Database Audit Workload Prioritization via Game Theory

CHAO YAN, Vanderbilt University, USA
BO LI, University of Illinois at Urbana–Champaign, USA
YEVGENIY VOROBEYCHIK, Washington University in St. Louis, USA
ARON LASZKA, University of Houston, USA
DANIEL FABBRI, Vanderbilt University, USA
BRADLEY MALIN, Vanderbilt University, USA

The quantity of personal data that is collected, stored, and subsequently processed continues to grow rapidly. Given its sensitivity, ensuring privacy protections has become a necessary component of database management. To enhance protection, a number of mechanisms have been developed, such as audit logging and alert triggers, which notify administrators about suspicious activities. However, this approach is limited. First, the volume of alerts is often substantially greater than the auditing capabilities of organizations. Second, strategic attackers can attempt to disguise their actions or carefully choose targets, thus hide illicit activities. In this paper, we introduce an auditing approach that accounts for adversarial behavior by 1) prioritizing the order in which types of alerts are investigated and 2) providing an upper bound on how much resource to allocate for each type.

Specifically, we model the interaction between a database auditor and attackers as a Stackelberg game. We show that even a highly constrained version of such problem is NP-Hard. Then we introduce a method that combines linear programming, column generation and heuristic searching to derive an auditing policy. On the synthetic data, we perform an extensive evaluation on the approximation degree of our solution with the optimal one. The two real datasets, 1) 1.5 months of audit logs from Vanderbilt University Medical Center and 2) a publicly available credit card application dataset, are used to test the policy-searching performance. The findings demonstrate the effectiveness of the proposed methods for searching the audit strategies, and our general approach significantly outperforms non-game-theoretic baselines.

CCS Concepts: • **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; **Database activity monitoring**; **Privacy protections**; • **Information systems** → *Autonomous database administration*;

Additional Key Words and Phrases: Anomaly Detection, Database Auditing, Game Theory, Data Privacy

ACM Reference Format:
Chao Yan, Bo Li, Yevgeniy Vorobeychik, Aron Laszka, Daniel Fabbri, and Bradley Malin. In Press. Database Audit Workload Prioritization via Game Theory. *ACM Trans. Priv. Sec.* 0, 0, Article 0 (In Press), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Chao Yan, Vanderbilt University, 2201 West End Ave, Nashville, 37235, USA, chao.yan@vanderbilt.edu; Bo Li, University of Illinois at Urbana–Champaign, Champaign, USA, bli89@illinois.edu; Yevgeniy Vorobeychik, Washington University in St. Louis, St. Louis, USA, yvorobeychik@wustl.edu; Aron Laszka, University of Houston, Houston, USA, alaszka@uh.edu; Daniel Fabbri, Vanderbilt University, Nashville, USA, daniel.fabbri@vanderbilt.edu; Bradley Malin, Vanderbilt University, Nashville, USA, b.malin@vanderbilt.edu.

2471-2566/In Press/0-ART0 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Modern computing and storage technology has made it possible to create *ad hoc* database systems with the ability to collect, store, and process extremely detailed information about the daily activities of individuals [34]. These database systems hold great value for society, but accordingly face challenges to security and, ultimately, personal privacy. The sensitive nature of the data stored in such systems attracts malicious attackers who can gain value by disrupting them in various ways (e.g., stealing sensitive information, commandeering computational resources, committing financial fraud, and simply shutting the system down) [1]. It is evident that the severity and frequency of attack events continue to grow. Notably, the most recent breach at Equifax led to the exposure of data on 143 million Americans, including credit card numbers, Social Security numbers, and other information that could be used for identity theft or other illicit purposes [37]. Even more of a concern is that the exploit of the system continued for at least two months before it was discovered.

While complex access control systems have been developed for database management, it has been recognized that in practice no database systems will be impervious to attack [44]. As such, prospective technical protections need to be complemented by retrospective auditing mechanisms, a notion that has been well recognized by the database community [30]. Though audits do not directly prevent attacks in their own right, they may allow for the discovery of breaches that can be followed up on before they escalate to full blown exploits by adversaries originating from beyond, as well as within, an organization.

In the general situation of database management, auditing relies heavily on the performance of a *threat detection and misuse tracking* (TDMT) module, which raises real-time alerts based on the actions committed to a system for further investigation by experts. In general, the alert types are specifically predefined by the administrator officials in *ad hoc* applications. For instance, in the healthcare domain, organizations are increasingly reliant on electronic medical record (EMR) systems for anytime, anywhere access to a patient's health status. Given the complex and dynamic nature of healthcare, these organizations often grant employees broad access privileges, which increases the potential risk that inside employees illegally exploit the EMR of patients [25]. To detect when a specific access to a patient's medical record is a potential policy violation, healthcare organizations use various triggers to generate alerts, which can be based on predefined rules (e.g., when an access is made to a designated very important person). As a consequence, the detected anomalies, which indicate deviations from routine behavior (e.g., when a pediatrician accesses the records of elderly individuals), can be checked by privacy officials [2]. As another example, consider the credit card provisioning domain. In this setting, individuals are interested in applying for credit cards, which might be used in a fraudulent manner. There may be many reasons why an application would trigger an alert for a credit risk analyst, who, in turn, would need to determine if the applicant is worth investigating.

Although TDMTs are widely deployed in database systems as both detection and deterrence tools, security and privacy have not been sufficiently guaranteed. The utility of TDMT in practice is currently limited by the fact that they often lead to a very large number of alerts, whereas the number of actual violations tends to be quite small. This is particularly problematic because the large quantities of false alarms can easily overwhelm the capacity of the administrative officials who are expected to follow-up on these, but have limited resources at their disposal [38]. One typical example is the observation from our evaluation dataset: at Vanderbilt University Medical Center, on any single workday, the volume of accesses to the EMR system is around 1.8 million, of which more than 30,000 alerts of varying predefined types are generated, which far beyond the capacity of privacy officials. Therefore, in lieu of an efficient audit functionality in the database systems, TDMTs are not optimized for detecting suspicious behavior.

Given the overwhelming number of alerts in comparison to available auditing resource and the need to catch attackers, the core query function invoked by an administrator must consider resource constraints. And, given such constraints, we must determine which triggered alerts should be recommended for investigation. One intuitive way to proceed is to prioritize alert categories based on the potential impact of a violation if one were to be found. However, this is an inadequate strategy because would-be violators can be strategic and, thus, reason about the specific violations they can perform so that they balance the chance of being audited with the benefits of the violation. To address this challenge, we introduce a model based on a Stackelberg game, in which an auditor chooses a randomized auditing policy, while potential violators choose their victims (such as which medical records to view) or to refrain from malicious behavior after observing the auditing policy.

Specifically, our model restricts the space of audit policies to consider two dimensions: 1) how to prioritize alert categories and 2) how many resources to allocate to each category. We show that even a highly restricted version of the auditor's problem is NP-Hard. Nevertheless, we propose a series of algorithmic methods for solving these problems, leveraging a combination of linear programming and column generation to compute an optimal randomized policy for prioritizing alert categories. We perform an extensive experimental evaluation with two real datasets—one involving EMR access alerts and the other pertaining to credit card eligibility decisions—the results of which demonstrate the effectiveness of our approach.

The remainder of the paper is organized as follows. In Section 2, we discuss related work on alert processing in the database systems, security and game theory, and audit games. In Section 3, we formally define the game theoretic alert prioritization problem and prove its NP-hardness. In Section 4, we describe the algorithmic approaches for computing a randomized audit policy. In Section 5, we introduce a synthetic dataset to show, in a controlled manner, the effectiveness of our methods for approximating the optimal solution with dramatic gains in efficiency. In Sections 6, we use two real datasets (from healthcare and finance) that rely upon predefined alert types to show that our methods lead to high-quality audit strategies. We discuss our findings and conclude this paper in Section 7.

2 RELATED WORK

The development of computational methods for raising and subsequently managing alerts in database systems is an active area of research. In this section, we review recent developments that are most related to our investigation.

Alert Frameworks. Generally speaking, there are two main categories by which alerts are generated in a TDMT: 1) machine learning methods – which measure the distance from either normal or suspicious patterns [26, 27, 33], and 2) rule-based approaches – which flag the occurrences of predefined events when they are observed [5, 17, 39]. Concrete implementations are often tailored to distinct application domains.

In the healthcare sector, methods have been proposed to find misuse of EMR systems. Boxwala et at. [9] treated it as a two-label classification problem and trained support vector machines and logistic regression models to detect suspicious accesses. Given that not all suspicious accesses follow a pattern, various techniques have been developed to determine the extent to which an EMR user [14] or their specific access [15] deviated from the typical collaborative behavior. By contrast, Fabbri et al. [20–22] designed an explanation-based auditing mechanism which generates and learns typical access patterns from an expert-, as well as data-driven, view. EMR access events by authenticated employees can be explained away by logical relations (e.g., a patient scheduled an appointment with a physician), while the residual can trigger alerts according to predefined rules (e.g., co-workers) or simply fail to have an explanation. The remaining events are provided to privacy officials for investigation; however, in practice, only a tiny fraction can feasibly be audited due to the resource limitation.

In the financial sector, fraud detection [35] in credit card applications assists banks in mitigating their losses and protecting consumers [19]. Several machine learning-based [6] models have been developed to detect fraud behavior. Some of the notable models include hidden Markov models[41], neural networks [12], support vector machines [13], etc. Rule-based techniques were also integrated into some detection frameworks [10, 42, 43, 47]. While these methods trigger alerts for investigators, they result in a significant number of false positives—a problem which can be mitigated through alert prioritization schemes.

