player is playing the best response to each other player.

More complicated games with sequences of actions are often modeled by extensive-form games (EFG), which can be represented by a game tree where each node corresponds to a unique history of actions taken by all players and chance from the root of the game, and each edge corresponds to possible actions available to the player (could be a chance player) who will choose an action at the node. Players get payoffs at the leaf nodes of the game tree and then the game terminates. In addition, each players' choice nodes can be partitioned into information sets to model the imperfect information in the game. A player cannot distinguish between nodes in the same information set. A pure strategy for player i in an EFG assigns one action for each information set of player i. Stochastic behavior can be modeled by introducing a nature, or chance player, who moves according to a fixed probability distribution. By enumerating pure strategies for all players, we can get an induced normal form game of an EFG and the NEs are preserved.

In complete information games, all players know the identity and payoff functions of all other players. Bayesian games relax this assumption by allowing multiple types of players and that at least one player is unsure of the type of another player. Formally, a Bayesian game extends the aforementioned normal-form game model by introducing $\Theta = \Pi_{i=1}^{n} \Theta_i$ where Θ_i is the set of possible types for player i and a common prior of joint probability distribution of players' types $\mathcal{P} : \Theta \rightarrow [0, 1]$. The utility function of a player is dependent on the players' joint type profile and joint action profile, i.e., $u_i : \mathcal{A} \times \Theta \rightarrow \mathbb{R}$. The Harsanyi transformation [12] converts a Bayesian game to a normal-form game.

Stackelberg games assume two kinds of players: a leader, and one or more followers. Rather than each player choosing a strategy simultaneously, in Stackelberg games, the leader acts first and commits to a strategy. The followers then observe the strategy of the leader and best respond. This game type is useful in a security context to capture the fact that a sophisticated attacker can often observe the defender's strategy before acting. Several approaches exist for efficiently solving Strong Stackelberg Equilibria (SSE), in both Bayesian and extensive form settings [17, 21, 30].

3 OBSERVABLE EVICTION GAME

Self-protecting systems dealing with APT scenarios need to automatically decide between attempting to evict an attacker versus attempting to gather more information about them, as explained in Section 2.2. To support designing these systems, a useful model of decision making should include the following elements:

(C-1) Multiple types of attackers, each with different goals and available TTPs
(C-2) A defender who must choose between attempting to evict the attacker or gathering more information, with or without active measures
(C-3) A defender who must balance thwarting the attacker with minimizing disruption to the system
(C-4) Eviction success should be predicated on the defender's knowledge about the attacker's identity
(C-5) An attacker who changes their behavior and becomes more difficult to evict after an unsuccessful eviction

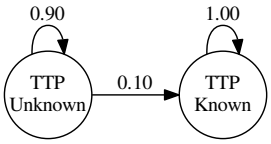

Figure 1: Markov process for TTP observability.

We present a novel game model called the Observable Eviction Game (OEG) that satisfies all of the above criteria. OEG is a Bayesian game with two players, i.e., $\mathcal{N}$={1,2}. Player 2 is an attacker who seeks to compromise the system, and player 1 is a defender who wants to minimize the attacker's success and disruptions to the system's operations. There is one defender type, but the attacker's type is drawn from a set supporting modeling different APT threat groups (C-1), denoted as $\Theta = \{\theta_1, ..., \theta_M\}$. OEG depicts sequential actions of players and therefore can be described as an EFG.

In the remainder of the section, Section 3.1 explains how the game proceeds and the actions available to each player, Section 3.2 explains the utility functions, Section 3.3 describes solving the game for equilibria, and Section 3.4 provides an example quantification of the OEG for use in the experiments described in Section 4. Together, these provide software engineers with a new and powerful tool for designing autonomic software systems resilient to APTs.

3.1 Actions

OEG models observability as a first-class concern and we leave out details that are less relevant to observability. As an overview of the game, OEG models the defender-attacker interaction in discrete time. The attacker chooses their attack plan, i.e., TTP, at the beginning of the game, which is initially unobserved or unknown to the defender. The game lasts τ time steps. Each time step the defender can perform an eviction attempt or perform some other observational tactic (such as waiting or taking active measures) to gain more information about the attacker (C-2). In addition, in each time step, nature, or the chance player, randomly determines whether the attacker's attack plan will become observable to the defender.

Rather than considering every possible path an attacker might take through an attack tree [26] to attack the system, we consider attacker strategies at a higher level of abstraction. The attacker's attack plan is described by a TTP, and the attacker may choose among several TTPs. This is consistent with the behavior of real APT groups (Section 2.2) since these groups often gain expertise in a particular set of techniques and operate in a particular way. In the remainder of the paper, we will use the terms TTP and attacker plan interchangeably. More concretely, in time step 0, the attacker chooses an action $\gamma_j \in \Gamma = \{\gamma_1, ...\gamma_Z\}$ that indicates their attack plan or TTP. They will not change it unless they observe that the defender makes an eviction attempt or takes an active measure.

