A Game-Theoretic Approach for Alert Prioritization

Aron Laszka and **Yevgeniy Vorobeychik** and **Daniel Fabbri** and **Chao Yan** and **Bradley Malin**
Vanderbilt University

Abstract

The quantity of information that is collected and stored in computer systems continues to grow rapidly. At the same time, the sensitivity of such information (e.g., detailed medical records) often makes such information valuable to both external attackers, who may obtain information by compromising a system, and malicious insiders, who may misuse information by exercising their authorization. To mitigate compromises and deter misuse, the security administrators of these resources often deploy various types of intrusion and misuse detection systems, which provide alerts of suspicious events that are worthy of follow-up review. However, in practice, these systems may generate a large number of false alerts, wasting the time of investigators. Given that security administrators have limited budget for investigating alerts, they must prioritize certain types of alerts over others. An important challenge in alert prioritization is that adversaries may take advantage of such behavior to evade detection - specifically by mounting attacks that trigger alerts that are less likely to be investigated. In this paper, we model alert prioritization with adaptive adversaries using a Stackelberg game and introduce an approach to compute the optimal prioritization of alert types. We evaluate our approach using both synthetic data and a real-world dataset of alerts generated from the audit logs of an electronic medical record system in use at a large academic medical center.

1 Introduction

The quantity of sensitive information that is collected and stored on computer systems grows steadily. At the same time, the number cybersecurity incidents also grows at an alarming rate (PricewaterhouseCoopers 2016). In recent years, we have seen a number of high-profile attacks, which have demonstrated that determined and resourceful attackers can penetrate even highly secure systems. The impact of these cyberattacks can be disastrous, as evidenced by large-scale data breaches, such as the 2015 Anthem data breach, which exposed the personal information of as many as 80 million individuals (Hiltzik 2015).

Further, computer systems are also threatened by insider attacks, in which individuals that are authorized to access a system misuse their authorization (Silowash et al. 2012; Farahmand and Spafford 2013). According to the 2014 U.S. State of Cybercrime Survey, which is based on responses from 557 organizations of various sizes, 28% of electronic crime events were caused by insiders, while 46% of the participating organizations reported that crimes perpetrated by insiders were more damaging (CSO Magazine et al. 2014). Insiders pose a serious threat because they harbor detailed knowledge and are authorized to access the target system. Specifically, they can exploit such capabilities to circumvent (or simply click through) technical security measures. For example, a CERT investigation of 23 insider attacks showed that in 78% of incidents, insiders were indeed authorized users with live computer accounts. Moreover, in 43% of the incidents, the insiders invoked their own usernames and passwords to perpetrate the attack (Randazzo et al. 2005).

To address the threats posed by intrusions and misuse, security administrators often deploy intrusion and misuse detection systems. These detectors can continuously monitor a computer system or network for signs of malicious activity, and raise an alert when they encounter such activity. These alerts can then be investigated by security administrators, who may mitigate the attacks through appropriate response measures. Further, the possibility of detection also deters insiders from mounting an attack since they now face the risk of being held accountable.

Unfortunately, practical detection systems also have the potential to generate a large number false alerts. This is because, in practice, it is challenging to differentiate unusual but non-malicious activities from attacks. This is problematic because the defenders of such systems (e.g., administrators who must work within the confines of their business) often have limited time and resources to spend on investigations. This budget is typically insufficient for investigating every alert that has been generated. (Rostad and Edsberg 2006). As a result, defenders must establish a policy for deciding which alerts to investigate using their limited budget. Since defenders do not know if an alert is false or not until they investigate it, this problem is inherently challenging.

Alert prioritization must consider a variety of factors. These include, but are not limited to, i) potential magnitude of loss from undetected attacks, ii) the available budget, and iii) the probability that an attack raises a certain alert. Since attackers can often reliably estimate these factors in practice,

they are also often able to infer the defenders' prioritization policies. Thus, an important challenge in alert prioritization stems from the fact that attackers may use such knowledge to evade detection. Specifically, they can do so by mounting attacks that trigger alerts that are less likely to be investigated by defenders.

It should be recognized that the notion of alert prioritization is not a novel concept in artificial intelligence, and considerable research has been performed on the topic (see, for example, the surveys (Hubballi and Suryanarayanan 2014; Salah, Maciá-Fernández, and Díaz-Verdejo 2013)). However, to the best of our knowledge, there has been no investigation into strategic adversaries, who may adapt to the prioritization chosen by a defender.

Thus, in this paper, we introduce a game-theoretic approach for alert prioritization. Note that intrusions by external attackers and misuse by insiders differ in both how malicious actions may be carried out, as well as how they may be detected. However, in the context of alert prioritization, we can study both threats using a single framework, in which attacks model either intrusions or misuse. For convenience, in the remainder of the paper, we will refer to potential perpetrators as adversaries, and to potential malicious actions as attacks.

Note that even though alert prioritization may resemble audit games (Blocki et al. 2013; 2015) at first glance, there are fundamental differences between the two. In audit games, the set of targets that can be audited is fixed, and both players know this set at the beginning of the game. In alert prioritization, the set of alerts raised is non-deterministic and its cardinality may be multiple orders of magnitude higher than the typical number of audit-game targets considered in prior work. Further, we cannot simply map alert types (see Section 2) to audit-game targets because 1) alert types are non-atomic in the sense that the defender might end up investigating only a subset of the alerts of a given type and 2) alert types contain a non-deterministic number of alerts.