Alert Burden Reduction. Various methods have been developed to reduce alert magnitude generated in database systems. Many focus on reducing redundancy and clustering alerts based on their similarity. In particular, a cooperative module was proposed for intrusion detection, which implemented the functions of alert management, clustering and correlation [18]. Xiao et al. proposed a multilevel alert fusion model to abstract high-level attack scenarios to reduce redundancy [46]. As an alternative, fuzzy set theory was applied by Maggi et al. to design robust alert aggregation algorithms [32]. Also, a fuzzy-logic engine to prioritize alerts was introduced by Alsubhi et al. by rescoring alerts based on a few metrics [3]. Njogu et al. built a robust alert cluster by evaluating the similarity between alerts to improve the quality of those sent to analysts [36]. However, none of these approaches consider the impact of alert aggregation and prioritization on decisions by potential attackers, especially as the latter may choose attacks that circumvent the prioritization and aggregation mechanisms.

Security Games. Our general model is related to the literature on Stackelberg security games [28], where a single or multiple defenders [45, 48] first commit to a (possibly randomized) allocation of defense resources, while the attacker chooses an optimal attack in response based on observation. Such models have been applied in a broad variety of security settings, such as airport screening [11], coast guard patrol scheduling [4], and even for preventing poaching and illegal fishing [23]. However, models used in much of this prior work are specialized to physical security and do not readily generalize to the problem of prioritizing alerts for auditing. This is the case even for several efforts specifically dealing with *audit games* [7, 8], which abstract the problem into a set of targets that could be attacked, so that the structure of the model remains essentially identical to physical security settings. In practical alert prioritization and auditing problems, in contrast, a crucial consideration is that there are many potential attackers and many potential victims or modes of attack for each of these. Moreover, auditing policies involve recourse actions where the specific alerts audited depend on the realizations of alerts of various types. Since alert realizations are stochastic, this engenders complex interactions between the defender and attackers, and results in a highly complex space of prioritization policies for the defender. In an early investigation on alert prioritization, it was assumed that 1) the identity of a specific attacker was unknown and 2) an exhaustive auditing strategy across alert types of a given order would be applied [31]. These assumptions were relaxed in the investigation addressed by our current study.

Recently, the problem of assigning alerts to security analysts has been introduced [24], with a follow-up effort casting it within a game theoretic framework [40]. In [40], there are two key limitations addressed by our framework: 1) it considers only single attacker, whereas auditing decisions in the context of access control policies commonly involve many potential attackers, with most never considering the possibility of an attack; 2) it assumes that the number of alerts in each category is known a priori to both the auditor and attacker. In this paper, we consider the scenario with multiple attackers and apply the practical situation where alert counts by category are stochastic and can exhibit high variance.

Table 1. A legend of the notation used in this paper.

Symbols	Interpretation
$\mathcal{T}$	Set of alert types
$\mathcal{E}$	Set of entities or users causing events
$\mathcal{V}$	Set of records or files available for access
$P^t(\langle e, v\rangle)$	Probability of raising type t alert by attack $\langle e, v\rangle$
C_t	Cost for auditing an alert of type t
B	Auditing budget
$F_t(n)$	Probability that at most n alerts are in type t
$\boldsymbol{O}$	Set of all alert prioritizations over T
Z_t	Number of alerts under type t
b_t	Budget threshold assigned for auditing type t
$R(\langle e, v\rangle)$	Adversary's gain when attack $\langle e, v\rangle$ is undetected
$M(\langle e, v\rangle)$	Adversary's penalty when attack $\langle e, v\rangle$ is captured
$K(\langle e, v\rangle)$	Cost of deploying attack $\langle e, v\rangle$
$p_{\boldsymbol{o}}$	Probability of choosing an alert prioritization $\boldsymbol{o}$
P_e	Probability that e is a potential adversary

3 GAME THEORETIC MODEL OF ALERT PRIORITIZATION

In environments dealing with sensitive data or critical services, it is common to deploy TDMTs to raise alerts upon observing suspicious events. By defining *ad hoc* alert types, each suspicious event can be marked with an alert label, or type, and put into an audit bin corresponding to this type. Typically, the vast majority of the raised alerts do not correspond to actual attacks, as they are generated as a part of a routine workflow that is too complex to accurately capture. Consequently, looking for actual violations amounts to looking for needles in a large haystack of alerts, and inspecting all, or even a large proportion of, alerts that are typically generated is rarely feasible. A crucial consideration, therefore, is how to *prioritize* alerts, choosing a subset that can be audited given a specified auditing budget from a vast pool of possibilities. The prioritization problem is complicated by the fact that intelligent adversaries—that is, would-be violators of organizational access policies—would react to an auditing policy by changing their behavior to balance the gains from violations, and the likelihood, and consequences, of detection.

We proceed to describe a formal model of alert prioritization as a game between an auditor, who chooses an alert prioritization policy, and multiple attackers, who determine the nature of violations, or are deterred from one, in response. In the described scenarios, we assume that the attackers have complete information, which is the worst case assumption[1]. For reference purposes, the symbols used throughout this paper are described in Table 1.

3.1 System Model

Let $\mathcal{E}$ be the set of potential adversaries, such as employees in a healthcare organization, some of whom could be potential violators of privacy policies, and $\mathcal{V}$ be the set of potential victims, such as patients in a healthcare facility. We define events, as well as attacks, by a tuple $\langle e, v\rangle$. A subset of these events will trigger alerts. Now, let $\mathcal{T}$ be the set of alert types or categorical labels assigned to different kinds of suspicious behavior. For example, a doctor viewing a record for a patient not

[1]We do not claim that the attacker actually has such information, but instead aim to be robust even if the attacker has complete information

assigned to them and a nurse viewing the EMR for another nurse (who is also a patient) in the same healthcare facility could trigger two distinct alert types. We assume that each event $\langle e, v\rangle$ maps to at most one alert type $t \in \mathcal{T}$. This mapping may be stochastic; that is, given an event $\langle e, v\rangle$, an alert with type t is triggered with probability $P^t(\langle e, v\rangle)$, and no alert is triggered otherwise (i.e., $p^{t'}_{\langle e,v\rangle} = 0$ for all $t' \neq t$). Typically, both categorization of alerts and corresponding mapping between events and types is given (for example, through predefined rules). If not, it can be inferred by generating possible attacks and inspecting how they are categorized by TDMT. Auditing each alert is time-consuming and the time to audit an alert can vary by alert type. Let C_t be the cost (e.g., time) of auditing a single alert of type t and let B be the total budget allocated for auditing.

We assume that the number of alerts triggered by normal events follows a distribution which reflects a typical workflow of the organization and can be learned based on historical data. We assume this distribution is known, represented by $F_t(n)$, which is the probability that at most n alerts of type t are generated. If we make the reasonable assumption that attacks are rare events and that the alert logs are tamper-proof by applying a certain technique [29], then this distribution can be obtained from historical alert logs. It is noteworthy that the probability that adversaries successfully manipulate the distribution in the sensitive practices (e.g., the EMR system or the credit card application program), to fool the audit model is almost zero. The cost of orchestrating and implementing such attacks is much higher than what could be gained from running a few undetected attacks.

3.2 Game Model

We model the interaction between the auditor and potential violators as a Stackelberg game. Informally, the auditor chooses a possibly randomized auditing policy, which is observed by the prospective violators, who in response choose the nature of the attack, if any. Both decisions are made *before the alerts produced through normal workflow are generated* according to a known stochastic process $F_t(n)$.

In general, a specific pure strategy of the defender (auditor) is a mapping from an arbitrary realization of alert counts of all types to a subset of alerts that are to be inspected, abiding by a constraint on the total amount of budget B allocated for auditing alerts. Even representing a single such strategy is intractable, let alone optimizing in the space of randomizations over these. We, therefore, restrict the defender strategy space in two ways. First, we let pure strategies involve an ordering $\boldsymbol{o} = (o_1, o_2, \ldots, o_{|\mathcal{T}|})$ ($\forall i, j \in \mathbb{Z}^+$ and $i, j \in [1, |\mathcal{T}|]$, if $i \neq j$, then $o_i \neq o_j$) over alert types, where the subscript indicates the position in the ordering, and a vector of thresholds $\mathbf{b} = (b_1, \ldots, b_{|\mathcal{T}|})$, with b_t being the maximum budget available for auditing alerts in category t. Let $\mathbf{O}$ be the set of feasible orderings, which may be a subset of all possible orders over types (e.g., the organizational policy may impose constraints, such as always prioritizing some alert categories over others). We interpret a threshold b_t as the maximum budget allocated to t; thus, the most alerts of type t that can be inspected is $\lfloor b_t/C_t \rfloor$. Second, we allow the auditor to choose a randomized policy over alert orderings, with $p_{\boldsymbol{o}}$ being the probability that ordering $\boldsymbol{o}$ over alert types is chosen, whereas the thresholds $\mathbf{b}$ are deterministic and independent of the chosen alert priorities.

We have a collection of potential adversaries $\mathcal{E}$, each of whom may target any potential victim $v \in \mathcal{V}$. We assume that the adversary will target exactly one victim (or at most one, if $\mathcal{V}$ contains an option of not attacking anyone). Thus, the strategy space of each adversary e is $\mathcal{V}$. In addition, we characterize the probability that an adversary $e \in \mathcal{E}$ performs an attack as P_e (i.e., e does not even consider attacking with probability $1 - P_e$).

Suppose we fix a prioritization $\boldsymbol{o}$ and thresholds $\mathbf{b}$. Let $o(t)$ be the position of alert type t in $\boldsymbol{o}$ and o_i be the alert type in position i in the order. Let $B_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z})$ be the budget remaining to inspect alerts of type t if the order is $\boldsymbol{o}$, the defender uses alert type thresholds $\mathbf{b}$, and the vector of realizations of benign alert type counts is $\mathbf{Z} = \{Z_1, \ldots, Z_{|\mathcal{T}|}\}$. Then we have

$$B_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z}) = \max\left\{B - \sum_{i=1}^{o(t)-1} \min\left\{b_{o_i}, Z_{o_i}C_{o_i}\right\}, 0\right\}. \tag{1}$$

Now, let us take a moment to unpack this expression for context. For the audited alert type t, we repeatedly compare the threshold b_t with Z_tC_t to determine how much budget will be left for the types that follow in the priority order. If the total budget that is eaten by inspecting alerts prior to t is larger than B, $B_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z})$ returns 0, and no alerts of type t will be inspected. Next, we can compute the number of alerts of type t that are audited as

$$n_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z}) = \min\left\{\lfloor B_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z})/C_t \rfloor, \lfloor b_t/C_t \rfloor, Z_t\right\}. \tag{2}$$

Suppose that an attack generates an alert of type t. As noted earlier, we assume that the number of alerts generated due to attacks is a negligible proportion of all generated alerts (e.g., when P_e are small). Then, the probability that an alert of type t generated through an attack is detected is approximately

$$P_{al}(\boldsymbol{o}, \mathbf{b}, t) \approx \mathbb{E}_{\mathbf{Z}}\left[\frac{n_t(\boldsymbol{o}, \mathbf{b}, \mathbf{Z})}{Z_t}\right]. \tag{3}$$

The approximation comes from the fact that we use the benign counts Z_t in the denominator to approximate the sum of the number of the false positive alerts and the true positive alerts in type t. This is because 1) the number of true positive alerts in each type is very small in practice and 2) the exact number true positives are unknown to the auditor.

