# Don't Bury your Head in Warnings: A Game-Theoretic Approach for Intelligent Allocation of Cyber-security Alerts

**Aaron Schlenker**[1]**, Haifeng Xu**[1]**, Mina Guirguis**[2]**, Chris Kiekintveld**[3]**, Arunesh Sinha**[4]
**Milind Tambe**[1]**, Solomon Sonya**[5]**, Darryl Balderas**[2]**, Noah Dunstatter**[2]

[1] University of Southern California, Los Angeles, CA 90089
[2] Texas State University, San Marcos, Texas 78666
[3] University of Texas at El Paso, El Paso, TX 79968
[4] University of Michigan, Ann Arbor, MI 48109
[5] United States Air Force Academy, USA Academy, CO 80840

## Abstract

In recent years, there have been a number of successful cyber attacks on enterprise networks by malicious actors. These attacks generate alerts which must be investigated by cyber analysts to determine if they are an attack. Unfortunately, there are magnitude more alerts than cyber analysts - a trend expected to continue into the future creating a need to find optimal assignments of the incoming alerts to analysts in the presence of a strategic adversary. We address this challenge with the four following contributions: (1) a *cyber allocation game* (CAG) model for the cyber network protection domain, (2) an NP-hardness proof for computing the optimal strategy for the defender, (3) techniques to find the optimal allocation of experts to alerts in CAG in the general case and key special cases, and (4) heuristics to achieve significant scale-up in CAGs with minimal loss in solution quality.

## 1 Introduction

Automated intrusion detection and prevention systems (IDPS) and security information and event management tools (SIEM) are important for computer network security. The alerts generated by these systems must be investigated by human cybersecurity analysts to assess whether they were generated by malicious activity, and if so, how to respond. Unfortunately, these automated systems are notorious for generating high rates of false positives [Spathoulas and Katsikas, 2013]. Expert analysts are in short supply, so organizations face a key challenge in managing the enormous volume of alerts they receive using the limited time of analysts. Failing to solve this problem can render the entire system insecure, e.g., in the 2013 attack on Target, IDPS raised alarms, but they were missed in the deluge of alerts [Riley *et al.*, 2014].

There are many approaches for mitigating this problem by reducing the number of alerts. IDPS can be carefully configured, alert thresholds can be tuned, and the classification methods underlying the detections can be improved [Sommer and Paxson, 2010; Barbara and Jajodia, 2002; Laszka *et al.*, 2016]. Other techniques include aggregating alerts [Zimmerman, 2014], and visualizing alerts for faster analysis [Patton

*et al.*, 2011]. Even when using all of these techniques, there are still too many alerts for the analysts to investigate all of them in depth. Our work focuses on the remaining problem of assigning limited analysts to investigate alerts *after* automated pre-processing methods have been applied.

The typical approach to managing alerts is either ad-hoc or first investigates alerts with the highest priority (e.g., risk). However, this fails to account for the adversarial nature of the cyber security setting. An attacker who can learn about a predictable alert management policy can exploit this knowledge to launch a successful attack. For example, if we had a policy that only inspects alerts from high valued assets from our network, an attacker who can learn this evades detection indefinitely by only attacking lower valued assets.

To address this shortcoming of the previous method, our first contribution is a *Cyber-alert Allocation Game* (CAG), a game-theoretic model for optimizing the assignment of cyber alerts to a limited group of cyber analysts. Using game theory allows us to explicitly model the strategies an attacker could take to avoid detection. By following a randomized, unpredictable assignment strategy the defender can improve the effectiveness of alert assignments against strategic attackers. Our model considers the characteristics of the alerts (e.g., criticality of origin system), as well as the capabilities of the analysts in formulating the optimal policy for the defender.

Our second contribution in this paper is to show that finding the optimal strategy for a CAG is NP-hard, posing a major computational challenge. Third, we present an algorithm for finding optimal, implementable CAG policies. Fourth, we devise novel heuristics to solve large CAGs, and we provide empirical evaluation of our algorithms and model.

## 2 Related Work

There is a large body of work on automated detection of cyber attacks using machine learning (e.g., [Hu *et al.*, 2003; Hofmeyr and Forrest, 1999; Wu *et al.*, 2015]). However, these methods have significant detection error and suffer from generating too many alerts [Sommer and Paxson, 2010]. In modern security operations a team of human cyber analysts work to investigate the alerts generated by automated detectors [DAmico and Whitley, 2008]. A recent line of work [Ganesan *et al.*, 2016] has used decision theory to optimize the scheduling of cyber-security analysts for screening

over multiple time periods, but this approach does not consider the response of a strategic attacker.

Our approach draws on the principles and modeling techniques of a large body of work that applies game theory to security problems [Tambe, 2011]. The existing work on security games focuses heavily on applications to physical security (e.g., patrolling), with some exceptions (e.g., [Laszka *et al.*, 2016; Durkota *et al.*, 2015]). However, CAG significantly differs from traditional security games [Tambe, 2011; Jain *et al.*, 2010b] due to the absence of an explicit set of targets, a large number of benign alerts and varying time requirements for inspections. A model that is closely related to ours is the Threat Screening Game (TSG) introduced in [Brown *et al.*, 2016] for screening passengers at airports. There are some crucial differences with the cybersecurity domain: (1) Screening in airports is a quick scan of a passenger; in CAGs, investigating a threat may take varying amounts of time leading to a different "non-implementability" [Korzhyk *et al.*, 2010] issue for CAG as compared to TSG and other security games which require novel techniques to resolve, (2) CAG does not consider teams of resources, and (3) in CAGs attacks result in a probability distribution over a set of alerts.