In time step $t \in \{1, ...\tau\}$, the defender can choose an action from the set of eviction attempts $\Omega = \{\omega_1, \omega_2, \ldots, \omega_L\}$. The success of an eviction attempt depends on the suitability of the eviction action to the TTP chosen by the attacker. We denote by $\chi_{jl} \in [0, 1]$ the effectiveness of the defender's eviction action $\omega_l \in \Omega$ to evict the attackers TTP γ_j, which describes the probability that the attacker can be successfully evicted, with 1 indicating always successful and 0 indicating always unsuccessful. We assume that for each TTP γ_j, there exists some eviction attempt ω_l such that $\chi_{jl} = 1$ and can successfully evict the attacker. If the defender performs a successful eviction attempt, the attacker is immediately evicted, ending the game. If the defender performs an unsuccessful eviction attempt, the attacker remains in the system and is alerted to the defender's knowledge of them, making it more difficult for the defender to evict them in a subsequent attempt (C-5). Instead of modeling the attacker's change of behavior and the defender's subsequent attempts in detail, we simply assume the attacker will stay in the system until the end of the $\tau + F$ time step without being interrupted by the defender. Equivalently in the game tree, the failed eviction leads to a leaf node as no more actions will be taken. This modeling approach is motivated by the high cost of a failed eviction attempt in practice, which can result in an attacker digging in and becoming more difficult to evict in the future (see Section 2.2). Since the game ends on a eviction action in either case, the defender can only take at most one eviction action throughout the game, and the defender can choose to not take an eviction action within τ time steps. This model reflects the fact that a defender's ability to evict an APT attacker depends on the defender's knowledge about the attacker (C-4). If the defender knows the attacker's TTP, the defender can choose the most suitable eviction action with $\chi_{jl} = 1$. On the other hand, if the defender uses an eviction action without knowing the TTP of the attacker, the defender may choose an ineffective action, resulting in a failed eviction attempt.

In addition to choosing an eviction attempt, the defender can also choose an action from the set of observational tactics $\Phi = \{\phi_{L+1}, \phi_{L+2}, \ldots, \phi_{L+Q}\}$ where ϕ_{L+1} is the default tactic of "wait" and $\phi_l, \forall l \in L + 2, ..., L + Q$ are active measures that can be applied. Whether or not the attacker's TTP is known to the defender is determined by nature and the observation tactics chosen by the defender. Intuitively, without any active measures taken by the defender, the longer the attacker stays in the system, the more observable the attacker's TTP, i.e., the more likely the defender learns the attacker's TTP. We model the observability of the attacker's TTP over time as a two-state Markov process. Figure 1 shows an example TTP observability model. Initially the defender does not know the attacker's choice of TTP, i.e., the attacker is in the "TTP Unknown" state. After each time step, the defender has some chance of learning the attacker's chosen TTP and the attacker may move to the "TTP Known" state. We use $q_j \in [0, 1]$ to denote the transition probability $P(\text{TTP Known}|\text{TTP Unknown})$ if the attacker chooses TTP j. A lower q_j means TTP j is stealthier and harder to observe. The attacker remains in the "TTP Unknown" state with probability $1 - q_j$. Therefore, in each time step $t \geq 1$, the chance player determines whether the attacker's TTP becomes known according to probability distribution $\langle q_j, 1 - q_j \rangle$ if the defender always choose to wait. If the defender learns the attacker's TTP, we assume that the defender will choose to evict the attacker immediately, ending the game. If the defender does not learn the attacker's TTP, the game continues.

If the defender uses an active measure, q_j will increase and the defender may learn the attacker's TTP earlier. However, the defender takes the risk of being noticed by the attacker, resulting in a change in the attacker's behavior. An observational tactic ϕ_l

is associated with a scalar representing the effectiveness of the tactic, denoted as x_l, and scalars representing how observable the tactic is to each attacker TTP, denoted as y_{jl}. The first element, $x_l \in [0, 1]$ describes the decrease in the attacker's probability of remaining hidden if the tactic is not noticed by the attacker. Let q_j^t to denote the transition probability at time t. If the defender applies ϕ_l at time t without being noticed by the attacker, then $q_j^t = 1 - x_l(1 - q_j^{t-1})$. In the second element, y_{jl} represents the probability that the attacker notices the defender's action ϕ_l, dependent on the TTP γ_j they are using. This allows the model to capture the fact that different TTPs may be more or less likely to observe the defender's countermeasures. In the game tree, the stochasticity of attacker noticing the defender's action can be represented by having the chance player determine the observability of action ϕ_l right after the defender taking action ϕ_l. If the attacker noticed the defender's action, the attacker changes their behavior and becomes more difficult to evict, resulting in the same outcome as a failed eviction attempt, and the game ending. The defender may take multiple active measures throughout the game. For the default tactic "wait", $x_{L+1} = 1$ and $y_{j(L+1)} = 0, \forall j$.

3.2 Utilities

When the game terminates, each player gets a utility or payoff. We model the attacker's utility as the amount of time they remain in the system multiplied by the appropriateness $\alpha_{ij} \in [0, 1]$ of their chosen TTP γ_j to their type θ_i. This modeling approach captures the fact that having more time in the system gives the attacker more opportunity to accomplish their goals. However, some TTPs may be more aligned with their goals than others. In addition, APT groups are often trained in a particular set of techniques amenable to their goals, and other TTPs are not available to them. Modeling the appropriateness of the attacker's TTP captures these facts, as a higher α_{ij} indicates more alignment between the TTP and their goal and $\alpha_{ij} = 0$ indicates that TTP γ_j is not available to the attacker of type i.

Similarly, we model the defender's utility as the negation of the amount of time the attacker remains in the system multiplied by $\delta_i \in [1, 10]$, a coefficient describing how disruptive the attacker type i is. Coefficients δ_i model the fact that different attackers may also be more or less disruptive to the defender. An adversary observing the number of orders being processed for intelligence purposes for example, is less disruptive to the defender than one that attempts to cause physical damage to the defender's resources. In addition, if the defender chooses an eviction attempt ω_l or an observational tactic ϕ_l, the defender pays a cost $\kappa_l \in [0, 1]$. This modeling choice allows the OEG to model an important practical reality of defense, that certain defensive measures that might be more effective, e.g., shutting down an infected server, might come at a high cost to the defender's operation (C-3).

Therefore, at a leaf node of the game tree, if the attacker stays in the system for T time steps, the attacker gets a utility of $T \cdot \alpha_{ij}$ and the defender gets a utility of $-T \cdot \delta_i - \nu$ where ν is the total cost of the defender actions. Note that if the game terminates with a successful eviction attempt, T is the number of time steps that the attacker has been in the system so far, and if the game terminates due to a failed eviction attempt or an observed active measure, $T = \tau + F$. In this utility model, the utility for the defender depends on the type of the attacker (through the coefficient δ_i), modeling that different attacker types may have goals resulting in varying degrees of damage to the defender.