The adversary e does not directly choose alert types, but rather the victims v (e.g., an EMR). The probability of detecting an attack $\langle e, v\rangle$ under audit order $\boldsymbol{o}$ and audit thresholds $\mathbf{b}$ is then

$$P_{at}(\boldsymbol{o}, \mathbf{b}, \langle e, v\rangle) = \sum_t P^t(\langle e, v\rangle) P_{al}(\boldsymbol{o}, \mathbf{b}, t). \tag{4}$$

We now have sufficient preliminaries to define the utility functions of the adversaries $e \in \mathcal{E}$. Let $M(\langle e, v\rangle)$ denote the penalty of the adversary when captured by the auditor, $R(\langle e, v\rangle)$ denote the benefit if the adversary is not audited, and $K(\langle e, v\rangle)$ the cost of an attack. The utility of the adversary is then

$$\begin{aligned} U_a(\boldsymbol{o},\mathbf{b}, \langle e, v\rangle) = {} & P_{at}(\boldsymbol{o}, \mathbf{b}, \langle e, v\rangle) \cdot M(\langle e, v\rangle) \\ & + (1 - P_{at}(\boldsymbol{o}, \mathbf{b}, \langle e, v\rangle)) \cdot R(\langle e, v\rangle) - K(\langle e, v\rangle). \end{aligned} \tag{5}$$

By assuming that the game is zero-sum, there is no difference between the Strong Stackelberg Equilibrium (SSE) and the Nash Equilibrium (NE) [16]. Under this assumption, the auditor's goal can be transferred into finding a randomized strategy $\boldsymbol{p}$ and type-specific thresholds $\mathbf{b}$ to minimize the expected utility of the adversary:

$$\min_{\boldsymbol{p}, \mathbf{b}} \sum_{e \in \mathcal{E}} P_e \max_v \sum_{\boldsymbol{o} \in \boldsymbol{O}} p_{\boldsymbol{o}} U_a(\boldsymbol{o}, \mathbf{b}, \langle e, v\rangle), \tag{6}$$

where $\boldsymbol{p} = \{p_{\boldsymbol{o}} \mid \boldsymbol{o} \in \boldsymbol{O}\}$. We call this optimization challenge the *optimal auditing problem (OAP).*

The optimal auditing policy can be computed using the following mathematical program, which directly extends the standard linear programming formulation for computing mixed-strategy Nash equilibria in zero-sum games:

$$\begin{aligned} \min_{\mathbf{b},\boldsymbol{p},\mathbf{u}} \quad & \textstyle\sum_{e\in\mathcal{E}} P_e u_e \\ s.t. \quad & \forall \langle e,v\rangle,\ u_e \geq \textstyle\sum_{\boldsymbol{o}\in\boldsymbol{O}} p_{\boldsymbol{o}} U_a(\boldsymbol{o},\mathbf{b},\langle e,v\rangle) \\ & \textstyle\sum_{\boldsymbol{o}\in\boldsymbol{O}} p_{\boldsymbol{o}} = 1, \\ & \forall \boldsymbol{o}\in\boldsymbol{O}, \quad 0 \leq p_{\boldsymbol{o}} \leq 1. \end{aligned} \tag{7}$$

An important issue in this formulation is that we do not randomize over the decision variables $\mathbf{b}$. However, if we restrict strategies to the decision variables $\boldsymbol{p}$ by fixing $\mathbf{b}$ first, then the resulting SSE and NE are identical. Indeed, if we fix $\mathbf{b}$, the formulation becomes a linear program. Nevertheless, since the set of all possible alert prioritizations is exponential, even this linear program has exponentially many variables. Furthermore, introducing decision variables $\mathbf{b}$ makes it non-linear and non-convex. Next, we show that solving this problem is NP-hard, even in a restricted special case. We prove this by reducing from the 0-1 Knapsack problem.

Definition 3.1 (0-1 Knapsack Problem). Let $\mathcal{I}$ be a set of items where each item $i \in \mathcal{I}$ has a weight w_i and a value v_i, with w_i and v_i integers. W is a budget on the total amount of weight (an integer). Question: given a threshold K, does there exist a subset of items $R \subseteq \mathcal{I}$ such that $\sum_{i\in R} v_i \geq K$ and $\sum_{i\in R} w_i \leq W$?

THEOREM 3.2. *OAP is NP-hard even when* $\mathbf{O}$ *is a singleton.*

PROOF. We reduce from the 0-1 Knapsack problem defined by Definition 3.1.

We begin by constructing a special case of the auditing problem and work with the decision version of optimization Equation 6, in which we decide whether the objective is below a given threshold θ. First, suppose that $Z_t = 1$ for all alert types $t \in \mathcal{T}$ with probability 1. Since the set of orders is a singleton, the probability distribution over orders $p_{\boldsymbol{o}}$ is not relevant. Consequently, it suffices to consider $b_t \in \{0,1\}$ for all t, and the actual order over types is not relevant because $Z_t = 1$ for all types. Consequently, we can choose $\mathbf{b}$ to select an arbitrary subset of types to inspect subject to the budget constraint B (i.e., type t will be audited iff $b_t = 1$). Thus, the choice of $\mathbf{b}$ is equivalent to choosing a subset of alert types $A \subseteq \mathcal{T}$ to audit.

Suppose that $\mathcal{V} = \mathcal{T}$, and each victim $v \in \mathcal{V}$ deterministically triggers some alert type $v \in \mathcal{V} = \mathcal{T}$ for any attacker e. Let $M(\langle e,v\rangle) = C(\langle e,v\rangle) = 0$ for all $e \in \mathcal{E}, v \in \mathcal{V}$, and suppose that for every e, there is a unique type $t(e)$ with $R(\langle e,v\rangle) = 1$ if and only if $v = t(e)$ and 0 otherwise. Then $\max_v U_a(\mathbf{o},\mathbf{b},\langle e,v\rangle) = 1$ if and only if $b_{t(e)} = 0$ (i.e., alert type $t(e)$ is not selected by the auditor) and 0 otherwise. Finally, we let $P_e = 1$ for all e.[2]

For the reduction, suppose we are given an instance of the 0-1 Knapsack problem. Let $\mathcal{T} = \mathcal{I}$, and for each $i \in \mathcal{I}$, generate v_i attackers with $t(e) = i$. Thus, $v_i = |\{e : t(e) = i\}|$. Let $C_i = w_i$ be the cost of auditing alerts of type i, and let $B = W$. Define $\theta = |\mathcal{E}| - K$. Now observe that the objective in Equation 6 is below θ if and only if $\min_{\mathbf{b}} \sum_{t:b_t=0} v_t \leq \theta$, or, equivalently, if there is R such that $\sum_{t\in R} v_t \geq K$. Thus, the objective of Equation 6 is below θ if and only if the Knapsack instance has a subset of items $R \subseteq \mathcal{I}$ which yield $\sum_{i\in R} v_i \geq K$, where R must satisfy the same budget constraint in both cases. □

[2]While this is inconsistent with our assumption that attackers constitute only a small portion of the system users, we note that this is only a tool for the hardness proof.

4 SOLVING THE ALERT PRIORITIZATION GAME

There are two practical challenges that need to be addressed to compute useful approximate solutions to the OAP. First, there is an exponential set of possible orderings of alert types that need to be considered to compute an optimal randomized strategy for choosing orderings. Second, there is a combinatorial space of possible choices for the threshold vectors **b**. In this section, we develop a column generation approach for the linear program induced when we fix a threshold vector **b**. We then introduce a search algorithm to compute the auditing thresholds.

4.1 Column Generation Greedy Search

By fixing the auditing threshold vector **b**, Equation 7 becomes a linear program, albeit with an exponential number of variables. However, since the number of constraints is small, only a limited number of variables will be non-zero. In other words, the number of effective orderings of alert types in the optimal solution is small compared to the exponential search space. The challenge is in finding this small basis. We solve this problem in a greedy manner by applying the column generation framework. In this approach, we iteratively solve a linear program, where we use a subset of the variables. Upon each iteration we add a new variable before the value of the objective function fails to reduce. By doing so, we can incorporate the orderings that contribute to reducing the value of the objective function. When no new orderings can be added, the process terminates. We refer to this method as Column Generation Greedy Search (CGGS), the pseudocode for which is in Algorithm 1.