## 3 Motivating Domain

While many organizations face the challenge of cyber alert allocation, we highlight a scenario developed in consultation with experts at the United States Air Force (USAF). The USAF relies on extensive global cyber systems to support its missions, which are monitored by IDPS to prevent attacks on the network by intelligent adversaries. The Air Force Cyber defense unit (AFCYBER) is responsible for investigating and resolving alerts generated by these IDPS [afc, 2017]. Due to the global scale of USAF computer systems, millions of alerts are generated every day, associated with different types of events. Prescreening of the alerts eliminates a large fraction of insignificant events, but thousands remain to be investigated. Any of these remaining alerts could indicate a malicious attack, but a large fraction are false positives.

Two primary features are used to prioritize the most critical alerts to investigate. First, each alert has a risk classification (e.g., high, medium, low) based on the type of event detected by the IDPS. Second, each alert has an origin location within the global network (e.g., a specific host, system); some locations (e.g., headquarters) are more critical to operations.

The AFCYBER has a limited number of Incident Response Team (IRT) cyber analysts who investigate significant alerts after prescreening [aft, 2016]. Each analyst has different areas of expertise, and may therefore be more effective and/or faster at investigating certain types of incidents. The USAF also must protect against an adaptive adversary who can observe strategies through beaconing and other techniques. The problem AFCYBER faces is an excellent example of our central analyst assignment problem in the real world.

## 4 Cyber-alert Allocation Games

We model the *Cyber-alert Allocation Game* (CAG) as a (zero-sum) Stackelberg game played between the defender (e.g., AFCYBER) and an adversary (e.g., hacker). The defender

commits to a mixed strategy to assign alerts to cyber analysts. We make the worst-case assumption that the attacker moves with complete knowledge of the defender's strategy and plays a best-response attack strategy [Kiekintveld *et al.*, 2009]. However, in a zero-sum game the optimal strategy for the defender is the same as the Nash equilibrium (i.e., when the attacker moves simultaneously) [Yin *et al.*, 2010], so the order of the moves is *not* consequential in the model.

*Systems and Alerts*: The defender responds to alerts originating from a set of systems $k \in K$. A "system" in our model could represent any level of abstraction, ranging from a specific server to a complete network.IDPS for each system generate alerts of different types, $a \in A$. The alert types correspond to levels of severity (e.g., high, medium, and low), reflecting the likelihood of a malicious event. We represent the combination of the alert type and the origin system as an alert category, $c \in C$, where $c = (k, a)$. The alerts in a given category are not differentiable, so the defender must investigate all alerts within a category with the same probability. The total number of alerts for a given category $c$ is denoted by $N_c$. We assume that the both the defender and attacker know the typical value of $N_c$ from historical averages (similar to [Ganesan *et al.*, 2016]).

*Attack Methodologies*: Attackers can choose from many attack methodologies. These fall into high-level categories such as denial of service attacks, malware, web exploitation, or social engineering. We represent these broad classes of attacks as attack methods $m \in M$. For every attack method there is a corresponding probability distribution $\beta_a^m$ which represents the probability that the IDPS generates an alert of type $a$ for an attack method $m$. For example, if the attacker chooses $m = DoS$ the corresponding alert probabilities could be $\beta_{High}^{DoS} = .8$, $\beta_{Medium}^{DoS} = .15$ and $\beta_{Low}^{DoS} = .05$.

*Cybersecurity Analysts*: Cybersecurity analysts $R$ are assigned to investigate alerts. The time required for an analyst to resolve an alert type $a$ varies, and is represented by $T_a^r$. Intuitively, $T_a^r$ represents the portion of a time period that an analyst needs to resolve an alert of type $a$. A time period may be a shift, an hour or other fixed scheduling period. For example, if an analyst needs half a time period to resolve $a$, then $T_a^r = 0.5$. In our model: $T_a^r \leq 1, \forall\, a \in A$, i.e., an analyst can address multiple alerts within a time period. In addition to $T_a^r$, we allow modeling of the effectiveness of an analyst against an attack method, representing her expertise, via a parameter $E_m^r$.

**Defender Strategies:** A pure strategy $P$ for the defender is a non-negative matrix of *integers* of size $|C| \times |R|$. Each $c,r$ entry is the number of alerts in category $c$ assigned to be investigated by cyber analyst $r$, denoted by $P_{c,r}$. The set of all pure strategies $\hat{P}$ is all allocations that satisfy the following constraints; $C_a$ denotes all categories with the alert type $a$:

$$\sum_{a \in A} \sum_{c \in C_a} T_a^r P_{c,r} \leq 1 \qquad \forall r \in R \qquad (1)$$

$$\sum_{r \in R} P_{c,r} \leq N_c \qquad \forall c \in C \qquad (2)$$

$$P_{c,r} \text{ are integers} \qquad (3)$$

|       | $r_1$ | $r_2$ |       | $r_1$ | $r_2$ |
|-------|-------|-------|-------|-------|-------|
| $c_1$ | 0     | 2     | $c_1$ | 0.5   | 2.5   |
| $c_2$ | 1     | 1     | $c_2$ | 0.8   | 0     |
| $c_3$ | 0     | 0     | $c_3$ | 0     | 0     |
| $c_4$ | 1     | 0     | $c_4$ | 0.2   | 0     |

(a) Pure Strategy     (b) Marginal Strategy

Figure 1: CAG Strategies for the defender.

