Game Theoretic Deception and Threat Screening for Cyber Security

by

Aaron Schlenker

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Computer Science)

July 2018

Contents

List Of Figures

List Of Tables

Abstract

Protecting an organization's cyber assets from intrusions and breaches due to attacks by malicious actors is an increasingly challenging and complex problem. Companies and organizations (hereon referred to as the defender) who operate enterprise networks employ the use of numerous protection measures to intercept these attacks, such as Intrusion and Detection Systems (IDS) and along with dedicated Cyber Emergency Readiness Teams (CERT) composed of cyber analysts tasked with the general protection of an organization's cyber assets. In order to optimize the use of the defender's limited resources and protection mechanisms, we can look to game theory which has been successfully used to handle complex resource allocation problems and has several deployed real-world applications in physical security domains. Applying previous research on security games to cybersecurity domains introduce several novel challenges which I address in my thesis to create models that deceive cyber adversaries and provide the defender with an alert prioritization strategy for IDS. My thesis provides three main contributions to the emerging body of research in using game theory for cyber and physical security , namely (i) the first game theoretic framework for cyber deception of a defender's network, (ii) the first game-theoretic framework for cyber alert allocation and (iii) algorithms for extending these frameworks to general-sum domains.

In regards to the first contribution, I introduce a novel game model called the Cyber Deception Game (CDG) model which captures the interaction between the defender and adversary during the recon phase of a network attack. The CDG model provides the first game-theoretic framework for deception in cybersecurity and allows the defender to devise deceptive strategies that deceptively alters system responses on the network. I study two different models of cyber adversaries and provide algorithms to solve CDGs that handle the computational complexities stemming from the adversary's static view of the defender's network and the varying differences between adversary models.

The second major contribution of my thesis is the first game theoretic model for cyber alert prioritization for a network defender. This model, the Cyber-alert Allocation Game, provides an approach which balances intrinsic characteristics of incoming alerts from an IDS with the defender's analysts that are available to resolve alerts. Additionally, the aforementioned works assume the games are zero-sum which is not true in many real-world domains. As such, the third contribution in my thesis extends CAGs to general-sum domains. I provide scalable algorithms that have additional applicability to other physical screening domains (e.g., container screening, airport passenger screening).

Acknowledgments

Completing the doctorate has been one of the most arduous, enjoyable, and bittersweet experiences in my life. I have learned an enormous amount about myself which has led to personal growth in unexpected ways. My personal feeling is that most Ph.D. students will share similar sentiments about completing the doctorate and that taking this path has not always led them to the place they expect. However, given the struggles faced during the Ph.D. it is impossible not to come out of it feeling a deeper appreciation of the personal learning process. For me this has been a beautiful, unexpected consequence. It is also important to say that – and I imagine nearly all Ph.D. students agree – completing the doctorate is not possible without the support and encouragement from a small village of people.

My family has been a constant source of support throughout my undergrad years until I finished the doctorate. I feel grateful and fortunate to have all of them in my life. First of all, my brothers David, Kevin, Chris and Scott have all been sources of inspiration giving me the fuel needed to finish those long nights in the office. My parents, Susan and Steve, have guided and encouraged me from the time I was just a young elementary school kid to a seasoned grad student. My grandmother, Shirley, and all of my extended family who gave strength and love from afar. I cannot begin to thank you enough for

providing such a positive influence through these last 26 years. Finishing a Ph.D. becomes exponentially harder without individuals like you all in someone's life.

The many advisors I have had throughout my years as a college student – given the privilege of learning under such wise, intelligent teachers – have given me the intellectual rigor to ask basic questions about the world. This has been a guiding principle used to drive my research and will be a focus of my future work in general. Dr. Bill Johnston (or Dr. J as everyone knows you) and Dr. Jon Sorenson, you are the first two great mentors that have helped shape me into the person I am today. From taking your classes in my first college years, I learned to gain a true appreciation of technical work and exploring abstract concepts and principles. Those valuable experiences first led me to pursuing a research and career oriented towards solving basic problems.

Dr. Milind Tambe, my Ph.D. advisor and good mentor, thank you for taking me under your wing and guiding me through the difficult and rewarding Ph.D. life. You have given me invaluable experience in regards to both research and identifying the most significant problems in the world today. In part, it is my time in the Teamcore research group that has shaped me into the person I am today. I want to thank all of the wonderful people and colleagues I have had the privilege of getting to know during my time in Teamcore, Amulya Yadav, Sara Mc Carthy, Matthew Brown, Debarun Kar, Shahrzad Gholami, Arunesh Sinha, Haifeng Xu, Ben Ford, Thanh Nguyen, Fei Fang, Eric Shieh, Bryan Wilder, Elizabeth Orrico, Elizabeth Bondi, Chao Zhang, Aida Rahmattalabi, Yundi Qian, Yasaman Abbasi, Omkar Thakoor, Han Ching Ou, Biswarup Bhattacharya, Donnabell Dmello, Aaron Ferber, Kai Wang, Subasree Sengupta, and Sarah Cooney. Lastly, I want to thank my many co-authors I have been fortunate enough to collaborate with and

learn a great deal from, Solomon Sonya, Dr. Chris Kiekintveld, Dr. Phebe Vayanos, Dr. Eugene Vorobeychik, Dr. Tran Thanh-Long, Dr. Ruta Mehta, Dr. Mina Guirguis, Noah Dunstatter, Darryl Balderas.

Finally, I would be remiss if I did not thank my rock during both the most difficult times of the Ph.D. and the best of times, Rosemary Feregrino. You have been a source of constant support and I could not imagine completing the Ph.D. without you. Your family has also been nothing but welcoming and warm to me. You helped make these last four years exciting and full of adventure.

Chapter 1

Introduction

Protecting an organization's cyber assets from intrusions and breaches due to attacks by malicious actors is an increasingly challenging and complex problem. This challenge is highlighted by several recent major breaches which have caused severe damage, such as the Equifax breach in 2017 and Yahoo in 2016 [1]. To protect from cyber breaches, companies and organizations employ the use of anti-virus softwares, Intrusion and Detection Systems (IDS), and Cyber Emergency Readiness Teams (CERT) composed of cyber analysts tasked with the general protection of an organization's network and cyber assets. Modern day cyber adversaries are persistent, targeted and sophisticated. This highlights the tremendous need for organizations protecting against such attacks to model these adversaries to optimize the application of the network defender's to protect targets and systems across the enterprise network. The Cyber Kill Chain [55, 8] encapsulates the necessary steps an adversary must complete to successfully breach the defender's enterprise

[1] 24th Air Force - AFCYBER: `http://www.24af.af.mil`; 688th Cyberspace Wing: `http://www.24af.af.mil/Units/688th-Cyberspace-Wing`

network.[2] Figure 1.1 shows the Cyber Kill Chain[3]. In the first phase of the Cyber Kill Chain the adversary spends a significant amount of time completing reconnaissance of the defender's enterprise network to learn about the vulnerabilities present and potential points of compromise. After recon, the adversary's next phases consist of weaponizing his exploit or malware, delivering it to the network through some medium and then exploiting a vulnerability in a system connected to the defender's network. To finish out his attack, the adversary's installs additional malware to ensure persistence and then establishes a command and control channel so he is able to act on his objectives, i.e., exfiltrate sensitive information, and complete his ultimate goal from breaching the network.

Figure 1.1: The Cyber Kill Chain developed by Lockheed Martin.

Stopping a cyber breach crucially depends on thwarting an adversary's attack during one of the phases occurring in the Cyber Kill Chain. To impede the adversary during the phases of his attack, network administrators use techniques such as the whitelisting

[2]The Cyber Kill Chain is a common framework for post analysis of a cyber breach. Recently, it was used to analyze the methods used by Russian actors targeting energy and other critical U.S. infrastructure sectors. `https://www.us-cert.gov/ncas/alerts/TA18-074A`

[3]Source: `https://www.eventtracker.com/tech-articles/siemphonic-cyber-kill-chain/`.

of applications, locking down permissions, and immediately patching vulnerabilities [42]. An interesting direction of research is the use of deception as a framework to improve cybersecurity defenses [4]. In particular, deception is extremely useful as a defense mechanism to impede an adversary during the recon phase of his attack. Criminals who target networks first map it out by using network scanning tools to ascertain network information through a suite of requests using tools such as NMap [53]. Deception is useful here as the adversary relies on the information received from network scanning tools to learn about the vulnerabilities he can exploit to compromise the defender's network. Instead of directly stopping an attack, deceptive techniques concentrate on diverting an adversary to attack non-critical systems or honeypots using deceptive views of the network state. Essentially, approaches for deception focus on making it difficult for the adversary to ascertain the true state of the network using network scanning tools like NMap. However, one drawback of most of these previous approaches is that they do not adequately model the adversarial nature of the cybersecurity domain.

After the recon phase, the adversary must complete phases 2 through 6 of his attack which include the delivery and exploitation phases of his attack. During these phases, automated intrusion detection and prevention systems (IDS) generate alerts for potentially malicious activity occurring on the defender's network where the generated alerts are aggregated into a central repository security information and event management tools (SIEM). To resolve the alerts generated by the IDS, human cybersecurity analysts on the CERT must investigate the alerts 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 [78]. Compounding this problem is the

fact that expert analysts are in short supply, so organizations face a key challenge in investigating and 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, IDS raised alarms, but they were missed in the deluge of alerts [69]. It is important to note these alerts give the defender the ability to stop the adversary during the later phases of the cyber kill chain, e.g., the delivery, exploitation, or installation phases, and prevents a network breach from occurring or catches the adversary earlier in his attack that reduces the damage from a network breach.

An overriding consideration throughout the Cyber Kill Chain is the strategic reasoning taking place between the network defender and a motivated adversary. Unfortunately, the defender is limited in her security resources and cannot protect all systems from an attack; further, the adversary could be conducting surveillance to learn about the defender's deceptive strategies and resolution strategies. A drawback of previous approaches to deception and the prioritization of cyber alert resolution is that they do not sufficiently consider the response of a strategic adversary. Failing to consider the actions of a strategic adversary can have detrimental effects on the effectiveness of the defender's deception and protection strategies as deterministic (non-randomized) strategies are open to exploitation by smart adversaries. Game theory provides a foundation for modeling the strategic interactions between two opposing parties. In this sense, my thesis looks to game theory as a foundation to improve cyber defense for enterprise networks and to provide another tool to the network defender to harness for protecting her network from constantly evolving cyber adversaries.

In recent years, the use of game theory has seen tremendous success in security domains for handling handling the complex scheduling and resource allocation problems of security resources. One model of interest for the cybersecurity problems studied in my thesis is the Threat Screening Game (TSG) model which was developed for the airport passenger screening domain. Airport passenger screening relates nicely to the problem of choosing which incoming alerts to screen with cyber analysts. However, it and other game theory models fail in three significant ways when being applied to cyber deception and alert prioritization. Namely, (1) previous game theory models consider an adversary who observes a mixed strategy from the defender, (2) they do not model defender resources with heterogenous screening times or attacks which present as probabilistic distributions over alert types, and (3) previous work assumes a game with zero-sum payoffs.

(a) Network Reconnaissance (b) Cyber Security Operations (c) TSA Screening

Figure 1.2: Domains of application for game theory in varying security domains and operations.

My thesis addresses these issues and advances the state of the art in the application of game theory to cybersecurity and the field of security games. The first major contribution of my thesis is the Cyber Deception Game (CDG) model which captures the interaction between defender and adversary during the reconnaissance phase of an enterprise network attack. The second major contribution of my thesis is the Cyber-alert Allocation Game

model which provides a game theoretic framework for prioritization of cyber alerts coming from IDS placed throughout the defender's network and tackles to second limitation of previous game theoretic models. The final major contribution of my thesis extends CAGs to domains with general-sum payoffs that is additionally applicable to physical security domains as it applies to TSGs as well.

1.1 Cyber Deception

Experienced attackers attempting to infiltrate a network spend a significant amount of time during the reconnaissance phase of their attack to find vulnerabilities throughout the network by mapping out the network through NMap scans, stealth SYN scans, TCP connections scans along with others [54, 42]. After gathering all of this information, the attacker then mounts their attack on a network. In the cyber domain, the network administrator has asymmetric information as she knows the true state of the network, i.e., properties of systems such as its operating system and applications running, and further, she can deceptively alter responses to network scans sent by an adversary [18, 2]. By hiding or lying about part of each system's configuration, the defender makes it significantly harder for the adversary to determine the true vulnerabilities present in systems on the network. Since exploits generally rely on specific vulnerabilities and versions of software, incorrectly identifying a system's software information decreases the likelihood of a successful attack and increases the amount of time it takes an adversary to compromise the defender's network. This type of interaction introduces an opportunity

for the defender to employ deceptive techniques at a network level to increase uncertainty during an adversary's reconnaissance activities.

The first contribution of my thesis concentrates on how the defender can maximize the benefit from deceiving cyber adversaries with a mix of true, false and obscure responses to network scans. To highlight the defender's advantage, consider a network with 1 system running Nginx and 2 running Tomcat. Suppose the adversary has a specific exploit for Nginx. The adversary uses nmap to determine the webserver for each system and deploy his exploit to the one running Nginx. However, if the defender can lie about the webserver, the adversary potentially has to deliver his exploit to all systems before infiltrating the network. This process increases the time spent by the adversary to infiltrate the network (which gives the defender time to mount a better defense) and increases the chances the defender identifies an ongoing attack. The main problem for the defender then is to determine how to alter the adversary's perception of the network to minimize her expected loss from an attack.

In this regard, I introduce a novel game theoretic model called the Cyber Deception Game (CDG) [72] which captures the interaction between the defender and adversary during the recon phase of network attacks and the cyber kill chain. The CDG model introduces a framework for a defender to deploy randomized deceptive strategies in a holistic fashion at a network wide level. I first consider a powerful adversary where I make a robust assumption about the information the adversary has to determine his optimal response to the defender's deceptive strategy In this domain, the adversary is assumed to view a static version of the network state, i.e., one observed configuration for each system on the network, which in turn causes the problem of computing the defender's optimal

strategy to be NP-hard. To alleviate these issues, I explore two separate algorithms for solving CDGs. First, I use a reformulation approach to convert the defender's non-linear optimization problem into a MILP. Second, I make use of a Bisection algorithm framework which solves a sequence of Mixed Integer Linear Programs (MILP) feasibility problems to obtain an ϵ-optimal approximate strategy for the defender. The second adversary model I explore in this research focuses on a naive adversary who is not aware of the deception and has a fixed set of preferences for systems (given an observed configuration) on the network. Extensive experimental evaluation is provided comparing the two approaches and the trade-off between the solving the optimal reformulated MILP and the bisection algorithm.

