GUARDS - Game Theoretic Security Allocation on a National Scale

James Pita, Milind Tambe, Chris Kiekintveld*, Shane Cullen**, Erin Steigerwald***
University of Southern California, Los Angeles, CA 90089
*University of Texas at El Paso, El Paso, TX 79968
**APEX STORE-Technology Lead: Science and Technology Directorate,
Department of Homeland Security
***Program Manager: Transportation Security Administration

ABSTRACT

Building on research previously reported at AAMAS conferences, this paper describes an innovative application of a novel game-theoretic approach for a *national scale* security deployment. Working with the United States Transportation Security Administration (TSA), we have developed a new application called GUARDS to assist in resource allocation tasks for airport protection at over 400 United States airports. In contrast with previous efforts such as ARMOR and IRIS, which focused on one-off tailored applications and one security activity (e.g. canine patrol or checkpoints) per application, GUARDS faces three key issues: (i) reasoning about hundreds of heterogeneous security activities; (ii) reasoning over diverse potential threats; (iii) developing a system designed for hundreds of end-users. Since a national deployment precludes tailoring to specific airports, our key ideas are: (i) creating a new game-theoretic framework that allows for heterogeneous defender activities and compact modeling of a large number of threats; (ii) developing an efficient solution technique based on general purpose Stackelberg game solvers; (iii) taking a partially centralized approach for knowledge acquisition and development of the system. In doing so we develop a software scheduling assistant, GUARDS, designed to reason over two agents — the TSA and a potential adversary — and allocate the TSA's limited resources across hundreds of security activities in order to provide protection within airports.

The scheduling assistant has been delivered to the TSA and is currently under evaluation and testing for scheduling practices at an undisclosed airport. If successful, the TSA intends to incorporate the system into their unpredictable scheduling practices nationwide. In this paper we discuss the design choices and challenges encountered during the implementation of GUARDS. GUARDS represents promising potential for transitioning years of academic research into a nationally deployed system.

Categories and Subject Descriptors

J.m [**Computer Applications**]: MISCELLANEOUS

General Terms

Security, Design, Performance

Cite as: GUARDS - Game Theoretic Security Allocation on a National Scale, James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen and Erin Steigerwald, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

Keywords

Applications, Game Theory, Security, Resource Allocation

1. INTRODUCTION

The United States Transportation Security Administration (TSA) is tasked with protecting the nation's transportation systems [2]. These systems are often large in scale and require many personnel and security activities to protect them. One set of systems in particular is the over 400 airports [2]. These airports serve approximately 28,000 commercial flights per day and up to approximately 87,000 total flights [1]. To protect this large transportation network, the TSA employs approximately 48,000 Transportation Security Officers [2]. These Security Officers are responsible for implementing security activities at each individual airport in order to provide security for the transportation network.

While many people are aware of common security activities, such as individual passenger screening, this is just one of many security layers TSA personnel implement to help prevent potential threats [2]. These layers can involve hundreds of heterogeneous security activities executed by limited TSA personnel leading to a complex resource allocation challenge. Unfortunately, TSA cannot possibly run every security activity all the time and thus must decide how to appropriately allocate its resources among the layers of security to protect against a number of potential threats.

To aid the TSA in scheduling resources in a risk-based manner, we take a multi-agent game-theoretic approach. Motivated by advantages of such an approach reported at AAMAS conferences (see Section 2.2), we utilize Stackelberg games where one agent (the leader) must commit to some strategy first and a second agent (the follower) can make his decision with knowledge of this commitment. Here, the TSA acts as a defender (i.e. the leader) who has a set of targets to protect, a number of security activities to protect each target, and a limited number of resources to assign to these security activities. This approach then models a motivated attacker's ability to observe the TSA's resource allocations before choosing a potential threat to execute in an attempt to attack an airport target. The advantage of our approach is in finding the optimal mixed strategy for the TSA to commit to in order to provide them with a risk-based, randomized schedule for allocating their limited resources. From the perspective of the underlying game-theoretic model, a crucial difference of our novel approach and previous approaches is this: we allow for both heterogeneous security activities and threats whereas previous "security games" approaches reported at AAMAS [14, 15] are only able to consider homogeneous security activities and threats, leading to a new game model called "Security Circumvention Games" (SCGs).

In conjunction with TSA subject matter experts, we developed a software system, Game-theoretic Unpredictable and Randomly Deployed Security (GUARDS), that utilizes a Stackelberg framework to aid in protecting the airport transportation network. From an application perspective, the fundamental novelty in GUARDS, compared to previous applications [14, 15] of such game-theoretic approaches, is the potential national scale deployment at over 400 airports. Given that previous approaches only dealt with a single standalone location, this scale raises three new issues. The first issue is in appropriately modeling the TSA's security challenges in order to achieve the best security policies (mixed strategy). Due to the complex nature of TSA's security challenges, traditional models of security games [17] are no longer appropriate models. Specifically, the TSA's domain has the following additional features beyond traditional security games: (i) heterogeneous security activities for each potential target; (ii) heterogeneous threats for each potential target; (iii) unique security activities for individual airports. The second issue is in efficiently solving the model we developed where, because we consider a national deployment, a special-purpose solver may not be appropriate. In fact, previous solution techniques [8, 9] for traditional security games are no longer directly applicable. The final issue is in knowledge acquisition for the many variables involved in TSA's security challenges.