The remainder of this paper is organized as follows. In Section 2, we introduce a game-theoretic model of alert prioritization in the presence of strategic adversaries. In Section 3, we present analytical results on our model and propose a solution approach for finding an optimal prioritization strategy. In Section 4, we present numerical results with our approach using synthetic and real-world datasets. Finally, in Section 5, we provide concluding remarks and recommendations for next steps in this line of research.

2 Model

In this work, we model the problem of alert prioritization as a Stackelberg security game between a defender and an adversary. For reference purposes, we provide definitions for the common notation used in this work in Table 1.

We assume that the defender has deployed an *intrusion detection system* (IDS), which may raise alerts for actual attacks, but which may also raise false alerts in the course of routine system behavior. The possible alerts that can be raised by the IDS are partitioned into a set of *alert types* T, such that alerts belonging to the same type appear equally important (before they have been investigated). We assume

Table 1: Legend of Common Symbols

Symbol	Description
T	set of alert types
O	set of possible prioritizations of the alert types T
A	set of possible attacks
B	defender's budget for investigating alerts
D_a	defender's loss when attack a is not detected
G_a	adversary's gain when attack a is not detected
C_t	cost of investigating an alert of type t
K_a	cost of mounting attack a
$R_{a,t}$	probability that attack a raises an alert of type t
$F_t(n)$	probability that there are at most n false alerts of type t

that the defender incurs a *cost* C_t when investigating an alert of type t (e.g., the manpower spent on the investigation). The defender is allocated a *fixed budget* B (e.g., the available manpower) to spend on investigating alerts. We further assume that the budget B as well as each cost C_t is an integer value. [1]

An adversary may mount any of the *possible attacks* A against the defender's computer system (e.g., compromising a database server or a web server), and each attack may raise multiple alerts (though it may also raise no alerts). The *probability* that attack $a \in A$ raises an alert of type $t \in T$ is denoted by $R_{a,t}$. Independently of the attack, the IDS may also generate any number of false alerts. We assume that the number of false alerts for each type $t \in T$ follows some known probability distribution. We let F_t denote the cumulative distribution function of the number of alerts of type t. Additionally, assuming that an adversarial alert is generated at a random point in time, we let F_t^* denote the cumulative distribution function of the number of false alerts that are generated before the true adversarial alert.

2.1 Strategy Sets

The adversary's pure-strategy choice is to select an attack a from the set of possible attacks A. The defender's pure strategies are *prioritizations* over the alert types T. We represent a prioritization as a vector $\boldsymbol{o} = (o_1, o_2, \ldots, o_{|T|})$, where $o_i \in T$, and $o_i \neq o_j$ if $i \neq j$ (i.e., each o_i is an alert type, and every alert type t is listed only once). We let the set of all possible prioritizations be denoted by O.

When the defender invokes prioritization $\boldsymbol{o}$, they investigate alerts of type o_1 first, then move on to types o_2, o_3, and so forth.[2] This proceeds until the defender has exhausted its

[1] Note that this is for computational convenience. Fractions of arbitrary precision can be represented by linearly scaling up B and every C_t.

[2] Note that alerts within a type can be investigated in a random order to prevent an attacker from minimizing the probability of de-

budget B or, in rare instances, until there are no more alert types left to investigate. Now, consider an alert of type t that was raised due to an attack. Suppose that type t is the kth element of the prioritization (i.e., $o_k = t$). Then, this adversarial alert will be investigated if and only if

$$C_{o_k} + \sum_{i=1}^{k} n_i \cdot C_{o_i} \leq B, \tag{1}$$

where n_i, $i = 1, \ldots, k-1$, is the number of false alerts of type o_i, and n_k is the number of false alerts of type o_k that were raised before the adversarial alert. Note that we have to add cost C_{o_k} since the defender's budget must also include investigating the adversarial alert itself.

As a result, the probability that an adversarial alert of type o_k will be investigated is

$$PI(\boldsymbol{o}, k) = \sum_{\substack{\boldsymbol{n}: \\ C_{o_k} + \sum_{i=1}^{k} n_i \cdot C_{o_i} \leq B}} \left[\left(F^*_{o_k}(n_k) - F^*_{o_k}(n_k - 1) \right) \cdot \prod_{i=1}^{k-1} \left(F_{o_i}(n_i) - F_{o_i}(n_i - 1) \right) \right] \tag{2}$$

since the defender investigates the alarm if and only if the numbers of false alerts satisfy Equation (1), and the probability of generating n_i false alerts of type o_i is $F_{o_i}(n_i) - F_{o_i}(n_i - 1)$ by definition.