1.2 Cyber Threat Screening

Many approaches for mitigating the problem of sifting through the deluge of alerts generated focus on reducing the number of overall alerts. IDS can be carefully configured, alert thresholds can be tuned, and the classification methods underlying the detections can be improved [77, 13, 48]. Other techniques include aggregating alerts [91], and visualizing alerts for faster analysis [61]. Even when using all of these techniques, there is still a large volume of alerts which makes it infeasible for analysts to investigate all of them in depth. My 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 and reduce the likelihood that they will be detected. For example, if the defender had a policy that only inspects alerts from high valued assets in her network, an attacker who can learn this evades detection indefinitely by only attacking lower valued assets.

To address shortcomings of the previous methods for cyber alert allocation, I introduce the Cyber-alert Allocation Game (CAG) [73] model which provides a game theoretic framework for the defender to prioritizes and assign alerts for resolution raised from IDPS placed on systems and nodes across the network. 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 resolution strategies against strategic attackers. The CAG model considers the characteristics of the alerts (e.g., criticality of origin system), as well as the capabilities of the analysts in determining the optimal policy for the defender. I develop techniques to find the optimal allocation of alerts to analysts on the Computer Emergency Readiness Teams (CERT) in general CAGs and identify special cases where the computation becomes easy. The main algorithm takes advantage of a compact marginal representation of the defender's strategy space that leverages a special type of constraint structure called a bihierarchy which provides special conditions for when optimizing over the defender's marginal strategy space is equivalent to the mixed strategy space. Further heuristics are developed to achieve significant scale-up in CAGs without significant trade-offs in solution quality.

Both the CDG and CAG models assume games in which the payoffs are zero-sum, however, in many real world security domains this assumption may not hold. Hence, the

last contribution of my thesis extends the CAG to domains with general-sum payoffs which I show makes the computation of the defender's optimal strategy NP-hard [71]. This work has additional applicability to physical security domains (e.g., shipping container screening or airport passenger screening) as it extends the work done on Bayesian Threat Screening Games (TSGs) which previously assumed the payoffs to be general-sum as well. To deal with these issues, I provide the GATE which combines branch-and-bound search with Marginal Guided Algorithm, introduced in [22] to solve zero-sum TSGs, to optimally solve general-sum TSGs and additional heuristics which improve the scalability of GATE to real-world domains.

1.3 Thesis Outline

In Section 2, I discuss two relevant game theoretic models which inspire the development of the game theoretic models in this thesis. In Section 3, related works are discussed for background in security games and cybersecurity research. Next, Section 4 I introduce the Cyber Deception Game (CDG) model and discuss the associated algorithms for solving for optimal deceptive defender strategies. Section 5 introduces the Cyber-alert Allocation Game (CAG) model and presents algorithms for determining defender alert allocation strategies to her cyber analysts on the floor. In Section 6, I present the GATE algorithm which solves General-sum Bayesian Threat Screening Games. Finally, in Section 7 I discuss relevant future work for the use of deception in cybersecurity domains, future work for using the alert prioritization strategies in the real world and conclude my thesis.

Chapter 2

Background

2.1 Security Games

A security game [70, 27, 60, 81, 43] is a two-player game player between a defender and an adversary, where the defender protects a set of N targets from the attacker. The defender is assumed to have K security resources at her disposal to prevent an attack. A pure strategy for the defender is an assignment of the K resources to eiher targets or patrols (which can consist of multiple targets) while an adversary's pure strategy consists of choosing a target to attack. Denote the k^{th} defender pure strategy as P_k, which is an assignment of all security resources. P_k is represented as a column vector $P_k = [P_{ki}]^T$, where $P_{ki} = 1$ determines whether target i is protected by P_k. For instance, consider a security game with 3 targets and 2 resources, then $P_k = [1, 0, 1]$ represents the pure strategy where the defender protects targets 1 and 3 while 2 is left uncovered. Each target $i \in N$ has a corresponding set of payoffs $\{U_d^c, U_d^u, U_a^c, U_a^u\}$: If an adversary attacks target i and it is protected by a resource, then he receives utility U_a^c and the defender receives utility U_d^c. If target i is not protected then the adversary receives utility U_a^u

while the defender receives U_d^u. For security games, it is assumed that $U_a^c < U_d^c$ and $U_a^u > U_d^u$, which means the defender strictly benefits from covering a target more often with a resource while it is disadvantageous to the adversary. In security domains, it is often true that there are significantly more targets to protect than security resources available to protect them, i.e., $K < N$, and hence, it is of tremendous importance to the defender to carefully design a protection strategy.

A major assumption in most work on security games is that the interaction between the defender and adversary can be captured via the Stackelberg assumption, i.e., the defender commits to a strategy first. The adversary is then assumed to conduct surveillance and learns the defender's strategy before selecting his best response. This type of game is called the Stackelberg Security Game (SSG) where the standard solution concept is the Strong Stackelberg Equilibrium (SSE). For SSE, the defender is assumed to select an optimal strategy based on the assumption that an adversary selects his best response, breaking ties in favor of the defender. In an SSG, the defender's optimal strategy is generally a mixed (randomized) strategy $\mathbf{q}$, which is a distribution over pure strategies $\mathcal{P}$, as an adversary can typically exploit any deterministic (pure strategy) played by the defender. The mixed strategy is represented as a vector $\mathbf{q} = [q_k]^T$, where $q_k \in [0, 1]$ is the probability of choosing a pure strategy P_k and $\sum_k q_k = 1$. The defender's strategy can also be represented in a compact manner using a "marginal" representation. Let $\mathbf{n}$ be the marginal strategy, then $n_i = \sum_{P_k \in \mathcal{P}} q_k P_{ki}$ is the probability that target i is protected. Hence, work on SSG typically focus on solving for the defender's optimal marginal strategy $\mathbf{q}$ or the marginal strategy $\mathbf{n}$.

2.2 Threat Screening Games

A *threat screening game* (TSG) is a Stackelberg game played between the screener (leader) and an adversary (follower) in the presence of a set of non-player screenees that pass through a screening checkpoint operated by the screener. While TSGs are applicable to many domains, given that readers may be familiar with passenger screening at airports, I use examples from that domain. A TSG is composed of several features beyond the traditional SSG. A TSG is played over *Time windows* $w \in W$ that capture the temporal aspect of screening. The incoming passengers are group together by implicit characteristics, e.g., risk level and flight, into *screenee categories* $c \in C$. N_c gives the total number of screenees arriving in a category while N_c^w denotes the number of screenees in category c arriving in time window w. The adversary is assumed to have an *adversary type* $\theta \in \Theta$ that defines implicit characteristics of an adversary which cannot be chosen, e.g., TSA-assigned risk level. This in turn restricts the adversary to select from screenee categories $C_\theta \subset C$. The adversary is assumed to know his type, but the defender only knows a prior distribution $\mathbf{z}$ over the types. The adversary then chooses an *attack methods* $m \in M$ to attack with, e.g., on-body explosive. The defender has *resource types* $r \in R$ that can be used to screen at most L_r^w in a given time window. Each resource has effectiveness E_m^r which is the probability of detecting attack method m. These resources can be combined to *team types* $t \in T$ that are used to screen a passenger, e.g., walk-through metal detector *and* x-ray machine. Each team has effectiveness $E_m^t = 1 - \Pi_{r \in t}(1 - E_m^r)$ which defines the probability of detecting m.

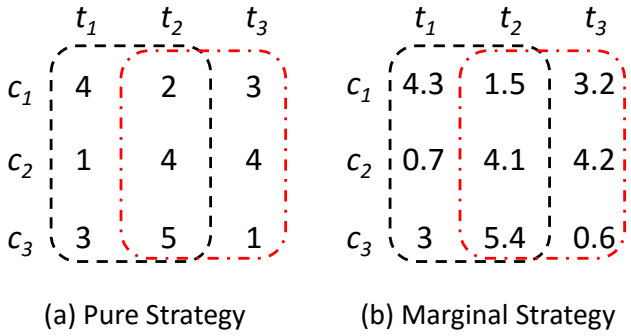


Figure 2.1: TSG Strategies

Pure strategy A pure strategy P for the screener can be represented by $|W|\times|C|\times|T|$ non-negative *integer-valued* numbers $P^w_{c,t}$, where each $P^w_{c,t}$ is the number of screenees in c assigned to be screened by team type t during time window w. Pure strategy P must assign every screenee to a team type while satisfying the resource type capacity constraints for each time window, via the following constraints: $\sum_{t\in T} I^t_r \sum_{c\in C} P^w_{c,t} \leq L^w_r \quad \forall w \in W, \forall r \in R$ and $\sum_{t\in T} P^w_{c,t} = N^w_c \quad \forall w \in W, \forall c \in C$ where I^t_r is an indicator function returning 1 if team type t contains resource type r and 0 otherwise. $\hat{P}$ denotes the set of all valid pure strategies and it is assumed $\hat{P} \neq \emptyset$, i.e., it is possible to assign every screenee to a team type. The pure strategies for the adversary types are denoted as $a^{\theta,w}_{c,m}$ which specifies that adversary type θ selects time window w, screenee category c, and attack method m.

For example, consider a game with one time window w_1 and two screening resources r_1, r_2 with capacity constraints $L_{r_1,r_2} = 20$, respectively. The resources are combined into three screening teams $t_1 = \{r_1\}$, $t_2 = \{r_1, r_2\}$ and $t_3 = \{r_3\}$. There are three categories c_1, c_2 and c_3 and each has $N_c = 9$ passengers. Figure 2.1(a) shows an example of a pure strategy allocation in this game.

Marginal strategy A marginal strategy $\mathbf{n}$ for the screener can be represented by $|W|\times|C|\times|T|$ non-negative *real-valued* numbers $n^w_{c,t}$, where $n^w_{c,t}$ is the number of screenees in c assigned to be screened by team type t during time window w. We would want this marginal strategy to be a valid mixed strategy (*implementable*), i.e., there should exist a probability distribution over $\hat{P}$ given by q_P (i.e., $\sum_{P\in\hat{P}} q_P = 1$, $0 \leq q_P \leq 1$) such that $n^w_{c,t} = \sum_P q_P P^w_{c,t}$. An example marginal screener strategy is shown in Figure 2.1(b) with the same game parameters described previously.

Utilities Since all screenees in category c are screened equally in expectation, we can interpret $n^w_{c,t}/N^w_c$ as the probability that a screenee in category c arriving during time window w will be screened by team type t. Then, the probability of detecting an adversary type in category c during time window w using attack method m is given by $x^w_{c,m} = \sum_t E^t_m n^w_{c,t}/N^w_c$. The payoffs for the screener are given in terms of whether adversary type θ chooses screenee category c and is either detected during screening, denoted as $U^d_{s,c}$, or is undetected during screening, denoted as $U^u_{s,c}$. Similarly, the payoffs for adversary type θ are given in terms of whether θ chooses screenee category c and is either detected during screening, denoted as $U^d_{\theta,c}$, or is undetected during screening, denoted as $U^u_{\theta,c}$. Given adversary type θ pure strategy $a^{\theta,w}_{c,m}$, the screener's expected utility is given by $U_s(a^{\theta,w}_{c,m}) = x^w_{c,m}U^d_{s,c} + (1 - x^w_{c,m})U^u_{s,c}$ and the expected utility for adversary type θ is given by $U_\theta(a^{\theta,w}_{c,m}) = x^w_{c,m}U^d_{\theta,c} + (1 - x^w_{c,m})U^u_{\theta,c}$.

Chapter 3

Related Work

3.1 Stackelberg Security Games

Stackelberg Security Games (SSG) have been well studied in previous literature [45, 50, 47, 34, 80, 41, 51, 46]. The early work in this area of research concentrated on providing the necessary theoretic and algorithmic foundation needed to solve general Stackelberg games and not on the problem of security. [82] is the first work to explore the concept of commitment to mixed (i.e., randomized) strategies in Stackelberg games. The earliest approach to solving general Stackelberg games termed Multiple LPs is introduced in [27] which shows that you can compute the optimal commitment by solving a single LP for each adversary action. The DOBSS algorithm presented in [60] improves on the Multiple LPs approach by showing the Strong Stackelberg Equilibrium (SSE) in Bayesian Stackelberg Games, i.e., where the leader may face multiple follower types, can be found using a single Mixed Integer Linear Program (MILP).

The formal Stackelberg Security Game model was introduced in [43] along with the ORIGAMI and ERASER algorithms for solving these games. ORIGAMI gives a polynomial time algorithm for solving security games without resource constraints, e.g., scheduling constraints in patrolling domains. ERASER, on the other hand, compactly represents the defender's strategy space which allows for significantly more scalability in security games with multiple resources. Additionally, ERASER handles resource constraints that are not considered in ORAGAMI. The first optimal approach to solving SSG with arbitrary resource constraints is ASPEN [37] which uses a branch-and-price framework that considers only the most relevant pure strategies to incrementally build up the defender's optimal mixed strategy. The HSBA algorithm [39] advanced the state-of-the-art in solving Bayesian SSG. HSBA breaks down the Bayesian game into smaller restricted games, i.e., games with a subset of the adversary types from the original problem, and solves the restricted games by combining an efficient branch and bound search with column generation. The solution information from these restricted games is then used to solve the original game more effectively.

Research on SSGs have led to several successfully deployed decision-support applications including ARMOR [64], IRIS [81], GUARDS [65], PROTECT [75], and TRUSTS [87]. These decision aids provide algorithms that generate strategies and patrols of defender security resources for the protection of physical targets such as airports, ports and metro systems. All of these works assume that an adversary observes a mixed strategy from the defender and assume the security resources are 100% effective at protecting a target if allocated to it. This is in contrast to the work in this thesis that accounts

for resources that are not 100% effective and an adversary who observes a pure strategy from the defender, which are crucial features of the cyber domain.

The problem of threat screening has been explored extensively in literature. The Threat Screening Game (TSG) model was introduced in [22] and looks to SSG for inspiration to devise a game theoretic model for the threat screening domain. As [22] points out, the TSG approach provides a significant improvement on prior non-game-theoretic models in domains such as, screening for shipping containers [6], stadium patrons [68], and airport passengers [57, 58] or simple game-based models such as [83]. Specifically, previous non-game-theoretic approaches fail to model a rational adversary who aims to take advantage of vulnerabilities in the screening strategies whereas TSGs take this into account when devising screening strategies. Furthermore, previous solution methods for solving security games fail to apply directly to airport passenger screening. This happens because TSGs (i) include a group of non-player screenees that all must be screened while a single adversary tries to pass through screening undetected, (ii) model screening resources with different efficacies and capacities that can be combined to work in teams and (iii) do not have an explicitly modeled set of targets. These are fundamental differences from traditional security games [43, 64, 75, 81] where the defender protects a set of targets against a single adversary. [22] provides MGA to solve for the defender's optimal screening strategy in Bayesian zero-sum TSGs, but these techniques fail to apply to TSGs with non-symmetric (i.e., general-sum) payoffs. HSBA represents an interesting approach for solving Bayesian general-sum TSGs, but as shown in [22] column generation is not a scalable approach for solving TSGs. Hence, novel algorithms are needed to tackle the challenges of scaling up solution methods to Bayesian general-sum TSGs.