Let Π_i be the set of pure strategies for player i in the game. Given the game model, the attacker's pure strategy set is the same as the set of TTPs, i.e., $\Pi_1 = \Gamma$. A pure strategy for the defender assigns an action for each information set at which the defender needs to take an action from $\Omega \cup \Phi$. Since the defender may choose to take an active measure at each time step without being noticed by the attacker, the set of pure strategies for the defender is exponential in size with respect to the number of active measures. When the attacker of type θ_i is following a pure strategy γ_j and the defender is following a pure strategy π_k (the k^{th} pure strategy in Π_2) the utilities for the players are nondeterministic due to the existence of the chance node. We use u_{jk}^1 and u_{ijk}^2 to represent the expected utility for the defender and the attacker respectively. To find the game equilibria, we derive the reduced normal form game [20] where actions at irrelevant information sets are omitted, and the computation of u_{jk}^1 and u_{ijk}^2 becomes necessary. Here we explain how these quantities can be computed.

A defender's pure strategy consists of a sequence of observational actions from Φ, followed by an eviction action from Ω. The set of pure strategies Π_2 can be enumerated by a recursive function $e(s, t, \tau)$, which, for each timestep $t \leq \tau$ adds the pure strategies where the defender uses an eviction action at timestep t to the set s. For expository purposes, we use a_k^t to denote the action that the defender plays in time step t when following π_k. Let ϵ_{ijk} represent the total expected time that the attacker will stay in the system. To compute ϵ_{ijk}, we introduce a helper quantity $\gamma_{ijk}^t \in \mathbb{R}$, which returns the probability that the game reaches timestep t, that is, the probability that the attacker is still in the system and has not noticed the defender's activity by timestep t. Since the defender takes at most one eviction action, let T denote the timestep the eviction action is taken ($T = \tau + 1$ if no eviction action is taken) where $T \leq \tau$. Then the probability γ_{ijk}^t can be obtained by the following equation:

$$\gamma_{ijk}^t = \begin{cases} \gamma_{ijk}^{t-1} \cdot (1 - q_j^{t-1}) \cdot (1 - y_j^{t-1}) & \text{if } 0 < t \leq T \\ 1 & \text{if } t = 0 \\ 0 & \text{if } t > T \end{cases}$$

Here we use y_j^{t-1} to denote the probability that the observational tactic used in timestep $t-1$ is observed by the attacker, i.e., y_{jl} where $a_k^{t-1} = \omega_l$. The equation can be interpreted as follows: Conditioned on that the game reached timestep t-1, the game can reach timestep t if (1) the defender has not taken an eviction action in previous timesteps (i.e., $t \leq T$); (2) nature does not reveal the attacker's true TTP to the defender (the second term); (3) the attacker does not notice the defender's observational tactic (the third term). Note that the second term depends on the defender's chosen strategy π_k, since an active measure might modify q.

The attacker's total expected time in the system is then given by summing the product of the probability of each possible outcome with the time the attacker would remain in the system if that

outcome were to occur. This is given by:

$$\epsilon_{ijk} = \sum_{t=1}^{T-1} \left(\gamma_{ijk}^{t} \cdot \left(y_j^t \cdot (F+\tau) + \left(1 - y_j^t\right) \cdot q_j^t \cdot t \right) \right) + \gamma_{ijk}^{T} \cdot \left(\chi_{jl} \cdot T + \left(1 - \chi_{jl}\right) \cdot (F+\tau) \right)$$

This expression sums the product of the probability of occurrence and the number of timesteps that the attacker remains in the system over each timestep. The inside of the summation handles all but the final timestep, which is treated differently since the defender's last action is always an eviction tactic, while the defender's other actions are always observational tactics. Inside the summation, the first term is the probability that the game has not already ended in previous timesteps. The game can end in the current timestep in two ways. The first is by the attacker noticing the defender's active measure, which results in the failed eviction penalty number of timesteps, handled by the term $y_j^t \cdot (F+\tau)$. The second way the game can end is by the defender learning the attackers TTP and evicting them on the current timestep, which is captured by the term $(1-y_j^t) \cdot q_j^t \cdot t$. Note that the this can only occur if the attacker has not noticed the observational tactic, hence the need to multiply by one minus the chance that the attacker observes the defender, and the term q_j^t depends on the defender's strategy since it may have been modified by an active measure in an earlier timestep. The term outside the summation handles the eviction action at the end of the defender's chosen strategy. This is the probability that the game does not end before this timestep, multiplied by both possible outcomes, either the eviction succeeds and the attacker is evicted on timestep T, given by $\chi_{jl} \cdot T$, or the eviction fails resulting in the failed eviction penalty, $(1-\chi_{jl}) \cdot (F+\tau)$.

The defender's utility u_{ijk}^1 is given by the expected time the attacker remains in the system, modified by how disruptive the attacker type i is, i.e., δ_i. The defender also pays a cost dependent on which defensive action is performed based on how disruptive it is κ_l.

Since a defender strategy that involves non-evicting actions may result in using different eviction actions depending on whether the attacker is observed by the defender, the expected cost can be found in a similar way to the expected time that the attacker remains in the system, summing the product of the probability of each outcome by the cost of that outcome, given by $\nu_{ijk} \in \mathbb{R}$. Due to the similarity of this equation to ϵ_{ijk}, we omit the details of this function. The defender's utility is then given by $u_{ijk}^1 = -\epsilon_{ijk} \cdot \delta_i - \nu_{ijk}$.