We assume that the defender *detects* an attack if it investigates any of the alerts raised due to the attack. Consequently, an attack is detected if the defender does not exhaust its alert-investigation budget before reaching the highest priority adversarial alert. Hence, if the set of alert types raised by the attack is $\hat{T}$, then the attack is detected with probability

$$PI\left(\boldsymbol{o}, \min\left\{ i \in \{1, \ldots, |T|\} \;\middle|\; o_i \in \hat{T} \right\}\right). \tag{3}$$

Therefore, the probability that the defender detects an attack $a \in A$ using prioritization $\boldsymbol{o}$ is

$$PD(\boldsymbol{o}, a) = \sum_{\hat{T} \subseteq T} \prod_{t \in \hat{T}} R_{a,t} \prod_{t \in T \setminus \hat{T}} (1 - R_{a,t}) \, PI\left(\boldsymbol{o}, \min\{i \mid o_i \in \hat{T}\}\right). \tag{4}$$

2.2 Payoffs

The adversary's expected gain from attack a when the defender uses prioritization $\boldsymbol{o}$ is

$$EG(\boldsymbol{o}, a) = (1 - PD(\boldsymbol{o}, a)) \cdot G_a - K_a, \tag{5}$$

while the defender's expected loss from invoking prioritization $\boldsymbol{o}$ when the adversary mounts attack a is

$$EL(\boldsymbol{o}, a) = (1 - PD(\boldsymbol{o}, a)) \cdot L_a. \tag{6}$$

A mixed strategy for the defender is a probability distribution over its pure strategies, that is, a probability distribution over the possible prioritizations O. We let $p_{\boldsymbol{o}}$ denote the probability that the defender chooses strategy $\boldsymbol{o}$. Then, the adversary's expected gain from mounting attack a when the defender uses mixed strategy $\boldsymbol{p}$ is

$$\sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot EG(\boldsymbol{o}, a). \tag{7}$$

Meanwhile, the defender's expected loss is

$$\sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot EL(\boldsymbol{o}, a). \tag{8}$$

2.3 Optimal Prioritization

Following Kerckhoffs's principle (Kerckhoffs 1883), we assume that an adversary can infer the defender's strategy (e.g., can obtain the same software or use the same algorithms as the defender). As a result, the adversary can adapt its attack to the defender's strategy. Therefore, we assume that an adversary will always choose a *best-response strategy*, which is formally defined as follows.

Definition 1 (Adversary's Best-Response Strategy)**.** The adversary's *best-response strategies* $BR(\boldsymbol{p})$ against a given mixed strategy $\boldsymbol{p}$ are the set of attacks that maximize the adversaries's expected gain. Formally,

$$BR(\boldsymbol{p}) = \operatorname*{argmax}_{a \in A} \sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot EG(\boldsymbol{o}, a). \tag{9}$$

In contrast, the defender cannot know in advance which attack the adversary will mount. However, the defender can anticipate that the adversary will choose a best response. As is typical in the security literature, we consider subgame perfect Nash equilibria as our solution concept (Korzhyk et al. 2011). As such, we will refer to the defender's equilibrium strategies as *optimal strategies* for the remainder of the paper.

Definition 2 (Defender's Optimal Strategy)**.** A mixed strategy is an *optimal strategy* if it minimizes the defender's expected loss given that an adversary will always choose a best response with tie-breaking in favor of the defender. Formally, a mixed strategy $\boldsymbol{p}^*$ is optimal if it minimizes

$$\min_{\boldsymbol{p}, a \in BR(\boldsymbol{p})} \sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot EL(\boldsymbol{o}, a). \tag{10}$$

Note that the effect of the tie-breaking rule is negligible in practice. The only purpose it serves is to avoid pathological mathematical cases where no optimal strategy would exist.

3 Analysis

Given the model design, we now derive its analytic aspects, with a particular focus on its computational components. First, in Section 3.1, we prove that finding an optimal prioritization is a computationally hard problem. Then, in Section 3.2, we show how to compute the detection probability $PD(\boldsymbol{o}, a)$ in polynomial time for a given prioritization $\boldsymbol{o}$ and attack a. Finally, in Section 3.3, we formulate the problem of finding an optimal prioritization as a set of linear programs and introduce a column-generation approach that works exceptionally well in practice.

tection by carefully timing its attack.

3.1 Computational Complexity

To establish our computational-complexity result, we first formulate the problem of finding an optimal prioritization as a decision problem.

Definition 3 (Optimal Prioritization Problem (OPP))**.** Given an instance of the prioritization game and a threshold loss value L^*, determine if there exists a mixed strategy $\boldsymbol{p}$ for the defender such that for every $a \in BR(\boldsymbol{p})$, the defender's expected loss is at most L^*.

Theorem 1. *OPP is NP-hard.*

We show that the OPP is NP-hard using a reduction from a well-known NP-hard problem, the Set Cover Problem.

Definition 4 (Set Cover Problem (SCP))**.** Given a base set U, a family $\mathcal{S}$ of subsets of U, and a threshold size k, determine if there exists a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of cardinality k whose union is U.

Proof sketch. Given an instance of SCP, we construct an instance of OPP as follows:

- let $T = \mathcal{S}$, $A = U$, and $B = k$;
- for each $a \in A$, let $D_a = G_a = 1$ and $K_a = 0$;
- for each $t \in T$, let $C_t = 1$;
- for each $a \in A$ and $t \in T$, let $R_{a,t} = 1$ if $a \in t$ and 0 otherwise (note that an attack a corresponds to an element of the set U and a type t corresponds to subsets from $\mathcal{S}$);
- for each $t \in T$, let $F_t(0) = 0$, $F_t(n) = 1$ for $n > 0$, and $F_t^*(n) = 1$ for all n;
- the threshold loss is $L^* = 0$.