3.2 Cyber-alert Allocation

Intrusion detection has been studied for over three decades, beginning with the early work of [7, 29]. A major concentration of research on intrusion detection has focused on developing automated techniques (e.g., machine learning) for identifying malicious activity [36, 35, 84, 30, 79]. However, these methods have significant detection error and suffer from generating a plethora of alerts for the defender to investigate [77]. As the volume of alerts increased significantly due to Intrusion and Detection systems across a network, later research [12, 77] focused on reducing the number of false positive alerts by developing automated alert reduction techniques. In this regard, there are both open source [21] and commercially available [91] Security and Event Management (SIEM) tools that take raw data input from sensors, aggregate and correlate them, and provides a central repository of alerts from the enterprise network that can then be assigned for investigation by the cyber security analysts. Most modern cybersecurity operations of large organizations house a team of human cyber analysts (e.g., Computer Emergency Response Team (CERT)[1]) who are tasked with investigating these alerts generated by the automated detectors [28]. A recent line of work [33] uses decision theory to optimize the scheduling of cyber-security analysts over 2 week periods to minimize the risk associated with investigating and needing to remediate a critical attack, but this approach does not consider the response of a strategic attacker. In contrast with this previous work, my thesis focuses on the problem of how to assign the alerts to analysts which are currently on

[1] As an example the United States Air Force employs the use of 688th Cyberspace Wing to protect Department of Defense networks and systems from priority threats. `http://www.afcyber.af.mil/About-Us/688th-Cyberspace-Wing/`

the operations floor given an adversary attempting to exploit weaknesses in the defender's allocation strategy.

My approach for cyber alert allocation draws on the principles and modeling techniques of the large body of work that applies game theory to security problems [80]. The existing work on security games focuses heavily on applications to physical security (e.g., patrolling), with some exceptions (e.g., [48, 31]). However, Cyber-alert Allocation Games (CAG) significantly differs from SSG due to the absence of an explicit set of targets, a large number of benign alerts and varying time requirements for inspections. TSGs relate nicely to CAGs, but 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" [45] 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 cybersecurity attacks result in a distribution of potential alert types.

3.3 Cyber Deception

The use of game theory has been studied before in the context cybersecurity problems [52, 5, 74, 49, 73]. [44, 63, 32, 31] study a honeypot selection game where the defender chooses the properties of the network, a sequel where the adversary can probe machines to ascertain the true state and a final version which models the adversary's actions as attack graphs. [24] studies a signaling game where the defender signals to an adversary if a system is either real or a honeypot when the adversary performs a scan. [62] extends the

signaling game to account for an adversary who can gain evidence about the true state of a system. In my work, I consider a game scenario in which the defender determines the optimal way to respond to scans sent by a potential adversary given a set of possible responses. Further, I explore different types of adversaries with varying awareness of deception.

Deception has also been widely studied as a means to improve the protection of enterprise networks from potential hackers and intruders [3, 1]. [2] uses a graph theoretic approach to confuse a potential attacker by manipulating his view of systems on the network. However, this work focuses on finding a view which is measurably different from the true state and does not adequately model the response of a strategic adversary. [40] is closely related to my work on deception. The authors study how to respond to an attacker's scan queries using an annotated probabilistic logic model. My research provides a complimentary view using game theory to determine how a defender manipulates scan responses to confuse an attacker's view of systems on the network. I also study varying adversary models, which can have significant impact on the defender's optimal strategy which is not explored in [40].

Chapter 4

Cyber Deception Games

4.1 Problem Domain

The first stage of the cyber kill chain, the reconnaissance phase, is one of the most critical stages for an adversary attempting to breach an enterprise network, and hence, a critical phase in which the defender can protect against network intrusions. . Motivated adversaries spend a significant amount of time completing reconnaissance on the enterprise network to learn about system vulnerabilities and the best points of compromise in the defender's network. Reconnaissance is completed through a suite of scan requests, such as Stealth SYN scans, OS scans, and services scans by using network scanning tools such as NMap. Importantly, the type of interaction between the defender and an adversary during the recon phase provides an opportunity for the defender to deceive a cyber adversary by altering and lying about the system information which is returned from from scanning activity.

This chapter explores a game theoretic model, the Cyber Deception Game (CDG), which provides a framework for the defender to deceptively set scan responses for systems

across the defender's enterprise network to deceive potential cyber adversaries. Two adversary models are explored that represent the various types of real-world adversaries that can be encountered in the network security domain. The first is a powerful adversary who is assumed to have a robust amount of information about the defender's deceptive strategy, e.g., nation-state level adversaries who may have access to insider information. The second is a naive adversary assumed to not be aware of the deception and who has a fixed set of preferences over observed systems on the network, e.g., script-kiddies who are not as advanced. I show that computing the optimal strategy for the network defender is NP-hard due when facing a powerful adversary due to either masking or cost constraints imposed on the defender's deceptive response strategy space. Additionally, I show computing the optimal strategy against the naive adversary to be NP-hard as well when both the masking and cost constraints are imposed on the defender. I develop three solution approaches to find the defender's optimal deceptive response scheme against a powerful adversary: (1) a Mixed Integer Linear Programming (MILP) approach, (2) a bisection algorithmic approach, and approaches to find the defender's optimal strategy against naive adversaries and (3) a greedy heuristic approach to quickly generate defender deceptive strategies. I also present experimental results showing the scalability of the algorithms to larger scale CDGs. Finally, technical deceptive techniques and approaches are discussed which could leverage the use of the CDG model to generate deceptive network views.

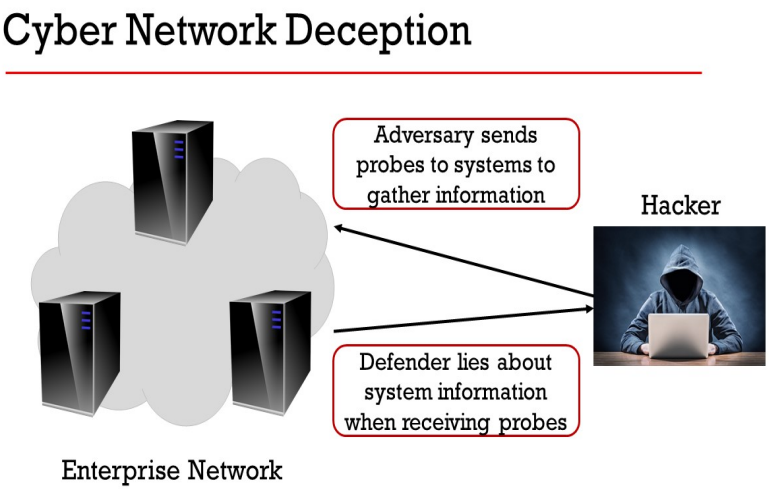


Figure 4.1: The network reconnaissance domain in which an adversary scans the defender's enterprise network and the defender deceptively alters the system's responses.

4.2 Cyber Deception Game

The *Cyber Deception Game* (CDG) is a zero-sum Stackelberg game between the defender (e.g., network administrator) and an adversary (e.g., hacker). The defender moves first and chooses how the systems should respond to scan queries from an adversary, and the adversary subsequently moves by choosing a system to attack based on the responses. Despite the similarities with game-theoretic models in security domains, such as [80, 15, 16], there are two key differences. First, the defender can only commit to a pure strategy and not an arbitrary mixed strategy. This is because, in these domains, network administrators modify the network very infrequently, and thus, the attackers' view of the network is static. Second, there are no explicit security resources for the defender in CDGs. Consequently, the existing approaches for solving standard Stackelberg games in

security domains, cannot be directly applied. The various components of the game and the aforementioned model characteristics are described in detail as follows:

Systems and True Configurations

The defender aims to protect a set K of systems, from possible exploits and intrusions. Each system has certain attributes, e.g., an operating system, an anti-virus protection mechanism, services hosted, etc. These attributes altogether constitute the *true configuration* (TC) of the system. The set of all possible TCs is denoted by F. Each system has an associated utility, which captures how much the adversary would get by attacking it. This utility solely depends on the TC of the system — each $f \in F$ induces a utility denoted by U_f to any system that is assigned f. U_f can be negative if the security level of the system is so high that the attacker's efforts end in vain or the attacker gets fake data from a seemingly successful attack, leading to a loss in the end. It follows that, the *true state of the network* (TSN) can be represented as a vector $N = (N_f)_{f \in F}$, where $N_f \in \mathbb{Z}_{>0}$ denotes the number of systems on the network which have a TC f and $\sum_{f \in F} N_f = |K|$ (Assume $N_f \neq 0$, since such a TC simply need not be considered).

Observed Configurations

The adversary attempts to gain information about every system on the network, via probes and scans. By scanning a system, the adversary observes certain attributes, which constitute the *observable configuration* (OC) of the system. The set of possible OCs is denoted by $\tilde{F}$. It is assumed that it is possible for the defender to make some of the observable attributes of a system appear different than what they truly are (e.g., altering

the TCP/IP stack of a system, spoofing a running service on a port). By means of such alterations at her disposal, the defender controls the OC an attacker sees when probing a system. Note that it may not be possible for an arbitrary TC f to be made to appear as an arbitrary OC $\tilde{f} \in \tilde{F}$ — such a constraint is called a *feasibility* constraint, and these are denoted by a (0,1)-matrix π. Iff $\pi_{f,\tilde{f}} = 1$, then f can be *covered*, or *masked* with $\tilde{f}$. The set of OCs which can mask a TC f, is denoted by $\tilde{F}_f = \{\tilde{f} \in \tilde{F} \mid \pi_{f,\tilde{f}} = 1\}$, and similarly, the set of TCs which can be masked by an OC $\tilde{f}$, by $F_{\tilde{f}} = \{f \in F \mid \pi_{f,\tilde{f}} = 1\}$.

From the adversary's perspective, two systems having the same $\tilde{f}$ as their OC are indistinguishable, and hence, his *observed state of the network* (OSN) can be represented as a vector $\tilde{N} = (\tilde{N}_{\tilde{f}})_{\tilde{f} \in \tilde{F}}$ where $\tilde{N}_{\tilde{f}} \in \mathbb{Z}_{\geq 0}$ denotes the number of systems which have an OC $\tilde{f}$. As is the case with the TSN N, we must have $\sum_{\tilde{f} \in \tilde{F}} \tilde{N}_{\tilde{f}} = |K|$.

It is assumed that masking a TC f with an OC $\tilde{f}$, has a cost of $c(f, \tilde{f})$ incurred by the defender, which typically captures the monetary costs for deploying network modifications necessary for such a deception. It is also useful to note that although honeypots are not explicitly discussed in the model they can represented with a true configuration f_{honey} and an observable $\tilde{f}_{honey}$. Systems on the network which do not appear for a specific system can also be modeled with an observable configuration $\tilde{f}_{hidden}$ which does not have a utility, and further, any systems masked with $\tilde{f}_{hidden}$ do not appear in the optimization problem formulated in Section 4.3.2.

Defender Strategies

Naturally, F, $\tilde{F}$, π, c and N are known to the defender. Given all this information, the defender must decide her strategy — for each TC f, she must decide how many of the N_f

systems having TC f, should be assigned the OC $\tilde{f}$, where $\tilde{f} \in \tilde{F}_f$. Thus, any possible strategy can be represented as a $|F| \times |\tilde{F}|$ matrix ϕ having non-negative integer entries, with $\phi_{f,\tilde{f}}$ representing the number of systems having TC f and OC $\tilde{f}$. Hence, ϕ must satisfy

$$\phi_{f,\tilde{f}} \in \mathbb{Z}_{\geq 0} \qquad \forall f \in F, \forall \tilde{f} \in \tilde{F} \tag{4.1}$$

Since the TSN N is fixed, ϕ must also satisfy

$$\sum_{\tilde{f} \in \tilde{F}} \phi_{f,\tilde{f}} = N_f \qquad \forall f \in F \tag{4.2}$$

Since feasibility constraints π are specified, ϕ must also satisfy

$$\phi_{f,\tilde{f}} \leq \pi_{f,\tilde{f}} N_f \qquad \forall f \in F,\ \forall \tilde{f} \in \tilde{F} \tag{4.3}$$

Finally, since setting any OC on a system has an associated cost, the defender's total cost cannot exceed a limit B, which is called the budget constraint. Formally, ϕ must also satisfy

$$\sum_{f \in F} \sum_{\tilde{f} \in \tilde{F}} \phi_{f,\tilde{f}}\ c(f, \tilde{f}) \leq B \tag{4.4}$$

The set of strategies ϕ which satisfy the constraints (4.1), (4.2), (4.3), and (4.4), is denoted by Φ.[1] When the defender plays $\phi \in \Phi$, the resulting OSN $\tilde{N}$ is given by $\tilde{N}_{\tilde{f}} = \sum_{f \in F} \phi_{f,\tilde{f}}\ \forall \tilde{f} \in \tilde{F}$.

[1]The feasibility constraints can simply be captured via the budget constraint by setting the costs of infeasible assignments to be higher than the budget. However, they are essential in the model, since, in some cases, having no budget constraint allows an efficient solution to the problem (e.g. Section 5), while still having the very practical feasibility constraints keeps the problem non-trivial.

Adversary Strategies

Depending on the defender's strategy, the adversary observes $\tilde{N}$ as described above. All systems having the same OC $\tilde{f}$ are indistinguishable to the adversary, and hence, he must be indifferent between all such $\tilde{N}_{\tilde{f}}$ systems when deciding which system to attack. As a result, the adversary is assumed to choose the OC $\tilde{f}$ which gives him the highest expected utility (described momentarily), and attack all the $\tilde{N}_{\tilde{f}}$ systems having this OC with an equal probability. In short, we say "the adversary attacks an OC $\tilde{f}$" to mean he attacks all the systems having OC $\tilde{f}$ with an equal probability. A general mixed strategy for the adversary is to attack the set of OCs with any probability distribution. However, since there always exists a pure best-response strategy in any game, it suffices to consider the adversary's strategies as simply attacking a particular $\tilde{f}$.