Inequality (1) ensures that each analyst is assigned a valid number of alerts, while inequality 2 ensures we do not assign more alerts than the total in a category.

**Example CAG.** Consider a CAG with two systems $K = \{k_1, k_2\}$, two alert levels $A = \{a_1, a_2\}$, and two analysts $r = \{r_1, r_2\}$. There are four alert categories $C = \{c_1, c_2, c_3, c_4\}$, where $c_1 = (k_1, a_1)$, $c_2 = (k_1, a_2)$, $c_3 = (k_2, a_1)$ and $c_4 = (k_2, a_2)$. For the alert categories we have $N_{c_1} = 3$, $N_{c_2} = 2$, $N_{c_3} = 0$, and $N_{c_4} = 1$. For $r_1$, assume $T_{a_1}^{r_1} = 1$ and $T_{a_2}^{r_1} = 0.5$; For $r_2$, assume $T_{a_1}^{r_2} = 0.4$ and $T_{a_2}^{r_2} = 0.2$. The analyst capacity constraint (Inequality (1)) for $r_1$ is instantiated as follows (the other columns are similar):

$$P_{c_1,r_1} + 0.5 \cdot P_{c_2,r_1} + P_{c_3,r_1} + 0.5 \cdot P_{c_4,r_1} \le 1$$

For $c_1$ the alert capacity constraint (Inequality (2)) we have (the other rows are similar):

$$P_{c_1,r_1} + P_{c_1,r_2} \le 3$$

An example of a pure strategy $P$ is given in Figure 1(a). The dashed boxes in Figure 1(a) represent the set of variables in the analyst capacity constraints, i.e. constraints of type (1). We show an example marginal strategy in Figure 1(b). This drops constraint (3), but satisfies constraints (1) and (2).

We define a mixed strategy **q** over pure strategies $P \in \hat{P}$ ($\sum_{P \in \hat{P}} q_P = 1, 0 \le q_P \le 1$). From the mixed strategy we can calculate the marginal (expected) number of alerts of category $c$ assigned to each analyst $r$, denoted by $n_{c,r} = \sum_P q_P P_{c,r}$. The *marginal* allocation is denoted by **n** with component $n_{c,r}$ representing the expected number of alerts in category $c$ assigned to analyst $r$. The adversary plays a best response to the defender's marginal strategy **n** which amounts to choosing a system $k$ to attack and an attack method $m$.

**Utilities** Since the alerts in a category are indistinguishable they are all investigated with the same probability $n_{c,r}/N_c$, which is the probability that an alert in category $c$ is investigated by analyst $r$. The probability of detecting an attack of type $m$ that results in an alert of type $c$ is calculated as: $x_{c,m} = \sum_{r \in R} E_m^r n_{c,r}/N_c$. The payoffs for the defender depend on the system $k$ that is attacked, the attack method $m$, and if the adversary is detected (or undetected) during investigation. This is denoted by $U_{\delta,c}^d$ and $U_{\delta,c}^u$, respectively, where $c$ refers to the category $(k, a)$ and $\delta$ is the defender. We formulate a CAG as a zero-sum game, hence the payoffs for the adversary ($\theta$) are $U_{\theta,c}^d = -U_{\delta,c}^d$ and $U_{\theta,c}^u = -U_{\delta,c}^u$. If the adversary chooses $k$, $m$, and given $\beta_a^m$, the defender's utility is:

$$U_s = \sum_{a \in A} \beta_a^m [x_{c,m} * U_{\delta,c}^d + (1 - x_{c,m}) U_{\delta,c}^u] \quad (4)$$

## 5 Defender's Optimal Strategy

We start with a linear program, denoted as $MixedStrategyLP$, that computes the defender's optimal mixed strategy (as the maximin strategy):

$$\max_{\mathbf{n,v}} \quad v \quad (5)$$

$$s.t. \quad v \le U_s \quad \forall k, m \quad (6)$$

$$x_{c,m} = \sum_{r \in R} E_m^r \frac{n_{c,r}}{N_c} \quad \forall c, m \quad (7)$$

$$n_{c,r} = \sum_{P \in \hat{P}} q_P P_{c,r} \quad \forall c, r \quad (8)$$

$$\sum_{P \in \hat{P}} q_P = 1, q_P \ge 0 \quad (9)$$

This LP requires *exponentially* many pure strategies $P \in \hat{P}$. The objective function in Equation 5 maximizes the defender's utility, $v$. Equation 6, which uses Equation (4), ensures the adversary selects a best response over all $m \in M$ and $k \in K$. Equation 7 calculates the detection probabilities **x** from the marginal strategy **n**, which is computed by Equation 8. Equation 9 ensures the mixed strategy is valid.