It is clear that the reduction can be performed in polynomial time. As a consequence, we only need to show that OPP has a solution if and only if SCP does.

To do so, first let us suppose that there exists a set cover $\mathcal{C}$ of cardinality k. Let $\boldsymbol{o}$ be a prioritization in which the first $B = k$ alert types are the subsets from $\mathcal{C}$ (i.e., $\{o_1, \dots o_k\} = \mathcal{C}$). Then, the mixed strategy $\boldsymbol{p}$ that always uses prioritization $\boldsymbol{o}$ (i.e., $p_{\boldsymbol{o}} = 1$) is a solution for OPP because it detects every attack with probability 1.

Second, let us suppose that there exists a mixed strategy $\boldsymbol{p}$ that achieves $L^* = 0$ loss. Let $\boldsymbol{o}$ be an arbitrary prioritization from the support of this strategy (i.e., $p_{\boldsymbol{o}} > 0$). Then, it follows that this prioritization $\boldsymbol{o}$ must detect every attack with probability 1; otherwise, the defender's expected loss would be non-zero. Hence, the subfamily $\mathcal{C}$ that comprises the first $B = k$ elements of $\boldsymbol{o}$ (i.e., $\mathcal{C} = \{o_1, \dots, o_B\}$) is a solution to SCP since it forms a set cover of U. □

3.2 Computing $PD(\boldsymbol{o}, a)$ in Polynomial Time

Next, we address the problem of computing the detection probabilities PD, which play a crucial role in the solution that we present in Section 3.3. Computing $PD(\boldsymbol{o}, a)$ for a given prioritization $\boldsymbol{o}$ and attack a is challenging because the right-hand sides of Equations (2) and (4), which together define $PD(\boldsymbol{o}, a)$, both comprise exponential numbers of terms. Consequently, it is imperative that we find a better way of computing the probabilities PD.

Firstly, we can estimate $PD(\boldsymbol{o}, a)$ in practice using simulations: repeatedly draw false-alert numbers and adversarial alert types from the distributions F, F^*, and the probabilities $R_{a,\cdot}$, and calculate the ratio of instances in which attack a is detected. However, for distributions with large variability, reliable estimation may require a very large number of repetitions. As a more efficient alternative, we propose a dynamic-programming algorithm that can compute the exact value of $PD(\boldsymbol{o}, a)$ for any prioritization $\boldsymbol{o}$ and attack a in polynomial time.

Algorithm 1 Computing $PD(\boldsymbol{o}, a)$

Input: prioritization game, prioritization $\boldsymbol{o}$, attack a
for $b = 0, 1, \dots, B$ **do**
  $PD(\boldsymbol{o}, a, |T|, b) \leftarrow R_{a,o_{|T|}} \cdot F^*_{o_{|T|}}\big(\lfloor b/C_{o_{|T|}} \rfloor - 1\big)$
end for
for $i = |T| - 1, \dots, 2, 1$ **do**
  for $b = 0, 1, \dots, B$ **do**
    $PD(\boldsymbol{o}, a, i, b) \leftarrow R_{a,o_i} \cdot F^*_{o_i}(\lfloor b/C_{o_i} \rfloor - 1)$
    $+ (1 - R_{a,o_i}) \sum_{j=0}^{\lfloor b/C_{o_i} \rfloor} \Big[(F_{o_i}(j) - F_{o_i}(j-1)) \cdot PD(\boldsymbol{o}, a, b - j \cdot C_{o_i}, i+1) \Big]$
  end for
end for
Return $PD(\boldsymbol{o}, a) := PD(\boldsymbol{o}, a, 1, B)$

Proposition 1. *For any $\boldsymbol{o} \in O$ and $a \in A$, Algorithm 1 computes $PD(\boldsymbol{o}, a)$, and its running time is $\mathcal{O}(B^2 \cdot |T|)$.*

Proof. We begin by introducing the values $PD(\boldsymbol{o}, a, b, i)$ and discussing how they can be computed. For any $b \in [0, B]$ and $i \in [1, |T|]$, let $PD(\boldsymbol{o}, a, b, i)$ denote the conditional probability that attack a is detected given that the defender has already investigated alert types $o_1, \dots, o_{i-1}$ (none if $i = 1$), has only budget b left for subsequent investigations, and has not detected the attack so far. First, we have by definition that $PD(\boldsymbol{o}, a) = PD(\boldsymbol{o}, a, B, 1)$. Second, for any b, we have that

$$PD(\boldsymbol{o}, a, b, |T|) = R_{a,o_{|T|}} \cdot F^*_{o_{|T|}}\big(\lfloor b/C_{o_{|T|}} \rfloor - 1\big) \quad (11)$$

since the defender will detect attack a using alert type $o_{|T|}$ *if and only if* there is an adversarial alert of type $o_{|T|}$, whose probability is $R_{a,o_{|T|}}$, and budget b is sufficient for investigating both the false alerts and the adversarial alert. The probability of the latter can be expressed as $F^*_{o_{|T|}}\big(\lfloor b/C_{o_{|T|}} \rfloor - 1\big)$ since budget b is sufficient for investigating $\lfloor b/C_{o_{|T|}} \rfloor$ alerts of type $o_{|T|}$, which must include both the false alerts and the adversarial alert.