Utilities

When the defender plays a strategy ϕ, the adversary's expected utility on attacking an OC $\tilde{f}$ with $\tilde{N}_{\tilde{f}} > 0$, denoted by $\bar{U}_{\tilde{f}}(\phi)$ — or, as $\bar{U}_{\tilde{f}}$ for simplicity, when the underlying ϕ is unambiguously understood — is given by

$$\tilde{U}_{\tilde{f}} = E[U_f|\phi, \tilde{f}] = \sum_{f \in F_{\tilde{f}}} P(f|\phi, \tilde{f}) U_f = \sum_{f \in F} \frac{\phi_{f,\tilde{f}}}{\tilde{N}_{\tilde{f}}} U_f \tag{4.5}$$

(4.5) follows from computing $P(f|\phi, \tilde{f})$ using the fact that out of $\tilde{N}_{\tilde{f}}$ systems having an OC $\tilde{f}$, $\phi_{f,\tilde{f}}$ have a TC f. Since the game is zero-sum, the defender's expected utility is $-\tilde{U}_{\tilde{f}}$ when $\tilde{f}$ is attacked. Note the attacker cannot attack an OC $\tilde{f}$ with $\tilde{N}_{\tilde{f}} = 0$, or equivalently, his expected utility is $-\infty$ if he does so.

Next, the model is illustrated using a simple example.

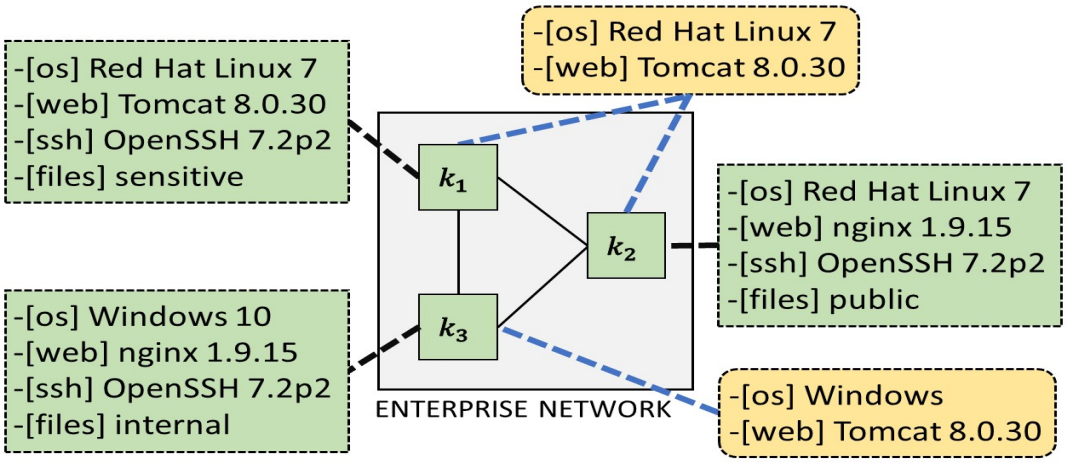


Figure 4.2: Simple example of an enterprise network.

Example Game 1: Figure 4.2 shows a simple example enterprise network which will be used as a running example. We have a set of systems $K = \{k_1, k_2, k_3\}$, set of TCs $F = \{f_1, f_2, f_3\}$ (shown in Figure 4.2 as the green boxes) and set of OCs $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2\}$ (shown in Figure 4.2 as the yellow boxes). Let the feasibility constraints be given by the sets $F_{\tilde{f}_1} = \{f_1, f_2\}$ and $F_{\tilde{f}_2} = \{f_2, f_3\}$. The TCs are as follows:

$$f_1 = [\text{[os] L, [web] T, [ssh] O, [files] S}]$$

$$f_2 = [\text{[os] L, [web] N, [ssh] O, [files] P}]$$

$$f_3 = [\text{[os] W, [web] N, [ssh] O, [files] I}]$$

For the TCs, the utilities are $U_{f_1} = 10$, $U_{f_2} = 0$, and $U_{f_3} = 6$. The OCs are as follows:

$$\tilde{f}_1 = [\text{[os] L, [web] T}] \qquad \tilde{f}_2 = [\text{[os] W, [web] T}]$$

For simplicity, let all the costs $c(f, \tilde{f})$ to be 0, so that there is essentially no budget constraint. Based on the TCs assigned as shown, the state of the network $(N_f)_{f \in F}$ is $(1, 1, 1)$. When the defender assigns OCs as shown in Figure 4.2, her strategy ϕ is given by

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cc} \tilde{f}_1 & \tilde{f}_2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

The expected utility of the adversary (loss of the defender) when he attacks $\tilde{f}_1$ or $\tilde{f}_2$ is respectively given by $\tilde{U}_{\tilde{f}_1} = (10 + 0)/2 = 5$ and $\tilde{U}_{\tilde{f}_2} = 6/1 = 6$. Thus, attacking $\tilde{f}_2$ leads to the highest expected utility for the attacker.

Adversary Knowledge and Utility Estimation

The attacker's awareness of the deception and the understanding of the defender's strategy may vary. Note that if the adversary is always able to find the OC with highest expected utility, it is the worst case scenario for the defender given the game is zero-sum. An attacker who is fully aware of how the defender sends the false responses to scan requests (via insider threats, information leakage, etc.) would have such an ability. Formally, a *powerful* attacker is defined as someone who knows F, $\tilde{F}$, π, U and ϕ and chooses to attack the OC with the (correct) highest expected utility $\tilde{U}_{\tilde{f}}$ computed through Equation (4.5). If the defender chooses a strategy that minimizes the expected utility of a powerful attacker, she gets a robust strategy as the defender can be assured that no matter the

extent of the adversary's knowledge, no strategy he plays can lead to a greater loss for the defender, in alignment with the minimax principle.

However, the attacker may not be so powerful. On the other end of the spectrum, if the attacker is unaware of the defender's precise deception scheme or has a very limited understanding of the situation such that he cannot make any meaningful inference, his decision making would be completely dependent on the observed configurations of the systems and some fixed preferences over OCs in terms of the estimated expected utility. Formally, a naive attacker is defined to be someone who chooses to attack an existing OC $\tilde{f}$ (i.e., one which has at least one system configured to it) with the highest $\bar{U}_{\tilde{f}}$ where $\bar{U}_{\tilde{f}}$ is not dependent on the defender's strategy and is known to the defender. This is also equivalent to the case where the attacker just has a fixed preference of the OCs. CDGs with powerful attackers are analyzed in Section 4.3, and CDGs with naive attackers in Section 4.4.

4.3 Optimal Defender Strategy against Powerful Adversary

In this section, it is shown how to compute the defender's optimal strategy in a CDG assuming a powerful adversary. The adversary attacks an OC from the set $\arg\max_{\tilde{f}\in\tilde{F}} \tilde{U}_{\tilde{f}}$ and gets an expected utility of $\max_{\tilde{f}\in\tilde{F}} \tilde{U}_{\tilde{f}}$, denoted in short as $\tilde{U}^*(\phi)$, where the negative value is the defender's expected loss. Hence, the defender aims to minimize her loss by choosing her ϕ from the set $\arg\min_{\phi\in\Phi} \tilde{U}^*(\phi)$.

4.3.1 Computational Complexity

The problem of finding optimal defender strategy against a powerful adversary in a CDG is called *CDG-Robust*.

First, it is useful to investigate a special case. The following proposition provides a tight lower bound on $\min_{\phi \in \Phi} \tilde{U}^*(\phi)$.

Lemma 4.1 *The expected loss of the defender when playing her optimal strategy, is no lower than the average utility of the systems, i.e.,*

$$\min_{\phi} \tilde{U}^*(\phi) \geq U^{Ave}(K) = \frac{\sum_{f \in F} N_f U_f}{|K|}$$

Proof 4.1 *Equivalently, it can be shown that,* $\tilde{U}^*(\phi) \geq \frac{\sum_{f \in F} N_f U_f}{|K|}$ *for all* ϕ*. Fix any* $\phi \in \Phi$*. We then have,*

$$\begin{aligned}
\tilde{U}^*(\phi) &\geq \tilde{U}_{\tilde{f}}(\phi) \quad \forall \tilde{f} && \text{(by definition of } \tilde{U}^*(\phi)) \\
\therefore \quad \sum_{\tilde{f} \in \tilde{F}} N_{\tilde{f}} \tilde{U}^*(\phi) &\geq \sum_{\tilde{f} \in \tilde{F}} N_{\tilde{f}} \tilde{U}_{\tilde{f}} \\
\therefore \quad |K| \cdot \tilde{U}^*(\phi) &\geq \sum_{\tilde{f} \in \tilde{F}} \sum_{f \in F} \phi_{f,\tilde{f}} U_f && \text{(using (4.5))} \\
&= \sum_{f \in F} \left(U_f \sum_{\tilde{f} \in \tilde{F}} \phi_{f,\tilde{f}} \right) && \text{(re-ordering terms)} \\
&= \sum_{f \in F} U_f N_f && \text{(by definition of } \phi, N_f) \\
\therefore \quad \tilde{U}^*(\phi) &\geq \frac{\sum_{f \in F} N_f U_f}{|K|}
\end{aligned}$$

Since the choice of ϕ *was arbitrary, the claim follows.*

Thus, even when the defender plays her optimal strategy, the attacker's expected utility is at least $U^{\text{Ave}}(K)$. Consequently, if the inequality becomes tight for a strategy ϕ, it must be an optimal strategy. It is easy to see that the bound becomes tight if and only if $\tilde{U}^*(\phi) = \tilde{U}_{\tilde{f}}(\phi)$, $\forall \tilde{f}$. Clearly, this is true if and only if $\bar{U}_{\tilde{f}}$ is the same for each $\tilde{f}$ set on any system, and trivially so, if only a single OC is set on all the systems. Thus,

Corollary 4.1 *If it is feasible for the defender to set the same OC on all the systems making them all indistinguishable to the adversary, doing so is an optimal strategy. Formally, if* $\exists \tilde{f}^*$ *s.t.* $\exists \phi^* \in \Phi$ *where* $\phi^*_{f,\tilde{f}^*} = N_f, \forall f$, *then* $\phi^* \in \arg\min_{\phi \in \Phi} \tilde{U}^*(\phi)$.

It is possible to efficiently check if such an OC exists, by enumeration. However, it may not exist, and next it is shown that *CDG-Robust* is NP-hard in general.

Proposition 4.1 *CDG-Robust is NP-hard.*

Proof 4.2 *The result is proven via a reduction from the Partition problem (PART) which is known to be NP-complete. Given a multiset* S *of* n *positive integers that sum up to* $2r$, *PART is the decision problem to determine if* S *can be partitioned into two subsets* S_1 *and* S_2 *such that the sum of integers in* S_1 *and* S_2 *are each* r. *It can be reduced to CDG-Robust as follows.*

Let the input to PART be a set of integers $S = \{s_1, \ldots, s_n\}$ *whose elements sum to* $2r$. *To construct a CDG, let the set of TCs be* $F = \{f_1, \ldots, f_n\} \cup \{f_{n+1}, f_{n+2}\}$, *with utilities* $U_{f_i} = s_i$ *for each* $i \in \{1, \ldots, n\}$ *and* $U_{f_{n+1}} = U_{f_{n+2}} = -r$. *Next, let there be* $n + 2$ *systems, each having a different TC. Let the set of OCs be* $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2\}$, *with* $\tilde{F}_{f_i} = \tilde{F}$ *for each* $i \in \{1, \ldots, n\}$, *and* $\tilde{F}_{f_{n+1}} = \{\tilde{f}_1\}$, $\tilde{F}_{f_{n+2}} = \{\tilde{f}_2\}$. *Let all the costs be* 0 *so*

that the budget constraint can be ignored. Assuming the adversary to be powerful, these components completely define a CDG-Robust problem.

Note that, by Corollary 4.1 and the fact that $\sum_f U_f = 0$*, the optimal strategy* ϕ *must have* $\tilde{U}^*(\phi) \geq 0$*. Now, suppose* S *can be partitioned into subsets* S_1 *and* S_2 *such that the numbers in each sum to* r*. Then, consider the strategy* ϕ *which masks the TCs in* $\{f_i | s_i \in S_1\}$ *and* f_{n+1} *with* $\tilde{f}_1$*, and masks the TCs in* $\{f_i | s_i \in S_2\}$ *and* f_{n+2} *with* $\tilde{f}_2$*. It is easy to check that* $\tilde{U}_{\tilde{f}_1}(\phi) = \tilde{U}_{\tilde{f}_2}(\phi) = 0 = \tilde{U}^*(\phi)$*, making* ϕ *an optimal strategy. On the other hand, suppose the defender's optimal* ϕ *yields* $\tilde{U}^*(\phi) = 0$*. Since* $\tilde{f}_1$ *must mask* f_{n+1}*, and* $\tilde{f}_2$ *must mask* f_{n+2}*, neither of the OCs are unused. Since* $\tilde{U}^*(\phi) = 0$*, w.l.o.g., assume* $\tilde{U}_{\tilde{f}_1} = 0$*. Hence, the sum of utilities of the TCs masked with* $\tilde{f}_1$ *must be 0. Therefore, the sum of utilities of TCs masked by* $\tilde{f}_2$ *is also 0. Then,* $S_1 = \{s_i | \phi_{f_i, \tilde{f}_1} = 1\}$*, and* $S_2 = \{s_i | \phi_{f_i, \tilde{f}_2} = 1\}$ *form a partition of* S*, each having sum of the elements* r*. It follows that,* $PART$ *should output* YES *iff* CDG*-Robust finds an optimal strategy* ϕ *with* $\tilde{U}^*(\phi) = 0$*. This reduction, being polynomial-time, proves the claim.*

4.3.2 The Defender's Optimization Problem

The defender's optimal strategy ϕ can be computed by solving the optimization problem given below.

$$\min_{u,\phi} \quad u \tag{4.6a}$$

$$\text{s.t.} \quad u \sum_{f \in F} \phi_{f,\tilde{f}} \geq \sum_{f \in F} \phi_{f,\tilde{f}} U_f \qquad \forall \tilde{f} \in \tilde{f} \tag{4.6b}$$

$$\text{Constraints (4.1)} \sim \text{(4.4)}$$

The objective function in Equation (4.6a) minimizes the utility u the adversary receives for the game. Equation (4.6b) enforces that the adversary chooses a best response to the defender's strategy ϕ, where the expected utility for attacking a given $\tilde{f}$ is given by (4.5). Constraints (4.1)$\sim$(4.4) represent a feasible defender strategy.