Specifically, we begin with a small subset of alert prioritizations $\boldsymbol{Q} \subseteq \boldsymbol{O}$. We solve the linear program induced after fixing **b** in Equation 7, restricted to columns in $\boldsymbol{Q}$. For reference purposes, we call this the *master problem*, which is generated by function $G_{lp}(*)$. Next, we check if there exists a column (ordering over types) that improves upon the current best solution. The column of parameter matrix of constraints can be denoted as $\boldsymbol{\Gamma}_{p_{\boldsymbol{o}}} = P_{at}(\boldsymbol{o}, \mathbf{b}, \langle e, v\rangle) - \mathbf{1}$ for the decision variable $p_{\boldsymbol{o}}$ or $\boldsymbol{\Gamma}_{u_e} = \mathbf{1}$ for the decision variable u_e. The corresponding reduced costs, computed by function $rc(*)$, are $\mathbf{C^r}_{p_{\boldsymbol{o}}} = 1 - \pi_{\boldsymbol{Q}} \cdot \boldsymbol{\Gamma}_{p_{\boldsymbol{o}}}$ and $\mathbf{C^r}_{u_e} = -\pi_{\boldsymbol{Q}} \cdot \boldsymbol{\Gamma}_{u_e}$, where $\pi_{\boldsymbol{Q}}$ is the solution of the dual problem. By minimizing the reduced costs, we generate one new column in each iteration and add it to the subset of columns $\boldsymbol{Q}$ in the master problem. Within the process of generating a new column, we use $\boldsymbol{\Gamma}'_{(\boldsymbol{o}'+\mathrm{t})}$ to denote the parameter column with the audit order $(\boldsymbol{o}' + t)$. This process is repeated until we can prove that the minimum reduced cost is non-negative.

The subproblem of generating the optimal column is itself non-trivial. We address this subproblem through the application of a greedy algorithm for generating a reduced-cost-minimizing ordering over alert types. The intuition behind CGGS is that, in the process of generating a new audit order, we greedily add one alert type at a time to minimize the reduced cost *given the order generated thus far*. We continue until the objective (reduced cost) fails to improve.

4.2 Iterative Shrink Heuristic Method

Armed with an approach for solving the linear program induced by a fixed budget threshold vector **b**, we now develop a heuristic procedure to find alert type thresholds.

Now, let us characterize the range of each element in **b**. First, it should be recognized that $\sum_t b_t \geq B$ because to allow otherwise would clearly waste auditing resources. Yet there is no explicit upper bound on the thresholds. However, given the distribution of the number of alerts Z_t for an alert type t, we can obtain an approximate upper bound on b_t, where $F_t(b_t/C_t) \approx 1$. This is possible because setting the thresholds above such bounds would lead to negligible improvement. Consequently, searching for a good solution can begin with a vector of audit thresholds, such that for each b_t, $F_t(b_t/C_t) \approx 1$. Leveraging this intuition, we design a heuristic method, which iteratively

ALGORITHM 1: Column Generation Greedy Search (CGGS)

```
Input  : The set $\boldsymbol{Q}$ with a single random pure strategy for the auditor.
Output : The set of pure strategies $\boldsymbol{Q}$.
1  while $\underline{True}$ do
2      $Z = G_{lp}(\boldsymbol{Q})$;                                  /* Construct LP using current $\boldsymbol{Q}$ */
3      $\boldsymbol{\pi_Q} = LP(Dual(Z))$;                            /* Solve dual problem */
4      $\boldsymbol{o'} = []$;
5      while $\underline{|\boldsymbol{o'}| < |T|}$ do
6          $\boldsymbol{o'} = \boldsymbol{o'} + \text{argmax}_{t \in T \setminus \boldsymbol{o'}} \boldsymbol{\pi_Q} \cdot \Gamma'_{(\boldsymbol{o'}+\text{t})}$;
7      end
8      if $\underline{\min rc(\boldsymbol{Q}) < 0}$ then
9          $\boldsymbol{Q} = \boldsymbol{Q} + \boldsymbol{o'}$;
10     else
11         break;
12     end
13 end
```

shrinks the values of a good[3] subset of audit thresholds according to a certain step size ϵ. We refer to this as the Iterative Shrink Heuristic Method (ISHM), the pseudocode for which is provided in Algorithm 2.

In each atomic searching action, ISHM first makes a subset of thresholds b_t strategically shrink. Next, it checks to see if this results in an improved solution. We introduce a variable l_h, which indicates the level (or the size) of the given subset of $\mathbf{b}$, and $\epsilon \in (0, 1)$, which controls the step size.

In the beginning, the vector of audit thresholds $\{\hat{H}_o\}$ is initialized with the approximate upper bounds. Then, by assigning $l_h = 1$, we consider shrinking each of the audit thresholds $\hat{H}_i$. The coefficient for shrinking is defined by the *ratio* in line 7, which is instantiated with the predefined step size ϵ; i.e., $i = 1$. If the best value for the objective function in the candidate subsets at $l_h = 1$ after shrinking shows an improvement, then the shrink is accepted and the shrinking coefficient is made smaller by increasing i. When no coefficient leads to improvement, we increase l_h by one, which induces tests of threshold combinations at the same shrinking ratio. This logic is described in line 6 through 20.

Once an improvement occurs, the search course resets based on the current $\mathbf{b}$. The search terminates once $l_h > |T|$.

Note that for a single improvement, the worst-case time complexity is $O(\lceil 1/\epsilon \rceil \cdot O(LP) \cdot 2^{|T|})$. Though exponential, our experiments show that ISHM achieves outstanding performance, both in terms of precision (of approaching the optimal solution) and efficiency.

5 CONTROLLED EVALUATION

To gain intuition into the potential for our methods, we evaluated the performance of the ISHM and CGGS approaches using a synthetic dataset, *Syn_A*. To enable comparison with an optimal solution, we use a relatively small synthetic dataset, but as will be clear, it is sufficient to illustrate the relationship between our methods and the optimal brute force solution.

To perform the analysis, we vary the audit budgets B and step size ϵ of ISHM. In addition, we evaluate a combination of CGGS+ISHM (since the former is also an approximation), by again comparing to the optimal.

[3]"Good" in this context means that shrinking the thresholds within the subset improves the value of the objective function.

ALGORITHM 2: Iterative Shrink Heuristic Method (ISHM)

```
Input   : Instance of the game, step size $\epsilon$.
Output : Vector of audit thresholds $\{\hat{H}_i\}$.
Initialize $\{\hat{H}_i\}$ with full coverages in $\{F_t\}$;
$l_h = 1; obj = +\infty$;
while $l_h <= |T|$ do
    $C_{l_h} = choose(|T|, l_h)$;                              /* Find combinations */
    $prgrs = 0$;
    for $i \leftarrow 1$ to $\lceil 1/\epsilon \rceil$ do
        $ratio = \max\{0, 1 - i * \epsilon\}$;
        $obj_r = +\infty; pst_r = 0$;
        for $j \leftarrow 1$ to $|C_{l_h}|$ do
            $temp = \{\hat{H}_i\}$;
            for $k \leftarrow 1$ to $l_h$ do
                $temp(1, C_{l_h(j,k)}) = temp(1, C_{l_h(j,k)}) * ratio$;
            end
            $obj' = LP(B, temp)$;                            /* Return LP objective value */
            if $obj' < obj_r$ then $obj_r = obj'; pst_r = j$;
        end
        if $obj_r < obj$ then
            $obj = obj_r$;
            $S_u = C_{l_h}(pst_r, :)$;                        /* Types in need of update */
            for $j \leftarrow 1$ to $|S_u|$ do $\hat{H}_{S_{u_j}} = \hat{H}_{S_{u_j}} * ratio$;
            break;
        end
        $prgrs = i$;
    end
    if $prgrs == \lceil 1/\epsilon \rceil$ then $l_h = l_h + 1$;
    else $l_h = 1$;
end
```

5.1 Data Overview

The dataset *Syn_A* consists of 5 potential attackers who perform accesses ($p_e = 1$[4]), 8 files, 4 predefined alert types, and a set of rules for triggering alerts if any access happens. Table 2 summarizes the information of *Syn_A* and related parameters in the corresponding scenario. We let the number of alerts for all types be distributed according to a Gaussian distribution with means and standard deviation as reported in Table 2a. Since the number of alerts for each alert type are integers, we discretize the x-axis of each alerts cumulative distribution function and use the corresponding probabilities for each possible alert count. We consider the 99.5 probability coverage for each alert type to obtain a finite upper bound on alert counts.

We assume alerts are triggered deterministically for each access, a common case in rule-based systems. The alert type that will be triggered for each potential access is provided in Table 2b, where "-" represents a benign access. This table is generated with a probability vector [0.07, 0.38, 0.23, 0.16, 0.16] for each employee, which corresponds to alert type vector [0, 1, 2, 3, 4]. Although in reality, benign accesses may be more frequent, we lower their probability to better differentiate the final value of

[4]The artificially high incidence of attacks here is merely to facilitate a comparison with a brute-force approach.

the objective function. The benefit of the adversary for a successful attack, the cost of an attack and the cost of an audit are all directly related to the alert type, which is shown in Table 2a. In addition, the penalty for being caught is set to a constant value of 4.

Table 2. Description of Dataset *Syn_A*.

(a) Parameters for alert types in the synthetic setting.

	Type 1	**Type 2**	**Type 3**	**Type 4**
Mean	6	5	4	4
Std	2	1.6	1.3	1
99.5% Coverage	+/-5	+/-4	+/-3	+/-3
Benefit	3.4	3.7	4	4.3
Attack Cost	0.4	0.4	0.4	0.4
Audit Cost	1	1	1	1

(b) Rules for alert types in the synthetic setting.

Employee	**Record**							
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
e_1	–	3	2	2	3	4	3	1
e_2	1	–	1	1	1	2	1	1
e_3	1	3	4	–	1	3	1	4
e_4	2	1	3	1	4	4	2	2
e_5	2	3	1	4	2	1	3	2

5.2 Optimal Solution with Varying Budget

Based on the given information, we can compute the optimal OAP solution. First, the search space for audit thresholds in this scenario is as follows: 1) for each alert type, the audit threshold $b_t \in \mathbf{N}$, 2) the sum of thresholds for all alert types should be greater than or equal to B, 3) for each type, the upper bound of the audit threshold b_t is where $F_t(b_t/C_t) \approx 1$. Concretely, we set vector $\mathcal{J} = Mean + |99.5\% Coverage|$ as the upper bound for finding the optimal solution. Thus, the space of the investigation of the optimal solution is $O(\prod_{i=1}^{|T|}(\mathcal{J}_i + 1))$. Note that 0 is also a possible choice, which means the auditor will not check the corresponding alert type. Thus, it is infeasible to directly solve the OAP in the instances with a large number of alert types or large $\mathcal{J}_i$.