Computing the maximin mixed strategy for the defender was shown to be NP-hard in the case of TSGs [Brown *et al.*, 2016]. The computational hardness arises from the underlying team formation of applying a group of screening resources to screen incoming passengers. However, in CAGs we do not have teams of analysts, we only need to assign the alerts to individual analysts. Thus, one might hope that this could simplify the problem and admit a polynomial time algorithm. Unfortunately, this turns out not to be the case. Specifically, we show in Theorem 1 that the problem is still NP-hard, where the hardness arises from a different domain feature, i.e., the time values, $T_a^r$, for the analysts. All proofs can be found in the on-line appendix[1].

**Theorem 1.** *Computing the defender maximin strategy is weakly NP-hard when there is only one resource, and is strongly NP-hard with multiple resources.*

In some special cases, it is possible to compute the optimal marginal strategy in polynomial time. Specifically, if all $T_a^r$ for a given analyst $r$ are identical $\forall a \in A$, then the optimal marginal strategy can be found with an LP which is stated in Proposition 1. This result is discussed further in Section 6.

**Proposition 1.** *When $T_{a_i}^r = T_{a_j}^r \; \forall a_i, a_j \in A$ for each resource, then there is a polynomial time algorithm for computing the maximin strategy.*

### 5.1 Defender's Optimal Marginal Strategy

In the security games literature, two approaches are commonly used to handle scale-up: marginal strategies [Kiekintveld *et al.*, 2009; Letchford and Conitzer, 2013] and column generation [Jain *et al.*, 2010a]. We adopt a marginal strategy based approach which finds the defender's marginal strategy **n** and does not need to explicitly enumerate the exponential number of pure strategies. We now introduce a relaxed version of LP (5)~(9) in LP (10)~(14). LP (10)~(14)

---

[1] https://teamcore.usc.edu/papers/2017/ars17.Appendix.pdf

is similar to LP (5)$\sim$(9) except that we replace equations (8) and (9) with equations (13) and (14) to model the relaxed marginal space. Recall that marginal strategies satisfy constraints (1)$\sim$(2) (which lead to Equations 13 and 14) but drop constraint (3). The optimal marginal strategy $\mathbf{n}$ for the defender can then be found by solving the following $MarginalStrategyLP$ (MSLP):

$$\max_{\mathbf{n},\mathbf{v}} \quad v \tag{10}$$

$$v \leq U_s \qquad\qquad \forall k, m \tag{11}$$

$$x_{c,m} = \sum_{r \in R} E_m^r \frac{n_{c,r}}{N_c} \qquad \forall c, m \tag{12}$$

$$\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1 \qquad \forall r \tag{13}$$

$$\sum_{r \in R} n_{c,r} \leq N_c, \quad n_{c,r} \geq 0 \qquad \forall r, c \tag{14}$$

Though $MarginalStrategyLP$ computes the optimal marginal strategy $\mathbf{n}$, it may not correspond to any valid mixed strategy $\mathbf{q}$, i.e., there may not exist a corresponding mixed strategy $\mathbf{q}$ such that $\mathbf{n} = \sum_{P \in \hat{P}} q_P P$, $\sum_{p \in \hat{P}} q_P = 1$. Marginal strategies of this type are called *non-implementable*. However, when $T_a^r$ have a particular structure, we can show the marginal strategy returned is the optimal for the defender. The intuition is that when $T_a^r = \frac{1}{w_a}$ where $w_a \in \mathbb{Z}^+$, the extreme points of the marginal polytope are all integer. In these cases, we can efficiently compute the defender's optimal implementable marginal strategy using the MSLP.

**Theorem 2.** *For any feasible marginal strategy $\mathbf{n}$ to MSLP, there is a corresponding mixed strategy $\mathbf{q}$ that implements $\mathbf{n}$ whenever $T_a^r = \frac{1}{w_a}$ where $w_a \in \mathbb{Z}^+$, $\forall r \in R, \forall a \in A$ and $N_c \geq \sum_{r \in R} \frac{1}{T_a^r}$, $\forall c \in C$ for a given CAG.*

## 6 CAG Algorithmic Approach

The problem of non-implementability of marginals in security games has been studied in previous research [Letchford and Conitzer, 2013; Brown *et al.*, 2016], but the non-implementability arose because of spatio-temporal resource constraints and constraints from combining resources into teams. For our problem, non-implementability arises from the presence of the $T_a^r$ coefficients (we discuss an example later). In this section, we present an algorithm that takes the initial constraints on a CAG and converts them to ensure the implementability of the marginal strategy. To that end, [Budish *et al.*, 2013] presents a useful approach, as they define a special condition on the constraints on the marginals called a *bihierarchy*. A bihierarchy captures a sufficient condition needed to guarantee the implementability of the defender's marginal strategy $\mathbf{n}$. Unfortunately, constraints on CAGs rarely satisfy the conditions for a bihierarchy and must be converted to achieve the bihierarchy condition.