This optimization problem is non-convex due to constraint (4.6b), which can be linearized, to convert the optimization problem to a MILP as follows. First, an alternate representation is devised for the defender's strategy ϕ, as a $|K| \times |\tilde{F}|$ (0,1)-matrix σ, where $\sigma_{k,\tilde{f}} = 1$ denotes system k is masked with $\tilde{f}$. Further, the TSN N is represented via a vector $\mathbf{x}$, where $x_k \in F$ represents the TC for system k. Then, for each TC f, we have $N_f = |K_f|$ where, $K_f = \{k \in K \mid x_k = F\}$, and $\phi_{f,\tilde{f}} = \sum_{k \in K_f} \sigma_{k,\tilde{f}} \ \forall f, \forall \tilde{f}$. Hence, the alternate representations are indeed equivalent. Then, constraints equivalent to (4.1)$\sim$(4.4) can be easily formulated for σ and x with an additional constraint $\sum_{\tilde{f} \in \tilde{F}} \sigma_{k,\tilde{f}} = 1 \ \ \forall k \in K$ to ensure feasibility. More importantly, equation (4.6b) can be reformulated as

$$u \sum_{k \in K} \sigma_{k,\tilde{f}} \geq \sum_{k \in K} \sigma_{k,\tilde{f}} U_{x_k} \quad \forall \tilde{f} \in \tilde{F} \tag{4.7}$$

The left-hand side of (4.7) can be seen as the sum of a set of terms $u\sigma_{k,\tilde{f}}$, each of which is the product of binary variable $\sigma_{k,\tilde{f}}$ and the continuous variable u. Such an expression can be linearized by introducing variables $z_{k,\tilde{f}}$ for each $k \in K$ and $\tilde{f} \in \tilde{F}$, and enforcing $z_{k,\tilde{f}} = u\sigma_{k,\tilde{f}}$. Consequently, (4.7) can be rewritten as:

$$\sum_{k \in K} z_{k,\tilde{f}} \geq \sum_{k \in K} \sigma_{k,\tilde{f}} U_{x_k} \tag{4.8}$$

To enforce $z_{k,\tilde{f}} = u\sigma_{k,\tilde{f}}$, consider $u \in [U^{min}, U^{max}]$ where $U^{min} = \min_{f \in F} U_f$ and $U^{max} = \max_{f \in F} U_f$. With these bounds on u, we then include the constraints for each z variable in the optimization problem as follows:

$$U^{min}\sigma_{k,\tilde{f}} \leq z_{k,\tilde{f}} \leq U^{max}\sigma_{k,\tilde{f}} \tag{4.9}$$

$$u - (1 - \sigma_{k,\tilde{f}})U^{max} \leq z_{k,\tilde{f}} \leq u - (1 - \sigma_{k,\tilde{f}})U^{min} \tag{4.10}$$

After this conversion the optimization problem becomes a MILP. The complete formulation is given below for clarity.

$$\min_{u,\sigma,z} \quad u \tag{4.11a}$$

$$\text{s.t.} \quad \sum_{k\in K} z_{k,\tilde{f}} \geq \sum_{k\in K} \sigma_{k,\tilde{f}} U_{x_k} \qquad \forall \tilde{f} \in \tilde{f} \tag{4.11b}$$

$$\sum_{\tilde{f}\in\tilde{F}} \sigma_{k,\tilde{f}} = 1 \qquad \forall k \in K \tag{4.11c}$$

$$\sigma_{k,\tilde{f}} \leq \pi_{x_k,\tilde{f}} \qquad \forall k \in F, \forall \tilde{f} \in \tilde{F} \tag{4.11d}$$

$$\sum_{\tilde{f}\in\tilde{F}} \sum_{k\in K} \sigma_{k,\tilde{f}} c(x_k, \tilde{f}) \leq B \tag{4.11e}$$

$$U^{min}\sigma_{k,\tilde{f}} \leq z_{k,\tilde{f}} \leq U^{max}\sigma_{k,\tilde{f}} \qquad \forall k \in F, \forall \tilde{f} \in \tilde{F} \tag{4.11f}$$

$$u - (1 - \sigma_{k,\tilde{f}})U^{max} \leq z_{k,\tilde{f}} \qquad \forall k \in F, \forall \tilde{f} \in \tilde{F} \tag{4.11g}$$

$$z_{k,\tilde{f}} \leq u - (1 - \sigma_{k,\tilde{f}})U^{min} \qquad \forall k \in F, \forall \tilde{f} \in \tilde{F} \tag{4.11h}$$

$$\sigma_{k,\tilde{f}} \in \{0, 1\} \qquad \forall k \in F, \forall \tilde{f} \in \tilde{F} \tag{4.11i}$$

4.3.3 MILP Bisection Algorithm

The reformulated MILP presented requires the addition of $|K||\tilde{F}|$ variables and $4|K||\tilde{F}|$ constraints to solve for ϕ. This conversion significantly increases the size of the optimization problem from the original number of $|F||\tilde{F}|$ decision variables in the original optimization problem and can create issues when solving larger CDG instances. The second approach develop for CDGs does not require the reformulation and instead solves a sequence of smaller MILPs (same size as (4.6a)) to find an ϵ-approximate solution for the defender [89]. This is done via a bisection algorithmic framework. The algorithm initially is given an interval that the optimal objective value $\tilde{U}^*$ lies in which for CDGs

is $U^* \in [U^{LB}, U^{UB}]$ where $U^{LB} = U^{AVE}(K)$ and $U^{UB} = \max_{f \in F} U_f$. For the algorithm, two variables $l = U^{LB}$ and $d = U^{UB}$ are introduced with the initial width as $\epsilon_0 = d - l$ that contains the optimal value U^* of the optimization problem. The main loop of the algorithm is repeated until the width $d - l \leq \epsilon$. The main loop has the following two steps:

1. Take $\tau = (u + l)/2$ and solve the feasibility problem in Equation (4.12a) to find if there exists a solution $\mathbf{n}$ that satisfies the constraints.

2. **if** *feasible*, take $u := \tau$; **if** *not feasible*, take $l := \tau$.

The algorithm is guaranteed to converge as after each update the interval $[l, u]$ contains the optimal U^* and the width is halved. The number of steps that are needed to find the ϵ-approximate optimal solution is $\lceil \log \frac{\epsilon_0}{\epsilon} \rceil$.

$$\max_{\phi} \quad 1 \tag{4.12a}$$

$$\text{s.t.} \quad \tau \sum_{f \in F} \phi_{f,\tilde{f}} \geq \sum_{f \in F} \phi_{f,\tilde{f}} U_f \qquad \forall \tilde{f} \in \tilde{f} \tag{4.12b}$$

$$\text{Constraints (4.1)} \sim \text{(4.4)}$$

While the bisection algorithm may need to solve on the order of a dozen MILPs to arrive at the approximate solution, as is shown in the experiments it can significantly outperform the reformulated MILP in computational speed.

4.3.4 Greedy-Minimax Algorithm

Although the optimal ϕ can be found via a MILP, it can still be computationally expensive for large instances. Hence, heuristic algorithms can be preferable as they may be suboptimal but run fast and perform well on average. In this section, a simple approach is described to sequentially assign OCs to the systems, by greedily minimizing attacker's maximum expected utility for the partially built strategy at each stage. Algorithm 1 gives the pseudo-code.

Algorithm 1: Greedy-Minimax

$minIndCost[] \leftarrow (\min_{\tilde{f}} c(f, \tilde{f}))_{f \in F}$
$minTotCost \leftarrow \sum_f N_f * minIndCost[f]$
initialize $minu^*$, σ_{best}
For $iter = 1 \dots numIter$
    $K_{list}[] \leftarrow shuffle(K)$
    initialize $remB \leftarrow B$, $reqB \leftarrow minTotCost$
    initialize $\sigma[]$, $\bar{N}[]$, $\bar{U}[]$
    For $i = 1 \dots |K|$
        $k \leftarrow K_{list}[i]$, $f \leftarrow x[k]$
        $\sigma[k] \leftarrow GMMAssign(f, \sigma[], \bar{N}, \bar{U}[])$
        $\bar{N}[\sigma[k]] \leftarrow \bar{N}[\sigma[k]] + 1$
        $update(\bar{U}[\sigma[k]])$
        $remB \leftarrow remB - c(f, \sigma[k])$
        $reqB \leftarrow reqB - minIndCost[f]$
    compute $u^* = \max_{\tilde{f}} \bar{U}[\tilde{f}]$
    $update(minu^*, u^*, \sigma_{best}, \sigma)$
return σ_{best}
Procedure $GMMAssign(f, \sigma[], \bar{N}, \bar{U}[])$
    initialize $newU^*[]$
    For $\tilde{f} \in \tilde{F}_f$
        If $(reqB - minIndCost[f] + c(f, \tilde{f}) > remB)$ **Then**
            Continue
        $\sigma[k] \leftarrow \tilde{f}$
        $newU^*[\tilde{f}] \leftarrow U^*(\sigma)$
    $\tilde{F}_{best} \leftarrow \arg\min_{\tilde{f}} newU^*[\tilde{f}]$
    generate $\tilde{f}_{best} \sim uniRand(\tilde{F}_{best})$
return $\tilde{f}_{best}$

Greedy-Minimax starts by computing for each $f \in F$, the minimum cost of masking f with any feasible OC, and subsequently, the minimum total cost of masking all the systems (Lines 1-2). Next, σ_{best} and $minu^*$ are initialized, which respectively denote the final output strategy of the algorithm and the corresponding utility (Line 3). Subsequently, the algorithm is conducted in a number of iterations. In each iteration, a random shuffle of the set of systems is obtained, referred to as K_{list} above. Subsequently, the strategy σ which is a candidate solution corresponding to this shuffle, the corresponding observed state of the network $(\bar{N}_{\tilde{f}})_{\tilde{f} \in \tilde{F}}$, and the corresponding utilities $(\bar{U}_{\tilde{f}})_{\tilde{f} \in \tilde{F}}$ are all initialized. These are constantly maintained as the algorithm loops through K_{list}, building the solution by assigning an OC to a system one by one (Lines 8-10). The OC to be assigned for a system is determined via the function $GMMAssign()$ which is the essence of this heuristic algorithm. The input to this function is the TC f of the system in question, and the currently built solution in terms of $\sigma, \bar{N}, \bar{U}, remB, reqB$. Given these, the function considers the candidate OCs in $\tilde{F}$ one by one, refutes those which lead to the violation of the budget constraint (i.e., make the resultant minimum required budget to exceed the resultant remaining budget). For every other $\tilde{f}$, it computes resultant $\bar{U}_{\tilde{f}}$ if the system is masked with $\tilde{f}$, and stores it in the array $newU^*$ (Lines 19, 23-24). Finally, based on these, it uniformly randomly chooses an OC from those which minimize the resultant utility $newU^*()$ (Lines 25,26). Once $GMMAssign()$ returns an OC $\tilde{f}$, it is assigned to the system in question, $\bar{N}_{\tilde{f}}$, $\bar{U}_{\tilde{f}}$ are updated accordingly, as well as the remaining budget and the minimum required (Lines 11-14). Once the loop through K_{list} is over and the full strategy σ is built, its utility $u*$ is computed, and compared with $minu^*$, to update $minu^*$ and σ_{best} appropriately (Lines 15-16).

It is possible to conceive examples where this heuristic approach does not yield a good solution on an arbitrary shuffle, even for problem instances with small parameters. Such an example with 4 systems, 4 TCs and 2 OCs is discussed next. Further, there are examples where the solution value is $\Theta(|K|)$ times as bad as the optimal, on exponentially many shuffles. This motivates getting candidate solutions for a large number of shuffles and choosing the best among them as described above. Since the greedy choice does not guarantee optimality, *Soft-GMM* is also proposed, a slight modification of GMM which makes assignment probabilistically, and not deterministically. It works exactly as GMM, except Lines 25,26 — it draws f_{best} from a distribution $P(\tilde{F})$ where, $P(\tilde{f}) \propto \exp(-newU^*[\tilde{f}])$.

Note that the adversary's utility $U^*(\phi)$ for any strategy ϕ can be at most $|K|$ times the optimal value $min_\phi U^*(\phi)$. This follows from observing that for any strategy ϕ, we have $\tilde{U}_{\tilde{f}} \leq \max_{f|N_f>0} U_f \forall \tilde{f}$ by definition, and thus, $U^*(\phi) \leq \max_{f|N_f>0} U_f$, whereas $min_\phi U^*(\phi)$ is at least the average of all the system utilities by Proposition 4.1. Since any choice a greedy heuristic makes can be potentially suboptimal, one may intuitively expect its performance to be worse for a higher number of choices to be made, that is, for larger sized inputs, and relatively better for smaller inputs. However, an example instance is shown next of a CDG where despite the input size ($|F|, |K|, |\tilde{F}|$) being very small, the (hard-)GMM algorithm in a particular iteration (i.e., when conducted on a particular shuffle of the systems) gives a highly suboptimal solution.

Consider the set of systems $K = \{k_1, k_2, k_3, k_4\}$, the set of TCs $F = \{f_1, f_2, f_3, f_4\}$ and the set of OCs $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2\}$. Let the feasibility constraints be given via the sets $F_{\tilde{f}_1} = \{f_1, f_2, f_3\}$ and $F_{\tilde{f}_2} = \{f_2, f_3, f_4\}$. Let each system k_i have the TC f_i, so that the

TSN $(N_f)_{f \in F}$ is $(1, 1, 1, 1)$. For the TCs, let the utilities be $U_{f_1} = 1$, $U_{f_2} = 2$, $U_{f_3} = 30$, and $U_{f_4} = 40$. For simplicity, let all the costs $c(f, \tilde{f})$ to be 0, so that there is essentially no budget constraint.

Consider the ordering of the systems on which GMM is performed to be: $\{k_1, k_2, k_3, k_4\}$. Then, the strategy σ computed by the GMM on this ordering is as follows:

$$\begin{array}{c} \\ k_1 \\ k_2 \\ k_3 \\ k_4 \end{array} \begin{array}{c} \begin{array}{cc} \tilde{f}_1 & \tilde{f}_2 \end{array} \\ \left[\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}\right] \end{array}$$

Accordingly, the expected utilities of OCs are $\tilde{U}_{\tilde{f}_1} = (1 + 2 + 30)/3 = 11$ and $\tilde{U}_{\tilde{f}_2} = 40/1 = 40$, and thus, adversary's utility is 40 for this strategy. The optimal solution, however, masks k_1, k_3 with $\tilde{f}_1$ and k_2, k_4 with $\tilde{f}_2$ giving the expected utilities of the OCs: $\tilde{U}_{\tilde{f}_1} = (40 + 2)/2 = 21$ and $\tilde{U}_{\tilde{f}_2} = (30 + 1)/2 = 15.5$, thus, the optimal being just 21.