To investigate the performance of the proposed audit model, we allocated a vector of audit budgets $\mathbf{B} = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$, which has a wide range with respect to the scale of the means of the alert types. We then apply a brute force search to discover an optimal vector of budget thresholds for each type. Table 3 shows the optimal solution of OAP for each candidate $\mathbf{B}$, including the optimal value of the objective function, optimal threshold (using the smallest optimal threshold whenever the optimal solution is not unique), pure strategies in the support of the optimal mixed strategy, and the optimal mixed strategy of the auditor. As expected, it can be seen that as the budget increases, the optimal value of the objective function (minimized by the auditor) decreases monotonically.

Table 3. The optimal solution for the auditor under various budgets.

ID	Budget	Optimal Objective Value	Optimal Threshold	Effective Pure Strategy	Optimal Mixed Strategy
1	2	12.2945	[1,1,1,1]	[2,3,4,1][4,1,3,2][4,2,3,1][4,3,2,1]	[0.3566, 0.3780, 0.1210, 0.1444]
2	4	7.7176	[2,1,1,2]	[1,2,3,4][2,1,3,4][4,2,1,3][4,2,3,1]	[0.4664, 0.0052, 0.0934, 0.4350]
3	6	3.2651	[2,2,2,2]	[2,1,3,4][4,1,3,2][4,2,1,3][4,2,3,1]	[0.2748, 0.2341, 0.3293, 0.1618]
4	8	-0.4517	[3,3,2,2]	[2,1,3,4][4,1,3,2][4,2,1,3][4,2,3,1]	[0.0762, 0.4600, 0.1329, 0.3309]
5	10	-2.1314	[3,3,3,3]	[1,2,3,4][1,4,3,2][4,1,2,3][4,1,3,2]	[0.3926, 0.0788, 0.4080, 0.1206]
6	12	-3.7345	[4,4,3,3]	[2,1,3,4][4,2,3,1][4,2,1,3][4,1,3,2]	[0.2028, 0.1554, 0.2076, 0.4342]
7	14	-5.1645	[5,4,3,3]	[2,1,3,4][4,2,3,1][4,2,1,3][4,1,3,2]	[0.3559, 0.2199, 0.3176, 0.1066]
8	16	-6.4510	[6,5,4,4]	[2,1,3,4][4,1,3,2][4,2,1,3][4,2,3,1]	[0.2431, 0.2636, 0.1728, 0.3205]
9	18	-7.4649	[7,6,5,5]	[2,1,3,4][4,1,3,2][4,2,1,3][4,2,3,1]	[0.2710, 0.2630, 0.2054, 0.2615]
10	20	-8.1561	[9,7,6,6]	[1,2,3,4][4,1,2,3][4,1,3,2][4,2,3,1]	[0.2398, 0.1742, 0.2275, 0.3585]

5.3 Findings

Our heuristic methods aim to find an approximate solution through major reductions in computation complexity. In this respect, the search step size ϵ is a key factor to consider because it could lead the search into a locally optimal solution. To investigate the gap between the objective function with the optimal solution, as well as the influence of ϵ on the gap, we performed experiments with a series of step sizes $\epsilon = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]$. Tables 4 and 5, summarize the results, where each cell consists of two items: 1) the minimized sum of the maximal utilities of all adversaries obtained using the heuristic method and 2) the corresponding audit threshold vector.

There are three findings worth highlighting. First, when ϵ is fixed, the approximated values of the objective function decrease as the budget increases. This is akin to the trend shown in Table 3. Second, when the budget B is fixed, the approximated values achieved through ISHM and ISHM+CGGS exhibit a general growth trend as ϵ increases. This occurs because larger shrink ratios increase the likelihood that the heuristic search will miss more of the good approximate solutions. Third, we find that the ISHM and ISHM+CGGS solutions are close to the optimal. To measure the solution quality as a function of ϵ, we use $\gamma_\epsilon = \frac{1}{|\mathbf{B}|}\sum_i^{|\mathbf{B}|} |\hat{S}_{\mathbf{B}_i,\epsilon} - S_{\mathbf{B}_i,\epsilon}|/|S_{\mathbf{B}_i,\epsilon}|$, where $\hat{S}_{\mathbf{B}_i,\epsilon}$ denotes the approximate optimal values in Tables 4 and 5 and $S_{\mathbf{B}_i,\epsilon}$ denotes the optimal values provided by Table 3.

In Table 6, it can be seen that ISHM (and solving the linear program to optimality) achieves solutions near 99% of the optimal (as denoted by γ_ϵ^1) when the step size $\epsilon \leq 0.2$. Even the approximately optimal solutions with $\epsilon = 0.5$ have a good approximation ratio (above 89%). As such, it appears that if we choose an appropriate ϵ, then ISHM can perform well.

When we combine ISHM+CGGS (denoted by γ_ϵ^2), the approximation quality drops compared to γ_ϵ^1, as we would expect, with the lone exception of ($\epsilon = 0.4$). However, γ_ϵ^2 is very close to γ_ϵ^1, which suggests that our approximate column generation method does not significantly degrade the quality of the solution.

Next, we consider the computational burden for ISHM to achieve an approximate target of the optimal solution. Table 7 provides the values of the threshold vectors under various B and ϵ. It can be seen that the number of threshold candidates explored decreases as the step size grows. For a given ϵ, the number of thresholds considered by the algorithm initially increases, but then drops as the audit budget increases. The reason that less effort is necessary at the extremes of the budget range is that the restart of the test for a single alert type (to find a better position) is invoked less frequently. By contrast, a larger amount of effort is required in the middle of the budget range due to more frequent restarts (although this yields only a small improvement).

Finally, we investigate the average number for the threshold vectors explored by the algorithm over the budget range **B**. For the various step sizes, we represent the results in vector form $\mathscr{T} = [403, 223, 156, 121, 93, 86, 68, 66, 61, 47]$. Dividing by the number of investigations needed to discover the optimal solution, the resulting ratio vector is $\mathscr{T}' = [0.0831, 0.0460, 0.0321, 0.0251, 0.0198, 0.0190, 0.0163, 0.0182, 0.0206, 0.0210]$. Thus, when $\epsilon = 0.2$ (when both γ^1_ϵ and γ^2_ϵ are greater than 0.99), the number of thresholds explored is only 2.51% of the entire space. As such, by applying ISHM, the number of investigated threshold candidates can be greatly reduced without significantly sacrificing solution quality.

Table 4. The approximation of the optimal solutions obtained by ISHM at various levels of B and ϵ.

B	**Approximation of Optimal Loss of the Auditor and corresponding thresholds by ISHM**									
	$\epsilon = 0.05$	$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$	$\epsilon = 0.45$	$\epsilon = 0.50$
2	12.2945 [10, 1, 1, 1]	12.2945 [9, 1, 1, 1]	12.2958 [9, 9, 1, 1]	12.2945 [8, 1, 1, 1]	12.2958 [8, 9, 1, 1]	12.3675 [7, 9, 7, 7]	12.3675 [7, 9, 7, 7]	12.2945 [6, 1, 1, 1]	12.3675 [6, 9, 7, 7]	12.3675 [5, 9, 7, 7]
4	7.7176 [2, 1, 1, 2]	7.7176 [2, 1, 1, 2]	7.7176 [2, 1, 1, 2]	7.7176 [2, 1, 1, 2]	7.7176 [2, 1, 1, 2]	7.7176 [2, 1, 1, 2]	7.7181 [2, 1, 7, 2]	7.8402 [1, 1, 7, 7]	7.8402 [1, 9, 1, 3]	7.9037 [11, 9, 1, 3]
6	3.2651 [2, 2, 2, 2]	3.2651 [2, 2, 2, 2]	3.2651 [2, 2, 2, 2]	3.2651 [2, 2, 2, 2]	3.2651 [2, 2, 2, 2]	3.2651 [2, 2, 2, 2]	3.3267 [3, 3, 2, 2]	3.2744 [2, 3, 2, 2]	3.4549 [11, 2, 3, 3]	3.4549 [11, 2, 3, 3]
8	−0.4517 [3, 3, 2, 2]	−0.4517 [3, 3, 2, 2]	−0.4517 [3, 3, 2, 2]	−0.4517 [3, 3, 2, 2]	−0.4517 [3, 3, 2, 2]	−0.3508 [4, 4, 2, 2]	−0.4517 [3, 3, 2, 2]	−0.4116 [11, 3, 2, 2]	−0.3730 [3, 4, 3, 3]	−0.2910 [5, 4, 3, 3]
10	−2.1314 [3, 3, 3, 3]	−2.1314 [3, 3, 3, 3]	−2.1314 [3, 3, 3, 3]	−2.1314 [3, 3, 3, 3]	−2.1314 [3, 3, 3, 3]	−1.9693 [4, 4, 4, 4]	−1.9996 [4, 3, 4, 4]	−2.0119 [3, 3, 4, 4]	−2.0755 [3, 4, 3, 3]	−2.0037 [5, 4, 3, 3]
12	−3.7345 [4, 4, 3, 3]	−3.7345 [4, 4, 3, 3]	−3.7345 [4, 4, 3, 3]	−3.7345 [4, 4, 3, 3]	−3.7345 [4, 4, 3, 3]	−3.5991 [4, 4, 4, 4]	−3.5627 [4, 5, 4, 4]	−3.4854 [6, 5, 4, 4]	−3.6533 [6, 4, 3, 3]	−3.6873 [5, 4, 3, 3]
14	−5.0713 [9, 4, 3, 5]	−5.0713 [9, 4, 3, 5]	−5.0430 [11, 5, 3, 3]	−5.0430 [11, 5, 3, 3]	−5.0713 [5, 4, 3, 5]	−5.0962 [7, 4, 4, 4]	−5.0350 [7, 5, 4, 4]	−5.0629 [6, 5, 4, 4]	−5.0713 [6, 4, 3, 7]	−5.0713 [5, 4, 3, 7]
16	−6.4510 [6, 5, 4, 4]	−6.4510 [6, 5, 4, 4]	−6.4363 [7, 5, 4, 4]	−6.4510 [6, 5, 4, 4]	−6.3823 [6, 6, 5, 5]	−6.4135 [7, 6, 4, 4]	−6.4363 [7, 5, 4, 4]	−6.4510 [6, 5, 4, 4]	−6.3225 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]
18	−7.4649 [7, 6, 5, 5]	−7.4649 [7, 6, 5, 5]	−7.4600 [7, 7, 5, 5]	−7.4490 [8, 7, 5, 5]	−7.4585 [8, 6, 5, 5]	−7.4490 [7, 6, 7, 7]	−7.4320 [7, 9, 7, 7]	−7.3956 [11, 9, 7, 4]	−7.3612 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]
20	−8.1561 [9, 7, 6, 6]	−8.1561 [9, 7, 6, 6]	−8.1548 [9, 7, 7, 7]	−8.1523 [8, 7, 7, 7]	−8.1520 [8, 9, 7, 7]	−8.1308 [11, 6, 7, 7]	−8.1138 [7, 9, 7, 7]	−7.6619 [11, 9, 7, 4]	−7.3612 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]

6 MODEL EVALUATION

The previous results suggest ISHM and CGGS can be efficient and effective in solving the OAP in a small controlled environment. Here, we investigate the performance of the proposed game-theoretical audit model on more realistic and larger datasets. This evaluation consists of comparing the quality of solutions of OAP with several natural alternative auditing strategies.