In consideration of national deployment for the TSA, we face two unique constraints. First, headquarters cannot do centralized planning where they create a single optimal mixed strategy (security policy) that will be applicable to all airports. Each airport is unique and thus will require its own individual security policy. Second, TSA wants to maintain a common standard of security among airports. This precludes an entirely decentralized approach where each individual airport is completely in charge of creating their security policy. Even so, due to the possibility of over 400 end-users, it is not practical to sit down with each location and tailor the system to their individual needs. This presents a challenge in acquiring the necessary domain knowledge for such a large network of airports to appropriately model their security challenge.

To address these issues, we developed both a new formal model of security games and techniques to solve this class of games. We also had to incorporate a new methodology for knowledge acquisition. To appropriately model the TSA's security challenges we created a novel game-theoretic model, which is referred to as Security Circumvention Games (SCGs), and cast the TSA's challenges within this model. In the creation of SCGs we provide the following contributions: (i) the ability for defenders to guard targets with more than one type of security activity (heterogeneous activities); (ii) the ability for attackers to choose threats designed to circumvent specific security activities. Given our new model, we designed an efficient solution technique in which we create a compact representation of SCGs. This allows us to avoid using a tailored Stackelberg solver and instead utilize a general purpose Stackelberg solver to compute solutions efficiently. Finally, we took a partially centralized approach to knowledge acquisition for the TSA domain. We integrated a two phase knowledge acquisition process in which we acquire common information, standards, and practices directly from TSA headquarters and then developed the GUARDS system itself to acquire the necessary information that is unique to individual airports.

These key issues present a novel and exciting problem in transitioning years of research from the AAMAS conference to a highly complex domain [9, 12, 14, 15, 17]. GUARDS is currently under evaluation by the TSA with the goal of incorporating its scheduling practices into their unpredictable security programs across airports nationwide.

2. BACKGROUND

Game theory is well known to be a useful foundation in multi-agent systems to reason about multiple agents each pursuing their own interests [7]. Game-theoretic approaches, specifically based on Stackelberg games, have recently become popular as approaches to address security problems (e.g. assigning checkpoints, air marshals, or canine patrols). These approaches reason about two agents pursuing opposing interests (i.e. a security force and an adversary) in an attempt to optimize the security force's goals. Specifically, they model the commitment a security force must make in providing security and the attacker's capability of observing this commitment before attacking. The objective is to find the optimal mixed strategy to commit to given that an attacker will optimize his reward after observing this strategy. At this point we will describe how security games, as defined in [17], fit into the Stackelberg paradigm. In Section 3.1 we will define SCGs to account for the challenges that the TSA faces.

2.1 Security Games

In a security game there are two agents – the defender (security force) and an attacker – who act as the leader and the follower in a Stackelberg game. There are also a set of targets, which the defender is trying to protect. Each of these targets has a unique reward and penalty to both the defender and attacker. Thus, some targets may be more valuable to the defender then others. To protect these targets the defender has a number, K, of resources at her disposal. There is a single security activity being considered and these resources can be allocated to execute this activity on any target. Once a resource is allocated to a target it is marked as covered, otherwise it is marked as uncovered. If the attacker attacks an uncovered target he gets his reward and the defender her corresponding penalty else vice versa. The defender's goal is to maximize her reward given that the attacker will attack with knowledge of the defensive strategy the defender has chosen. In most cases, the optimal strategy for the defender is a randomized strategy in which she chooses a mixed strategy over all her possible resource assignments.

There exist a number of algorithms and techniques for solving security games [6, 8, 9, 12]. DOBSS, a mixed-integer linear program, and the Multiple Linear Programs methods are the most general and are capable of solving any Stackelberg game optimally [6, 12]. The other algorithms are tailored to security games specifically and are much faster in practice for these games.

2.2 Assistants for Security Games

A number of tools have been designed to assist in security problems that fall under the security game paradigm. ARMOR and IRIS are two such tools which take a game-theoretic approach for scheduling checkpoints and canine patrols (ARMOR) and Federal Air Marshals (IRIS). In fact, ARMOR and IRIS have been deployed to aid with security operations for the Los Angeles World Airport Police at Los Angeles International airport and for the Federal Air Marshals Service respectively [14, 15]. These systems offer two sets of advantages. The first set of advantages deal with solution quality: (i) they provide an optimal mixed strategy for the single security activity they consider such as assigning checkpoints or air marshals; (ii) the randomized solutions produced both avoid deterministic strategies that are easily exploitable and remove the human element in randomization since humans are well known to be poor randomizers [16]; (iii) they reason over difficult problems that are often impossible for humans to reason over optimally. These advantages are useful for any tool being utilized in the field to help with randomized resource allocation in security problems and we incorporate them into GUARDS as well.

The second set of advantages are specific to the problems they address: (i) they develop unique and useful preference elicitation systems and knowledge acquisition techniques for the specific problem they address; (ii) based on practical requirements, they apply state of the art algorithms tailored to solving the Stackelberg games they consider efficiently. Unfortunately, previous methods are too specific to the standalone location they consider and thus cannot directly be applied in GUARDS; indeed GUARDS requires us to address a novel set of challenges described in the next section.

3. NATIONAL DEPLOYMENT CHALLENGES

We now describe in detail the three major issues in potentially deploying game-theoretic randomization for airport security on a national scale, including modeling, computational, and knowledge acquisition challenges and our solutions to them.