Further, the following is an example of a CDG which shows the GMM algorithm can perform $\Theta(|K|)$ as bad as the optimal solution on exponentially many shuffles.

Consider the CDG instance with the set of systems $K = \{k_1, \ldots, k_m\}$, so that $|K| = m$. Let the set of TCs $F = \{f_1, f_2, f_3\}$ and the set of OCs $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2\}$. Let the true state of the network be: $\mathbf{x} = (1, 2, 3, \ldots)$ Let the feasibility constraints be given by the sets $F_{\tilde{f}_1} = \{f_1, f_3\}$ and $F_{\tilde{f}_2} = \{f_2, f_3\}$. For the TCs, the utilities are $U_{f_1} = 1$, $U_{f_2} = 2000$, and $U_{f_3} = \epsilon$. For simplicity, let all the costs $c(f, \tilde{f})$ to be 0, so that there is essentially no budget constraint.

The optimal solution to this CDG is to assign systems $k_2, \ldots, k_m$ to be masked by $\tilde{f}_2$ with k_1 being masked with $\tilde{f}_1$. This gives the following expected utilities: $\tilde{U}_{\tilde{f}_1} = 1/1 = 1$ and $\tilde{U}_{\tilde{f}_2} = \frac{2000+(m-2)\epsilon}{m-1} = \frac{2000}{m-1} + \frac{(m-2)\epsilon}{m-1}$. Consider any shuffle which orders the systems such that k_1 is first and k_2 is last (of which there are $(m-2)!$). Given any ordering of this type, GMM assigns assigns systems $k_3, \ldots, k_m$ to be masked with $\tilde{f}_1$ and would assign k_2 to be masked with $\tilde{f}_2$. The expected utilities given this assignment is the following: $\tilde{U}_{\tilde{f}_1} = \frac{1+(m-2)\epsilon}{m-1} = \frac{1}{m-1} + \frac{(m-2)\epsilon}{m-1}$ and $\tilde{U}_{\tilde{f}_2} = 2000/1 = 2000$. The loss in this case is $\approx \frac{2000}{\frac{2000}{m-1}} = \frac{1}{m-1}$ which is a $\Theta(|K|)$ loss.

4.3.5 Solving for an Optimal Marginal Assignment n

The prior analysis focuses on finding the optimal pure strategy ϕ for the defender to commit to in the game. This is due to the assumption that adversaries view a fixed (static) version of the network when completing reconnaissance. However, it can also be useful to find the optimal mixed strategy $\mathbf{q}$ for the defender in the game. Formally, a mixed strategy is defined as a probability distribution over all possible defender pure strategies $\phi \in \Phi$ where $\sum_{\phi \in \Phi} q_\phi = 1$ and $0 \leq q_\phi \leq 1$. For this game, enumerating the set of pure strategies is infeasible, but it is possible to find the defender's optimal marginal strategy $\mathbf{n} = \sum_{\phi \in \Phi} q_\phi \phi$ due to compactly representing the defender's strategy space. The optimal marginal strategy can be found using the same optimization as (4.6a) and replacing all instances of $\phi_{f,\tilde{f}}$ with $n_{f,\tilde{f}}$. The optimization problem for finding the defender's optimal marginal strategy can be seen as a generalized fractional linear program.

As in Section 4.3, generalized linear fractional programs are solved efficiently using a bisection algorithmic approach which solves a sequence of linear programming feasibility

problems to get an ϵ-approximate optimal solution [10]. Similarly to the MILP bisection algorithm, this algorithm is given an interval that U^* lies in which is $U^* \in [U^{LB}, U^{UB}]$. The variables $l = U^{LB}$ and $d = U^{UB}$ are introduced with the initial width $\epsilon_0 = d - l$ that contains the optimal value U^* of the optimization problem. The main loop is repeated until the width $d - l \leq \epsilon$. The main loop has the following two steps:

1. Take $\tau = (u + l)/2$ and solve the feasibility problem in Equations (4.13)$\sim$(4.18) to find if there exists a solution $\mathbf{n}$ that satisfies the constraints.

2. **if** *feasible*, take $u := \tau$; **if** *not feasible*, take $l := \tau$.

The algorithm is guaranteed to converge as after each update the interval $[l, u]$ contains the optimal U^* and the width is halved. The number of steps that are needed to find the ϵ-approximate optimal solution is $\lceil \log \frac{\epsilon_0}{\epsilon} \rceil$.

$$\max_{u,\sigma} \quad 1 \tag{4.13}$$

$$s.t. \quad \tau \sum_{f \in F} n_{f,\tilde{f}} \geq \sum_{f \in F} n_{f,\tilde{f}} U(f) \qquad \forall \tilde{f} \in \tilde{f} \tag{4.14}$$

$$\sum_{\tilde{f} \in \tilde{f}} n_{f,\tilde{f}} = N_f \qquad \forall f \in F \tag{4.15}$$

$$\sum_{\tilde{f} \in \tilde{F}} \sum_{f \in F} n_{f,\tilde{f}} c(f, \tilde{f}) \leq B \tag{4.16}$$

$$n_{f,\tilde{f}} \leq \pi_{f,\tilde{f}} N_f \qquad \forall f \in F,\ \forall \tilde{f} \in \tilde{F} \tag{4.17}$$

$$n_{f,\tilde{f}} \geq 0 \qquad \forall f \in F,\ \forall \tilde{f} \in \tilde{F} \tag{4.18}$$

4.4 Optimal Defender Strategy against Naive Adversary

The robust approach to solving CDGs, i.e., assuming a powerful adversary with knowledge of ϕ, can cause the defender to not fully realize the benefit of her informational advantage when faced with a less powerful attacker. In particular, the adversary may value OCs in a fixed manner that is known to the defender.[2] In this case, the values $\bar{U}_{\tilde{f}}$ are fixed and the defender's strategy does not affect the adversary's expected utility for attacking some $\tilde{f}$. Importantly, if there is no budget constraint one can solve for the defender's optimal strategy ϕ in polynomial time using Algorithm 2. W.l.o.g. it is assumed the adversary has a strict preference ordering over $\tilde{F}$ as if $\bar{U}_{\tilde{f}}$ is equal for any two OCs, the sets could be merged from the defender's perspective, with the feasibility constraint and cost adjusted accordingly.

Algorithm 2 begins by initializing ϕ, Γ^* (which stores the TCs the adversary attacks) and $\tilde{f}^*$ (the OC the adversary attacks given ϕ). In Line 3 the matrix $minUtil[]$ is computed which stores the lowest utility achievable for each TC which is $\min_{\tilde{f} \in \tilde{F}_f} \bar{U}_{\tilde{f}}$. The **For** loop in Line 4 iterates over all $\tilde{f} \in \tilde{F}$ which is sorted descending by $\bar{U}_{\tilde{f}}$ (Line 2) and determines for each $\tilde{f}$ the best set of TCs to mask if $\tilde{f}$ is attacked by the adversary in Lines 5 through 12. To do this, F is split into 4 separate sets P_1, P_2, P_3 and P_4 and the set of TCs to be masked with $\tilde{f}_i$ is stored in Γ'. Note that for each f, N_f copies for the algorithm are enumerated. P_1 contains all TCs which cannot be masked with an $\tilde{f}$ that has $\bar{U}_{\tilde{f}} < \bar{U}_{\tilde{f}_i}$. Intuitively, if this set is non-empty it means the defender is not able to devise a strategy ϕ such that the adversary prefers to attack $\tilde{f}_i$, and hence, all

[2] As an example, the adversary could estimate his utility according to values derived from the NIST National Vulnerability Database [59].

subsequent $\tilde{f}_i$ will never be preferred by the adversary. P_2 (P_4) contain TCs f which must be masked (cannot be masked) with $\tilde{f}_i$. P_3 then contains all TCs f which can be masked with $\tilde{f}_i$ but may also be masked with another OC $\tilde{f}_j \neq \tilde{f}_i$. The function $update(\Gamma^{'}, P_3)$ sorts the TCs in ascending order and iterates over the TCs $f \in P_3$ and masks all TC f with $\tilde{f}_i \iff U_f \leq EU(\Gamma^{'})$. In Line 13 $update(\Gamma^*, \Gamma^{'}, \tilde{f}^*, \tilde{f}_i)$ sets $\Gamma^* = \Gamma^{'}$ and $\tilde{f}^* = \tilde{f}_i$ if $EU(\Gamma^{'}) < EU(\Gamma^*)$. Finally, the function $update(\phi, \Gamma^*, \tilde{f}^*)$ in Line 14 determines the OCs $\tilde{f}^{'}$ for all $f \notin \Gamma^*$ given $\bar{U}_{\tilde{f}^{'}} < \bar{U}_{\tilde{f}^*}$ and the strategy ϕ is returned.

Algorithm 2: Compute defender's optimal ϕ with fixed $\bar{U}_{\tilde{f}}$.

```
1  initialize φ, Γ*, f̃*
2  sort(F̃) //descending by utility Ū_f̃
3  minUtil[] := (min_f̃ Ū_f̃)_f
4  For i = 1, ..., |F̃|
5      initialize Γ'
6      P1:={f | minUtil[f] > Ū_f̃i}
7      If P1 ≠ ∅
8          break
9      P2:={f | minUtil[f] = Ū_f̃i}
10     P3:={f | minUtil[f] < Ū_f̃i and f̃i ∈ F̃f}
11     P4:={f | minUtil[f] < Ū_f̃i and f̃i ∉ F̃f}
12     Γ' := P2
13     update(Γ', P3)
14     update(Γ*,Γ',f̃*,f̃i)
15 update(φ,Γ*,f̃*)
16 return φ
```

Proposition 4.2 *Given fixed utilities $\bar{U}_{\tilde{f}}$ and no budget constraint, Algorithm 2 computes the optimal strategy ϕ in $O(|F||\tilde{F}|)$.*

Proof 4.3 *First, I show that for each $\tilde{f} \in \tilde{F}$, Lines 5 through 13 in Algorithm 2 computes the set $\Gamma^{'}$ with the minimum average value. To see this, note that all TCs $f \in P_2$ must*

be in $\Gamma^{'}$ *while all TCs* $f \in P_4$ *cannot be included. In update*$(\Gamma^{'}, P_3)$ *(note* P_3 *is given in sorted order) the defender decides for each* $f \in P_3$ *to include the* N_f *TCs in* $\Gamma^{'} \iff U_f \leq EU(\Gamma^{'})$*. At the end of this update, tt follows that* $\Gamma^{'}$ *must be the minimum average set for* $\tilde{f}_i$*. Given that the for loop in Line 4 iterates through all* $\tilde{f} \in \tilde{F}$*, it must be the case that the optimal* Γ^* *is returned for some* $\tilde{f}$*.*

In Line 2, sorting $\tilde{F}$ *takes* $O(|\tilde{F}| \log |\tilde{F}|)$ *time and calculating minUtil*[] *takes* $O(|F||\tilde{F}|)$ *time. For each iteration of the for loop in Line 4, it takes* $O(|F|)$ *time to split* F *into the sets the four sets* P_1*,* P_2*,* P_3 *and* P_4*. It takes the function update*$(\Gamma^{'}, P_3)$ *at most* $|F|$ *operations to update* $\Gamma^{'}$ *while update*$(\Gamma^*, \Gamma^{'}, \tilde{f}^*, \tilde{f}_i)$ *takes* $O(1)$ *time. Hence, each iteration it takes* $O(|F|)$ *time and hence,* $O(|F||\tilde{F}|)$ *time for the for loop. Lastly, update*$(\phi, \Gamma^*, \tilde{f}^*)$ *takes at* $O(|F||\tilde{F}|)$ *time to return the defender's strategy* ϕ *as it must find an OC* $\tilde{f}_j$ *for each* $f \notin \Gamma^*$ *with* $\bar{U}_{\tilde{f}_j} < \bar{U}_{\tilde{f}_i}$*.*

It is possible to efficiently compute the defender's optimal strategy when there is no budget constraint. When the defender has a budget constraint, however, the question arises if her optimal strategy can be found efficiently as well. This problem is called *CDG-Fixed* and next it is shown to be NP-Hard.

Proposition 4.3 *CDG-Fixed is NP-hard.*

Proof 4.4 *The proposition is proved via a reduction from the 0-1 Knapsack problem (0-1 KP), which is a classical NP-hard problem. Given a budget* B *and a set of* m *items each with a weight* w_i *and value* v_i*, 0-1 KP is the optimization problem of finding the subset of items* Y *which maximizes* $\sum_{i \in Y} v_i$ *subject to the budget constraint* $\sum_{i \in Y} w_i \leq B$*. Now I*

show that 0-1 KP can be reduced to CDG-Fixed. For convenience, $[m]$ *is used to denote the set* $\{1, 2..., m\}$ *and* $S = \sum_{i\in[m]} v_i$ *denote the sum of all utilities.*

Given a 0-1 KP instance as described above, construct a CDG instance as follows. Let the set of TCs be $F = \{f_1, \ldots, f_m\} \cup \{f_{m+1}\}$*, with utilities* $U_{f_i} = v_i, \forall i \in [m]$ *and* $U_{f_{m+1}} = -W$ *for some fixed constant* V*. Note* $N_f = 1\ \forall f \in F$*. Let the set of OCs be* $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2\}$*, with* $\tilde{F}_{f_i} = \tilde{F}\ \forall i \in [m]$ *and* $\tilde{F}_{f_{m+1}} = \{\tilde{f}_1\}$*. Set the costs as* $c(f_i, \tilde{f}_1) = 0$*,* $c(f_i, \tilde{f}_2) = w_i$ *for all* $i \in [m]$ *and* $c(f_{m+1}, \tilde{f}_1) = 0$*. Set* $\bar{U}_{\tilde{f}_1} > \bar{U}_{\tilde{f}_2}$*. Assuming a naive adversary, these components completely define a CDG-Fixed problem. Since* f_{m+1} *is bound to be masked by* $\tilde{f}_1$*, and* $\bar{U}_{\tilde{f}_1} > \bar{U}_{\tilde{f}_2}$*, attacking* $\tilde{f}_1$ *is a dominant strategy for the adversary.*

Observe that $\sum_{f\in F} U_f$ *is* $\sum_{i\in[m]} v_i - V = S - V$*. It is claimed that the optimal objective of the 0-1 KP instance is greater than* $S - V$ *if and only if the optimal defender utility in the constructed CDG-Fixed problem, i.e.,* $U^*(\phi)$*, is negative. The* $\Leftarrow$ *direction is proven next as the* $\Rightarrow$ *is a similar proof. Let* ϕ^* *be the optimal solution to the CDG-Fixed problem. By definition, the set* $Y = \{i : \phi^*_{f_i,\tilde{f}_2} = 1\}$ *is a feasible solution to the 0-1 KP since the cost of mapping* f_i *to* $\tilde{f}_2$ *is* w_i*. The sum of all utilities of all systems is* $S - V$ *whereas* $U^*(\phi^*) < 0$ *means the total utilities of systems mapped to* $\tilde{f}_1$ *is less than 0, this implies that the total utilities of systems mapped to* $\tilde{f}_2$ *is at least* $S - V$*. Note each system mapped to* $\tilde{f}_2$ *corresponds to an item and hence, the optimal objective of the 0-1 KP is also at least* $S - V$*.*

The above claim shows that for any constant V*, one can check whether the optimal objective of the 0-1 KP is greater than* $S - V$ *by solving a CDG-Fixed instance. Using this procedure as a black-box, a binary search can be performed to find the exact optimal*

objective of the 0-1 KP with integer values within $O(poly(\log(S)))$ steps (both S and weights are machine numbers with input size $O(\log(S))$). As a result, a polynomial time reduction has been constructed from computing the optimal objective of any given 0-1 KP to solving the CDG-Fixed problem. This implies the NP-hardness of the CDG-Fixed problem.