3.3 Computing Equilibria

In this paper, we are interested in both the Nash equilibrium and the Strong Stackelberg equilibrium since it is often hard to know how much the attacker knows about the defender's strategy. To solve the game when neither side knows the strategy profile of the other, we use the Gambit software tools for game theory [19] to obtain the Nash Equilibrium. If we assume that the defender acts first as the leader in a Stackelberg game, and the attacker then best responds to the defender's strategy, we formulate the problem as a mathematical program to obtain the Strong Stackelberg Equilibrium [6], which we solve using the Gurobi Optimizer [11]. The AI community has made significant progress towards solving Stackelberg games efficiently [17, 21, 30], and we leave the investigation of more efficient solution approaches for the OEG to future work.

Solving these equilibria in a Bayesian setting requires knowing or estimating the likelihood of each attacker type, the prior probability distribution p. Since the optimal strategy may change based on this value, and it is likely not possible to know the exact prior probabilities in practice, the defender may wish to know how sensitive their strategy is to variations in the prior probabilities. This may be determined by solving the equilibrium for a range of different probability distributions.

3.4 Example Quantification

As an example, we will consider a game consisting only of two time steps, i.e., $\tau = 2$. We will consider two attacker types: Attacker 1, inspired by a nation-state adversary, and Attacker 2, inspired by an organized crime group. The first attacker type will possess more sophisticated capabilities and be primarily interested in intelligence gathering, while the second attacker type possesses less sophisticated capabilities and is interested in gathering personal information to sell on the dark web. We also consider the case where there is no attacker present. To model this case, we introduce a third attacker type that represents the absence of an attacker. For the no attacker case, we assume that the observability of this attacker is always zero (modeling that there is no way to observe and evict a nonexistent attacker).

Initially, we consider two actions for the attacker, $TTP1$ and $TTP2$, representing two possible bins of TTPs. We consider $TTP1$ to be an advanced, less observable technique amenable to intelligence gathering; and $TTP2$ to be a more basic, observable approach developed for stealing personal information. $TTP1$ is then appropriate for the nation state attacker, and $TTP2$ is more appropriate for the criminal organization. While there is a TTP that is most appropriate for each attacker, an attacker might still choose to use a different TTP. For example, a nation state might choose to use less sophisticated techniques to deceive the defender or avoid revealing new exploits, while the criminal organization could potentially obtain services from a collaborative government for a cost. The defender observes $TTP1$ 10% of the time, and $TTP2$ 90% of the time. For the purposes of this example, $\delta_1 = 1$, $\delta_2 = 2$, $\alpha_{11} = \alpha_{22} = 1$, and $\alpha_{12} = \alpha_{21} = 0.5$. For the no attacker case, $\alpha_{3j} = 0$.

On the defender's side, the defender has three eviction actions, $\Omega = \{\omega_1, \omega_2, \omega_3\}$. ω_1 and ω_2 can fully evict TTP 1 and 2 respectively, and ω_3 is a pass tactic, an eviction action that never evicts the attacker, but comes with no additional cost. For expository purposes, we will refer to these actions as $e1$, $e2$, and p in tables and figures. The defender also has two observational actions, $\Phi = \{\phi_4, \phi_5\}$. ϕ_4 is the default action of waiting and ϕ_5 is an active measure. We will refer to them as w and a in tables and figures. Each of the defender's eviction actions successfully evicts the corresponding attacker TTP only, that is, $\chi_{jl} = 1$ when $j = l \in \{1, 2\}$. Otherwise, $\chi_{jl} = 0$. The costs of the eviction actions are 1 for $e1$ and 0.5 for $e2$, and 0 for p. The defender's first observational action is to wait, w, which does not alter the attackers observability, but is undetectable by any attacker TTP $x_4 = 1$, $y_{j4} = 0$, and has no cost to the defender. The defender's second action represents a camouflaging tool that

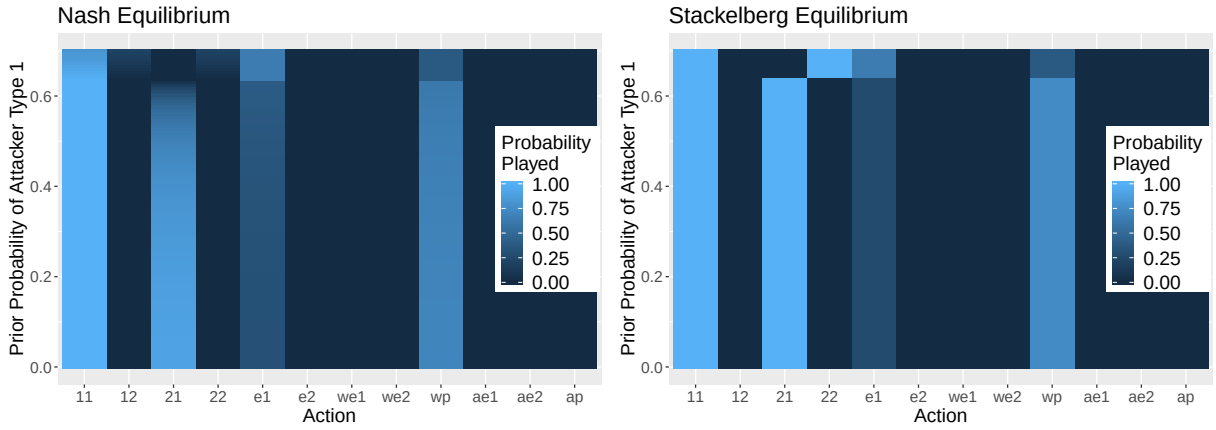

Figure 2: Nash and Stackelberg equilibria by attacker type distribution. Similar defender strategies permit a robust approach.

the defender can enable for a cost of 0.1, that makes the attacker more observable. We set $x_5 = 0.95$, $y_{15} = 0.9$, $y_{25} = 0.1$. By solving equilibria for this scenario for a range of prior probability values, a defender could identify a robust defensive strategy.