The first dataset, *Rea_A*, corresponds to the EMR access logs of Vanderbilt University Medical Center (VUMC). This dataset is notable because VUMC privacy officers rely on this data to conduct retrospective audits to determine if there are accesses that violate organizational policy. The central goal in this use case is to preserve patient privacy. Given that this is not a publicly available dataset, we conducted experiments with a second dataset, *Rea_B*, which consists of public observations of credit card applications. It labels applicants as having either low or high risk of fraud. We provide an audit mechanism to capture events of credit card fraud based on the features in this dataset. We use *Rea_B* to demonstrate the broad applicability of the proposed approaches and enable replication of our results.

Table 5. The approximation of the optimal solutions obtained by ISHM + CGGS at various levels of B and ϵ.

B	**Approximation of Optimal Loss of the Auditor and corresponding thresholds by ISHM + CGGS**									
	$\epsilon = 0.05$	$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$	$\epsilon = 0.35$	$\epsilon = 0.40$	$\epsilon = 0.45$	$\epsilon = 0.50$
2	12.2967 [1, 1, 1, 1]	12.2967 [1, 1, 1, 1]	12.3096 [9, 9, 1, 1]	12.2967 [1, 1, 1, 1]	12.3096 [8, 9, 1, 1]	12.3677 [7, 9, 7, 7]	12.3677 [7, 9, 7, 7]	12.2967 [1, 1, 1, 1]	12.3677 [6, 9, 7, 7]	12.3677 [5, 9, 7, 7]
4	7.7214 [2, 1, 1, 2]	7.7214 [2, 1, 1, 2]	7.7346 [2, 9, 1, 2]	7.7214 [2, 1, 1, 2]	7.7346 [2, 9, 1, 2]	7.7346 [2, 9, 1, 2]	7.7346 [2, 9, 1, 2]	7.9151 [1, 1, 1, 7]	7.8402 [1, 9, 1, 3]	7.9045 [11, 9, 1, 3]
6	3.2755 [2, 2, 2, 2]	3.2755 [2, 2, 2, 2]	3.2755 [2, 2, 2, 2]	3.2755 [2, 2, 2, 2]	3.2755 [2, 2, 2, 2]	3.2755 [2, 2, 2, 2]	3.3628 [3, 3, 2, 2]	3.3267 [2, 3, 2, 2]	3.4897 [11, 2, 3, 3]	3.3099 [2, 2, 3, 3]
8	−0.4422 [3, 3, 2, 2]	−0.4422 [3, 3, 2, 2]	−0.4422 [3, 3, 2, 2]	−0.4422 [3, 3, 2, 2]	−0.2761 [5, 2, 2, 7]	−0.3300 [4, 4, 2, 2]	−0.4006 [4, 3, 2, 2]	−0.4422 [3, 3, 2, 2]	−0.3404 [3, 4, 3, 3]	−0.2761 [5, 2, 3, 3]
10	−2.1203 [3, 3, 3, 3]	−2.1203 [3, 3, 3, 3]	−2.1203 [3, 3, 3, 3]	−2.1203 [3, 3, 3, 3]	−2.1203 [3, 3, 3, 3]	−1.9503 [4, 4, 4, 4]	−1.9873 [4, 3, 4, 4]	−2.0091 [3, 3, 4, 4]	−2.0612 [3, 4, 3, 3]	−1.9508 [5, 4, 3, 3]
12	−3.7215 [4, 4, 3, 3]	−3.7215 [4, 4, 3, 3]	−3.7215 [4, 4, 3, 3]	−3.7215 [4, 4, 3, 3]	−3.7215 [4, 4, 3, 3]	−3.5832 [4, 4, 4, 4]	−3.5448 [4, 5, 4, 4]	−3.4326 [6, 5, 4, 4]	−3.6383 [6, 4, 3, 3]	−3.6768 [5, 4, 3, 3]
14	−5.0709 [5, 9, 3, 4]	−5.1529 [5, 4, 4, 4]	−5.0430 [9, 4, 3, 3]	−5.0700 [6, 5, 3, 4]	−5.0698 [6, 4, 3, 7]	−5.0857 [7, 4, 4, 4]	−5.0125 [7, 5, 4, 4]	−5.0494 [6, 5, 4, 4]	−5.0698 [6, 4, 3, 7]	−5.0706 [5, 4, 3, 7]
16	−6.4394 [6, 5, 4, 4]	−6.4394 [6, 5, 4, 4]	−6.4258 [7, 5, 4, 4]	−6.4394 [6, 5, 4, 4]	−6.3683 [6, 6, 5, 5]	−6.4008 [7, 6, 4, 4]	−6.4258 [7, 5, 4, 4]	−6.4394 [6, 5, 4, 4]	−6.3038 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]
18	−7.4524 [7, 6, 5, 5]	−7.4524 [7, 6, 5, 5]	−7.4465 [7, 7, 5, 5]	−7.4363 [8, 7, 5, 5]	−7.4472 [8, 6, 5, 5]	−7.4359 [7, 6, 7, 7]	−7.4171 [7, 9, 7, 7]	−7.3825 [11, 5, 7, 7]	−7.3612 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]
20	−8.1448 [9, 7, 6, 6]	−8.1448 [9, 7, 6, 6]	−8.1433 [9, 7, 7, 7]	−8.1398 [8, 7, 7, 7]	−8.1388 [8, 9, 7, 7]	−8.1207 [11, 6, 7, 7]	−8.1043 [7, 9, 7, 7]	−7.6619 [11, 9, 7, 4]	−7.3612 [6, 9, 7, 7]	−6.1149 [5, 9, 7, 7]

Table 6. The average precision over the budget vector **B** by applying ISHM and ISHM+CGGS.

ϵ	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
γ_ϵ^1	0.9982	0.9982	0.9973	0.9974	0.9970	0.9634	0.9830	0.9680	0.9549	0.8982
γ_ϵ^2	0.9943	0.9959	0.9932	0.9940	0.9560	0.9562	0.9684	0.9700	0.9452	0.8966

Table 7. The number of threshold vectors checked by ISHM with a given budget B and step size ϵ.

ϵ	B									
	2	4	6	8	10	12	14	16	18	20
0.10	251	267	255	243	235	227	199	207	191	171
0.20	128	144	148	140	132	124	108	108	92	84
0.30	65	109	101	93	85	85	81	77	69	65
0.40	74	66	78	70	70	62	62	62	50	50
0.50	35	43	47	47	47	47	43	35	35	35

6.1 Data Overview

Rea_A consists of the VUMC EMR access logs for 28 continuous workdays during 2017. There are 48.6M access events, 38.7M (79.5%) of which are repeated accesses.[5] We filtered out the repeated accesses to focus on the distinct user-patient relationships established on a daily basis. The mean and standard deviation of daily access events was 355,602.18 and 195,144.99, respectively. The features for each event include: 1) timestamp, 2) patient ID, 3) employee ID, 4) patient's residential address, 5) employee's residential address, 6) employee's VUMC department affiliation and 7) indication of if a patient is an employee. We focus on the following alert types: 1) employee and patient share the same last name, 2) employee and patient work in the same VUMC department, 3)

[5]We define a repeated access as an access that is committed by the same employee to the same patient's EMR on the same day.

Table 8. Description of the EMR alert types.

ID	Alert Type Description	Mean	Std
1	Same Last Name	183.21	46.40
2	Department Co-worker	32.18	23.14
3	Neighbor ($\leq$ 0.5 miles)	113.89	80.44
4	Last Name; Same address	15.43	14.61
5	Last Name; Neighbor ($\leq$ 0.5 miles)	23.75	11.07
6	Same address; Neighbor ($\leq$ 0.5 miles)	20.07	11.49
7	Last Name; Same address; Neighbor ($\leq$ 0.5 miles)	32.07	16.54

employee and patient share the same residential address, and 4) employee and patient are neighbors within a distance threshold.

In certain cases, the same access may generate multiple alerts, each with a distinct type. For example, if a husband, who is a BMRC employee, accesses his wife's EMR, then two alert types may be triggered: 1 (same last name) and 3 (same address). We, therefore, redefine the set of alert types to also consider combinations of alert categories. The resulting set of alert types is detailed in Table 8.

We label each access event in the logs with a corresponding alert type or as "benign" (i.e., no alerts generated). To evaluate our methods, we choose a random sample of 50 employees and patients who generate at least one alert. This set of employees and the set of patients then results in 2500 *potential* accesses, where each employee can access each patient.