CDG-Fixed can be solved with Algorithm 2 via a modification to the function $update(\Gamma^{'}, P_3)$ in Line 13. Given $\Gamma^{'}$, one computes the minimum budget $B^{'}$ required to mask all TCs $f \in \Gamma^{'}$ with $\tilde{f}_i$ and mask all TCs $f \in P_3$ and $f \in P_4$ with $\tilde{f}_j$ such that $\bar{U}_{\tilde{f}_j} < \bar{U}_{\tilde{f}_i}$. If $\Gamma^{'} = \emptyset$, then for $f \in P_3$ mask f with $\tilde{f}_i$ if $c(f, \tilde{f}_i) < B^{'}$. Assuming P_3 is sorted ascending, once the defender assigns $\tilde{f}_i$ to a TC f she is done. If $\Gamma^{'} \neq \emptyset$, the defender must solve multiple MILPs, with $n = n_{\Gamma'}, \ldots, |K|$ to find the best $\Gamma^{'}$. Denote $u_{\Gamma'} = EU(\Gamma^{'})$.

$$\min_{\phi} \quad n_{\Gamma'} u_{\Gamma'} + \sum_{f} \phi_{f,\tilde{f}} U_f \tag{4.19a}$$

$$\text{s.t.} \quad \sum_{f} \phi_{f,\tilde{f}_i} \leq n - n_{\Gamma'} \tag{4.19b}$$

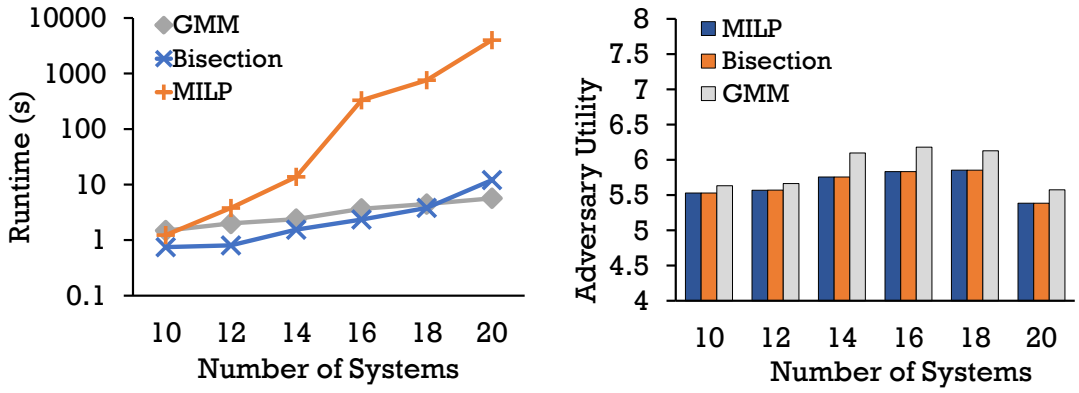
$$\text{Constraints (4.1)} \sim \text{(4.4)}$$

4.5 Experiments

The CDG model and solution techniques are evaluated using synthetically generated game instances. The game payoffs are set to be zero-sum, and for each TC, the payoffs U_f are uniformly distributed in $[1, 10]$. Each OC $\tilde{f}$ is randomly assigned a set of TCs it

can mask, while ensuring each TC can be masked with at least one OC. To generate the TSN, each system is randomly assigned a TC uniformly at random. The costs $c(f, \tilde{f})$ are uniformly distributed in $[1, 100]$ with the budget B uniformly distributed in-between the minimum cost assignment and maximum cost assignment. All experiments are averaged over 30 randomly generated game instances.

4.5.1 Powerful Adversary - Scalability and Solution Quality Loss



(a) Runtime

(b) Solution Quality

Figure 4.3: Runtime Comparison and Solution Quality Comparison (20 Observables) - Reformulated MILP (MILP), the bisection algorithm with $\epsilon = .0001$ (Bisection) and Greedy MaxiMin (GMM) with 1000 random shuffles.

When solving for the defender's optimal strategy ϕ strategy for enterprise networks, it is important to have solution techniques which can scale to large instances of CDGs. The first experiment compares the scalability of the reformulated MILP, the bisection algorithm and the Greedy Minimax (GMM) algorithm with 1000 random shuffles along with the solution quality of the approaches. In Figure 4.3(a) the runtime results are shown with the runtime in seconds on the y-axis and the number of systems varied on the x-axis. As can be seen, the runtime for solving the reformulated MILP increases dramatically

# OCs	2	4	6	8	10
10 systems	0	0.092%	0.015%	0.028%	0.512%
Optimal Instances	30	29	29	29	25
20 systems	0	0.028%	0.615%	1.91%	3.18%
Optimal Instances	30	28	17	12	9

Table 4.1: Solution Quality % loss and number of optimal instances for GMM versus MILP.

as the number of systems increases while both GMM and the bisection algorithm finish in under 10 seconds in all cases. The results from the bisection algorithm compared to the reformulated MILP are quite surprising given it provides the ϵ optimal solution and highlights the benefit from solving smaller MILP for larger CDG instances.

While GMM is much faster than the reformulated MILP (but comparable to the bisection algorithm), it is not guaranteed to provide the optimal solution or an ϵ-approximate solution. However, the experimental results show that empirically the solution quality loss is very small. In Figure 4.3(b) the solution quality of the MILP is compared to GMM, where the attacker's utility is given on the y-axis and the number of OCs are varied on the x-axis. Importantly, GMM shows a low solution quality loss for the defender compared to the MILP with a minimum loss of 1.68% for 12 systems and a maximum loss of 5.93% for 16 systems. This experiment highlights the scalability of GMM and shows the loss in solution quality from GMM can give a reasonable trade-off between computational efficiency and solution quality.

An interesting feature of GMM is how often it returns the optimal solution for the defender as the CDG game size changes. Table 4.1 compares the solution quality of GMM (with 1000 random shuffles) versus the MILP for several game sizes with 10 and 20 systems where the number of OCs are varied from 2 to 10. Interestingly, for CDGs with

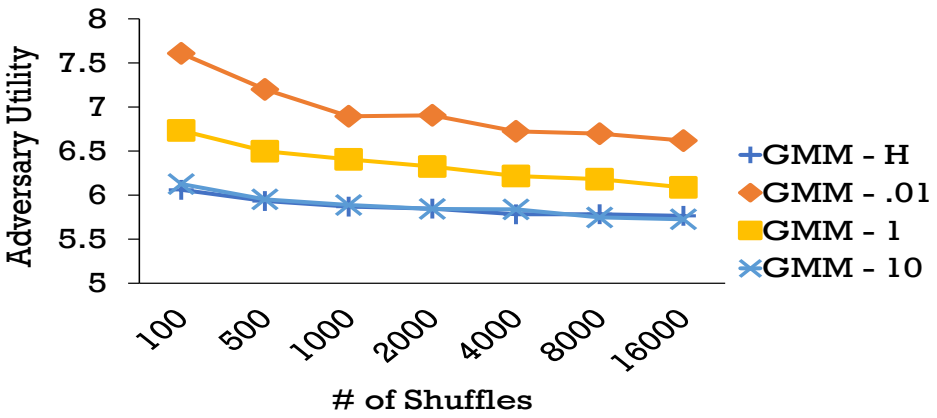


Figure 4.4: Solution Quality Comparison (20 systems and 20 OCs) - Comparison of Hard-GMM (GMM - H) and Soft-GMM (GMM - λ) varying the number of shuffles.

10 systems, Hard-GMM is able to find the optimal solution in a vast majority of instances (142 out of 150 instances). However, for CDGs with 20 systems, GMM fails to recover the optimal solution in about a third of the instances (96 out of 150). Nevertheless, the loss of solution quality still remains low (3.18%) even when GMM returns the optimal solution a third of the time.

The solution quality of a variation of GMM, called Soft-GMM or GMM$-\lambda$, was tested as well. Instead of greedily choosing the OC with minimax expected utility at the stage, a soft-min function [19] is applied with parameter λ controlling the greediness of the next choice. Figure 4.4 shows the solution quality of GMM (denoted as GMM-H) and GMM$-\lambda$ with varying λ values. GMM$-.01$ is very close to randomly choosing OCs for the systems and performs poorly compared to larger λ values, indicating that GMM is an effective heuristic and performs much better than random assignment. Importantly, the randomness in GMM$-\lambda$ leads to a potential of finding better strategies than GMM since GMM-Hard is restricted to a limited strategy space and GMM$-\lambda$ is not. This can be seen by comparing the results for GMM-Hard and GMM-10 where the latter

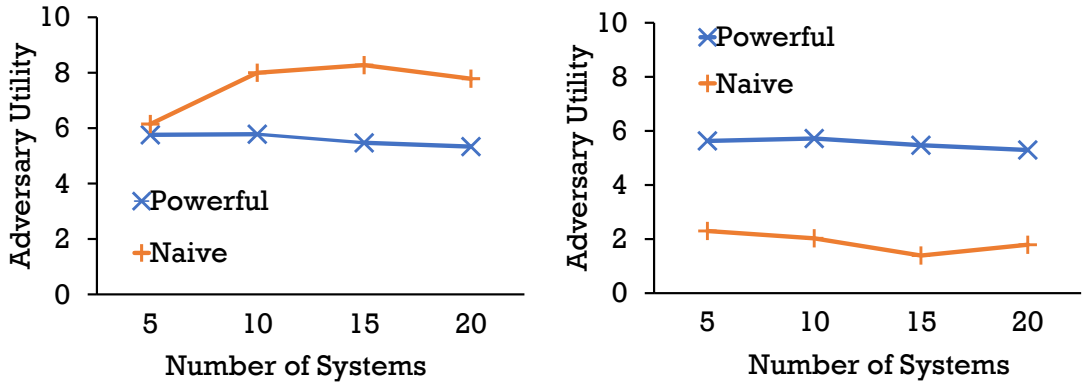
outperforms the solution quality achieved with GMM-Hard at 8000 and 16000 shuffles. Further investigation is deferred to future work.

4.5.2 Comparing Solutions for Different Types of Adversaries

The last experiment compares how the optimal strategies for the two adversary models (powerful versus naive) perform in the opposite case. Figure 4.5(a) compares the solution quality of the MILP in Section 4.3 to Algorithm 2 when the adversary is assumed to know ϕ with the attacker's utility on the y-axis and the number of systems varied on the x-axis. This figure highlights that for the powerful adversary the MILP performs significantly better than Algorithm 2 (except for 5 systems) and shows the risk of underestimating the adversary's information when devising the defender's strategy ϕ. In Figure 4.5(b) the solution quality of Algorithm 2 is compared to the MILP when the adversary is assumed to have fixed utilities. As the figure shows, the improvement in utility is dramatically higher for Algorithm 2 compared to the MILP. The reason for this difference lies in Algorithm 2 leveraging the adversary's fixed preferences over OCs and minimizes the value of systems masked with the OC the adversary will attack. The MILP, however, minimizes the worst case utility given the adversary may attack any OC and hence, fails to leverage the defender's advantage to a high benefit.

4.6 Real World Applicability

In this section, I will highlight several recent works that have provided technical solutions to achieving the deception outlined in this chapter. This section is not meant to be a complete reference of all technical deceptive techniques available, but rather, a catalog



(a) Powerful Adversary

(b) Naive Adversary

Figure 4.5: Solution Quality Comparison (10 OCs) - In (a) the solution quality of the two types of defender strategies is shown against a powerful adversary. In (b) the solution quality of the strategies is shown against a naive adversary.

of applicable deceptive methods for a network administrator using the Cyber Deception Game model. There are three main areas of active research in developing deceptive techniques to thwart adversarial network reconnaissance: (1) Operating System and application/service obfuscation against fingerprinting, (2) deceptive network topology alteration and (3) honeypots and honey-* defenses.

4.6.1 OS and Application Fingerprinting & Obfuscation

One of the first pieces of information an adversary must gain when completing reconnaissance on a defender's network is determining the operating systems deployed on systems in the network. This important first step provides crucial information about the types of exploits available to be used and potentially the difficulty in attacking the defender's network. Operating System (OS) fingerprinting is an area of significant interest as it provides the tools necessary for an adversary to determine the OSs that systems across the enterprise network are running. OS fingerprinting can be accomplished in two ways

(i) *active fingerprinting* which involves sending carefully crafted packets to the target system and analyzing the results and (ii) *passive fingerprinting* which sniffs and analyzes network network traffic traveling between systems.

Active fingerprinting techniques are generally more sophisticated than passive fingerprinting. In some cases, an adversary can disregard stealthier approaches and simply attempt to connect to the host system to learn about the OS of the system by establishing a connection via the Telnet or SSH protocol which sends the OS version as part of the welcome message. For network recon tools, active fingerprinting techniques trigger a target system to send a series of responses that are then analyzed by the adversary's network tools to determine the type and version of the OS a system is running. The ICMP, TCP and UDP packets sent to a system are specially crafted to observe how the system responds to both valid and invalid packets. For example, some features of TCP probes received from a system can distinguish in-between different operating systems (e.g., order of the TCP options, the TCP sequence number).

Passive fingerprinting consists of using a packet sniffer that passively collects and analyzes packets traveling between systems in a network. One simple method of passive fingerprinting uses the Time To Live (TTL) field in the IP header and the TCP Window Size of the SYN or SYN+ACK packet in a TCP session to determine the OS of a system. This is due to the values for TTL and the TCP Window Size depending on the OS implementation as the RFC specifications only define intervals of values and recommended values, and does not mandate specific values to be used.

