Optimal Strategies for Detecting Data Exfiltration by Internal and External Attackers

Karel Durkota[1](✉), Viliam Lisý[1], Christopher Kiekintveld[2], Karel Horák[1], Branislav Bošanský[1], and Tomáš Pevný[1,3]

[1] Deptartment of Computer Science, FEE, Czech Technical University in Prague, Prague, Czech Republic
{karel.durkota,viliam.lisy,karel.horak,branislav.bosansky, tomas.pevny}@agents.fel.cvut.cz

[2] Computer Science Department, University of Texas at El Paso, El Paso, USA
cdkiekintveld@utep.edu

[3] Cisco Systems, Inc., Prague, Czech Republic

Abstract. We study the problem of detecting data exfiltration in computer networks. We focus on the performance of optimal defense strategies with respect to an attacker's knowledge about typical network behavior and his ability to influence the standard traffic. Internal attackers know the typical upload behavior of the compromised host and may be able to discontinue standard uploads in favor of the exfiltration. External attackers do not immediately know the behavior of the compromised host, but they can learn it from observations.

We model the problem as a sequential game of imperfect information, where the network administrator selects the thresholds for the detector, while the attacker chooses how much data to exfiltrate in each time step. We present novel algorithms for approximating the optimal defense strategies in the form of Stackelberg equilibria. We analyze the scalability of the algorithms and efficiency of the produced strategies in a case study based on real-world uploads of almost six thousand users to Google Drive. We show that with the computed defense strategies, the attacker exfiltrates 2–3 times less data than with simple heuristics; randomized defense strategies are up to 30% more effective than deterministic ones, and substantially more effective defense strategies are possible if the defense is customized for groups of hosts with similar behavior.

Keywords: Data exfiltration detection · Game theory · Network security

1 Introduction

A common type of cyber attack is a *data breach* which involves the unauthorized transfer of information out of a system or network in a process called *information exfiltration.* Information exfiltration is a major source of economic harm from cyber attacks, including the loss of credit card numbers, personal information, trade secrets, unreleased media content, and other sensitive data. Many recent attacks on high-profile companies (e.g., Sony Pictures and Target) have involved large amounts of data theft over long periods of time without detection [7]. Improving strategies for detecting information exfiltration is therefore of great importance for improving cybersecurity.

We focus on methods for detecting information exfiltration activities based on detecting anomalous patterns of behavior in user upload traffic. An important advantage of this class of detection methods is that it does not require knowledge of user data or the ability to modify this data (e.g., to use honey tokens). To better understand the strategic aspects of anomaly detection and information exfiltration we introduce a two-player game model that captures the defender and attacker decisions in a sequential game. While we focus mainly on the information exfiltration example, note that raising alerts based on detecting anomalous behavior is a common strategy for detecting cyber attacks, so our model and results are relevant beyond just information exfiltration.

One of the novel aspects of our game model is that we consider both insider and outsider threats. A recent McAfee report [1] states that that 40% of serious data breaches were caused by insiders trusted by the organization, while the remaining 60% are due to outside attackers infiltrating the enterprise. Since both types of attacks are prevalent we consider both cases. There are significant differences between insiders and outsiders for information exfiltration. One difference is that an insider *knows* his typical behavior already and can use this knowledge to evade detection, while an outsider must *learn* this behavior from observation. A second difference is that insiders may be able to *replace* their normal activity with malicious activity, while an outsider's actions will be observed *in addition* to the normal activity. We model both of these key differences and examine how they affect both attacker and defender behavior in information exfiltration.

We introduce a sequential game model in which the objective of the attacker is to exfiltrate as much data as possible before detection, and the objective of the defender is to minimize the data loss before detection. The defender monitors the amount of data uploaded to an external location (e.g., Dropbox or Google Drive) and raises an alert if the traffic exceeds a (possibly randomized) threshold in a give time period. Some network companies use only uploaded data volume as feature to detect the data exfiltration. In our paper we follow this approach, however, our algorithm allows using more features as well. The defender is constrained to policies that limit the expected number of false positives that will be generated. The attacker chooses the amount of data to exfiltrate in each time period. We model both insider and outsider attackers, and both additive and replacing attacks. In the additive attack the total traffic observed is the sum of the normal user activity and the attack traffic, while in the replacing attack only the attack traffic is observed by the defender. Outsider attackers also receive an observation of the user traffic in each time period that can be used to learn the behavior pattern (and therefore infer something about the likely threshold for detection).

Our first main contribution is the exfiltration game model that considers the differences between insider and outsider attackers. Our second contribution is a set of algorithms for approximating the optimal strategies for both defender and attacker players in these games. For outsider attackers, we use *Partially Observable Markov Decision Process* (POMDP) to model the learning process for the attacker. We also consider randomized policies for the defender, since it has been shown that static decision boundaries can be quickly learned [4] and randomizing can mitigate successful attacks [11]. Our third main contribution is an experimental analysis of a case study based on real-world data from a large enterprise with 5864 users connecting to a Google drive service for 12 weeks. We compute optimal strategies against different classes of attackers, and examine the characteristics of the strategies, the effects of randomization and attacker learning, and the robustness of strategies against different types of attackers. We show that with the computed defense strategies, the attacker exfiltrates 2–3 times less data than with simple heuristics; randomized defense strategies are up to 30% more effective than deterministic ones, and substantially more effective defense strategies are possible if the defense is customized for groups of hosts with similar behavior.

2 Related Work

Several previous works focus on detection and prevention of data exfiltration. A common approach is anomaly detection, e.g., a system can automatically learn the structure of standard database transactions on the level of SQL queries and raise alerts if a new transaction does not match this structure [5,10]. An alternative option is to create signatures of the sensitive data based on their content and detect if this content is sent out [14]. The signatures should be resilient against the addition of noise or encryption, such as wavelet coefficients for multimedia files, which are resilient against added noise. Data exfiltration can also be partially mitigated by introducing automatically generating honey-tokes, a bait documents that rise alarm when are opened or otherwise manipulated [2]. These works do not consider volume characteristics of the traffic as means of detecting data exfiltration and do not study the learning process of the external attacker, which are the focus of this paper. A commonly studied option of exfiltration is to use a covert channel and hide the communication in packet timing differences of DNS requests [19]. If the covert channel increases the volume of traffic to some service, the methods presented in this paper can help with its detection. More general data exfiltration motivations and best practices to protect the data are described in [13].

Data exfiltration and similar security problems have been previously studied in the framework of game theory. Liu et al. [12] propose a high level abstract model of insider threat in the form of partially observable stochastic game (POSG). They propose computing players' strategies using generic algorithms developed for this class of games, which have very limited scalability. The instance of the game they analyze in the case study focuses on data corruption and not exfiltration. Our work can also be seen as a special instance of POSG, but we provide more scalable algorithms to solve it and analyze the produced strategies in the context of data exfiltration.

Similar to our work, [9] investigates selecting thresholds for intrusion detection systems protecting distinct subsets of a network. The goal is to find optimal trade-offs between false positives and the likelihood of detection of an attack, which is simultaneously executed on a several subsets of the network. The attacker cannot decide what action to execute, only which systems to attack; nor he has an ability to learn the possible thresholds before the attack is conducted. McCarthy et al. [15] use POMDP to compute defender's optimal sequence of (imperfect) sensors to accumulate enough evidence whether data exfiltration over Domain Name System queries is happening in the network or not. However, unlike our paper, they assume non-adaptive attackers. Lisý et al. [11] investigated the effect of randomization of detection thresholds to strength of attacks and their overall cost to the defender. Our modeling of insider attacks is similar to this work. In contrast to our work, the attacker has perfect knowledge about the detector and the attacked system before the attack.