3.1 Modeling the TSA Resource Allocation Challenges

While we are motivated by an existing model of security games [17], there are three critical aspects of the new TSA domain that raise new challenges. First, the defender now reasons over heterogeneous security activities for each potential area within an airport[1]. For example, airports have ticketing areas, waiting areas, and cargo holding areas. Within each of these areas, TSA has a number of security activities to choose from such as perimeter patrols, screening cargo, screening employees and many others. Second, given the multiple possible security activities, the defender may allocate more than one resource per area (i.e. areas are no longer covered or uncovered). Finally, the defender now considers an adversary who can execute heterogeneous attacks on an area. The TSA must reason about a large number of potential threats in each area such as chemical weapons, active shooters, and bombs. The key challenge is then how to allocate limited TSA security resources to specific activities in particular areas, taking into account an attacker's response.

To address this challenge it is necessary to create a more expressive model than outlined in security games; one that is able to reason over the numerous areas, security activities, and threats within an individual airport. We refer to this new class of security games as Security Circumvention Games (SCGs). SCGs are more expressive than traditional security games and thus can represent both traditional security games and the games we considered for the TSA. In SCGs, the TSA must choose some combination of security activities to execute within each area and the attacker must reason over both which area to attack and which method of attack to execute based on the defender's strategy. At this time we elaborate on the defender's and attacker's possible strategies.

3.1.1 Defender Strategies

We denote the defender by Θ, and the set of defender's pure strategies by $\sigma_\Theta \in \Sigma_\Theta$. The TSA is able to execute a variety of security activities, which we denote by $S = \{s_1, \ldots, s_m\}$. Each security activity has two components. The first is the type of activity it represents, and the second is the area where the activity is performed. We denote the set of areas by $A = \{a_1, \ldots, a_n\}$.

The defender has K resources available and thus can run any K security activities. The TSA's task is to consider how to allocate these resources among security activities in order to provide the optimal protection to their potential areas. An assignment of K resources to K security activities represents a single strategy $\sigma_\Theta \in \Sigma_\Theta$. For example, if there are three security activities, $S = \{s_1, s_2, s_3\}$ and two resources available, one possible pure strategy for the defender is to assign these two resources to s_1 and s_3. Given that the number of possible combinations of K security activities at an airport can be on the order of 10^{13} or greater for the TSA, we develop a compact representation of the possible strategies that we present in Section 3.2. The defender's mixed strategies $\delta_\Theta \in \Delta_\Theta$ are the possible probability distributions over Σ_Θ. Similar to previous work, a mixed strategy (randomized solution) is typically the optimal strategy.

3.1.2 Attacker Actions

Defending a target against terrorist attacks is complicated by the diversity of the potential threats. For example, an attacker may try to use a vehicle borne explosive device, an active shooter, a suitcase bomb, and many others in any given area. Not all methods of attack would make sense in all areas. For example, using a vehicle borne explosive device in the checked baggage screening area in some airport configurations would not be a viable method of attack. We denote the attacker by Ψ, and the set of pure strategies for the attacker is given by $\sigma_\Psi \in \Sigma_\Psi$. Each pure strategy for the attacker corresponds to selecting a single area $a_i \in A$ to attack, and a specific mode of attack. However, given that each airport considers its own potential threats, enumerating all threats for each individual airport through the software may not be practical. To handle the national deployment challenge we face and avoid this difficulty, we developed a novel way to represent threats for TSA's domain that we describe in Section 3.2.1.

3.2 Compact Representation for Efficiency

While we have developed a model that appropriately captures the TSA's security challenge, one issue with this model is that both the attacker and defender strategy spaces grow combinatorially as the number of defender security activities increases. Also, listing such a large number of potential threats would lead to extreme memory and runtime inefficiencies. Furthermore, existing solution techniques that have been developed for security games [8, 9] are not directly applicable to Security Circumvention Games (SCGs).

With this in mind, we looked at an alternate approach to finding optimal solutions efficiently. Specifically, we looked at representing threats in a more intelligent manner and creating a compact representation for the defender strategy space. By utilizing both of these techniques, we achieved large reductions in run-time. We utilized a general Stackelberg solver known as DOBSS [12] to solve our compact representation and avoided creating a tailored algorithm for each specific airport. At this time we will explain both how we model threats and how we achieve a compact representation of the defender's full strategy space.

3.2.1 Threat Modeling for TSA

While it is important that we reason over all the security activities that are available to an individual airport, enumerating all of the large number of potential threats they face can lead to severe memory and runtime inefficiencies. Thus, the problem we face is how to model attack methods in a way that limits the number of threats GUARDS needs to reason over, but appropriately captures both an attacker's capabilities and his goals. In particular, we automatically generate attack methods for the adversary that capture two key goals: (i) an attacker wants to avoid the security activities that are in place; (ii) an attacker wants to cause maximal damage with minimum cost.

In order to achieve these goals an intelligent adversary will ob-

[1]Due to the nature of the TSA's security challenge, we will refer to targets in the TSA's domain as areas henceforth.

serve security over time and design his attack method based on his observations. The attacker's plan will be designed to avoid security activities that he believes will be in place. We will refer to this as circumventing security activities. For example, imagine there is a single area with three security activities such as passenger screening, luggage screening, and perimeter patrol. In this example, TSA only has one resource available and thus can only execute one of these activities at a time. While passenger screening may have the highest probability of success, if TSA never screens luggage or patrols the perimeter, the adversary can choose an attack path that avoids passenger screening such as utilizing a suitcase bomb or an attack from the perimeter.