We let the probability that an employee could be malicious be 1, which is artificially high, but enables us to clearly compare the methods in the experiments. The benefit vector for the adversary is $[10, 12, 12, 24, 25, 25, 27]$ for the corresponding categories of alert types (1-7 in Table 8). The penalty for capture is set to 15. We set the cost of both an attack and an audit to 1. We acknowledge that the model parameters are *ad hoc*, but this does not affect the results of our comparative analysis. In practice, this would be accomplished based on expert opinion, but is outside the scope of this study.

Rea_B is the Statlog (German Credit Data) dataset available from the UCI Machine Learning Repository. *Rea_B* contains 1000 credit card applications. It is composed of 20 attributes describing the status of the applicants pertaining to their credit risk. Before issuing a credit card, banks would determine if it could be fraudulent based on the features in the data. Nevertheless, no screening process is perfect, and given a large number of applications, applications will require retrospective audits to determine whether specific applications should be canceled. Thus, alerts in this setting aim to indicate potential fraud and a subset of such alerts are chosen for a time-consuming auditing process. Leveraging the provided features, we define 5 alert types, which are triggered by the specific combinations of attribute values and the purpose of the application. The 8 selected purposes of application are the "victims" in our audit model. Table 9 summarizes how alerts are triggered. In the description field, italicized words represent the purpose of the application, while the other words represent feature values.

We used the 5 alert categorizations discussed above to label the 1000 applications with alert types, excluding any that fail to receive a label. Among these, we randomly selected 100 applicants who may choose to "attack" one of the 8 purposes of credit card applications, for a total of 800 possible events. The benefit vector for the adversary is $[15, 15, 14, 20, 18]$ for each of the alert types generated, respectively. We set the penalty for detection to 20 and costs for attack and audit were both set to 1. Again, to facilitate comparison we set $p_e = 1$ in all cases.

Table 9. Description of the defined alert types.

ID	Alert type Description	Mean	Std
1	No checking account, *Any purpose*	370.04	15.81
2	Checking < 0, *New car, Education*	82.42	7.87
3	Checking > 0, Unskilled, *Education*	5.13	2.08
4	Checking > 0, Unskilled, *Appliance*	28.21	5.25
5	Checking > 0, Critical account, *Business*	8.31	2.96

6.2 Comparison with Baseline Alternatives

The performance of the proposed audit model was investigated by comparing with several natural alternative audit strategies as baselines. The first alternative is to randomize the audit order over alert types, which we call *Audit with random orders of alert types*. Though random, this strategy mimics the reality of random reporting (e.g., where a random patient calls a privacy official to look into alleged suspicious behavior with respect to the use of their EMR). In this case, we adopt the thresholds out of the proposed model with $\epsilon = 0.1$ to investigate the performance. The second alternative is to randomize the audit thresholds. We refer to this policy as *Audit with random thresholds*. For this policy, we assume that 1) the auditor's choice satisfies $\sum_i b_i \geq B$ and 2) the auditor has the ability to find the optimal audit order after deciding upon the thresholds. The third alternative is a naive greedy audit strategy, where the auditor prioritizes alert types according to their utility loss (i.e., greater consequence of violations). In this case, the auditor investigates as many alerts of a certain type as possible before moving on to the next type in the order. For our experiments, when the alert type order is based on the loss of the auditor, which is the benefit the adversary receives when they execute a successful attack. Thus, we refer to this strategy as *Audit based on benefit*.

The following performance comparisons are assessed over a broad range of auditing budgets. For our model, we present the values of the objective function with three different instances of the step size ϵ in ISHM: $[0.1, 0.2, 0.3]$. Figures 1 and 2 summarize the performance of the proposed audit model and three alternative audit strategies for *Rea_A* and *Rea_B*, respectively.

For dataset *Rea_A*, the range of B was set to 10 through 100. The budget of 100 covers about 1/4 of the sum of the means of the seven alert types. In reality, such coverage is quite high. By applying the proposed audit model, we approximately solve the OAP given B and ϵ. For *Audit with random orders of alert types*, we assign the audit thresholds using ISHM with $\epsilon = 0.1$. The randomization is repeated 2000 times without replacement. As for *Audit with random thresholds*, we randomly generate the audit thresholds to solve the corresponding LP, which are repeated 5000 times. For *Audit based on benefit*, we randomly sample 2000 instances of $\mathbf{Z}$ based on the distributions of alert types learned from the dataset.

Based on Figure 1, there are several findings we wish to highlight. First, in our model, as the audit budget increases, the auditor's loss decreases. At the high end, when $B \geq 90$, the auditor's loss is zero, which, in the VUMC audit setting, implies that all the potential adversaries are deterred from an attack. This valuation of B is smaller than 1/4 of the sum of distribution means of all alert types. The reason for this phenomenon stems from the fact that when the audit budget increases, the audit model finding better approximations of the optimal audit thresholds, which, in turn, enables the auditor to significantly limit the potential gains of the adversaries. Second, our proposed model significantly outperforms all of the baselines. Third, even though *Audit with random orders of alert types* uses approximated audit thresholds, the auditor's loss is substantially greater than our proposed approach. However, the auditor's losses for the alternatives approach ours when $B = 20$.

This is because the thresholds are [0, 0, 0, 7, 0, 11, 8], such that the audit order is less of a driver than in other situations. Fourth, *Audit based on benefit* tends to have very poor performance compared to other policies. This is because when the audit order is fixed (or is predictable), adversaries have greater evasion ability and attack more effectively. Fifth, *Audit with random thresholds* tends to outperform the other baselines but is still significantly worse than our approach. The is because the auditor has the ability to search for the optimal audit policy, but the thresholds are randomly assigned such that they are hampered in achieving the best solution.

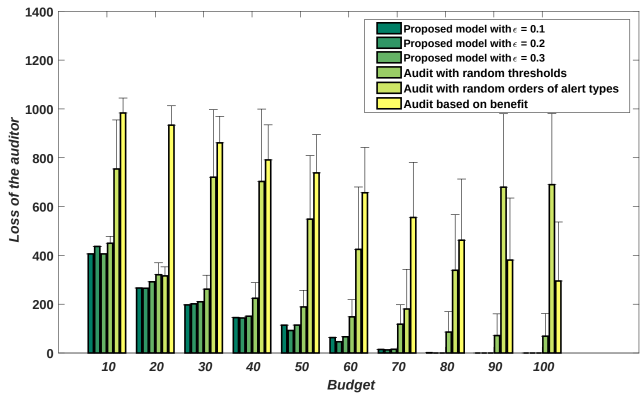

Fig. 1. Auditor's loss in the proposed and baseline models in the *Rea_A* dataset.

For the credit card application scenario, Figure 2 compares the auditor's loss in our heuristics and the three baselines. For dataset *Rea_B*, the range for B is 10 to 250 with a step size of 20. As expected, as the budget increases, the auditor sustains a decreasing average loss. It can be seen that the proposed audit model significantly outperforms the alternative baselines. Specifically, as the auditing budget increases, the auditor's loss trends towards, and becomes, 0 in our approach. This means that the attackers are completely deterred. For the alternatives, as before, *Audit with random thresholds* outperforms other strategies. And, just as before, the strategy that greedily audits alert types (in order of loss) tends to perform quite poorly.

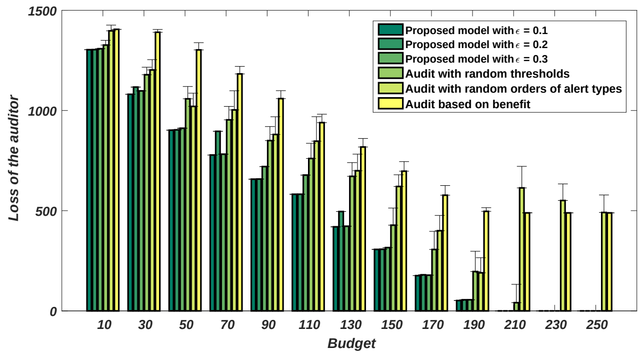

Fig. 2. Loss of the auditor in the proposed and alternatives audit model in the *Rea_B* dataset.

7 DISCUSSION AND CONCLUSIONS

TDMTs are usually deployed in database systems to address a variety of attacks that originate from within and beyond an organization. However, an overwhelming alert volume is far beyond the capability of auditors with limited resources. Our research illustrates that policy compliance auditing, as a significant component of database management, can be improved by prioritizing which alerts to focus on via a game theoretic framework, allowing auditing policies to make the best use of limited auditing resources while simultaneously accounting for the strategic behavior of potential policy violators. This is notable because auditing is critical to a wide range of management requirements, including privacy breach and financial fraud investigations. As such, this model and the effective heuristics we offer in this study fill a major gap in the field.

There are several limitations of our approach that we wish to highlight as opportunities for future investigations. First, there are limitations to the parameterization of the game. One notable aspect is that we assumed that the game has a zero-sum property. Yet in reality, this may not be the case. For example, an auditor is likely to be concerned less about the cost incurred by an adversary for executing an attack and more concerned about the losses that arise from successful violations Additionally, while our experiments show the proposed audit model outperforms natural alternatives, it is unclear how sensitive this result is to parameter variations. Thus, a fruitful direction of research is in the payoff structures and how they influence the performance of the model.

A second set of limitations stems from the assumptions we rely upon. In particular, we assumed that each attack is instantaneous, which turned the problem into a one-shot two-stage game. However, attacks in the wild may require multiple cycles to fully execute, such that the auditor may be able to capture the attacker before they complete their exploit. To address such a setting, a temporal audit model may complement the approach introduced in this paper. Furthermore, our model is predicated on an environment in which the auditor has complete knowledge, including the identities, about the set of potential adversaries. However, in practice, one player can hardly know everything about the other. Thus, a natural follow-up investigation is to relax such a strong assumption by involving uncertainty in the knowledge of the players.

A third limitation is in the economic premise of the attack. Specifically, we expected the interaction between the auditor and adversaries as fully rational. In reality, adversaries may be bounded in their rationality, and an important extension would be to generalize the model consider such behavior.