3 Game Theoretic Model

We model the problem of data exfiltration as a dynamic (sequential) game between the defender (network administrator) and the attacker trying to exfiltrate data over the network. We first discuss the basic setting of the game and focus on the fundamental differences between the insider and outsider and whether their activity is added to or replaces the normal traffic of the host. Then, we define the exact interaction between the attacker and the defender.

The **defender** monitors the volume of data uploaded by each network host[1] to a specific service over time, in time windows of constant length, e.g., 6 h. His action is to select a detection threshold θ from the set of available thresholds Θ. If the volume of the host's upload surpasses θ in a time window, an alarm is raised and the activity of the user is inspected by the administrator.

The **attacker** controls one of the users and tries to upload as much data as possible to the selected service before being detected. His actions are to choose the amount of data $a \in A \subseteq \mathbb{N}_0$ he exfiltrates in the next time window. If this amount (possibly) combined with the host's standard activity is below θ, the attacker immediately receives reward a and the defender suffers a penalty proportional to a. In this latter case, the attacker can act again in the following time window.

Since each **host** in a company may have different pattern of standard activity, the defender might want to set the threshold for each of them individually. However, this approach can be laborious in big companies and individual users rarely produce enough data for creating high quality models of their behavior. Therefore, it is common to create groups of hosts with similar behaviors and

[1] Hosts are non-strategic actors in the game considered to be part of the environment.

reason about these groups instead. In our models we refer to each group as a host type λ from a set of all types Λ. We assume both players know the probability $P(\lambda)$ that a randomly selected host in the network is of type λ. Each host type is characterized by its common activity pattern in the form of the probability $P(o|\lambda)$ that a host of type λ transfers the amount of data $o \in O \subseteq \mathbb{N}_0$ in a time interval. We call these amounts observations, since they are the information observed by the external attackers.

The standard host's activity can sometimes surpass the selected threshold even without any attacker's activity and the host is still inspected. These *false positives* take a lot of time for the administrator to investigate and are typically a key concern in designing IDS. To capture this constraint we require the defender's strategies to have an expected number of false positives bounded by a constant FP.

3.1 Outsider Vs. Insider

The outsider is an external attacker who compromises a host in the computer network to exfiltrate data. Although the outsider may know what types (groups) there are in the company (secretaries, IT admins, etc.), they often *do not know* which host type they compromised. However, they can observe the activity of the compromised host in each time window and update their belief about its type. Starting an aggressive exfiltration is likely not the best strategy, since once attacker surpasses a threshold, he is detected and the attack is stopped. However, conducing too much observation may cause that the host is disconnected or turned off before any exfiltration was conducted; or that the user's normal traffic surpasses the threshold, in which case the host is inspected and the attacker may be detected; or the data may become useless. We model this risk by discounting future rewards t time steps ahead with γ, where $\gamma \in (0, 1)$. The outsiders must cautiously weigh how much to exfiltrate at the current time step versus how long to learn the host type to increase future rewards. Typically, he would first emphasize learning with little data exfiltration, and proceed to more aggressive exfiltration when he is more certain about the host type.

The insiders are the regular users of the network and they *know* their host type and the deployed defenses. If the defender sets a fixed threshold for each host type, an insider can exfiltrate exactly at that threshold (we assume that the amount of data has to surpass the threshold to trigger the alarm). Such a defense strategy is not optimal, and the defender should minimize the insiders certainty about the threshold by randomization of her choices.

3.2 Additivity Vs. Replacement

Consider a situation where the host uploads o MB and has set threshold θ. Then the attacker can exfiltrate at most $\theta - o$, if he does not want to surpass the threshold. Additivity is important mainly for the external attacker operating on the host without its user's knowledge. However, we allow additivity even for the insider in our model so that we can analyze the effect of incomplete knowledge of the external attacker with all other conditions equal. Assuming that the attacker completely replaces the existing host traffic with the exfiltration is more natural for the insider. However, even the external attacker can, in principle, throttle or to completely block the standard user's traffic to increase his own bandwidth for exfiltration. We analyze combinations of scenarios when the attacker is insider/outsider and when the user's normal traffic is and is not present.

3.3 Formal Definition of Game Model

We have introduced the following components: Λ is the set of host types and $P(\lambda)$ the probability of their occurrence; O (resp. A) is the set of possible amounts of data that the hosts (resp. attackers) can upload; $P(o|\lambda)$ describes host's standard activity; Θ is the set of thresholds the defender can choose; FP is the defender's maximal false positive rate; γ is the discount factor.

In our model, we assume that the network administrator models the user's normal traffic using discrete representation, e.g., histograms. In that case, the attacker and defender's action are also discrete, as they have no incentive to choose actions between the discrete values.

Defender's Strategy. We allow *mixed* (or randomized) strategies in form of $\sigma(\theta|\lambda)$, where the defender chooses a probability distribution of thresholds θ given host-type λ. As a special case, the defender may choose a *pure* strategy $\psi : \Lambda \rightarrow \Theta$, a threshold for each host-type λ. Let Ψ and Σ be the set of all pure and mixed strategies, respectively. A valid defender strategy σ must satisfy the false positive constraint $\sum_{\lambda\in\Lambda}\sum_{\theta\in\Theta}\sigma(\theta|\lambda)P(\lambda)FP(\theta|\lambda) \leq FP$, where $FP(\theta|\lambda) = \sum_{o\in O:o>\theta} P(o|\lambda)$ is type λ's amount of false positives if threshold is θ.

Attacker's Strategy. In the course of the attack, the attacker follows a policy which prescribes what action he should take when he played actions $a_1, \ldots, a_k$ and saw observations $o_1, \ldots, o_k$ so far [3]. We assume, that the defender chooses his threshold strategy first, and the attacker acts afterwards, knowing the defender's strategy (we will discuss it in section Solution Concept. In such a case, the attacker acts only against the nature, without adversarial actor, and Partial Observable Markov Decision Processes (POMDPs) can be used to reason about (approximately) optimal attacker's policies for the attacker. In the POMDP the attacker is not required to remember the entire history of his actions and observations. Instead, he can capture all relevant information he has acquired in the course of the interaction in the form of a belief state $b \in \Delta(\Lambda \times \Theta)$, which is a probability distribution over possible host types and threshold settings. We can then define attacker's policy based on his belief as $\pi : \Delta(\Lambda \times \Theta) \rightarrow A$. In the course of the interaction, the attacker keeps track of his belief b using a Bayesian update rule when he takes the last action and observation into account. Based on his current belief, he chooses action $\pi(b)$ to play. We denote the set of all belief-based policies as Π.

Note that the insider knows the host type from which he exfiltrates the data (it is his own host machine), therefore, there is no need to update belief based on the observation. Therefore, for the insider the attack policy is to choose an action from A. Mixed strategies are probability distribution among these choices.

Utilities. We define the attacker's expected utility as $u_a(\sigma, \pi)$, which is the discounted total expected amount of exfiltrated data using policy π against the defense strategy σ.

We define the defender's utility as $u_d(\sigma, \pi) = -Cu_a(\sigma, \pi)$, where $C > 0$. That means, that players have opposing objectives and their payoffs are proportional. Typically $C > 1$, which means that the defender suffers more than the attacker gains.