On the defender side, we know that dedicating more resources to security activities in an area increases the security afforded to that area. However, even with more resources, we want to avoid being predictable since attackers can exploit this predictability; avoiding the security activities they know will be in place. Thus, we needed to represent threats in a way that accounts for the attacker's ability to observe security in advance and avoid specific security activities, but still represents the benefit of dedicating more resources.

A naïve approach is to represent only a single threat per area and decrease the likelihood of success for that threat as more security activities are put in place. This captures the increase in security for additional security activities, however, it does not account for the attacker's ability to circumvent security activities. With this method you would simply choose security activities in the order of their relative success making it predictable and exploitable.

The alternative that we chose is to create a list of potential threats that circumvent different combinations of specific security activities. By basing threats on circumventing particular combinations of security activities, we avoid the issue of enumerating all the possible potential threats. Instead the threats are automatically created based on the security activities in an area. However, we also incorporate a cost to the attacker for circumventing more activities to capture the idea of causing maximal damage at minimal cost. Each individual activity has a specific circumvention cost associated with it and more activities circumvented leads to a higher circumvention cost. This cost reflects the additional difficultly of executing an attack against increased security. This difficulty could be due to requiring additional resources, time and other factors for executing an attack. Since attackers can now actively circumvent specific security activities, randomization becomes a key factor in the solutions that are produced because any deterministic strategies can be circumvented.

3.2.2 Compact Representation

We introduce a compact representation that exploits similarities in defender security activities to reduce the number of strategies that must be enumerated and considered when finding an optimal solution to SCGs. First, we identify security activities that provide coverage to the same areas, and have the same circumvention costs (i.e. have identical properties). Let $\gamma_i \in \Gamma$ represent the sets of security activities that can be grouped together because they have identical properties. Now, instead of reasoning over individual security activities, we reason about groups of identical security activities $\gamma_i \in \Gamma$. A strategy $\sigma_\Theta \in \Sigma_\Theta$ is represented by the number of resources assigned to each set of identical security activities γ_i.

To illustrate this new representation, we provide a concrete example of the full representation versus the compact representation in Tables 1 and 2. In this example there are 4 security activities and 2 resources. Here, s_1 and s_2 have identical circumvention costs and affect a_1 while s_3 and s_4 have identical circumvention costs and affect a_2. Table 1 presents the full representation with corresponding payoffs and Table 2 represents the compact form of the same where γ_1 represents the group s_1 and s_2 and γ_2 represents the group s_3 and s_4. In both tables, each row represents a single pure strategy for the defender and each column the same for the attacker. Notice in Table 1 each strategy $\sigma_\Theta \in \Sigma_\Theta$ is represented by the exact security activities being executed while in Table 2 it is only which set $\gamma_i \in \Gamma$ each resource has been allocated to.

The key to the compact representation is that each of the security activities from a set $\gamma_i \in \Gamma$ will have the same effect on the payoffs. Therefore, it is optimal for the defender to distribute probability uniformly at random across all security activities within a set γ_i, so that all security activities are chosen with equal probability in the solution. Given that the defender strategy uniformly distributes resources among all security activities $s_j \in \gamma_i$ we also know that it does not matter which specific security activities the attacker chooses to circumvent from the set γ_i. For any given number of security activities circumvented, the expected payoff to the attacker is identical regardless of which specific activities within the set are chosen. This is because we are selecting security activities uniformly at random within the set γ_i. Therefore, we can use a similar compact representation for the attacker strategy space as for the defender, reasoning only over the aggregate number of security activities of each type rather than specific security activities.

Given this, we only need to know how many security activities are selected from each set in order to compute the expected payoffs for each player in the compact representation. For example, examining the second row and second column of Table 2 we see that the reward to the defender is -2 and the reward to the attacker is 0. In this case, the defender strategy is to assign 1 resource to activities in γ_1 and 1 resource to activities in γ_2. Given that she is uniformly distributing these resources, it follows that she will execute s_1 half of the time and s_2 the other half. On the attacker side, we know that the attacker is circumventing one security activity from the set γ_1. If he circumvents either s_1 or s_2 he will only succeed half of the time. Thus, half of the time the defender receives 4 and the other half -8 for an expectation of -2 $(4 * .5 + (-8) * .5)$. We compute the attacker's reward in the same manner.

	$a_1 : \emptyset$	$a_1 : s_1$	$a_1 : s_2$	$a_2 : \emptyset$	$a_2 : s_3$	$a_2 : s_4$
s_1, s_2	2, -1	4, -3	4, -3	-20, 10	-17, 7	-17, 7
s_1, s_3	2, -1	-8, 3	4, -3	5, -5	-17, 7	8, -8
s_1, s_4	2, -1	-8, 3	4, -3	5, -5	8, -8	-17, 7
s_2, s_3	2, -1	4, -3	-8, 3	5, -5	-17, 7	8, -8
s_2, s_4	2, -1	4, -3	-8, 3	5, -5	8, -8	-17, 7
s_3, s_4	-10, 5	-8, 3	-8, 3	5, -5	8, -8	8, -8

Table 1: Example payoffs for sample game.

	$a_1 : \emptyset$	$a_1 : \gamma_1$	$a_2 : \emptyset$	$a_2 : \gamma_2$
γ_1, γ_1	2, -1	4, -3	-20, 10	-17, 7
γ_1, γ_2	2, -1	-2, 0	5, -5	-4.5, -5
γ_2, γ_2	-10, 5	-8, 3	5, -5	8, -8

Table 2: Example compact version of sample game.