Next, for any b and $i < |T|$, we have that

$$\begin{aligned} PD(\boldsymbol{o}, a, b, i) =& R_{a,o_i} \cdot F^*_{o_i}(\lfloor b/C_{o_i} \rfloor - 1) \\ &+ (1 - R_{a,o_i}) \sum_{j=0}^{\lfloor b/C_{o_i} \rfloor} \Big[(F_{o_i}(j) - F_{o_i}(j-1)) \\ &\qquad \cdot PD(\boldsymbol{o}, a, b - j \cdot C_{o_i}, i+1) \Big] \quad (12) \end{aligned}$$

since the defender will detect attack a *if and only if* it is either 1) detected using alert type o_i or 2) not detected using alert type o_i but detected using a lower-priority type. Clearly, the probability of the former can be computed the same way as for alert type $o_{|T|}$ (see Equation (11)). To compute the probability of the latter, we iterate over possible numbers j of false alerts (we do not have to consider $j > \lfloor b/C_{o_i} \rfloor$ since the defender exhausts its budget in those cases). For each number j, we multiply the probability of having that many false alerts, which is equal to $F_{o_i}(j) - F_{o_i}(j-1)$ by definition, with the probability that the defender detects the attack using the remaining budget and alert types, which is equal to $PD(\boldsymbol{o}, a, b - j \cdot C_{o_i}, i+1)$.

Now, we can prove the correctness of Algorithm 1. First, the algorithm computes $PD(\boldsymbol{o}, a, b, |T|)$ for every $b \in [0, B]$ using Equation (11). Note that this is possible since Equation (11) depends only on the input of the algorithm. Second, the algorithm iterates i backwards from $|T| - 1$ to 1, and computes $PD(\boldsymbol{o}, a, b, i)$ for $b \in [0, B]$ using Equation (12). Note that this is possible since Equation (11) depends only on the input and previously computed values $PD(\boldsymbol{o}, a, b', i+1)$, $b' < b$. Finally, the algorithm outputs $PD(\boldsymbol{o}, a, B, 1)$, which is equal to $PD(\boldsymbol{o}, a)$ by definition. Since the algorithm iterates over all combinations of $b \in [0, B]$ and $i \in [1, |T|]$, and computes each $PD(\boldsymbol{o}, a, b, i)$ using $\lfloor b/C_{o_i} \rfloor \leq B$ steps, the running time of the algorithm is clearly $\mathcal{O}(B^2 \cdot |T|)$. □

3.3 Finding an Optimal Prioritization

A natural solution approach for the problem of alert prioritization is by using multiple linear programs. Specifically, for each attack $a \in A$, we solve the following linear program for $\boldsymbol{p}$, a probability distribution over possible prioritizations $\boldsymbol{o} \in O$, which we denote by $LP(a)$:

$$\max_{\boldsymbol{p}} \sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot PD(\boldsymbol{o}, a) \tag{13a}$$

subject to

$$\forall\, a' \in A: \quad \sum_{\boldsymbol{o} \in O} p_{\boldsymbol{o}} \cdot D(\boldsymbol{o}, a') \geq \Delta\left(K_{a'}\right), \tag{13b}$$

where $D(o, a') = [(1 - PD(\boldsymbol{o}, a))G_a - (1 - PD(\boldsymbol{o}, a'))G_{a'}]$ and $\Delta\left(K_{a'}\right) = K_a - K_{a'}$. Once each $LP(a)$ is solved, we can choose the solution $\boldsymbol{p}^*$ from these which minimizes the defender's expected loss.

The key challenge for each LP is that the set of possible prioritizations O is exponential, making this intractable to represent, let alone solve. We propose to address this challenge using column generation. Specifically, we start with a small subset of prioritizations $\bar{O}$. Let $y(\bar{O})$ be the optimal dual solution of LP (13) for a fixed subset of prioritizations $\bar{O}$, with $y(\bar{O}, a')$ denoting the component of the dual solution corresponding to attack strategy $a' \in A$. In each iteration of the column generation algorithm, we aim to find a new prioritization $\boldsymbol{o} \in O$ to add to $\bar{O}$ that maximizes reduced cost $\bar{c}(\boldsymbol{o})$, where

$$\bar{c}(\boldsymbol{o}) = PD(\boldsymbol{o}, a) + \sum_{a' \in A} y(\bar{O}, a') D(\boldsymbol{o}, a'). \tag{14}$$

Once we find that $\max_{\boldsymbol{o} \in O} \bar{c}(\boldsymbol{o}) \leq 0$, the solution of the LP for a restricted set of prioritizations $\bar{O}$ generated so far is optimal. Otherwise, we repeat with $\bar{O} = \bar{O} \cup \{\boldsymbol{o}^*\}$, where $\boldsymbol{o}^* \in \operatorname{argmax}_{\boldsymbol{o}} \bar{c}(\boldsymbol{o})$.