Solution Concept. Game theory provides a variety of solution concepts and algorithms for analyzing games with different characteristics. In zero-sum games and their slight generalizations, such as our payoff structure, many of these solution concepts lead to the same strategies. We use Kerckhoffs's principle, which assumes that the attacker knows the defender's algorithm or can conduct surveillance of the defender's behavior, therefore, knows his strategy. In game theory, Stackelberg equilibrium corresponds to such assumptions, where *leader* (the defender) acts first, by choosing strategy σ. Then, *follower* (the attacker), plays any *best response* strategy, which maximizes the attacker's utility against leader's strategy σ.

Definition 1 (best response). *Attacker plays* best response *if it maximizes the attacker's expected utility, taking the defender's strategy as given. Formally,* $\pi \in BR_a(\sigma)$ *iff* $\forall \pi' \in \Pi : u_a(\sigma, \pi) \geq u_a(\sigma, \pi')$.

In zero-sum games, all attacker's best responses have the same expected utility to the defender and the attacker, therefore, there is no need to distinguish between specific best responses. Because we use approximative algorithm to compute the attacker's policy, we focus on finding approximate ϵ-SE. The defender's strategy in ϵ-SE guarantees, that the defender's utility cannot be improved by a factor of more than $1 + \epsilon$ in the exact SE.

Definition 2 (ϵ-Stackelberg equilibrium (ϵ-SE)). *Let* $\epsilon \in (0, 1]$. *Solution profile* (σ^*, π^*) *where* $\pi^* \in BR_a(\sigma^*)$ *belongs to ϵ-SE, if* $\forall \sigma \in \Sigma, \forall \pi \in BR_a(\sigma)$: $\frac{u_d(\sigma,\pi)-u_d(\sigma^*,\pi^*)}{|u_d(\sigma^*,\pi^*)|} \leq \epsilon$.

Note, that we use *multiplicative* definition of approximate solution concept [6], rather then more typical *additive* approximation. In our opinion, the multiplicative approximation is slightly more reasonable for our domain. However, the algorithm can be easily modified to return additive ϵ-SE.

4 Algorithms

In this section, we present two algorithms. First algorithm computes exact SE against the insider. Second algorithm finds ϵ-SE against the outsider.

4.1 Optimal Defense Strategy Against Insiders

Since we assume that the insider knows from which user type he exfiltrates data (they have complete information), we can model the interaction between the attacker and a host type as a normal-form game, where the attacker chooses a probability distribution over actions A for each host type and the defender chooses probability distribution over thresholds Θ for each host type. We formalize the game between all host types and the attacker as one problem by extending the zero-sum normal-form linear program (LP) [17] with multiple host types and a false-positive constraint.

$$\min_{\sigma(\theta|\lambda)} U_a \tag{1a}$$

$$\text{s.t.} \ : (\forall \lambda \in \Lambda, \forall a \in A) : \sum_{\theta \in \Theta} u_a(\theta, a, \lambda)\sigma(\theta|\lambda) \leq U_{a,\lambda} \tag{1b}$$

$$\sum_{\lambda \in \Lambda} P(\lambda) U_{a,\lambda} \leq U_a \tag{1c}$$

$$(\forall \lambda \in \Lambda) : \sum_{\lambda \in \Lambda} \sigma(\theta|\lambda) = 1 \tag{1d}$$

$$(\forall \lambda \in \Lambda \forall \theta \in \Theta) : \sigma(\theta|\lambda) \geq 0 \tag{1e}$$

$$\sum_{\lambda \in \Lambda} \sum_{\theta \in \Theta} P(\lambda)\sigma(\theta|\lambda) FP(\theta|\lambda) \leq FP \tag{1f}$$

The variables in the LP are: $\sigma(\theta|\lambda), U_a$ and $U_{a,\lambda}$. The objective (1a) minimizes the attacker's expected utility U_a, which consists of expected utilities $U_{a,\lambda}$ of each type, weighed by its probability (1c). Constraints (1b) ensure that the attacker plays a best response in each host type against the given defense strategy; (1d) and (1e) ensures that the defender's strategy is proper probability distribution; and (1f) makes sure the strategy meets the false-positive rate.

In LP, we need to compute the attacker's payoff $u_a(\theta, a, \lambda)$ when the defender plays action θ and the attacker attacks host type λ with action a. For the attacker with replacement, we compute $u_a(\theta, a, \lambda)$ as follows:

$$u_a(\theta, a, \lambda) = \begin{cases} \frac{a}{1-\gamma} & \text{if } \theta \geq a \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and for the attacker with additivity as follows:

$$u_a(\theta, a, \lambda) = \frac{aP(o + a \leq \theta|\lambda)}{1 - \gamma P(o + a \leq \theta|\lambda)} \tag{3}$$

where $P(o + a \leq \theta|\lambda) = \sum_{o \in O : a+o \leq \theta} P(o|\lambda)$ is the probability that the user's action o combined with the attacker's action a is below threshold θ for type λ. To compute the defender's pure strategy, we replace (2d) by $(\forall \lambda \in \Lambda \forall \theta \in \Theta)$: $\sigma(\theta|\lambda) \in \{0, 1\}$.

4.2 Optimal Defense Strategy Against External Attacker

The outsider observes the activity of the host in an attempt to learn and infer it's type. Due to the learning process, the strategies of the attacker are more complex, compared to the insider case, as the strength of the attack can vary over time. We can reason about attacker's behavior under this uncertainty using Partially Observable Markov Decision Processes (POMDPs) and his optimal, best-response strategy can be computed by algorithms for solving POMDPs. Originally, POMDPs were designed to reason about actions of a single decision maker. However, since the defender only decides the initial belief of the POMDP and the defender then has no influence on the dynamics of the system, we can extend the POMDP framework to solve our game-theoretic problem.

The POMDP framework assumes that in every time step, the player chooses an action and receives an observation from the environment as a result. Based on this observation he updates his belief over the possible current states of the environment. Additionally, in each time step the player obtains a reward which depend on the state of the environment and the action chosen. A solution of the POMDP is a policy which prescribe an action to use given every possible belief state. Here, we extend a well-established algorithm for solving POMDPs, Heuristic Search Value Iteration (HSVI) [18] to find an ϵ-Stackelberg Equilibrium, with key ideas inspired by [8]. The main idea of the algorithm is to iteratively compute the attacker's and defender's best response strategies, which will eventually converge to a Stackelberg equilibrium.

The structure of this section is as follows: first, we define POMDP models formally; then we explain the main ideas of the HSVI algorithm; and lastly, we present our contribution, the Adversarial HSVI algorithm, aimed to find ϵ-SE in our game.