Given this compact representation for both the defender and attacker, we can compute an optimal mixed strategy of assigning resources over Γ. Once we have this mixed strategy, we will need to determine an actual strategy for the TSA to execute by sampling one of the possible strategies from the mixed strategy we have determined for our compact representation (e.g. one sample may be $\gamma_2\gamma_2$). Once sampled, we will know exactly how many resources are available to each set $\gamma_i \in \Gamma$. Given this resource assignment,

we can then sample security activities by selecting k uniformly at random where k is the number of resources assigned to $\gamma_i \in \Gamma$. This specific set of security activities for each area under the current resource assignment is a full strategy for the TSA to execute.

3.3 Knowledge Acquisition

One of the most difficult issues we faced from a potential national deployment perspective was in acquiring the appropriate knowledge for the security challenge being considered. In the past, tools such as ARMOR and IRIS [14, 15] have been developed to be used for a single security activity in a standalone location. That approach gave the advantage of being able to sit down with domain experts who will be using the system and develop a knowledge acquisition process for the specific domain at hand. Unfortunately, with hundreds of airports to consider, it is not possible to sit down at each location and acquire the exact needs for each of them. To overcome this obstacle, in close collaboration with TSA headquarters we developed a two phase knowledge acquisition process.

In phase one, we take an approach similar to previous centralized approaches. In particular, we met with domain experts to acquire knowledge that is common among all airports. This included area definitions, defining security activities, and determining resource capabilities among others. In collaboration with headquarters, we then decided how individual airports can customize these components in their individual games while maintaining standards set forth by headquarters (as discussed below). Additionally, we collaborated with headquarters to limit the amount of customization inputs so users at individual airports are not overwhelmed – a key to organizational acceptance as discussed in Section 6.

In phase two of our knowledge acquisition, we took a decentralized approach where it is the responsibility of individual airports to input customized information. For this phase we could not create a rigid system that was designed with one specific game instance in mind for a single airport. Instead, we rely on SCGs and developed a system in collaboration with headquarters that allows individual airports to manipulate specific components within this framework to create unique game instances. These inputs are designed to ensure that individual airports maintain standards set forth by headquarters in phase one. For example, individual airports are responsible for determining the unique reward and penalty associated with each area for the defender and attacker given a successful or unsuccessful attack. However, TSA headquarters requires a standardized method for determining these values to ensure that resources are being appropriately distributed. To this end, we designed an input module within GUARDS to reflect a risk evaluation process developed by the TSA where a series of quantifiable questions are answered for each area by individual airports. These questions include such things as the number of fatalities that may result from an attack in an area, whether the area has access control, and others. The answers to these questions are then combined in a mathematical formula to decide the values for a particular area for both the defender and attacker[2]. This input process ensures that airports are appropriately valuing the areas they protect within an airport according to headquarters guidelines. In general, using the customizable input airports generate, we can then create the unique game instance for that particular airport.

Our two phase knowledge acquisition process follows a partially centralized approach and provides the following advantages: (i) it allows domain experts from TSA headquarters to assure that the system meets the required needs of the challenge being considered; (ii) it focuses on creating customizable inputs instead of a system tailored to a highly specific problem instance; (iii) it allows TSA headquarters control while still enabling individual airports to customize the system to meet their individual needs. For the third advantage, there is an important trade-off between system customization and standardization among airports. Determining this trade-off is an important part of the first phase in this two phase knowledge acquisition process.

[2]Previous work [14, 15, 17] has shown that security problems are not necessarily zero-sum for a number of potential reasons. In GUARDS, for similar reasons, games are not necessarily zero-sum.

4. SYSTEM ARCHITECTURE

The GUARDS system consists of three modules. First, there is an input module designed to acquire the necessary information for one unique instance of the complex security game we consider. Second, there is a back-end module that is designed to both create and solve the unique game instance based on the inputs. Finally, there is a display/output module that presents a sample schedule to TSA officials based on the optimal solution. We now describe each individual module and its operations in TSA's airport domain.

Input Module: The input module is composed of three classes of inputs that are required by the system in order to generate a representative Stackelberg game and create an optimal allocation of resources. All inputs are quantifiable and tangible so that headquarters is able to maintain standards and guidelines on the way security policies are created. The first input is the area data. First, airports must input each of their potential areas. Second, for each area the airport must go through the risk evaluation process (see Section 3.3) which involves answering a series of quantifiable questions. The second set of input is the security activities data. For each area the airport must list all of the security activities that are available to execute in that area. While there is a standard list of activities airports can select from, they are also able to input new security activities that may be unique to that airport. The third input is the resource data. This includes the number of days to create a schedule for and the number of resources available each day.

Back-end Module: The back-end module has three primary components. These are generating the game, solving the game, and returning one sample schedule for TSA's use. First, based on the inputs from the input module, there is a component that creates a compact representation of the specific game instance the system is considering. This game instance is based on the compact form of the model we presented in Section 3.1. Second, we compute the solution to the Stackelberg game model using DOBSS, a general Stackelberg solver [12]. This produces a solution, which is a mixed strategy over the possible action space as defined in Section 3.2. Finally, using the optimal mixed strategy we sample one possible resource assignment that can be implemented by TSA.