**Definitions and Notation** The marginal assignments $\mathbf{n}$ for the defender form a $|C| \times |R|$ matrix. The assignment constraints on the defender's marginal strategy, namely Equations 13 and 14, are a summation of $n_{c,r}$ over a set $S \subset |C| \times |R|$ with an integral upper bound. For example, based on Equation 14, $\{\{c_1, r_1\}, \{c_1, r_2\}\}$ forms a constraint subset for the example CAG. The collection of all such $S$ form a *constraint structure $H$* when all coefficients in the constraints are *unitary*, as they are in Equation 14.
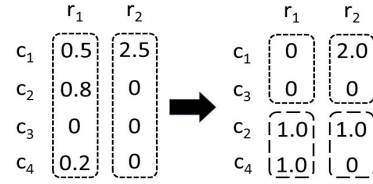


Figure 2: Conversion of Column Constraints on CAG

A marginal strategy $\mathbf{n}$ is said to be *implementable* with respect to $H$ if there exists a distribution (a.k.a., mixed strategy) $\mathbf{q}$ such that $\mathbf{n} = \sum_{P \in \hat{P}} q_P P$. A constraint structure $H$ is said to be a *hierarchy* if, for any two constraint sets in $H$, we have that either one is a subset of the other or they are disjoint. More concretely, we have the following: $\forall S_1, S_2 \in H$, $S_1 \subset S_2$, $S_2 \subset S_1$ or $S_1 \cap S_2 = \emptyset$. $H$ is said to be a *bihierarchy* if there exists hierarchies $H_1$ and $H_2$, such that $H = H_1 \cup H_2$ and $H_1 \cap H_2 = \emptyset$.

For any CAG, the row constraints $\sum_{r \in R} n_{c,r} \leq N_c$ form a hierarchy $H_1$. However, the column constraints, one for each resource $r \in R$, do not form a hierarchy: $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1$. As mentioned earlier, the culprit lies in the $T_a^r$ coefficients, as they can be non-unitary, and to achieve a hierarchy $H_2$ on the column constraints, and thus give us a bihierarchy, all $T_a^r$ coefficients must be removed.

**Constraint Conversion** The $T_a^r$ coefficients admits possibly non-implementable marginal strategies to be returned. For instance, in Figure 1(b) the marginal strategy is non-implementable, because it is impossible to get $n_{c_1,r_2} = 2.5$ by mixing pure assignments. This is because constraints (1) and (3), force the relevant pure strategy $P_{c_1,r_2} \leq \lfloor 1/0.4 \rfloor = 2$. We aim to convert the column constraints, namely: $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1$ into a hierarchy by removing the $T_a^r$ coefficients. The conversion can be completed by grouping together all $n_{c,r}$ which have the same $T_a^r$ and introducing a new constraint on these sets of $n_{c,r}$. Specifically, each column constraint (equation 13) is replaced with $|A|$ constraints:

$$\sum_{c \in C_a} n_{c,r} \leq L_r^{C_a} \tag{15}$$

This conversion must be done for all analysts $r \in R$ for the column constraints to form a hierarchy $H_2$. $L_r^{C_a}$ gives an upper bound on the number of alerts of type $a$ that an analyst can solve. The choices of $L_r^{C_a}$ must satisfy the original capacity constraint, namely: $\sum_{a \in A} T_a^r L_r^{C_a} \leq 1$ and $L_r^{C_a} \in \mathbb{Z}$.

**Conversion Example** We refer to the example CAG where the marginal strategy is given in Figure 2. We must convert the column constraints to a hierarchy. We highlight how this conversion is done for $r_1$ (as $r_2$ is converted in the same manner). Initially, for $r_1$ we have the following constraint:

$$T_{a_1}^{r_1} n_{c_1,r_1} + T_{a_2}^{r_1} n_{c_2,r_1} + T_{a_1}^{r_1} n_{c_3,r_1} + T_{a_2}^{r_1} n_{c_4,r_1} \leq 1$$

We remove the $T_a^r$ coefficients by grouping together all $n_{c,r}$ which share $T_a^r$ and introducing two new constraints like (15). This leads to two new constraints:

$$n_{c_1,r_1} + n_{c_3,r_1} \leq L_{r_1}^{C_{a_1}} \qquad n_{c_2,r_1} + n_{c_4,r_1} \leq L_{r_1}^{C_{a_2}}$$

These new constraints are shown for $r_1$ in Figure 2 on the right of the arrow. Next, we must set the $L_r^{C_a}$ variables. One possible combination is $H_2 = \{n_{c_1,r_1} + n_{c_3,r_1} \leq 0, n_{c_2,r_1} + n_{c_4,r_1} \leq 2\}$ ($H_2$ also includes constraints on $r_2$ which are not shown). This satisfies the original the original analyst capacity constraints as: $L_{r_1}^{C_{a_1}} + 0.5 \cdot L_{r_1}^{C_{a_2}} \leq 1$. However, there is another choice for $L_r^{C_a}$, $H_2 = \{n_{c_1,r_1} + n_{c_3,r_1} \leq 1, n_{c_2,r_1} + n_{c_4,r_1} \leq 0\}$. Given either of the two hierarchies $H_2$, we now have a bihierarchy. The original marginals shown in Figure 2 do not satisfy these new constraints; but solving the MSLP with these additional constraints in $H_2$ is guaranteed to give an implementable marginal.