POMDP Model. Let us now define a POMDP model formally, for a given defense strategy σ, as a tuple $\langle S, A, T, R, O, \gamma, \sigma\rangle$, where:

- S is set of states, where each state $s \in S$ is defined as $s = (\lambda_s, \theta_s)$, where λ is host-type and θ is the chosen detection threshold. We also define a terminal state s_T, which denotes that the attacker got detected and the attack was deflected.
- A is the set of attacker's actions;
- O is the set of observations about the traffic on host attacker tries to exfiltrate;
- $T(s, a, s')$ is the probability that action a in state s leads to new state s'. In our case, when additivity is considered $\forall s \in S \setminus \{s_T\} : T(s, a, s) = P(a + o \leq \theta_s)$, and $T(s, a, s_T) = 1 - P(a + o \leq \theta_s)$. If there is no additivity, then $\forall s \in S \setminus s_T : T(s, a, s') = 1_{a \leq \theta_s}$ and $T(s, a, s_T) = 1_{a > \theta_s}$ otherwise, where $1_A = 1$ if A is true and $1_A = 0$ otherwise is the indicator function.
- $R(s, a, s')$ is the immediate reward the attacker obtains for performing action a in state s. In our case $R(s, a, s') = a$ had the attacker not been detected yet, $R(s, a, s') = 0$ otherwise;
- $P(o|a, s)$ is the probability of observing $o \in O$ when action a is taken in state s. In our case $P(o|a, s = (\lambda, \theta)) = P(o|\lambda)$.
- $\gamma \in (0, 1)$ is the discount factor.

With $\mathscr{B}$ we denote the attacker's **belief space**, i.e. the set of all probability distributions over the states S. We derive the initial belief $b_0 \in \mathscr{B}$ according to the prior distribution over the host types $P(\lambda)$ and the strategy of the defender, i.e. $b_0(s) = P(\lambda)\sigma(\theta|\lambda)$ for state $s = (\lambda, \theta)$.

POMDP models are usually solved by approximating the optimal **value function** $v^* : \mathscr{B} \rightarrow \mathbb{R}$. This value function represents the utility $v^*(b)$ the attacker can obtain when the current distribution over the states is $b \in \mathscr{B}$ and he follows his optimal policy. We can then derive the optimal action to play in each belief state, i.e. the action $\pi(b)$, by solving the following equation

$$\pi(b) = \operatorname*{argmax}_{a \in A} \left[\sum_{s \in S} \sum_{s' \in S} \Pr[s, s'|b, a] R(s, a, s') + \gamma \sum_{o \in O} \Pr[o|b, a] \cdot v^*(\tau(b, a, o)) \right] \quad (4)$$

where we account for the immediate rewards (expectation over $R(\cdot)$) as well as the expectation over future rewards (represented by the value function v^*). $\tau(b, a, o)$ stands for a Bayesian update of the belief b based on receiving the observation o when action a was used by the attacker.

HSVI Algorithm. We now provide an explanation of basic ideas of the HSVI algorithm, which we complement with illustrations in Fig. 1. For detailed explanation of the HSVI algorithm, we refer the reader to [18]. The algorithm maintains the upper and lower bounds on the optimal value function v^* for each point in the belief space $\mathscr{B}$, as depicted in Fig. 1a. The horizontal axis represents the belief space $\mathscr{B}$ and the vertical axis represents the expected utility the attacker can achieve (or lower and upper bounds on this utility, respectively). In each iteration, HSVI performs a single simulation of depth D, in the course of which the attacker plays D actions and obtains D observations. This simulation is conducted according to a forward-exploration heuristic, which aims to select beliefs which can be reached using the play starting from the initial belief b_0, and for which the approximation using the lower and upper bounds is excessively inaccurate. For these beliefs, we compute the optimal action of the attacker (see Eq. 4) and based on that we refine the bounds on v^*. In Fig. 1b we illustrate the way the lower and upper bounds get refined.

Let us use notation $LB(b)$ and $UB(b)$ to refer to values of the lower and upper bounds, respectively, in belief $b \in \mathscr{B}$. The original HSVI algorithm terminates, when $UB(b_0) - LB(b_0) < \epsilon_{hsvi}$, where ϵ_{hsvi} is the desired approximation error.

Finding ϵ−SE. Recall that the initial belief of the POMDP problem, $b_0(s) = P(\lambda)\sigma(\theta|\lambda)$, can be directly mapped to the defender's strategy σ (and vice versa). Therefore, we search such initial belief b_0 for the defender, that it meets maximal false positive constraint and minimizes the attacker's expected utility (POMDP upper bound value at b_0). In high lever, our approach iteratively alternates between selecting a promising initial belief b_0 (strategy for the defender) and solving POMDP at that belief b_0. In Fig. 1c we illustrate a subset of valid initial beliefs that meets the false-positive constraint.

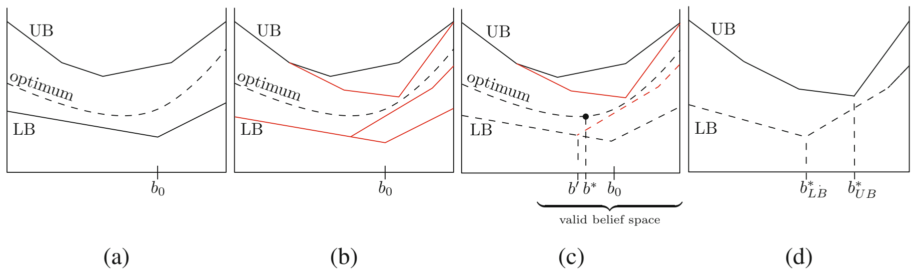

Fig. 1. Original and Adversarial HSVI algorithm: (a) initial upper bound (UB) and lower bound (LB) on the (unknown) optimal value function v^*. HSVI aims to minimizes the gap between UB and LB in the initial belief b_0. (b) After one HSVI iteration, tighter approximation using LB and UB is computed. (c) In Adversarial HSVI the defender chooses a new belief b' where LB has minimal value in every iteration. (d) A possible scenario when algorithm is converged and a conservative strategy for the defender, based on b^*_{UB}, is returned.

In detail, to find initial belief in ϵ-SE and the strategy σ of the defender, the Adversarial HSVI algorithm extends the original HSVI algorithm in two ways (the modified algorithm is presented in Fig. 2). First, instead of having fixed initial belief b_0, our algorithm chooses a new belief b' in every iteration. This belief, $b' = \text{argmin}_b\, LB(b)$, is chosen to minimize attacker's lower bound value. Second, we limit the depth D of the HSVI simulation by $\sqrt{iter}$, where $iter$ is the current iteration number. We do this to emphasize the exploration of the belief space first, and then focus on the computation of more accurate bounds later on (Fig. 2). The rest of the algorithm follows the ideas of the original HSVI algorithm. We refer the reader to Sects. 3.3 and 3.4 of [18] for details about the implementation of UpdateLB() and UpdateUB() procedures, and the forward exploration heuristic (lines 3–4 of the Explore procedure).

Let $b_{LB} = \text{argmin}_b\, LB(b)$ and $b_{UB} = \text{argmin}_b\, UB(b)$ be the beliefs with minimal value of lower and upper bounds. We ensure that the algorithm finds ϵ-SE, by terminating when defender's and attacker's strategies have maximum relative error ϵ and we then return a secure strategy implied by belief b_{UB}. The attacker can guarantee that he will obtain at least $LB(b_{LB})$, while the defender can guarantee that he will not lose more than $UB(b_{UB})$. Based on these numbers, we compute an upper bound on the relative improvement of defender's strategy (i.e. if he plays b_{LB} instead of b_{UB}) as $\frac{UB(b_{UB})-LB(b_{LB})}{UB(b_{UB})}$.