4 RESULTS

We evaluate our modeling approach by analyzing the example game provided in Section 3.4. Performing a sensitivity analysis, we illustrate how the defender can choose a robust strategy effective for a range of threat landscapes (Section 4.2). We quantify the defender's gain from using observability information by comparing the defender's utility at the equilibria compared to a random strategy (Section 4.3). Additionally, we consider a simple design problem and show how the OEG can enable the defender to choose the most effective design (Section 4.4). Lastly, we introduce an extended model to demonstrate scalability to practically useful problem sizes (Section 4.5).

4.1 Example Equilibria

Given prior probabilities for each attacker type, we can compute Nash and Stackelberg equilibria to obtain optimal strategy profiles for each side. Recall from Section 2.3 that Nash equilibrium assumes that both sides act simultaneously, while the Stackelberg equilibrium assumes that the defender acts first and commits to a strategy, which the attacker observes and then plays the best response. We first consider the case where the prior probability of attacker type one is 0.35, attacker type two is 0.35, and no attacker is 0.30. In the Nash setting, the first attacker type always plays $TTP1$, while attacker type two plays $TTP1$ 81% of the time and $TTP2$ the other 19%. The defender plays a mixed strategy of using $e1$ 33% of the time and wp the remaining 67%. The Strong Stackelberg equilibrium on the other hand, is for both attackers to play a pure strategy of $TTP1$, while the defender adopts a similar mixed strategy to the NE, playing wp 74% of the time, and $e1$ the other 26%.

4.2 Sensitivity Analysis

While solving equilibria for a specific attacker type distribution is useful, the defender may not know the exact prior probability values in practice. In this case, a sensitivity analysis enables the defender to choose a strategy robust to a range of possible prior values.

Figure 2 shows resulting optimal Nash and Stackelberg strategy profiles for various values of prior probabilities of attacker type one and two. For ease of visualization, values are shown for where the prior probability of no attacker is 0.30. The first four values on the horizontal axis are two-digit representations of the attacker's strategy, the attacker type followed by the attacker's TTP. For example, 11 denotes attacker type 1 plays $TTP1$, while 12 denotes attacker type 1 plays $TTP2$. The remaining values indicate the defender's strategy. The shading of the figure indicates with what probability each action is played in the mixed strategy at equilibrium.

For the Nash equilibrium, both attackers primarily play $TTP1$, with attacker type two occasionally using $TTP2$. As the prior probability of attacker type one increases, attacker type two increasingly switches from playing $TTP1$ to $TTP2$, with attacker type one also starting to use $TTP2$ some of the time for very high prior probabilities. The defender has two mixed strategies consisting of wp and $e1$. For the Stackelberg equilibrium, there are two optimal strategy profiles depending on the prior probability distribution. Attacker type one always adopts a pure strategy of $TTP1$, while attacker type two mostly plays $TTP1$, except for when the prior probability of attacker type is high. The defender's strategy is very similar to the NE case, consisting of wp and $e1$.

Interestingly, the defender frequently plays the wp strategy, showing the power of waiting to reduce uncertainty about the attacker. Additionally, the defender only needs to choose from two possible strategies for the entire range of the attacker type distribution (for when there is no attacker 30% of the time) when considering both Nash and Stackelberg equilibria, allowing the defender to adopt an optimal approach even when the prior probabilities are not fully known. The defender's optimal strategies are also similar between Nash and Stackelberg equilibria, meaning that

the defender may be able to choose an optimal strategy that is robust to the attacker knowing the defender's strategy.

Since the OEG has many parameters that must be set, we performed an exploratory analysis to understand how the game parameters affect the equilibrium strategies for each side. In this study, we randomly sampled 400000 concrete games from the parameter space. Each game used the same set of tactics and number of attackers as described in Section 3.4, but with different parameter values. The game parameters were chosen randomly between $0 - 1$, with the exception of the δ parameters which were chosen between $1 - 10$. We performed this sweep for both the Nash and Stackelberg equilibriums, with half of the trials solved for Nash and half for Stackelberg equilibria. Timesteps ranged between $1 - 10$. After recording the equilibrium strategies for each side for each trial, we constructed decision trees to predict the probability that each pure strategy is played in the mixed strategy equilibrium, using the game parameters as predictors. Decision trees were constructed using the `ctree` [13] package for the R language for statistical computation [28].

The decision tree shown in Figure 4 predicts the number of timesteps that the defender chooses to wait on average, for the trials that used $\tau = 10$. Each node in the tree represents a decision point, and is labeled with a decision variable. Edges are labeled with the decision criteria. The leaves of the tree show the distribution of the predicted variable from data instances that satisfy the criteria on every edge from the root to the leaf. This tree shows that the defender should primarily consider the prior probability that there is no attacker in the system (the root node), and the observability of the attacker's TTPs (the remaining nodes) when deciding how long to wait. The leaves on the right side of the tree are generally higher, indicating that the defender waits longer when the chance that their is no attacker in the system is higher. This makes sense intuitively since the defender can minimize the chance that they pay an unnecessary eviction cost by not taking premature action when the presence of an attacker is less likely. The defender also waits more often when the attackers TTPs are more observable, since the defender is more likely to successfully evict the attacker by observation in these cases. We also observe that the defender waits less on average when the observability for the two available TTPs are different. This is likely due to the defender being able to exploit their knowledge about the attackers observability, for example, if TTP1 is very observable and TTP2 is very stealthy, and the defender cannot observe the attacker after waiting a few timesteps, the defender concludes that if an attacker is present they are likely using the stealthier TTP and can thus make an educated eviction attempt.