## 6.1 Branch-and-Bound Search

So far, we have seen that a marginal strategy $\mathbf{n}$ for a CAG output from the MSLP may be non-implementable. Our goal is to ensure that the marginal strategy output by MSLP is implementable by adding new column constraints, i.e., by realizing a bihierarchy. The addition of new constraints as outlined above gives us a bihierarchy, but there are multiple ways to set the values of $L_r^{C_a}$ variables (as shown in the above example), creating a choice of what bihierarchy to create. Indeed, we may need to search through the combinatorially many ways to convert the constraints of CAG to a bihierarchy. Previous work [Brown *et al.*, 2016] proposed the Marginal Guided Algorithm (MGA) for creating bihierarchies, but MGA does not apply to CAGs as it does not deal with the non-unitary coefficients present in CAGs.

Here we propose a novel branch-and-bound search: out of the set of constraints that could be added to MSLP, find the best that would give the defender the optimal utility $v^*$. At the root node, we have the original constraints (13) and (14); running MSLP potentially yields a non-implementable marginal strategy $\mathbf{n}$. Then we branch from this root, where at each level in the tree, we add new constraints for an analyst $r$, and the children are expanded with the following rules:

1. Substitute $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1$ with $|A|$ constraints: $\sum_{c \in C_a} n_{c,r} \leq L_r^{C_a}$ for all $a \in A$. The $|A|$ new constraints form a set $H_2(r)$. A branch is created for all combinations of $L_r^{C_a}$ which satisfy $\sum_{a \in A} T_a^r * L_r^{C_a} \leq 1$.

2. Solve the $MarginalStrategyLP$ at each node with the modified constraints.

Thus, at each level of the tree, we have substituted the capacity constraint of some analysts, and for these, we have constraints of type (15), but for others, we still have constraint (13). This set of constraints does not form a hierarchy $H_2$ as $T_a^r$ coefficients are present in some analyst constraints. Still, at an intermediate node we have upper bound on the defender's utility $v$ which is stated in Proposition 2, as each conversion from (13) to (15) introduces new constraints on the defender's strategy space.

**Proposition 2.** *Each intermediate node in the tree gives an upper bound on the defender's utility $v$ for all subsequent conversions for the remaining analyst capacity constraints.*

A leaf in the search tree has column constraints only of the form: $\sum_{a \in A} n_{c,r} \leq L_r^{C_a}$. Hence, they form a hierarchy $H_2$ as all $n_{c,r}$ have unitary coefficients and an integer upper bound. At a leaf, we can then solve the MSLP with the resulting bihierarchical constraints to find a lower bound on the defender's utility $v$. Combining this with Proposition 2 gives the components needed for a branch-and-bound search tree which returns the optimal bihierarchy for the defender.

**Heuristic Search** The full branch-and-bound procedure struggles with large CAG. To find good bihierarchies, we can take advantage of the optimal marginal strategy $\mathbf{n}^*$ returned from $MSLP$ at an intermediate node to reduce the amount of branching done. The intuition for this strategy, is that the optimal bihierarchy either contains, or is near, $\mathbf{n}^*$. For example, in the conversion done in Figure 2, we could set the $L_r^{C_a}$ values close to $\mathbf{n}$. We set $L_{r_2}^{C_{a_1}} = \lfloor 1/.4 \rfloor = 2$, while the leftover capacity for $r_2$ is used to set $L_{r_2}^{C_{a_2}} = 1$. $L_{r_2}^{C_{a_1}}$ could be set to another value, but our choice must stay close to $\mathbf{n}^*$.

For the heuristic search, we use the following rules to expand child nodes which must set the $L_r^{C_a}$ values for an analyst $r$: (1) $L_r^{C_a} = \lceil n_{C_a,r} \rceil$, (2) $L_r^{C_a} = \lfloor n_{C_a,r} \rfloor$ or (3) $L_r^{C_a} = \lfloor \frac{1 - \sum_{a \in A} T_a^r * L_r^{C_a}}{T_a^r} \rfloor$, where $n_{C_a,r} = \sum_{c \in C_a} n_{c,r}$. The third rule is used whenever an $L_r^{C_a}$ value cannot be set to the roof or floor of $\mathbf{n}^*$, and is set to be the max value given the leftover analyst capacity. These choices are done in an attempt to capture the optimal marginal strategy $\mathbf{n}^*$. The set of all valid combinations of the $L_r^{C_a}$ values using the above rules which satisfy $\sum_{a \in A} T_a^r L_r^{C_a} \leq 1$ constitute the search space at each intermediate node. These rules then significantly reduce the branching at intermediate nodes in the search tree.

**Convex Hull Extension** The above searches return a set of good bihierarchies for obtaining a high value of $v^*$ for the defender when solving MSLP, as each leaf contains a bihierarchy $H_i$. Each bihierarchy $H_i$ contains a portion of the defender's mixed strategy space (due to new constraints). Thus, taking a convex hull over these bihierarchies increases the size of the defender's strategy space and hence, will only improve the defender's utility. Note, as each bihierarchy is implementable, the convex hull will also be implementable [Brown *et al.*, 2016].