Proposition 1. *Adversarial HSVI (Fig. 2) returns ϵ-SE.*

Proof. Without loss of generality, we assume the game is exactly zero-sum (i.e., $C = 1$). When the algorithm terminates and returns $\sigma(\theta|\lambda)$ induced by b_{UB}, we know that the best response of the attacker to the defender's strategy induced by b_{UB} cannot gain more than $UB(b_{UB})$, hence the defender's cost $-u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda)) \leq UB(b_{UB})$. If the defender played any alternative strategy σ', we know that the attacker would always be able to exfiltrate at

```
1: procedure FIND $\varepsilon$-SE($\varepsilon$)
2:     $b_{LB} \leftarrow$ minimal Lower bound
3:     $b_{UB} \leftarrow$ minimal Upper bound
4:     $iter \leftarrow 1$
5:     while $\frac{UB(b_{UB})-LB(b_{LB})}{UB(b_{UB})} > \varepsilon$ do
6:         Explore($b_{LB}, 1, iter$)
7:         $b_{LB} \leftarrow$ minimal Lower bound
8:         $b_{UB} \leftarrow$ minimal Upper bound
9:         $iter \leftarrow iter + 1$
10:    end while
11:    return $\sigma(\theta|\lambda)$ induced by $b_{UB}$
12: end procedure
```

```
1: procedure EXPLORE($b, depth, iter$)
2:     if $depth < \sqrt{iter}$ then
3:         $a^* \leftarrow$ best action to explore
4:         $o^* \leftarrow$ best observation to explore
5:         $b' \leftarrow \tau(b, a^*, o^*)$
6:         Explore($b', depth + 1, iter$)
7:     end if
8:     UpdateLB()
9:     UpdateUB()
10: end procedure
```

Fig. 2. Adversarial HSVI algorithm to find ϵ-SE.

least $LB(b_{LB})$ by definition of b_{LB}, hence $-u_d(\sigma', BR_a(\sigma')) \geq LB(b_{LB})$. If the termination condition is satisfied, $\frac{UB(b_{UB})-LB(b_{LB})}{UB(b_{UB})} \leq \epsilon$. Therefore, it is sufficient to show that the relative error of the computed strategy

$$\frac{u_d(\sigma', BR_a(\sigma')) - u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))}{|u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))|} \leq \frac{UB(b_{UB}) - LB(b_{LB})}{UB(b_{UB})}.$$

Since the defender's utility is always negative, we know $|u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))| = -u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))$. Hence, the above is equivalent to

$$1 - \frac{u_d(\sigma', BR_a(\sigma'))}{u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))} \leq 1 - \frac{LB(b_{LB})}{UB(b_{UB})} \text{ and } \frac{-u_d(\sigma', BR_a(\sigma'))}{-u_d(\sigma(\theta|\lambda), BR_a(\sigma(\theta|\lambda))} \geq \frac{LB(b_{LB})}{UB(b_{UB})}.$$

This is true, because from left to right in the last inequality, the nominator can only decrease and the denominator can only increase.

5 Real-World Data

From a large network security company we obtained anonymized data capturing the volumes of upload of 5864 active Google drive users uploaded during 12 weeks. For each user we computed the amount of data that the user uploads in 6 h windows. Next, we created histograms showing how often the user uploaded certain number of bytes per 6 h, which can be understood as user's upload probability distribution.

We used the Partitioning Around Medoids algorithm to find clusters of similar behavior of the users where similarity was measured by Earth Mover's Distance [16] metric. In Fig. 3 we present 7 histograms corresponding to user's average behavior in each cluster and their relative membership. The clusters (in order) contain 25.6%, 5.5%, 17.2%, 8.9%, 11.7%, 11.5% and 19.6% of the total users.

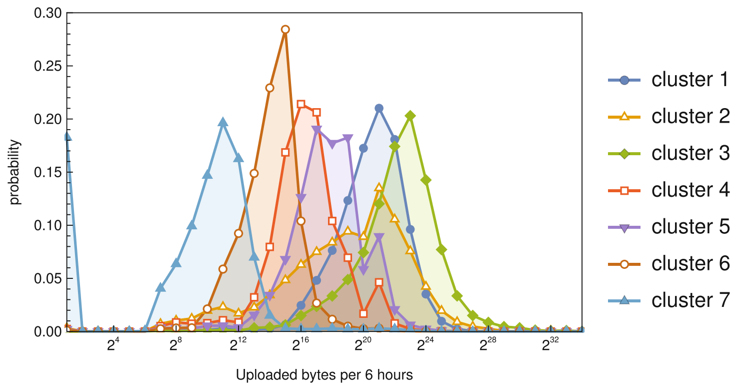

Fig. 3. Mean upload size histograms of the identified clusters of users.

6 Experiments

We now demonstrate our framework for a case study based on the real-world data. In all settings we choose: false positive rate $FP = 0.01$, the relative error of the strategy $\epsilon = 0.2$, and discount factor $\gamma = 0.9$. We chose the set of attacker actions $|A|$, the defender's thresholds $|\Theta|$, and the observations $|O|$ to be the set of $\{2^0, 2^2, 2^4, \ldots, 2^{34}\}$ bytes.

The structure of this section is as follows: In Sect. 6.1 we evaluate how much an optimal attacker can exfiltrate under various condition, in Sect. 6.2 we present a visualization of what optimal defense strategies look like, in Sect. 6.3 we evaluate defender strategies against different attacker models. In Sect. 6.4 we show how the presence of additivity influences the defense strategy, and finally, in Sect. 6.5 we present scalability results for computing the defense strategies.

6.1 Defender and Attacker Utilities

We now examine how much various attacker types can exfiltrate in our case study. In Table 1 we present a summary of the attacker's expected utilities (attacker maximizes and defender minimizes the value) for different types of attackers. Columns indicate whether the attacker is an insider or outsider and whether the attack is with replacement or with additivity. The rows indicate whether the defender plays a mixed or pure defense strategy, or a baseline defense. We present utilities against the outsider as minimal lower bound and minimal upper bound values from HSVI algorithm.

Note that the insider with replacement can exfiltrate up to 6 times more compared to insider with additivity. In in the case with additivity, the typical traffic of a user is added to the traffic of the attacker; hence, the attacker must choose a less aggressive strategy (i.e., upload less data) so that the total data upload does not exceed the threshold. Although the attacks with additivity are disadvantageous to the attacker, in some cases the additivity is unavoidable, e.g.,

Table 1. Attacker's utility for different scenarios: (columns) insider or outsider, with replacement or additivity and (rows) whether the defender plays optimal pure, optimal mixed or baseline strategy.

	Insider [MB]		Outsider [MB]	
	Replacement	Additivity	Replacement	Additivity
Mixed defense	23.71	3.56	(18.68, 24.81)	(3.11, 3.43)
Pure defense	33.58	5.03	(24.17, 29.87)	(3.78, 4.49)
Baseline single-quantile (mixed)	65.32	12.31	(54.85, 57.58)	(10.39, 10.84)
Baseline single-threshold (mixed)	68.86	14.54	(65.45, 68.86)	(13.96, 14.56)
One cluster (mixed)	63.29	12.95	N/A	N/A

when different detectors detect whether the user runs standard processes (which generate a standard traffic). Next, we see that the user type uncertainty caused around a 7%–12% decrease in the utility (computed from the upper bounds). To verify that this outcome does not rely on the fact that some of the host-types have higher prior probability than the others, we additionally ran experiments with uniform prior probability of the users, and the outsider had about 16% lower utility compared to the insider. Finally we note that if the defender must choose a pure strategy (e.g., due to practical deployment reasons) the amount of data exfiltrated by the attacker can be 24%–42% higher compared to randomized strategies.