Display/Output Module: The actual resource assignment selected is presented to the user via the display/output module. The schedule created is shown in the interface first as a summary of the number of resources assigned to each area similar to the mockup in Figure 1[3]. Once the schedule is created, TSA personnel can proceed to a more in depth report of the schedule. This report lists each of the specific security activities that were chosen for each location along with specific details of these security activities. After reviewing the report, TSA personnel can also choose to examine the distribution of resources over areas that the optimal mixed strategy provides as in Figure 2.

5. EVALUATION

[3]We are unable to show actual screen shots from our system due to security concerns. For the remainder of this paper we will show only basic visual representations of what GUARDS displays.

Locations	8/15	8/16	8/17	8/18	8/19	8/20	8/21
A1	1	3	1	1	3	0	0
A2	0	2	1	0	0	0	0
A3	2	2	0	1	0	3	1
A4	1	0	3	2	0	1	2
...	...	...	...	...	...	...	...
Total	10	10	10	10	10	10	10

Figure 1: Summary of sample schedule

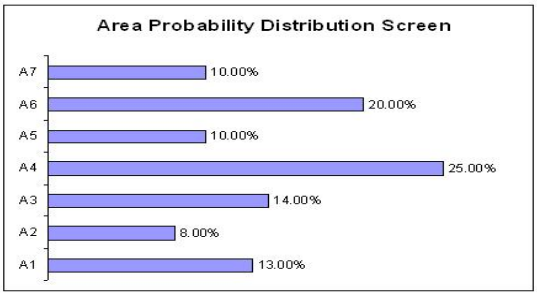


Figure 2: Summary of probability distribution over areas

When evaluating a system like GUARDS there are two important issues that are raised. The first issue is with scalability and run-times. To be useful in practice, the system needs to be able to solve real world challenges. The second issue is evaluating the value of the security policies generated against alternative approaches. In the following sections we present each of these evaluations.

5.1 Run-time Analysis

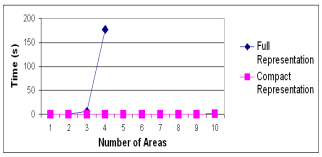
We present simulation results focusing on the computational efficiency of our compact method versus the full representation. All experiments are run on a system with an Intel 2 GHz processor and 1 GB of RAM. We used a publicly available linear programming package called GLPK to solve optimization problems as specified in the original DOBSS procedure. For the compact version we use a slightly altered version of DOBSS that is designed specifically for efficiency in the compact representation. The solver was allowed to use up to 700 MB of memory during the solution process. For larger game instances, solving the problem with the full representation runs out of memory and solutions cannot be found. In the results presented below we exclude results for cases where the full representation was not able to produce a result using the allotted memory. We also note that in all experiments both the solution found by the full representation and the solution found by the compact representation are optimal.

To test the solution methods we generated random game instances by randomly selecting payoff values from 1 to 50 and circumvention costs from 1 to 5 for each area. For each experiment we generated 20 random game instances and averaged the results (there is little variance in the run-times for different problem instances). We considered three different scenarios. The first scenario presents results for the case where there is an increasing number of areas, and each area has exactly 3 security activities associated with it. There are 5 resources available for the defender, and each security activity has identical properties (i.e. no security activity has a higher cost for circumvention or higher probability of success) for the area it is associated with. Given the M possible areas, for the full representation there are $\binom{3*M}{5}$ possible defender pure strategies and $8 * M$ possible attacker pure strategies. Thus, in the 10 area case there are 142,506 defender pure strategies and 80 attacker pure strategies. Examining Figure 3 (a), we show the improvement in run-time of our compact representation over the full representation. For more than 4 areas, the full representation failed to achieve a solution within the memory bounds. For 4 areas, the compact representation runs much faster than the full representation, with a run-time of less than 1 second versus the 177 seconds required by the full representation. In fact, for 10 areas, the compact representation has an average run-time of approximately 1 second, which is still much faster than the full representation for only 4 areas. Even if the number of security activities associated with each area is a relatively small constant our compact representation provides substantial benefits. As the number of similar security activities associated with an area increases, this advantage grows.

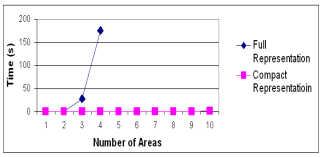
In our second scenario, we considered a situation where security activities are distributed randomly across possible areas. The total number of security activities is set similarly to the previous experiment, in that that the total number of security activities is three times the number of areas. However, we randomly assigned security activities to areas (with each area having at least one security activity) so the number is no longer uniform across areas. Once again the defender has 5 resources available and security activities have identical properties within an area. It follows that in the full representation, the number of defender pure strategies and attacker pure strategies are identical to the previous scenario. However, the number of strategies in the compact representation for both the defender and attacker may vary. Looking at Figure 3 (b), we see similar benefits for the compact representation in this case as in the previous experiment with a uniform distribution of activities.

In the final scenario, we considered a situation in which there are 10 areas to protect, each area has 3 identical security activities, and we increased the number of resources available to distribute between these areas. Thus, in the full representation, assuming there are K resources available, the defender has $\binom{30}{K}$ possible pure strategies and the attacker has 80 possible pure strategies. In Figure 4, we increase the number of resources available along the x-axis and show the time to compute a solution in seconds on the y-axis. The full representation is unable to compute a solution for more than 4 resources under these conditions within the allotted memory. On the other hand, the compact representation is able to arrive at a solution for 10 available resources in less than 30 seconds.