Finally, we discuss the problem of finding prioritizations that maximize reduced cost. Since the number of possible prioritizations $|O|$ is exponential in the number of alert types $|T|$, exhaustive search is infeasible for larger problem instances. In fact, using an argument similar to the one used in the proof of Theorem 1, it can be shown that finding a cost-maximizing prioritization is an NP-hard problem in general.

Algorithm 2 Greedy Column Generation

Input: prioritization game, reduced cost function $\bar{c}$
1: $\boldsymbol{o} \leftarrow \emptyset$
2: **while** $\exists\, t \in T \setminus \boldsymbol{o}$ **do**
3: $\quad \boldsymbol{o} \leftarrow \boldsymbol{o} + \operatorname{argmax}_{t \in T \setminus \boldsymbol{o}} \bar{c}(\boldsymbol{o} + t)$
4: **end while**
5: Return $\boldsymbol{o}$

To generate near-optimal columns in practice, we propose Algorithm 2, a polynomial-time greedy algorithm. For this algorithm, we generalize our model to consider *truncated* prioritizations $\boldsymbol{o}$, which have less than $|T|$ elements. Specifically, given a prioritization $\boldsymbol{o}$ of arbitrary length (i.e., a vector of at most $|T|$ alert types), the defender will investigate alert types in $\boldsymbol{o}$ one-by-one (the same way as in the original definition), but will stop investigating after the last element of $\boldsymbol{o}$, even if the remaining budget is greater than zero.

Now, we can formulate a greedy algorithm as follows. First, begin with an empty prioritization vector $\boldsymbol{o} = \emptyset$. Next, add alert types to the end of the vector one-by-one (i.e., $\boldsymbol{o} \leftarrow \boldsymbol{o} + t$). In each iteration, choose an alert type t that leads to maximal increase in reduced cost. In the following section, we demonstrate using numerical results that this algorithm performs exceptionally well in practice.

4 Numerical Results

In this section, we numerically evaluate the proposed column-generation approach with Algorithm 2 using synthetic and real-world datasets. Our evaluation will focus on two metrics: 1) how close to optimal the strategies obtained using our approach are in terms of the defender's expected loss and 2) the running time of our approach. To compute optimal solutions, we use the linear programs LP (13) with the full sets of prioritizations O.

4.1 Synthetic Datasets

First, we evaluate our solution approach using randomly generated instances of the alert-prioritization game. We control the size of the randomly-generated instances using a size parameter N. For a given N, we generate an instance of the alert-prioritization game as follows:

- let $T = \{1, \ldots, N\}$, $A = \{1, \ldots, N\}$, and $B = 5 \cdot |T|$;
- for each $a \in A$, D_a and G_a are drawn uniformly at random from $[0.5, 1]$;
- for each $a \in A$, $K_a = 0$;
- for each $t \in T$, $C_t = 1$;

- for each $t \in T$ and $a \in A$: with probability $\frac{2}{3}$, $R_{a,t} = 0$, and with probability $\frac{1}{3}$, $R_{a,t}$ is drawn uniformly at random from $[0, 1]$;
- for each $t \in T$, F_t follows a Poisson distribution whose mean is drawn uniformly at random from $[5, 15]$, and F_t^* follows a Poisson distribution whose mean is half of the mean of F_t.

For each size N, we generated 50 random instances, and plotted the averages of the expected losses and running times over these 50 instances. To compute the probabilities $PD(\boldsymbol{o}, a)$, we used the polynomial-time Algorithm 1.

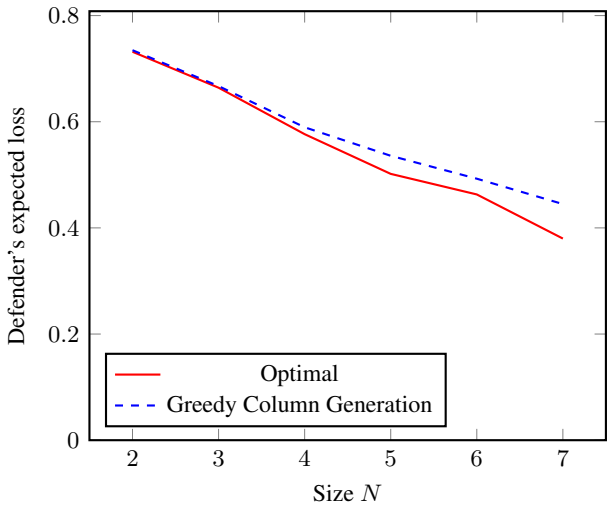

Figure 1: Defender's expected loss in the synthetic instances with optimal strategies (—) and with strategies computed using greedy column generation (Algorithm 2) (- -).

Figure 1 compares the strategies found using our greedy column-generation approach (- -) to optimal ones (—). We can see that the strategies are close to each other in terms of expected loss, even for larger problem instances.

Figure 2 compares the running time required to solve the linear programs (13) with full sets of prioritizations O (—) to that with our greedy column-generation algorithm (- -). For $N > 3$, our column-generation approach is clearly superior. For example, in the case $N = 7$, the average running time is over 9 minutes with full sets of prioritizations, while it is less than 17 seconds with Algorithm 2.