We compare our strategies against two baseline approaches: (i) *single-quantile* and (ii) *single-threshold*. In (i), the defender sets for each host type a threshold at quantile $(1 - FP)$ of their upload probability distribution. Since we have discrete thresholds, the defender's strategy randomizes between two consecutive thresholds to reach the exact $(1 - FP)$ false positive rate. The experimental results show that the attacker can exploit this straightforward strategy and can exfiltrate about 3-times more data than against the optimal solution. The main reason is that this strategy chooses high thresholds for the users with large data upload (e.g., cluster 3) in order to satisfy the false positive constraint. In (ii) the defender chooses a single threshold for all host types such that the false positive rate requirement is satisfied. Although the strategy is quite different, it also performs poorly. The utility is even worse than the single-quantile strategy. This strategy is exactly contrary to the previous one: it sets the threshold for passive users (e.g. cluster 7) is too high, and attackers easily exfiltrate from them.

Additionally, we show that it is worth developing different defense strategies for different user types. We computed the optimal defense strategy where all users belong to one cluster (instead clustering them into 7 clusters), and utilities were 2x–3x worse than the optimal defense strategy where users were clustered. Since there is only one cluster, the attacker does not need to learn the cluster of the attacked host. Therefore, there is no difference between insiders and outsiders in this case.

6.2 Defender's and Attacker's Strategies

Insiders. In this section we present the computed optimal strategies of the defender. Figure 4a shows the defender's (cumulative) probability of selecting thresholds (x-axis) for each host type against the insider with replacement. The cumulative distributions show the probability that an attacker exfiltrating data at a certain rate is detected. In Fig. 4b we present the attacker's expected utility for different attacks on different host types when the defender is using the strategy depicted in Fig. 4a. The defender's strategy is computed in such a way that it makes the attacker indifferent between intervals of actions (e.g., for host type 1 the attacker is indifferent between actions 2^{22} through 2^{30}), which is typical for stable strategies. The attacker's best response is to choose any of the attack actions that have the highest expected utility. We also note that the host types with the highest activity (e.g., host type 3 and 1) result in the highest

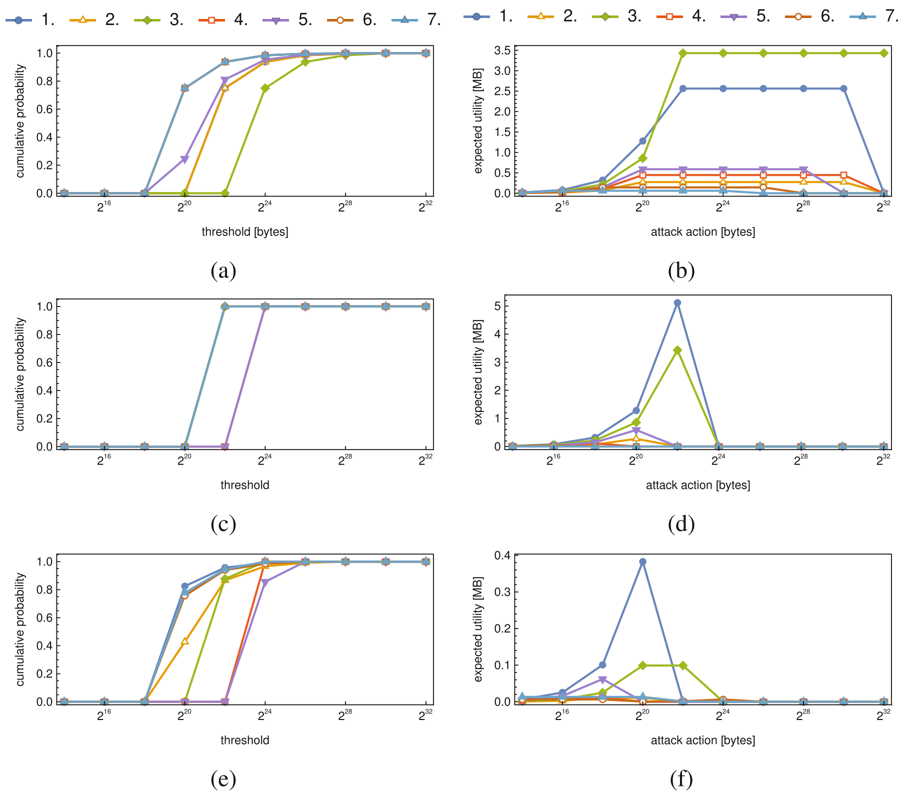

Fig. 4. The defender's strategies and the attacker's expected utilities for individual attack actions for: (a,b) mixed defense strategy against insider with replacement; (c,d) pure defense strategy against insider with replacement; and (e,f) mixed defense strategy against insider with additivity.

expected reward for the attacker. Therefore, we suggest that these hosts should be monitored thoroughly to avoid potentially large data loss.

Figure 4c shows the optimal strategy of the defender when restricted to pure strategies. For host types 3, 6 and 7, corresponding to the largest and the two smallest mean upload sizes, the defender chooses threshold 2^{22}. The threshold of 2^{24} is chosen for all the other types. The expected utility of the attacker depicted in Fig. 4b has peaks up to 5 MB, since with pure defense strategies it is impossible to make the attacker indifferent between multiple actions. By randomizing non-trivially between multiple thresholds the defender can significantly increase his expected utility.

The defense strategy against the insider with additivity (Fig. 4e) is quite different to the previous ones. With the additivity, the optimal defense strategy lowers the thresholds of the most active users (host types 1 and 3) to restrict their large loss, and increases the thresholds of the less active users to compensate the false-positives.

Outsiders. Optimal defense strategies against the outsider with additivity (Fig. 5a) (resp. with replacement (Fig. 5b)) are more complex than the strategies against the insiders. The strategies consider how the attacker attacks and learns from the observations each time step as well as the fact that the value of the data decreases over time due to the discount factor. None of the above was considered against the insider attacker.

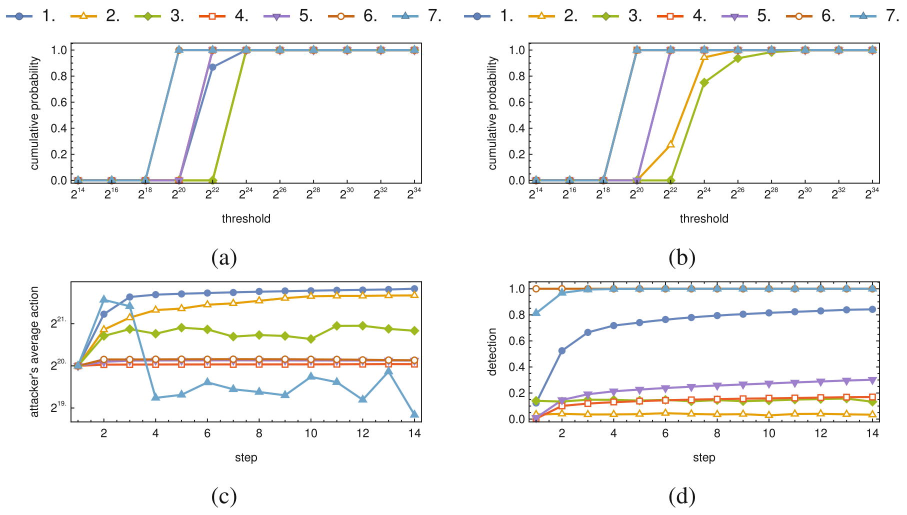

Fig. 5. Defender's strategy agaist outsider with (a) additivity and (b) replacement; the (near) optimal attacker's respone to (a) characterized by (c) the average action played at certain time step and (d) the probability of reaching the time step when attacking the given host type.