To take the convex hull, first notice each bihierarchy $H_i$ is a set of linear constraints and can be written as $D_i n \leq b_i$ for matrix $D_i$ and vector $b_i$. Hence, by definition $\mathbf{n}(H_i) = \{\mathbf{n}|D_i\mathbf{n} \leq b_i\}$. Using a result from [Balas, 1985] that represents the convex hull using linear constraints, we can write: $conv(\mathbf{n}(H_1), \ldots, \mathbf{n}(H_l)) = \{\mathbf{n}| \sum_i \mathbf{n}_i, D_i\mathbf{n}_i \leq \lambda_i b_i, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$. The convex hull of the bihierarchies can then be computed efficiently using an LP similar to MSLP.

In terms of the convex hull we have two options available: (1) Take the convex hull of all bihierarchies or (2) build the convex hull iteratively. In some cases, the set of bihierarchies available to the defender can be very large and hence, optimizing over all bihierarchies is not feasible. To alleviate this issue, the convex hull can be built iteratively. This is done by first sorting the bihierarchies by the defender utility $v$. Next, we take the convex hull of the top two bihierarchies which gives a utility $v'$ to the defender. We continue adding bihierarchies to the convex hull while the utility $v'$ returned increases by at least some $\epsilon$, and stop otherwise.

## 7 Evaluation

We evaluate the CAG model and solution algorithms with experiments inspired by the operations of the AFCYBER. The game payoffs are set to be zero-sum, i.e. $U_{\delta,c}^u = -U_{\theta,c}^u$, and the defender's payoffs are randomly generated with $U_{\delta,c}^u$ uniformly distributed in $[-1, -10]$. The rest of the game payoffs, $U_{\delta,c}^d$ and $U_{\theta,c}^d$, are set to be zero. For each experiment we average over 30 randomly generated game instances.
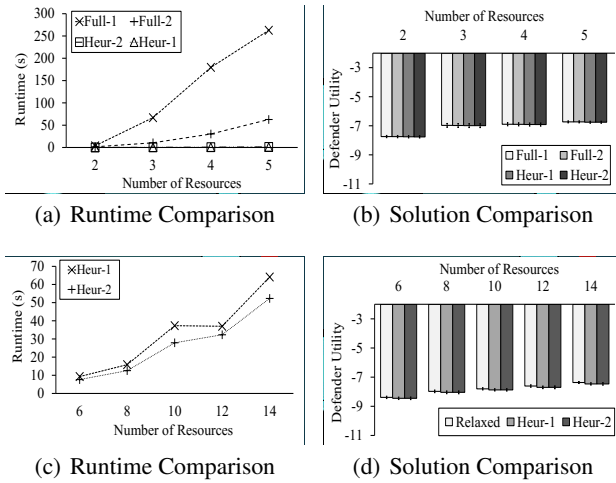


(a) Runtime Comparison      (b) Solution Comparison



(c) Runtime Comparison      (d) Solution Comparison

Figure 3: Experimental Results for CAG instances.

**Full vs Heuristic Search** Whether the heuristic approach of staying close to $\mathbf{n}^*$ would yield the right solution quality-speed tradeoff remains to be seen. To test this, we compare the performance of the full branch-and-bound search (Full) to the heuristic search (Heur). For this experiment we test two variations of the full search: Full-1 which uses the full convex hull and Full-2 which uses the iterative convex hull. For the Heuristic search we test the same two variations, labeled as Heur-1 and Heur-2. For these instances we have 20 systems, 3 attack methods, and 3 alert types.



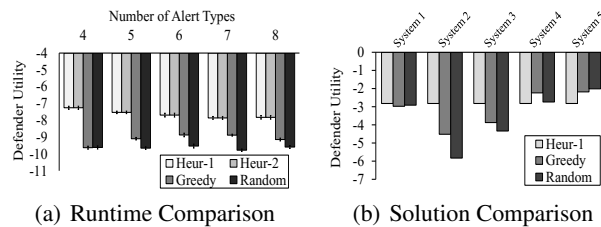(a) Runtime Comparison      (b) Solution Comparison

Figure 4: Allocation Approach Comparison.

In Figure 3(a) we vary the number of resources on the x-axis and we show the runtime in seconds on the y-axis. As can be seen the runtime of the full search explodes exponentially as the number of resources is increased. However, the average runtime of the heuristic approach is under 1 second in all cases and provides up to a 100x runtime improvement for 5 resources. In Figure 3(b) the number of resources are on the x-axis while the y-axis shows the defender's expected

utility. This graph shows that all variations perform similarly, with the heuristic suffering less than 1% solution in defender utility compared to the full search for all game sizes. Hence, these results show that our heuristic significantly improves runtime without sacrificing solution quality.

**Solving large CAG** Another important feature of real-world domains are the larger number of cybersecurity analysts available to investigate alerts. Accordingly, our next experiment tests the scalability of our heuristic approach to large CAG instances. The parameters for these experiments are 100 systems, 10 attack methods, and 3 alert levels.

In Figure 3(c) we show the runtime results with the number of analysts on the x-axis and the runtime in seconds on the y-axis. For example, Heur-1 takes an average of 40 seconds to solve a CAG with 10 analysts. This graph shows the heuristic runs in under a minute, even as we increase the analysts from 6 to 14. In Figure 3(d) we show the solution quality results with the number of analysts on the x-axis and the defender's expected utility on the y-axis. We compare the solution quality to the (potentially non-implementable) MSLP solution. This graph highlights that the heuristic approach achieves a utility close to the theoretical optimal value. Therefore, this experiment shows that our approaches scale to large CAG without sacrificing much solution quality.