While we only show one decision tree in this paper due to space constraints, we also constructed trees to predict the probability that the defender should use an active measure or pass, and the probability that an attacker should use a particular TTP. We found that the attacker primarily considered the amenability of each TTP to themselves and the other attacker, as well as the prior probability values. The defender's probability to pass rather than evict blindly could be predicted primarily with the prior probability that no attacker is present, the observability of each TTP, and the cost of eviction actions. Using an active measure was predicted by considering the effectiveness of the active measure and the observability of the attackers TTPs. Interestingly, whether the game was solved for Nash or Stackelberg equilbria did not appear as a predictor in the decision trees, supporting the idea that defenders can generate strategies that apply to both contexts. While used for interpretation purposes in this paper, we note that the general approach of constructing machine learning models over a parameter space could also be used to approximate equilibrium strategies. Such an approach could mitigate scalability concerns by enabling systems to quickly adapt when the parameters are not known until runtime.

4.3 Comparison to Uniform Random

To evaluate the utility of considering this observability information to the defender, we compare the defender's utility from playing the strategy specified by the equilibria to an uninformed uniform random strategy (that is, the mixed strategy where the defender chooses all available actions with equal probability). When considering the simultaneous move case, we test against an attacker who plays according to the NE. When considering the Stackelberg case, the attacker best responds to the defender's uniform random strategy. The left side of Figure 3 shows the defender's utility from playing according to the equilibrium, and according to a uniform random approach for both Nash and Stackelberg assumptions. The vertical axis shows the defenders utility, while the horizontal axis shows the prior probability of attacker type one.

These graphs show a significant utility benefit from an informed eviction policy. When operating under the Nash assumptions, there is a 20% difference between the equilibrium strategy and the uniform random approach. Under Stackelberg conditions, utility gain ranges between $22 - 32\%$.

4.4 Informing Architectural Choices

We evaluate how modeling observability can improve utility through informing system design choices that can be modeled as changes to the parameters of the Observable Eviction Game. The concrete illustrative design problem we consider is how many honeypot decoy systems the defender should build into the system, assuming that each decoys has costs and benefits. The defender can deploy at most 17 decoys. With each additional decoy, the defender pays a flat cost of 0.03 to their utility, but the probability that the attackers' TTPs are observed is increased. Each additional decoy makes $TTP1$ 5% more observable, and $TTP2$ 0.5% more observable.

The right side of Figure 3 shows the optimal number of decoys to build into the system based on the attacker prior probability for Nash and Stackelberg assumptions (left), and the defender's change in utility from using the decoys (right). When using Nash assumptions, the defender should use all 17 decoys when the prior probability of attacker type one is $< 65\%$, and use zero otherwise. When considering Stackelberg assumptions, the defender should always use all decoys.

The right side of Figure 3 shows the improvement the defender obtains by building the optimal number of decoys into the system for various prior probabilities, for both Nash and Stackelberg equilibria. Where the cost of building decoys into the system outweighs the benefits of greater attacker observability, zero decoys are chosen and the fitness change is zero. When beneficial however, the defender can gain up to 0.3 utility, a percentage difference of up to 28%.

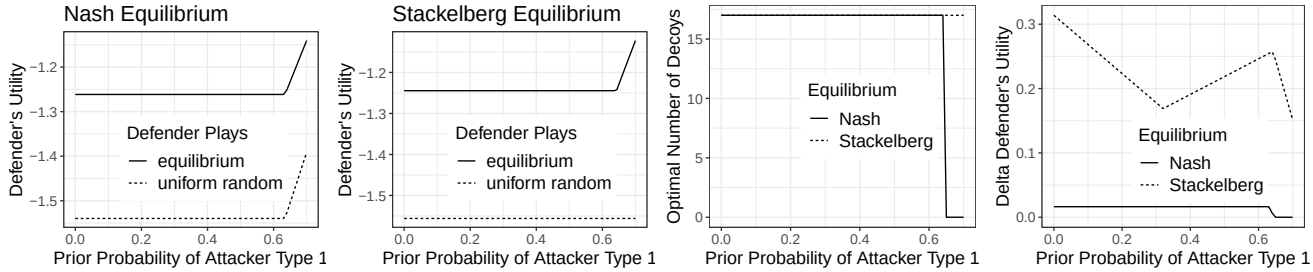

Figure 3: Left two graphs: Defender's utility by attacker type distribution for Nash and Stackelberg equilibria, showing utility improvement from playing the equilibria. Right two graphs: Optimal system design and change in defender's utility by attacker type distribution, showing the utility improvement the defender obtained by selecting an optimal design.

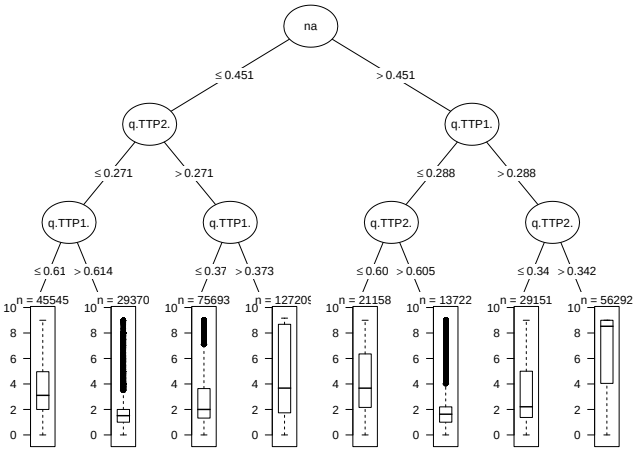

Figure 4: Decision tree predicting the number of timesteps the defender waits on average in an equilibrium mixed strategy.

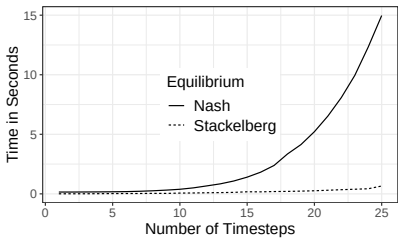

Figure 5: Runtime versus number of timesteps for Nash and Stackelberg equilibria.