To explain how the defense strategy takes into account these aspects we first examine the attack strategy of the outsider with additivity. In Fig. 5c we show the attacker's average action in every time step *given* that the observations are drawn from a certain host type and that the attacker was not detected until the previous time step. First, the attacker plays safe action 2^{20} and after the observation, the attacker strengthens his attack as he learns about the host type their possible thresholds. Since the defender knows that the first attack action is 2^{20}, he prefers lowering the threshold for host types 6 and 7 (at 2^{20}), which causes the defender to almost certainly detect the attacker during his first attack action on 31.1% of hosts (see Fig. 5d). Not only does it increase the detection probability, it also prevents the attacker from obtaining information at the beginning, when it is most valuable due to the discounting. This generates a lot of false positives, so for highly active users the strategy uses higher thresholds (host types 2 and 3). The attacker will exfiltrate from these hosts aggressively in the later phase of the game but the loss will be less important by that time due to discounting.

This example shows how sophisticated the outsider's strategies can be as they must consider a complex behavior of the attacker and all possible sequences of observations and attacks. To minimize the loss, the defender aims to detect the attacker as soon as possible. In Fig. 5d we show the probability that the attacker is detected until given time step. Using the optimal defense strategies, host types 1, 6 and 7 (56.7% of the users) detect the attackers until his third time step with higher probability than 0.5.

6.3 Different Attacker Models

Computing a defense strategy against the insiders can be done using linear programming, which is computationally more efficient than the Adversarial HSVI algorithm used for outsiders. It rises a question of whether strategies against the insiders applied against the outsiders are significantly worse than the strategies optimized against the outsiders. In Table 2 we show the expected attacker's utilities of various defense strategies against different attacker models. The attacker

Table 2. Discounted expected amount of data that the attacker can exfiltrate if defense strategy (row) is optimized against the attacker in the "Strategy against" column, played against the different type of attackers (columns). The intervals for the outsiders represent lower and upper bounds of the optimal value.

	Strategy against	Insider, additive	Insider, replacement	Outsider, additive	Outsider, replacement
Mixed	insider, add	3.56 MB	26.58 MB	(3.56, 4.84) MB	(23.51, 29.5) MB
	insider, rep	5.91 MB	23.71 MB	(5.59, 8.33) MB	(23.71, 32.72) MB
	outsider, add	3.69 MB	27.59 MB	(2.67, 3.32) MB	(20.24, 27.59) MB
	outsider, rep	3.71 MB	27.59 MB	(2.42, 3.4) MB	(18.59, 26.35) MB
Pure	insider, add	5.03 MB	33.59 MB	(3.58, 4.43) MB	(24.54, 29.95) MB
	insider, rep	5.03 MB	33.59 MB	(3.58, 4.43) MB	(24.54, 29.96) MB
	outsider, add	5.03 MB	33.59 MB	(3.72, 4.49) MB	(24.55, 29.95) MB
	outsider, rep	5.03 MB	33.59 MB	(3.58, 4.42) MB	(24.17, 29.87) MB

of type (columns) plays a best response against the defense strategy (rows), where each defense strategy was optimized against attackers listed in column "Strategy against". For example, if we apply insider with additivity (resp. with replacement) to the outsider with additivity (resp. replacement), than the loss is between 24% and 45%. However, if we compute a defense strategy against the insider with replacement and apply it against the outsider with additivity, than the defender can lose up to 150% (comparing upper bounds) more than if the appropriate strategy is used, which is significantly worse. Therefore, it is beneficial for the network administrator to apply appropriate mixed defense strategies against different attacker models. The pure strategies do not have such big utility difference between various attacker models, due to the high similarity of all defense strategies. However, all of them have quite high loss compared to mixed strategies.

6.4 Effect of the Additivity

We now analyze how the uncertainty of the attacker's behavior affects the defender's strategy for the choice of thresholds. We created users with behavior of a normal distribution with varying standard deviation parameter. In Fig. 6 we show how the defense strategy against the insider with additivity changes given that the standard deviation of the user's behavior increases. Note that in the case where the user's behavior is constant (low standard deviation), it is optimal to choose a pure strategy with threshold at the user's mean behavior. If the attacker chooses any non-zero action, the sum of the observation and action will exceed the threshold and attacker is detected. If the defender's behavior is spread, the pure strategy is ineffective. It would have to be set at quantile $1 - FP$ due to the false-positive rate, and the attacker has a single best response action with highest expected reward. By mixing the thresholds, the defender can decrease the attacker's expected utility for a specific action and make the expected utility equal for an interval of actions (similarly, as was done for host type 3 in Fig. 4f).

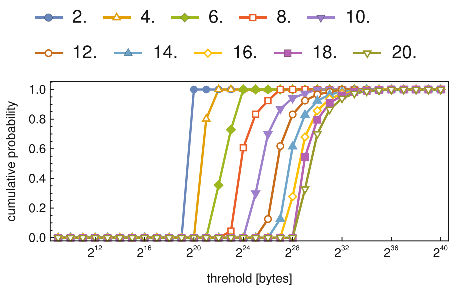

Fig. 6. Optimal defense strategies for users with standard activity of Normal distribution with $\mu = 2^{20}$ and standard deviations $\sigma = 2^i$, where i is given parameter.

This suggests that the users with close to constant behavior can be considered as the safest, as any exfiltration can be easily detected. For the users with uncertain behavior, the optimal defense strategies is to randomize among several thresholds, which forces the attacker to attacker weaker.

6.5 Algorithm Scalability

In Fig. 7a we present runtimes (note the logarithmic scale) for different numbers of host-types. All experiments were run on an Intel Xeon E5-2650 2.6 GHz with time limit 2 h. Strategies against the insider were computed in under 1 s, as they require single linear program computation. Adversarial HSVI, which iteratively improves the solution runs for between ten seconds and two hours, depending on the parameters of the problem. Even if problem was smaller (see outsider with two host types), the runtime could take longer than for seven host types. The reason is that the algorithm can temporarily get stuck in sequences of solutions with no or very little improvements. This is similar with the original HSVI. In Fig. 8 we show the relative error ϵ in each iteration for one and four host types. Our algorithm suffers with plateaus even more than HSVI, as the defender chooses initial belief with the lowest lower bound point every iteration. Despite the fact that Adversarial HSVI with $|\Lambda| = 4$ has 4 times more states, it is able to escape the plateau earlier than with $|\Lambda| = 1$. In Fig. 7b we show that the algorithm scales exponentially (note logarithmic y axis) with increasing number of actions, thresholds and observations.

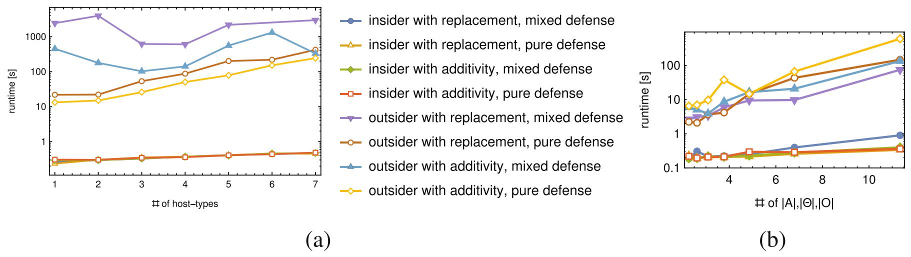

Fig. 7. (a) Runtime for increasing number of host types. (b) Runtime for increasing number of actions, thresholds, and observations.