These results show the benefits of our compact representation in terms of efficiency. We obtained further efficiency gains by caching results: specifically, the inputs into the game do not change on a daily basis. Thus, we can cache the resulting mixed strategy, and present results from sampling this mixed strategy, as long as the program users have not changed the inputs. When they do change inputs, we resolve the game using our compact representation.



(a) Three activities per area



(b) Random activities per area

Figure 3: X-axis: Areas, Y-axis: Run-time

5.2 Security Policy Analysis

For this analysis we examined the security policies generated by our game representation against two other possible solution strategies. The first strategy is a solution concept where resources are distributed uniformly among areas (uniformly random), an approach

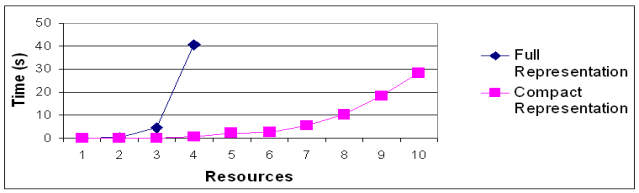


Figure 4: Run-time: Increasing resources for 10 areas with 3 security activities per area

sometimes used in lieu of a game-theoretic approach. The second strategy uses our new representation, however, it does not allow attackers to circumvent security activities (SCGs without circumvention). That is, we allow the attacker only a single attack strategy per area and simply reduce the value of that strategy as the number of security activities increases. This is a simplified model of an attacker as mentioned in Section 3.2.1. Finally, we included our new representation and allow an intelligent attacker to circumvent specific security activities when planning his mode of attack (SCGs).

We generated 20 random game instances with 10 areas and 3 security activities per area. In each game instance the payoff value of each area for both the defender and attacker are randomly selected from 1 to 50 and the circumvention costs are similarly selected from 1 to 5. We then calculated the optimal solution under the current solution strategy (i.e. uniformly random, SCGs without circumvention, and SCGs). After finding the optimal solution, we determined the expected reward for each solution given the assumptions made in SCGs (i.e. attackers are allowed to circumvent specific security activities when planning their attack). For each game instance, we computed the optimal solution varying the number of resources available from 1 to 10 as seen on the x-axis of Figure 5. On the y-axis, we present the average expected reward obtained by each solution strategy across all 20 game instances. In Figure 5 we see that the uniform policy is outperformed by both game-theoretic approaches with the approach accounting for circumvention strategies performing the best. In fact, an approach that accounts for circumvention strategies is the only one that was able to obtain a positive reward for the defender in the 20 randomly generated game instances and in the 10 resource case obtains a 200% improvement in reward over any other strategy. This shows the benefits of reasoning about an intelligent attacker who will research and exploit deterministic security activities.

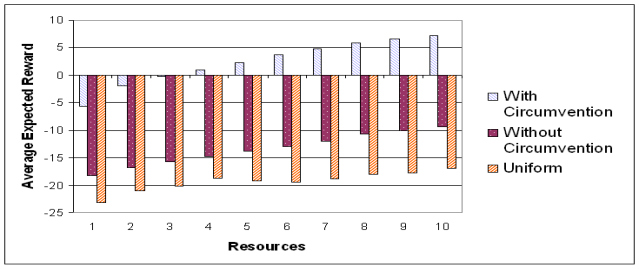


Figure 5: Policy Analysis: Increasing resources for 10 areas with 3 security activities per area

6. LESSONS IN TRANSITIONING RESEARCH INTO PRACTICE

GUARDS is the result of a unique collaboration where university researchers worked directly with a security agency for the purpose of creating a useful product to potentially deploy outcomes of research on a national scale. This collaboration to transition research to such a large-scale deployment has presented valuable lessons. This section outlines the three areas of insights we have gained in the process: (i) acceptance of GUARDS at headquarters; (ii) acceptance of GUARDS by a variety of end-users at numerous airports; (iii) obtaining correct input from users. Some of these insights are contrary to accepted wisdom in the research community.

In a large organization like the TSA, it is important that they are able to provide quality guarantees. A key implication is that a system such as GUARDS must be very clear-cut in terms of its assumptions and its solution quality guarantees based on these assumptions. Researchers often assume that speedy heuristic solutions that are on average high quality may be adequate "in the field", but we have learned in contrast that when dealing with security agencies it is important that we provide guarantees on this solution quality. More importantly, these guarantees may even be required to be optimal (i.e. even if we can guarantee solutions within some bound of the optimal solution it may not be enough). Without guarantees, the TSA may be unable to justify the use of any particular security strategy. In accordance with this requirement, we use a solver known as DOBSS, which provides game-theoretic optimal solutions in Stackelberg games.

With respect to acceptance of GUARDS at individual airports, one major lesson learned is bridging the culture gap in academic research and real-world operations. Indeed, what researchers may consider small uninteresting issues may nullify all their major research advances. For example, in an initial version of GUARDS, we displayed the final probabilities of our mixed strategies, but truncated the presentation of real numbers (i.e. truncating all decimal values). Unfortunately, this single display issue turned out to be a major headache for users who assumed incorrectness on part of GUARDS when the distribution of resources appeared to be less than 100%. Specifically, instead of considering the truncation of all real values, users might assume that some resources were not being utilized. A second major lesson learned is the continued need for efficiency of game-theoretic algorithms. While significant research has gone into speeding up these algorithms, we are still not able to get off-the shelf algorithms and deploy; GUARDS required the use of new compact representations. We have outlined our key advances in this regard in Section 5.1; including the need for caching.