4.5 Scalability

To evaluate the scalability of the Observable Eviction Game to real-world sizes, we extended the provided quantification from Section 3 by relaxing the limit on the number of timesteps τ. For scalability experiments, we assume a uniform distribution between the three attacker types. To evaluate scalability, we solve the game for an increasing number of timesteps until the game could not be solved in a time budget of one hour for both Nash and Stackelberg equilibria. Each runtime is the median of five executions. Scalability experiments were performed on a desktop computer running Ubuntu 18.04 with a 3.5GHz CPU and 8GB of RAM. Figure 5 shows the results of this experiment. While it is expected that computing Nash equilibria for large games is much more computationally difficult, the results show that solving for 25 timesteps can be done quickly (under 15 seconds). However, solving the Nash Equilibrium for 26 timesteps could not be computed in the time budget of one hour.

Stackelberg games are known to be computationally easier to solve, and we found that games of up to 65 timesteps can be solved in seconds, while 300 timesteps can be solved in under an hour. Our purpose in this paper is to introduce a model that captures important aspects of APT defense that cannot be handled by existing approaches, and we leave investigating efficient solutions strategies to future work, though we note that the OEG can model zero-sum situations which are easier to solve, and the AI community has made significant progress in efficiently solving large games [4, 27, 30].

4.6 Discussion

This section provides a discussion of limitations (Section 4.6.1) and opportunities for further work (Section 4.6.2).

4.6.1 Limitations. For this investigation, we make a number of simplifying assumptions, such as restricting the number of attacker types and available TTPs, and limiting the defender to a small set of actions. We do not claim that this example quantification generalizes exactly to the real world security landscape. Rather, we attempted to capture the fundamental concepts in the interaction, a defender that defends against multiple attacker types, a choice between gathering more information or attempting to evict, a defender with limited observability in the strategy of the attacker, an attacker that observes the defender's eviction attempt and reacts, and a general-sum payoff structure where the defender balances between evicting the attacker and avoiding disruption to the system. While the results will be sensitive to the values we choose for the payoff structure, we argue that the results nonetheless show the utility of the OEG as a model of observability in APT defense,

and demonstrate how this modeling approach facilitates reasoning about the choice between information gathering and evicting attackers in self-* systems, as well as design choices involving observability that can be modeled as parameters to the OEG.

4.6.2 Future Work. This work opens the door for a number of promising research directions. While we assume optimal behavior from the attacker, in practice humans do not often behave optimally. The attackers might have subjective preferences in TTPs. This captures the fact that attacker groups have familiarity with certain tools, and are accustomed to operating in a certain way. Deviating from these normal behaviors would be costly to the attacker, so they might choose to use a suboptimal strategy from the defender's perspective. Applying a model of bounded rationality from game theory could help model this dynamic.

Another direction for investigation is how the defender's strategy should change in multiple interactions with the same attacker groups. For example, a defender may choose to allow the attacker to remain in their system even when they know that an eviction would be successful to gain more information about the attacker for use in future interactions. Future work could also investigate defending against concurrent attacks, or attackers that can use multiple TTPs at once. Additionally, future work should model attacker TTPs at a lower level of abstraction, such as paths through an attack tree.

5 RELATED WORK

Work by Pawlick et al. [22] is the most similar to our work, also balancing the defender's choice between evicting an attacker versus waiting for more information. Their approach uses an infinite horizon MDP, and the defender gains information about the attacker by keeping them in honeypots. However, this work does not consider multiple attacker types with different goals and capabilities. Their approach also does not model how the information gained by the defender improves the defender's eviction ability, and assumes that an eviction attempt would always be successful. The OEG explicitly models the fact that the defender's ability to evict the attacker depends on the information that they have obtained through observation.

FlipIt [29] is a two-player game that reasons about security scenarios where an attacker may periodically gain full control of an asset, with each side trying to maintain control as much as possible. Uncertainty is modeled by not revealing the owner of a resource until a player actually moves, which comes at a cost. This work is similar to ours since there is some uncertainty about the attacker's actions. The FlipLeakage extension [10] by Farhang et al. is most similar to our work. In this model, the attacker gains information about the defender while they control assets, and this information improves the attacker's capability. By contrast, we consider multiple types of attackers with different goals and capabilities, and we also model how information is leaked from the attacker to the defender, with information about the attacker's identify and methods improving the defensive capabilities of the defender.

Many other approaches also apply game theory to different dimensions of the problem of advanced persistent threats, including using prospect theory to model attackers with bounded rationality [31], modeling attackers with continuous payoff distributions [15] to facilitate reasoning about the defender's uncertainty of the attacker's payoffs in Bayesian Stackelberg Security Games, and using a risk mitigation approach [23] in a zero-sum game to manage uncertainty in attacking payoff values, actions, and system state. Other work addressing APTs includes modeling defense-in-depth as a finite sequence of nested two-person zero-sum games [24]. Fang et al. [9] model an APT as a security game with the attacker choosing a path through the system. This model can be used to determine the attacker's optimal path through the system, but does not explicitly address uncertainty in observability on either side or consider multiple attacker types. Marecki et al. [18] investigate dealing with multiple adversary types over multiple rounds of Stackleberg games. While these approaches represent important steps towards well defended self-protecting systems, our approach is the first to include the value of increased information over time in both attack and defense, specific tactics to gain information for both sides, and the need to reason about the effects of observability of actions on the behavior of the other player.

6 CONCLUSION

Advanced persistent threats represent a significant challenge for self-* systems and security software engineering more generally. Uncertainty about the identity of the attacker, including their TTPs and goals, complicates the defender's task. Information flow between the attacker and defender influences the results of the interaction for both sides, yet the observability of both sides in the APT interaction is a neglected area in the automated defense of systems. We presented a model of APT interaction, the Observable Eviction Game, that captures uncertainty about the kind of attacker, the attacker's actions in the system, and the attacker's limited observability into the knowledge of the defender. We evaluated this model with an example that we compare to an uninformed defensive strategy, as well as an empirical sensitivity analysis showing how our model enables self-* systems to make robust decisions in uncertain security environments. A comparison of using equilibria generated form the model versus a random approach showed that the model results in a significant improvement of 20-32%. Lastly, we showed that the model has potential to scale to practically useful time horizons, such as 300 timesteps in the Stackelberg case. These results are a step towards designing self-* software systems that can automatically defend themselves from an array of APT attackers with awareness of information exchange on both sides.