4.2 Real-World Dataset

Next, we evaluated our solution approach in a real setting. To do so, we worked with five consecutive weekdays of access logs during 2016 from the electronic medical record (EMR) system, StarPanel (Giuse, Williams, and Giuse 2010), in place at Vanderbilt University Medical Center – a system that is well ingrained in clinical operations with over 25 years of continuous use. This study was approved by the medical center's institutional review board.

In preparation for this study, we integrated the EMR system with human-resources data to document i) which medical department each system user was affiliated with, ii)

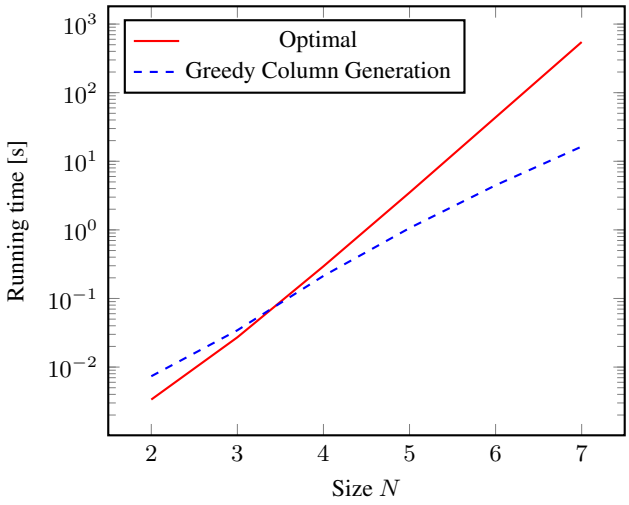

Figure 2: Running time of the linear programs with full sets of prioritizations (—) and with greedy column generation (Algorithm 2) (- -). Please note the logarithmic scale on the vertical axis.

which patients were also employees, and iii) home residential information for each system user. The resulting data was then subject to an explanation-based auditing system (EBAS) (Fabbri and LeFevre 2011; 2013) to annotate the access logs for six types of alerts: EMR user and patient 1) have the same surname, 2) are coworkers in the same department of the medical center, 3) have residential addresses within 0.25 miles, 4) have department tags related to "Primary Care Physicians," 5) have department tags related to "Pediatrics Housestaff," and 6) have department tags related to "Internal Medicine."

During this week, there were 8,481,767 accesses made by 14,531 users to 161,426 patient records, leading to a total of 863,989 alerts. To ensure that finding optimal strategies is numerically tractable, we restricted our analysis to three departments and a random sample of 12 patients. The complete description of the dataset used in our experiments can be found in the Supplementary Material.

First, we estimated the distribution of the number of false alerts for each alert type. Since most of the alerts on each workday are generated in the time interval between 8am and 4pm, the detection problem is by far the most challenging during these time intervals. Consequently, we focused our analysis on the numbers of false alert generated between 8am and 4pm. Table 2 shows the actual numbers for each workday and alert type. Based on these numbers, we approximated the actual probability distributions with Poisson distributions, which we then used as F_t in our model.

Second, we selected a random sample of 12 patients, and determined which alert types may be generated by accessing their records. Table 3 shows the occurrence of alerts for each patient (i.e., attack in our model) and alert type. We then used these values as $R_{a,t}$ in our solution approach. Note that we limited the size of the problem to ensure that we can compute the optimal strategies, which we use as baselines to

Table 2: Numbers of False Alerts

	Alert type 1	Alert type 2	Alert type 3	Alert type 4	Alert type 5	Alert type 6	Sum
Day 1	2467	2544	2671	4340	3451	6152	21625
Day 2	2434	2072	2446	5002	4277	6304	22535
Day 3	2495	2538	2418	4192	3491	5461	20595
Day 4	3175	2842	2366	3745	3181	4920	20229
Day 5	2923	2597	2641	3064	2487	4280	17992
Sum	13494	12593	12542	20343	16887	27117	102976
Average	2698.8	2518.6	2508.4	4068.6	3377.4	5423.4	

Table 3: Alert Types for Each Patient

	Alert type 1	Alert type 2	Alert type 3	Alert type 4	Alert type 5	Alert type 6
Patient 1	0	0	1	0	0	0
Patient 2	1	1	0	0	0	0
Patient 3	0	0	0	1	0	0
Patient 4	0	0	0	1	1	1
Patient 5	1	1	0	0	0	0
Patient 6	1	1	0	0	0	0
Patient 7	0	0	0	0	1	1
Patient 8	0	0	0	1	1	1
Patient 9	0	0	0	0	1	1
Patient 10	1	1	0	0	0	0
Patient 11	0	0	0	1	1	1
Patient 12	1	1	0	0	0	0

evaluate our proposed algorithm.