8 CONCLUSION

Prioritizing the alerts raised by TDMT modules can enable effective auditing of privacy- and security-related incidents. This paper introduced a game theoretic model to represent the strategic interactions between an auditor and a set of potential adversaries. We showed that discovering the optimal prioritization of alerts is NP-hard, but that several efficient search heuristics can be designed to solve the problem. Using a controlled, synthetic, dataset, we proved that the heuristics can achieve a performance that is close to the optimal solution. And, using several different types of datasets illustrated that the heuristics are substantially more effective at prioritization than typical auditing strategies invoked in practice. We did, however, make several simplifying assumptions regarding the behavior of the adversaries and the parameterization of the variables in the model, but believe that this research provides a foundation for further investigation in alert prioritization games.

9 ACKNOWLEDGEMENT

This work was supported, in part by grant R01LM10207 from the National Institutes of Health, grants CNS-1526014, CNS-1640624, IIS-1649972 and IIS-1526860 from the National Science Foundation, grant N00014-15-1-2621 from the Office of Naval Research and grant W911NF-16-1-0069 from the Army Research Office.

REFERENCES

[1] Lillian Ablon, Martin C. Libicki, and Andrea A. Golay. 2014. *Markets for cybercrime tools and stolen data: hackers' bazaar.*

[2] Rakesh Agrawal and Chris Johnson. 2007. Securing electronic health records without impeding the flow of information. *International Journal of Medical Informatics* 76, 5-6 (2007), 471–479.

[3] Khalid Alsubhi, Issam Aib, and Raouf Boutaba. 2012. FuzMet: A fuzzy-logic based alert prioritization engine for intrusion detection systems. *International Journal of Network Management* 22, 4 (2012), 263–284.

[4] Bo An, Fernando Ordóñez, Milind Tambe, Eric Shieh, Rong Yang, Craig Baldwin, Joseph DiRenzo III, Kathryn Moretti, Ben Maule, and Garrett Meyer. 2013. A deployed quantal response-based patrol planning system for the US Coast Guard. *Interfaces* 43, 5 (2013), 400–420.

[5] Ron Ben-Natan. 2008. System and methods for nonintrusive database security. (Oct. 14 2008). US Patent 7,437,362.

[6] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50, 3 (2011), 602–613.

[7] Jeremiah Blocki, Nicolas Christin, Anupam Datta, Ariel Procaccia, and Arunesh Sinha. 2014. Audit games with multiple defender resources. *arXiv preprint arXiv:1409.4503* (2014).

[8] Jeremiah Blocki, Nicolas Christin, Anupam Datta, Ariel D Procaccia, and Arunesh Sinha. 2013. Audit games. *arXiv preprint arXiv:1303.0356* (2013).

[9] Aziz A Boxwala, Jihoon Kim, Janice M Grillo, and Lucila Ohno-Machado. 2011. Using statistical and machine learning to help institutions detect suspicious access to electronic health records. *Journal of the American Medical Informatics Association* 18, 4 (2011), 498–505.

[10] R Brause, T Langsdorf, and Michael Hepp. 1999. Neural data mining for credit card fraud detection. In *Proceedings of the International Conference on Tools with Artificial Intelligence.* 103–106.

[11] Matthew Brown, Arunesh Sinha, Aaron Schlenker, and Milind Tambe. 2016. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 425–431.

[12] Philip K Chan, Wei Fan, Andreas L Prodromidis, and Salvatore J Stolfo. 1999. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and their Applications* 14, 6 (1999), 67–74.

[13] Rong-Chang Chen, Tung-Shou Chen, and Chih-Chiang Lin. 2006. A new binary support vector system for increasing detection rate of credit card fraud. *International Journal of Pattern Recognition and Artificial Intelligence* 20, 02 (2006), 227–239.

[14] You Chen, Steve Nyemba, and Bradley Malin. 2012. Detecting anomalous insiders in collaborative information systems. *IEEE Transactions on Dependable and Secure Computing* 99 (2012), 1–1.

[15] You Chen, Steve Nyemba, Wen Zhang, Bradley Malin, HY Shahir, U Glässer, R Farahbod, P Jackson, H Wehn, K Glass, et al. 2012. Specializing network analysis to detect anomalous insider actions. *Security Informatics* 1, 1 (2012), 5.

[16] Vincent Conitzer. 2016. On Stackelberg mixed strategies. *Synthese* 193, 3 (2016), 689–703.

[17] William R Cook and Martin R Gannholm. 2004. Rule based database security system and method. (Nov. 16 2004). US Patent 6,820,082.

[18] Frédéric Cuppens and Alexandre Miege. 2002. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the IEEE Symposium on Security and Privacy.* 202–215.

[19] Linda Delamaire, HAH Abdou, and John Pointon. 2009. Credit card fraud and detection techniques: A review. *Banks and Bank Systems* 4, 2 (2009), 57–68.

[20] Daniel Fabbri and Kristen LeFevre. 2011. Explanation-based auditing. *Proceedings of the VLDB Endowment* 5, 1 (2011), 1–12.

[21] Daniel Fabbri and Kristen LeFevre. 2013. Explaining accesses to electronic medical records using diagnosis information. *Journal of the American Medical Informatics Association* 20, 1 (2013), 52–60.

[22] Daniel Fabbri, Ravi Ramamurthy, and Raghav Kaushik. 2013. Select triggers for data auditing. In *Proceedings of the IEEE International Conference on Data Engineering.* 1141–1152.

[23] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Brian C Schwedock, Milind Tambe, and Andrew Lemieux. 2017. PAWS – a deployed game-theoretic application to combat

poaching. *AI Magazine* 38, 1 (2017), 23–36.

[24] Rajesh Ganesan, Sushil Jajodia, Ankit Shah, and Hasan Cam. 2016. Dynamic Scheduling of cybersecurity analysts for minimizing risk using reinforcement learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 4.

[25] Carl Gunter, David Liebovitz, and Bradley Malin. 2011. Experience-based access management. *IEEE Security and Privacy Magazine* 9 (2011), 48–55.

[26] Ashish Kamra and Elisa Ber. 2009. Survey of machine learning methods for database security. *Machine Learning in Cyber Trust* (2009), 53–71.

[27] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. 2008. Detecting anomalous access patterns in relational databases. *the International Journal on Very Large Data Bases* 17, 5 (2008), 1063–1077.

[28] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 689–696.

[29] Peter Kieseberg, Bernd Malle, Peter Frühwirt, Edgar Weippl, and Andreas Holzinger. 2016. A tamper-proof audit and control system for the doctor in the loop. *Brain Informatics* 3, 4 (2016), 269–279.

[30] Horacio D Kuna, Ramón García-Martinez, and Francisco R Villatoro. 2014. Outlier detection in audit logs for application systems. *Information Systems* 44 (2014), 22–33.

[31] Aron Laszka, Yevgeniy Vorobeychik, Daniel Fabbri, Chao Yan, and Bradley Malin. 2017. A Game-Theoretic approach for alert prioritization. In *AAAI Workshop on Artificial Intelligence for Cyber Security*.

[32] Federico Maggi, Matteo Matteucci, and Stefano Zanero. 2009. Reducing false positives in anomaly detectors through fuzzy alert aggregation. *Information Fusion* 10, 4 (2009), 300–311.

[33] Sunu Mathew, Michalis Petropoulos, Hung Q Ngo, and Shambhu J Upadhyaya. 2010. A data-Centric approach to insider attack detection in database systems. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses*. 382–401.

[34] Andrew McAfee and Erik Brynjolfsson. 2012. Big data: The management revolution. *Harvard Business Review* Oct (2012), 3–9.

[35] EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. *Decision Support Systems* 50, 3 (2011), 559–569.

[36] Humphrey Waita Njogu and Luo Jiawei. 2010. Using alert cluster to reduce IDS alerts. In *Proceedings of the International Conference on Computer Science and Information Technology*, Vol. 5. 467–471.

[37] Sara Ashley O'Brien. 2017. Giant Equifax data breach: 143 million people could be affected. http://money.cnn.com/2017/09/07/technology/business/equifax-data-breach/index.html. (2017).

[38] Lillian Rostad and Ole Edsberg. 2006. A study of access control requirements for healthcare systems based on audit trails from access logs. In *Proceedings of the Annual Computer Security Applications Conference*. 175–186.

[39] Sriram Samu, Namit Jain, and Wei Wang. 2002. Database system event triggers. (June 11 2002). US Patent 6,405,212.

[40] Aaron Schlenker, Milind Tambe, Christopher Kiekintveld, Haifeng Xu, Mina Guirguis, Arunesh Sinha, Solomon Sonya, Noah Dunstatter, and Darryl Balderas. 2017. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[41] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. 2008. Credit card fraud detection using hidden Markov model. *IEEE Transactions on Dependable and Secure Computing* 5, 1 (2008), 37–48.

[42] Michael Sternberg and Robert G. Reynolds. 1997. Using cultural algorithms to support re-engineering of rule-based expert systems in dynamic performance environments: a case study in fraud detection. *IEEE Transactions on Evolutionary Computation* 1, 4 (1997), 225–243.

[43] Mubeena Syeda, Yan-Qing Zhang, and Yi Pan. 2002. Parallel granular neural networks for fast credit card fraud detection. In *Proceedings of the IEEE International Conference on Fuzzy Systems*. 572–577.

[44] Chee-Wooi Ten, Govindarasu Manimaran, and Chen-Ching Liu. 2010. Cybersecurity for Critical Infrastructures: Attack and Defense Modeling. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 40, 4 (2010), 853–865.

[45] Liang Tong, Sixie Yu, Scott Alfeld, and Yevgeniy Vorobeychik. 2018. Adversarial Regression with Multiple Learners. In *Proceedings of the 35th International Conference on Machine Learning*. 4946–4954.

[46] Shisong Xiao, Yugang Zhang, Xuejiao Liu, and Jingju Gao. 2008. Alert fusion based on cluster and correlation analysis. In *Proceedings of the International Conference on Convergence and Hybrid Information Technology*. 163–168.

[47] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2 (2009), 2473–2480.

[48] Sixie Yu, Yevgeniy Vorobeychik, and Scott Alfeld. 2018. Adversarial Classification on Social Networks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 211–219.

Received May 2018; accepted March 2019