A third lesson learned in user acceptance is careful design of the user interface so as to reduce the amount of user workload to provide inputs: this must be kept at a manageable level. For instance, if users are required to directly enter values into the generated game matrix it can require thousands of inputs. Instead, it is important to provide a user-friendly method of conveying the necessary information. We used a simple interface where users are only required to input the base information that is then used to generate the larger game matrix. By base information, we mean such things as the areas and security activities. This is information that they have direct access to and can easily be input by the individual airports.

Finally, in any collaboration, it is important that researchers are able to obtain the appropriate input from their collaborators. This includes understanding what information is available versus what is not and accounting for this in modeling of the problem. For the available information, often end-users will not understand the techniques being applied and thus are prone to providing vague or incorrect information. For example, when asking a security agency such as the TSA to provide a utility for an attacker and for themselves as a defender on a successful attack, they may always say that is it very bad for themselves and very good for the attacker. Specifically, if there are 5 areas and they provide a utility for each

on a 10 point scale, they may always claim that it is -10 for the defender and 10 for the attacker. In practice, this feedback may not be useful because attacks on different areas may actually have very different impact in terms of economic damage, casualties, and many other factors. To aid in preventing this scenario, it is important to convey the impact that inputs will have on outputs; aiding their understanding of how their inputs will affect the results.

7. RELATED WORK AND SUMMARY

To the best of our knowledge, this paper presents the first-ever effort to transition any research reported at AAMAS conferences to an application designed for potential national scale deployment to hundreds of locations. This contrasts with previous efforts, including efforts that focus on application of game-theoretic approaches such as ARMOR and IRIS [14, 15], as detailed earlier in the paper. It also contrasts alternative models based on Markov Decision Processes (MDPs), queuing theory, or game theoretic approaches that would enumerate all possible defender actions and attacker threats [11, 13]. To accomplish this transition, we outlined novel contributions to game modeling and compact representations of games, because of the scale-up in defender and attacker strategies. This research complements other solution techniques for Stackelberg games [5, 10], which have traditionally not focused on such a scale-up. Our work also complements research actually applied to randomize patrolling strategies in robot patrol [3, 4], given our emphasis on modeling adversaries in a game-theoretic setting.

TSA is charged with protecting over 400 airports in the US. The key challenge is how to intelligently deploy limited security resources to unpredictable security activities within the airport in a risk-based manner to provide the maximum possible protection. These decisions may be made on a daily basis, based on the local information available at each airport.

This paper describes a scheduling assistant for TSA, GUARDS, which takes a game-theoretic approach to this resource allocation task. In creating GUARDS, we address three key issues that arise from a potential national deployment case. These issues are: (i) knowledge acquisition for hundreds of end-users under one organization; (ii) appropriately modeling TSA's security challenge to achieve the best security policies; (iii) efficiently finding solutions to the problem we consider. We addressed the first challenge by using a two phase knowledge acquisition process in which we acquire common information, standards, and practices directly from TSA headquarters. We then constructed the GUARDS system itself to reflect a risk evaluation process designed by TSA to acquire the necessary information that is unique to individual airports. To address the second challenge we developed a novel game-theoretic model, which we refer to as Security Circumvention Games (SCGs), and cast TSA's security challenge within this model. In creating this model we provided the following contributions: (i) the ability for defenders to guard targets with more than one type of security activity (heterogeneous activities); (ii) the ability for attackers to choose threats designed to mitigate specific security activities. Finally, we designed an efficient solution technique for reasoning over our new game model where we rely on creating a compact representation of each game instance and solving it using a general purpose Stackelberg solver. This is in contrast to tailored algorithms of the past that are designed for specific problem instances for standalone locations. To conclude, we present results demonstrating the benefits of our contributions along with lessons learned in creating GUARDS. The scheduling assistant has been delivered to the TSA and is currently under evaluation and testing for unpredictable scheduling practices at an undisclosed airport.

8. ACKNOWLEDGMENTS

The development of GUARDS has only been possible due to the exceptional collaboration with the Transportation Security Administration. This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) under grant number 2007-ST-061-000001. However, any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the United States Department of Homeland Security.

9. REFERENCES

[1] Air traffic control: By the numbers. In *http://www.natca.org/mediacenter/bythenumbers.msp#1*.

[2] TSA | Transportation Security Administration | U.S. Department of Homeland Security. In *http://www.tsa.gov/*.

[3] N. Agmon. On events in multi-robot patrol in adversarial environments. In *AAMAS*, 2010.

[4] N. Agmon, S. Kraus, G. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *IJCAI*, 2009.

[5] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *IAT*, 2009.

[6] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, 2006.

[7] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[8] M. Jain, E. Kardes, C. Kiekintveld, M. Tambe, and F. Ordóñez. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 2010.

[9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordóñez. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.

[10] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[11] R. C. Larson. A hypercube queueing model for facility location and redistricting in urban emergency services. *Computers and OR*, 1(1):67–95, 1974.

[12] P. Paruchuri, J. Marecki, J. Pearce, M. Tambe, F. Ordóñez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *AAMAS*, 2008.

[13] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Security in multiagent systems by policy randomization. In *AAMAS*, 2006.

[14] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport. In *AAMAS*, 2008.

[15] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - a tool for strategic security allocation in transportation networks. In *AAMAS*, 2009.

[16] W. A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. 1972.

[17] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, 2010.