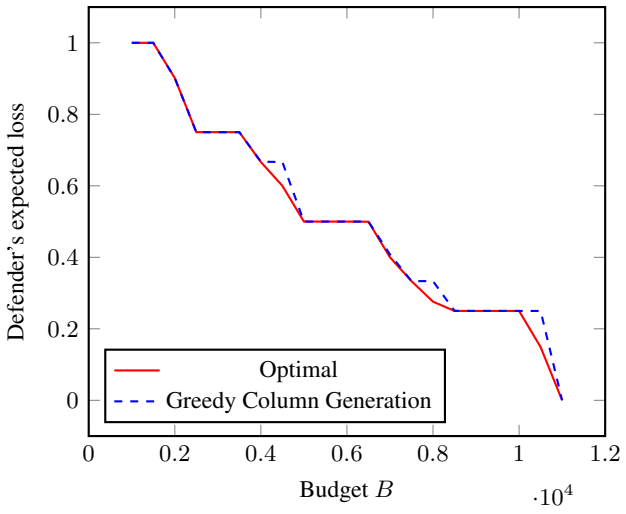

Figure 3: Defender's expected loss in the real-world dataset with optimal strategies (—) and with strategies computed using greedy column generation (Algorithm 2) (- -).

Figure 3 compares the strategies found using our greedy column-generation approach (- -) to optimal ones (—) for various budget values B. We can see that our algorithm performs exceptionally well as the defender's expected loss is very close to optimal in all cases.

5 Conclusion

The prioritization of alerts is of crucial importance to the effectiveness of intrusion and misuse detection. Even though considerable research has been performed in this area, there has been no investigation into strategic adversaries to the best of our knowledge. In this work, we modeled strategic adversaries, who may adapt to the prioritization chosen the defender, as a Stackelberg security game. We showed that finding an optimal prioritization against strategic adversaries is a computationally hard problem, and we have proposed a column-generation based approach, as well as a greedy column-generation algorithm, for solving this problem in practice. Using numerical results, we have demonstrated that our solution approach performs well in practice with respect to both expected losses and running time.

There are multiple natural future research directions. In this paper, we have shown using numerical results that Algorithm 2 performs very well in practice. However, it remains an open question if this algorithm – or any other polynomial-time algorithm – can achieve a constant approximation ratio for reduced cost. As another direction, the effectiveness of alert prioritization could be further increased by considering multiple adversary types, with different attack costs and gains. This extension could be modeled most naturally as a Bayesian Stackelberg game.

Acknowledgment

This research was supported, in part, by grants from the National Science Foundation (CNS-1238959, CNS-1640624, CNS-1526014, IIS-1526860), the Army Research

Office (W911NF1610069), the Office of Naval Research (N00014-15-1-2621), and the National Institutes of Health (R01LM010207).

References

Blocki, J.; Christin, N.; Datta, A.; Procaccia, A. D.; and Sinha, A. 2013. Audit games. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 41–47. AAAI Press.

Blocki, J.; Christin, N.; Datta, A.; Procaccia, A. D.; and Sinha, A. 2015. Audit games with multiple defender resources. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 791–797. AAAI Press.

CSO Magazine; U.S. Secret Service; CERT Division of the Software Engineering Institute; and Price Waterhouse Cooper. 2014. 2014 U.S. state of cybercrime survey.

Fabbri, D., and LeFevre, K. 2011. Explanation-based auditing. *Proceedings of the VLDB Endowment* 5(1):1–12.

Fabbri, D., and LeFevre, K. 2013. Explaining accesses to electronic medical records using diagnosis information. *Journal of the American Medical Informatics Association* 20(1):52–60.

Farahmand, F., and Spafford, E. H. 2013. Understanding insiders: An analysis of risk-taking behavior. *Information Systems Frontiers* 15(1):5–15.

Giuse, N. B.; Williams, A. M.; and Giuse, D. A. 2010. Integrating best evidence into patient care: A process facilitated by a seamless integration with informatics tools. *Journal of the Medical Library Association* 98(3):220.

Hiltzik, M. 2015. Anthem is warning consumers about its huge data breach. Los Angeles Times, `http://www.latimes.com/business/la-fi-mh-anthem-is-warning-consumers-20150306-column.html`. Accessed: September 14th, 2016.

Hubballi, N., and Suryanarayanan, V. 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications* 49:1–17.

Kerckhoffs, A. 1883. La cryptographie militaire. *Journal des Sciences Militaires* 9:5–83.

Korzhyk, D.; Yin, Z.; Kiekintveld, C.; Conitzer, V.; and Tambe, M. 2011. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research* 41(2):297–327.

PricewaterhouseCoopers. 2016. Global state of information security survey. `http://www.pwc.com/gx/en/issues/cyber-security/information-security-survey.html`. Accessed: September 14th, 2016.

Randazzo, M.; Keeney, M.; Kowalski, E.; Cappelli, D.; and Moore, A. 2005. Insider threat study: Illicit cyber activity in the banking and finance sector. Technical Report CMU/SEI-2004-TR-021, Carnegie Mellon University.

Rostad, L., and Edsberg, O. 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.

Salah, S.; Maciá-Fernández, G.; and Díaz-Verdejo, J. E. 2013. A model-based survey of alert correlation techniques. *Computer Networks* 57(5):1289–1317.

Silowash, G.; Cappelli, D.; Moore, A. P.; Trzeciak, R. F.; Shimeall, T. J.; and Flynn, L. 2012. Common sense guide to mitigating insider threats, 4th edition. Technical Report CMU/SEI-2012-TR-012, Software Engineering Institute, Carnegie Mellon University.