ACKNOWLEDGMENT

This research supported in part by the National Science Foundation (CCF-1618220). This material is based upon work supported by the NSA under Award No. H9823018D0008. Co-author Fang is supported in part by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). This research supported in part by a grant from the CyLab Security and Privacy Institute at Carnegie Mellon University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

[1] [n.d.]. Advanced Persistent Threat Groups. https://www.fireeye.com/current-threats/apt-groups.html. Accessed: 2018-04.

[2] [n.d.]. Inside the cyberattack that shocked the US government. https://www.wired.com/2016/10/inside-cyberattack-shocked-us-government/. Accessed: 2018-04.

[3] [n.d.]. Target: 40 million credit cards compromised. http://money.cnn.com/2013/12/18/news/companies/target-credit-card/index.html. Accessed: 2018-02-15.

[4] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.

[5] Shang-Wen Cheng, David Garlan, and Bradley Schmerl. 2009. Evaluating the Effectiveness of the Rainbow Self-adaptive System. In *Proc. of Workshop on Soft. Eng. for Adaptive and Self-Managing Syst. (SEAMS '09)*. IEEE Computer Society, Washington, DC, USA.

[6] Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*. ACM, 82–90.

[7] Premkumar T Devanbu and Stuart Stubblebine. 2000. Software engineering for security: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. ACM, 227–239.

[8] Ahmed Elkhodary and Jon Whittle. 2007. A survey of approaches to adaptive application security. In *Software Engineering for Adaptive and Self-Managing Systems, 2007. ICSE Workshops SEAMS'07. International Workshop on*. IEEE, 16–16.

[9] Xupeng Fang, Lidong Zhai, Zhaopeng Jia, and Wenyan Bai. 2014. A game model for predicting the attack path of APT. In *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*. IEEE.

[10] Sadegh Farhang and Jens Grossklags. 2016. FlipLeakage: a game-theoretic approach to protect against stealthy attackers in the presence of information leakage. In *Int. Conf. on Decision and Game Theory for Security*. Springer.

[11] LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[12] John C Harsanyi and Reinhard Selten. 1972. A generalized Nash solution for two-person bargaining games with incomplete information. *Management Science* 18, 5-part-2 (1972).

[13] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. 2015. ctree: Conditional inference trees. *The Comprehensive R Archive Network* (2015), 1–34.

[14] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, 1 (2003).

[15] Christopher Kiekintveld, Janusz Marecki, and Milind Tambe. 2010. Methods and algorithms for infinite Bayesian Stackelberg security games. In *International Conference on Decision and Game Theory for Security*. Springer.

[16] Richard Kissel. 2011. *Glossary of key information security terms*. Diane Publishing.

[17] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. 2017. Robust Stackelberg Equilibria in Extensive-Form Games and Extension to Limited Lookahead. arXiv:cs.GT/1711.08080

[18] Janusz Marecki, Gerry Tesauro, and Richard Segal. 2012. Playing repeated Stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems.

[19] Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. [n.d.]. Gambit: Software Tools for Game Theory, Version 16.0.1. http://www.gambit-project.org. Accessed: 2018-02.

[20] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. 2007. *Algorithmic game theory*. Vol. 1. Cambridge University Press Cambridge.

[21] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing Games for Security: An Efficient Exact Algorithm for Solving Bayesian Stackelberg Games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.

[22] Jeffrey Pawlick, Thi Thu Hang Nguyen, and Quanyan Zhu. 2017. Optimal Timing in Dynamic and Robust Attacker Engagement During Advanced Persistent Threats. *CoRR* abs/1707.08031 (2017). arXiv:1707.08031

[23] Stefan Rass, Sandra König, and Stefan Schauer. 2017. Defending against advanced persistent threats using game-theory. *PloS one* 12, 1 (2017).

[24] Stefan Rass and Quanyan Zhu. 2016. GADAPT: A Sequential Game-Theoretic Framework for Designing Defense-in-Depth Strategies Against Advanced Persistent Threats. In *Decision and Game Theory for Security*, Quanyan Zhu, Tansu Alpcan, Emmanouil Panaousis, Milind Tambe, and William Casey (Eds.). Springer International Publishing, Cham.

[25] Bradley Schmerl, Gabriel A Moreno, Andrew Mellinger, Javier Camara, and David Garlan. 2014. *Architecture-based self-adaptation for moving target defense*. Technical Report. Carnegie Mellon University Pittsburgh United States.

[26] Bruce Schneier. 1999. Attack trees. *Dr. Dobb's journal* 24, 12 (1999).

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[28] R Core Team et al. 2013. R: A language and environment for statistical computing. (2013).

[29] Marten Van Dijk, Ari Juels, Alina Oprea, and Ronald L Rivest. 2013. FlipIt: The game of "stealthy takeover". *Journal of Cryptology* 26, 4 (2013).

[30] Jakub Černý, Branislav Boýanský, and Christopher Kiekintveld. 2018. Incremental Strategy Generation for Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the 2018 ACM Conference on Economics and Computation (EC '18)*. ACM, New York, NY, USA, 151–168. https://doi.org/10.1145/3219166.3219219

[31] Liang Xiao, Dongjin Xu, Caixia Xie, Narayan B. Mandayam, and H. Vincent Poor. 2017. Cloud Storage Defense Against Advanced Persistent Threats: A Prospect Theoretic Study. *IEEE J.Sel. A. Commun.* 35, 3 (March 2017).