**Allocation Approach** Our last experiment aim to show that our game theoretic apporach for CAGs outperform approaches used in practice. In addition to our heuristic, we compare against a greedy approach which investigates the highest priority alerts from the most critical bases first and a random approach for the allocation. The parameters for this experiment are 20 systems, 5 attack methods, and 10 analysts. In Figure 4(a) we show the solution quality results. On the x-axis we vary the number of alert types and on the y-axis we show the defender's utility. For example, with 4 alert types the heuristics achieve a utility of -7.52 while the greedy and randomized allocations give -9.09 and -9.65, respectively. This difference is statistically significant ($p < 0.05$). In Figure 4(b), we show a solution comparison for a specific CAG instance. This graph gives intuition for why our approach performs so well. The greedy and random approaches tend to overprotect some systems (system 4) while leaving others without adequate protection (system 2).

## 8 Conclusion

In this paper we address the pressing problem in cyber security operations of how to allocate cyber alerts to a limited number of analysts. We introduce the Cyber-alert Allocation Game (CAG) to analyze this problem and show computing optimal strategies for the defender is NP-hard. To solve CAG, we present a novel approach to address implementability issues in computing the defender's optimal marginal strategy. Finally, we give heuristics to solve large CAGs, and give empirical evaluation of the CAG model and solution algorithms.

# References

[afc, 2017] *24th Air Force - AFCYBER*, 2017. http://www.24af.af.mil.

[aft, 2016] *688th Cyberspace Wing*, 2016. http://www.24af.af.mil/Units/688th-Cyberspace-Wing.

[Balas, 1985] Egon Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486, 1985.

[Barbara and Jajodia, 2002] Daniel Barbara and Sushil Jajodia. *Applications of data mining in computer security*, volume 6. Springer Science & Business Media, 2002.

[Brown *et al.*, 2016] Matthew Brown, Arunesh Sinha, Aaron Schlenker, and Milind Tambe. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *AAAI conference on Artificial Intelligence (AAAI)*, 2016.

[Budish *et al.*, 2013] Eric Budish, Yeon-Koo Che, Fuhito Kojima, and Paul Milgrom. Designing random allocation mechanisms: Theory and applications. *The American Economic Review*, 103(2):585–623, 2013.

[DAmico and Whitley, 2008] Anita DAmico and Kirsten Whitley. The real work of computer network defense analysts. In *VizSEC 2007*, pages 19–37. Springer, 2008.

[Durkota *et al.*, 2015] Karel Durkota, Viliam Lisỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. Approximate solutions for attack graph games with imperfect information. In *GameSec*, pages 228–249. Springer, 2015.

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

[Hofmeyr and Forrest, 1999] Steven Andrew Hofmeyr and Stephanie Forrest. *An immunological model of distributed detection and its application to computer security*. PhD thesis, Citeseer, 1999.

[Hu *et al.*, 2003] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289, 2003.

[Jain *et al.*, 2010a] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 2010.

[Jain *et al.*, 2010b] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, 2010.

[Kiekintveld *et al.*, 2009] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. AAMAS, 2009.

[Korzhyk *et al.*, 2010] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[Laszka *et al.*, 2016] Aron Laszka, Jian Lou, and Yevgeniy Vorobeychik. Multi-defender strategic filtering against spear-phishing attacks. In *AAAI*, 2016.

[Letchford and Conitzer, 2013] Joshua Letchford and Vincent Conitzer. Solving security games on graphs via marginal probabilities. In *AAAI*, 2013.

[Patton *et al.*, 2011] Robert M Patton, Justin M Beaver, Chad A Steed, Thomas E Potok, and Jim N Treadwell. Hierarchical clustering and visualization of aggregate cyber data. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 1287–1291. IEEE, 2011.

[Riley *et al.*, 2014] Michael Riley, Ben Elgin, Dune Lawrence, and Carol Matlock. Missed alarms and 40 million stolen credit card numbers: How target blew it. http://www.zdnet.com/article/anatomy-of-the-target-data-breach-missed-opportunities-and-lessons-learned/, 2014. Accessed: 2016-11-10.

[Sommer and Paxson, 2010] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[Spathoulas and Katsikas, 2013] Georgios Spathoulas and Sokratis Katsikas. Methods for post-processing of alerts in intrusion detection: A survey. 2013.

[Tambe, 2011] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[Wu *et al.*, 2015] Jianfa Wu, Dahao Peng, Zhuping Li, Li Zhao, and Huanzhang Ling. Network intrusion detection based on a general regression neural network optimized by an improved artificial immune algorithm. *PloS one*, 10(3):e0120976, 2015.

[Yin *et al.*, 2010] Zhengyu Yin, Dmytro Korzhyk, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, pages 1139–1146. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[Zimmerman, 2014] Carson Zimmerman. Ten strategies of a world-class cybersecurity operations center. *MITRE corporate communications and public affairs. Appendices*, 2014.