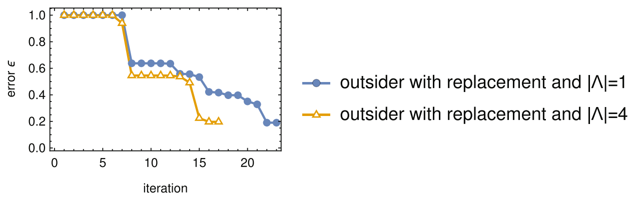

Fig. 8. ϵ error progress during computation strategies for outsider with replacement with 1 and 4 number of types.

All our case study strategies were computed with $|A| = |\Theta| = |O| = 15$, error $\epsilon = 20\%$ and number of host types $|\Lambda| = 7$ within 2 h. This demonstrates that the algorithm can be used to solve practically large problems. Moreover, HSVI and therefore Adversarial HSVI algorithms are easy to parallelize, which further improves the applicability of the presented approach.

7 Conclusion

Since computer networks, deployed defenses, and attacks are becoming more complex, developing effective decision support tools is critical for improving security. It is particularly difficult to consider the impact of all possible attacker's counteractions when the network administrator applies new defenses. Game theory provides a means to model these interactions and algorithms to compute the optimal strategies of the involved parties. We use the framework of game theory to model the problem of data exfiltration as a sequential game between the attacker and the network administrator. Sequential modeling allows us to model the decreasing value of data and the increasing chance of detection over time, as well as the development of attacker's knowledge about the network and user behavior and evolving attack strategy.

We propose two algorithms for computing (near) optimal defender strategies and bounds on their performance. For the case that the attacker does not need to learn the behavior of the attacked host, we specify a linear programming formulation for computing the optimal strategies. This situation typically corresponds to attacks by insiders, such as employees, who know the standard behavior of the hosts in the network. For the more complex situation with an attacker learning the upload behavior, we developed an algorithm based on recent results in solving single-player sequential decision-making problems. The algorithm computes strategies that optimally weigh whether to attack aggressively from the beginning and risk detection or to carefully learn the host type from observations and focus on exfiltration afterward.

Using real-world user traffic, we validate that the proposed algorithms are sufficiently scalable to analyze realistic problems. The results of our case study show that richer models produce substantially better strategies. For example, when facing the external attacker that does not replace the original traffic, the simple heuristic defense strategies let the attacker exfiltrate three times more data than the strategy optimized against a perfectly informed attacker using the linear program. Similarly, this strategy performs worse than the strategy optimized by Adversarial HSVI against the learning opponent. Our results further show that randomized defense strategies are up to 30% more effective in preventing data exfiltration compared to deterministic strategies. This is especially important when the attacker keeps the existing traffic intact, and the amounts of data transferred by the hosts vary substantially.

The attackers that know the exact behavior of the compromised host can exfiltrate by 7%–12% more data than the external attackers who have to learn it. Furthermore, a substantially more effective mitigation of data exfiltration is possible if the users are clustered into groups with similar behavior and a different detection strategy is used for each group. In our case study, the optimal strategy without the clustering allows the attacker to exfiltrate approximately three times more data than the optimal strategy using the clusters. Finally, we show that regardless of any other considered assumptions, if the attacker can replace the standard traffic of the compromised host, he can exfiltrate up to 6 times more data than the attacker who merely mixes his exfiltration traffic into the host's typical behavior. Therefore, monitoring the presence of the standard traffic (e.g., by expecting fake pre-scheduled transfers) may be a very effective countermeasure for decreasing the possible harm of data exfiltration.

Acknowledgments. This research was supported by the Czech Science Foundation (grant no. 15-23235S).

References

1. Grand theft data data exfiltration study: Actors, tactics, and detection. Technical report, McAfee, Inc. (2015). https://www.mcafee.com/us/resources/reports/rp-data-exfiltration.pdf
2. Bowen, B.M., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: Baiting inside attackers using decoy documents. In: Chen, Y., Dimitriou, T.D., Zhou, J. (eds.) SecureComm 2009. LNICST, vol. 19, pp. 51–70. Springer, Heidelberg (2009). doi:10.1007/978-3-642-05284-2_4
3. Braziunas, D.: Pomdp solution methods. University of Toronto, Technical Report (2003)
4. Comesana, P., Pérez-Freire, L., Pérez-González, F.: Blind newton sensitivity attack. IEE Proc.-Inf. Secur. **153**(3), 115–125 (2006)
5. Fadolalkarim, D., Sallam, A., Bertino, E.: Pandde: provenance-based anomaly detection of data exfiltration. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 267–276. ACM (2016)
6. Feder, T., Nazerzadeh, H., Saberi, A.: Approximating nash equilibria using small-support strategies. In: Proceedings of the 8th ACM Conference on Electronic Commerce, pp. 352–354. ACM (2007)
7. Wikipedia foundation: List of data breaches. https://en.wikipedia.org/wiki/List_of_data_breaches
8. Horák, K., Bošanský, B.: A point-based approximate algorithm for one-sided partially observable pursuit-evasion games. In: Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W. (eds.) GameSec 2016. LNCS, vol. 9996, pp. 435–454. Springer, Cham (2016). doi:10.1007/978-3-319-47413-7_25
9. Laszka, A., Abbas, W., Sastry, S.S., Vorobeychik, Y., Koutsoukos, X.: Optimal thresholds for intrusion detection systems. In: Proceedings of the Symposium and Bootcamp on the Science of Security, pp. 72–81. ACM (2016)
10. Lee, S.Y., Low, W.L., Wong, P.Y.: Learning fingerprints for a database intrusion detection system. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 264–279. Springer, Heidelberg (2002). doi:10.1007/3-540-45853-0_16
11. Lisý, V., Kessl, R., Pevný, T.: Randomized operating point selection in adversarial classification. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014. LNCS, vol. 8725, pp. 240–255. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44851-9_16

12. Liu, D., Wang, X., Camp, J.: Game-theoretic modeling and analysis of insider threats. Int. J. Crit. Infrastruct. Prot. **1**, 75–80 (2008)
13. Liu, S., Kuhn, R.: Data loss prevention. IT Prof. **12**(2), 10–13 (2010)
14. Liu, Y., Corbett, C., Chiang, K., Archibald, R., Mukherjee, B., Ghosal, D.: Sidd: a framework for detecting sensitive data exfiltration by an insider attack. In: 42nd Hawaii International Conference on System Sciences, HICSS 2009, pp. 1–10. IEEE (2009)
15. Mc Carthy, S.M., Sinha, A., Tambe, M., Manadhata, P.: Data exfiltration detection and prevention: virtually distributed POMDPs for practically safer networks. In: Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W. (eds.) GameSec 2016. LNCS, vol. 9996, pp. 39–61. Springer, Cham (2016). doi:10.1007/978-3-319-47413-7_3
16. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**(2), 99–121 (2000)
17. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations. Cambridge University Press (2008)
18. Smith, T., Simmons, R.: Heuristic search value iteration for pomdps. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pp. 520–527. AUAI Press (2004)
19. Zander, S., Armitage, G., Branch, P.: A survey of covert channels and countermeasures in computer network protocols. IEEE Commun. Surv. Tutorials **9**(3), 44–57 (2007)