There are numerous tools available for fingerprinting today. Due to the variety of the network mapping tools, no single deception approach can be used to defeat all of them,

but with a combination of approaches it is possible to significantly increase the difficulty in the reconnaissance efforts of an adversary. Below is a list of some of the network scanning tools available to give the reader a small overview of this area. A more in-depth and exceptional technical breakdown of network scanning tools and their specifications can be found in [1].

1. Nmap [53]: Nmap is a network security scanner used to discover hosts and services on a computer network that builds a "map" of the network. It works as an active tool by sending specially designed packets to a host which it then analyzes to identify the OS and applications running on different ports.

2. SinFP3: SinFP was developed in order to complete OS fingerprinting of a host under the worst-case network conditions [9]. This includes a remote host having only one port or the traffic to all other TCP and UDP ports is dropped by filtering services. Once the response packets have been received SinFP uses a matching algorithm to determine the OS of a particular system.

3. Xprobe: Xprobe is an active OS fingerprinting tool that uses fuzzy signature matching, probabilistic guesses and a signature database to determine the OS for a given host [86]. Many of the techniques in Xprobe have been built into nmap over time.

4. p0f3: p0f v3 [90] is a tool that utilizes passive traffic analysis to fingerprint hosts behind any TCP/IP communications in a network without interfering in any way. The techniques it uses for the fingerprinting are sophisticated and this tool can be used when Nmap probes may be blocked or the adversary wants to be stealthy.

5. amap: amap [67] is a application mapper tool which determines the applications a host on a network is running and the specific versions of those applications by interrogating network services and sockets.

6. Nessus: Nessus [17] provides a wide-variety of network scanning tools in order to map out a network. It is extremely useful as a vulnerability scanner by looking for types of vulnerabilities such as misconfigurations, default passwords, and denials of service against TCP/IP stack by using malformed packets. Additionally, it uses many of the other network and application fingerprinting tools in conjunction to provide results to a Nessus user.

For the fingerprinting of applications running on a port of a host system, a typical approach relies on retrieving the service banner to gather information on the application and version along with using the port number (e.g., webservers typically have HTTP on port 80). In this regard, approaches to combat application fingerprinting rely on altering the service banner sent back in a packet for a given probe. There are a few potential issues from altering the banner of a service which causes it to not be applicable to some services. For example, services that use the banner information during the connection process (like SSH) require a non-transparent approach.

A recent paper [1] provides a technical approach for OS and application obfuscation against nmap reconnaissance efforts. Their approach works by altering important information contained in the TCP/IP header's in order to fool nmap scans into misclassifying the OS of a particular system, and potentially, the application and services running on

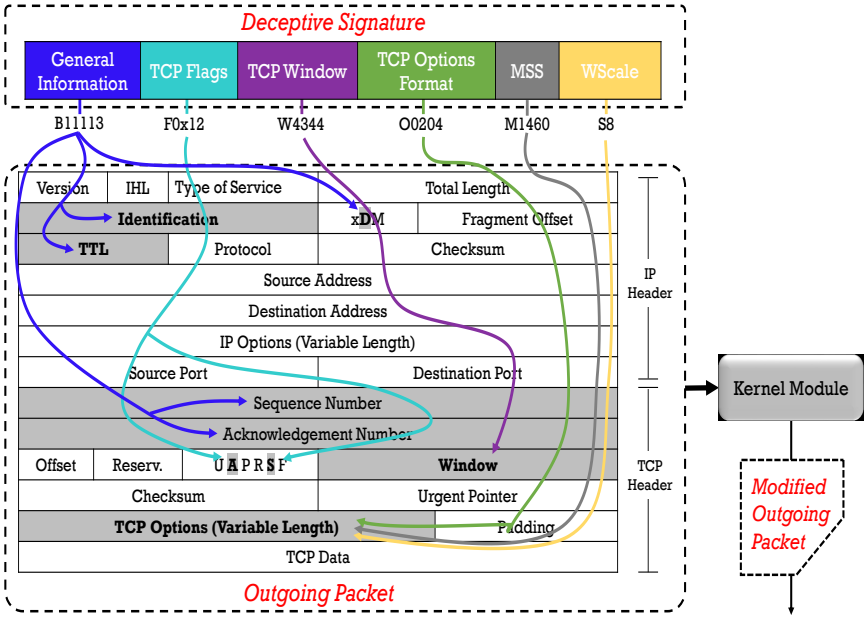


Figure 4.6: The alteration of an outgoing packet to mimic a certain desired deceptive signature.

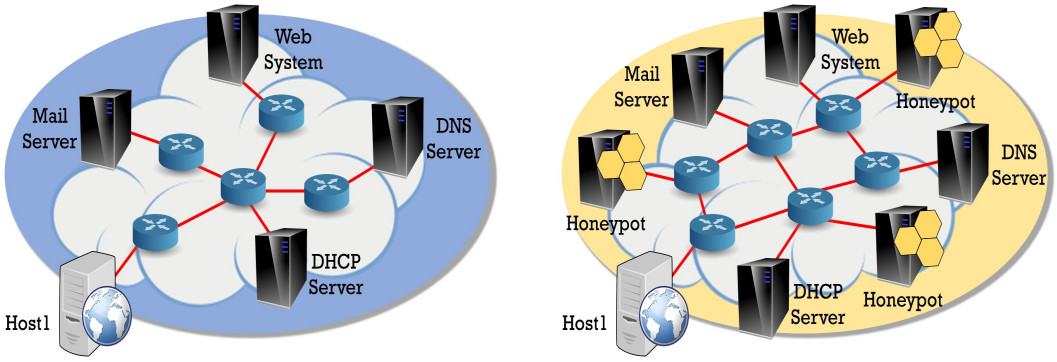
that system as well. Figure 4.6 shows how an outgoing packet is modified in order to provide a deceptive signature of the host machine through the alteration of specific properties in the packet header. This approach is useful as it can thwart both active and passive OS fingerprinting tools, such as Nmap and p0f3, while also altering the reported services by altering the header files sent back from a particular service scan (achieved by scanning the socket). For the obfuscation of a service hosted on a system, the techniques presented in [1] relies on altering the banner message sent back when establishing a connection or in the header of each appliance-level protocol data unit.

4.6.2 Deceptive Network Topologies

Another area of interest for recent research focuses on altering the network view each host system on a network observes from reconnaissance efforts. A recent line of work [26]

provides the the Adaptive Cyber Deception System (ACyDS) that gives the network administrator an incredibly powerful tool. ACyDS provides each network host a unique virtual view of the enterprise network that alters the subnet topology and IP address assignments of reachable hosts and servers and does not reflect the physical network configurations.

ACyDS works by altering the *network view* for each host (system) on the network which consists of the *network entities* and *network topology*. The network entities viewable from a system are those which it is permitted to communicate with on the network. The network topology view of a system's network view consists of the routers connecting the host and other systems present on the network from the network entities view. For a given system, the network view is then the composite of the network entities and network topology views. An example is given in Figure 4.7 which shows Host1's network view.



(a) True Network (b) Deceptive Network

Figure 4.7: Network views for a host connected to the defender's network. In 4.7(a) is the true network state while in 4.7(b) is an altered state with additional network connections and honeypots.

ACyDS works by deceptively setting the network view for a system via altering the network entities or network topology views. This is achieved with the use of *Software-Defined Networking* (SDN) along with *OpenFlow* to correctly manage the traffic in the network. ACyDS leverages SDN controllers, SDN switches, and other components to create the individual deceptive network views for systems. It is recommended the reader reference [26] for an in-depth breakdown of the incredible ACyDS tool.

4.6.3 Honeypots and Network Tools

Honeypots are used ubiquitously in cybersecurity as a means of identifying adversary attacks and learning about adversary behavior. Although I do not talk about the use of honeypots extensively in this chapter, they can be incorporated into the deception model along with their behaviors. Additionally, software applications have been developed which can mimic network personalities of host systems to increase the perceived number of host machines on a network to an adversary. HoneyD [66] establishes virtual network daemons that listen for requests sent to certain IP addresses on a defender's network and can answer these requests for the virtualized machines. HoneyD provides the ability to set the network topology, host machines and each the machine's configuration on a network. This provides the network administrator with the ability to fake additional services on the real hosts while also emulating additional honeypots that may appear the same as the real host systems. Using an approach of this type creates vastly more uncertainty for an adversary and forces them to expend significant effort in order to identify real versus fake systems on a network. Figure 4.8 gives an example of the how HoneyD can be used within a network.

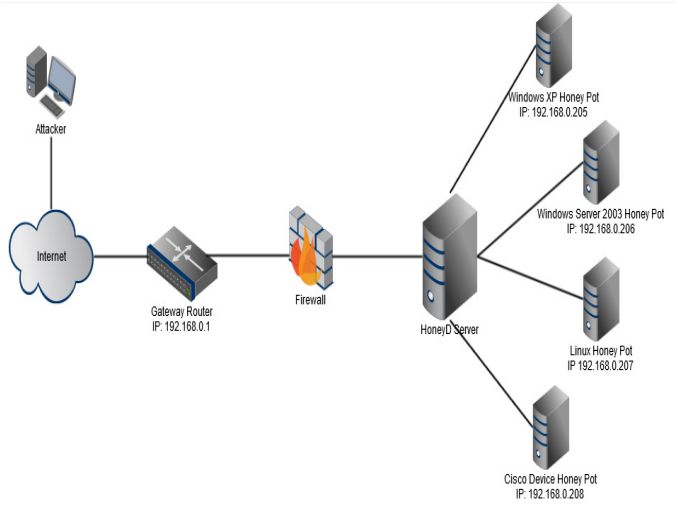


Figure 4.8: The HoneyD server initializes network daemons to respond to pings and scans for various IP addresses not used by a network.

Beyond HoneyD, another recent tool which has been developed to identify adversarial network reconnaissance in Project Nova[3]. This application works by first analyzing a defender's network through the use of nmap and then developing a deceptive 'haystack' that is deployed using HoneyD. The haystack is essentially a set of virtual systems to create in addition to the current real systems operating on a network to increase the difficulty to an adversary learning about systems on the network. In addition to increasing the effort expended by an adversary, the haystack is useful in helping determine adversarial nodes that have been compromised in a network which are being used for reconnaissance or attacking efforts. The CDG model can be leveraged with the use of HoneyD or inside of an application like Project Nova to determine the network state to show and autonomously change the views showed to the adversary.

[3] `http://www.novanetworksecurity.com/index.html`

4.6.4 Leveraging the CDG Model

The technical approaches presented in this section highlight much of the current state-of-the-art in developing network technologies to fool adversarial reconnaissance efforts. The CDG model represents a high-level approach to reasoning about the deception schemes a network administrator can employ when given an enterprise network environment and available technical deception tools. One question that arises for a network administrator from this research is, "Why respond at all?" It turns out that many organization's enterprise networks must be open to outside users through the DMZ portion of their network. These situations require systems on the network to respond to requests for connections and scanning activity which is part of their function on the network. Indeed, the CDG model is particularly useful as a tool to protect the perimeter of an organization's network, but it is general enough to also capture reconnaissance activities in the organization's intranet. In order to gain a concrete understanding of the applicability of CDGs, it is useful to go through an example similar to the example network in Section 4.2.

Consider an example scenario where the network administrator has 4 systems, e.g., acting as webservers in this scenario, on the network which have an operating system and a webserver software. For the systems, assume systems k_1 and k_2 have the Windows Server 2012 OS with k_3 and k_4 having the Ubuntu Linux 12.04 (Linux 3.2.x Distribution) OS. Assume all are running Apache webserver, with k_1 running version 2.2, k_2 and k_3 running version 2.3, and k_4 running version 2.4. The TCs for these systems are then as follows: $k_1 = \{[os]\ WS2012,\ [web]\ Apache2.2\}$, $k_2 = \{[os]\ WS2012,\ [web]\ Apache2.3\}$, $k_3 =$

$\{[os]\ Linux3.2.x,\ [web]\ Apache2.3\}$ and $k_4 = \{[os]\ Linux3.2.x,\ [web]\ Apache2.4\}$. For this scenario the technical approach presented in [1] is considered which alters outgoing packets to obfuscate the OS and applications for a system.

Assume the network administrator has developed several obfuscation personas for the operating systems which allows them to make an OS of a system to appear as Windows Server 2008, Windows Server 2012, Linux 2.6.x, and Linux 3.2.x. Additionally, the network administrator can obfuscate the webserver version of Apache to appear as 2.2, 2.3 or 2.4. The combination of all obfuscation personas make up the set of OCs available for the CDG model to optimize given constraints on which personas can be used for a given system, e.g., Windows Server 2012 can only be obfuscated as Windows Server 2008.

A possible deception scheme for the network administrator - from optimizing the CDG model - is deploying the following personas for the systems: $k_1 = \{[os]\ WS2008,\ [web]\ Apache2.2\}$, $k_2 = \{[os]\ WS2008,\ [web]\ Apache2.2\}$, $k_3 = \{[os]\ Linux2.6.x,\ [web]\ Apache2.4\}$ and $k_4 = \{[os]\ Linux2.6.x,\ [web]\ Apache2.4\}$. In this scheme, k_1 and k_2 are made to appear the same and in a similar way k_3 and k_4 are made to appear the same. From here the network administrator only needs to tell the OS and service obfuscator to alter packets to make each system appear as the desired persona. Switching to new personas is also easy as it only requires changing the attached personas in the network switch (kernel module in Figure 4.6) responsible for altering the packets sent out from systems on the network, i.e., one could have k_1 and k_2 have Apache 2.3 as their webserver without much overhead.

As mentioned in Section 4.6.1, the deception is not always transparent to legitimate users. Presenting false personas for systems requires the alteration of all outgoing packets

in some cases (given a passive network scanner), and hence, latency can become an issue along with reductions in the data transfer speed caused from altering values such as the Maximum Segment Size (MSS) or Window Size (WS). It is recommended the reader refer to [1] for more in-depth details and analysis of the cost to a legitimate user from their technical deception approach.

4.7 Chapter Summary

In this chapter, I study the problem of a network administrator should respond to scan requests from an adversary attempting to infiltrate her network. I show that computing the optimal defender strategy against a powerful adversary is NP-hard and provide two main techniques, a MILP approach and a bisection algorithmic approach, to solve for the defender's optimal strategy against a powerful adversary. Additionally, a greedy algorithm is provided which quickly finds good defender strategies and performs well empirically. I then show that computing the optimal strategy against a naive attacker is still NP-hard given a budget constraint. Extensive experimental analysis is given demonstrating the effectiveness of the approaches. Finally, a section covering technical deception approaches shows applications and tools that could be used in leveraging the CDG model and algorithms to generate deceptive network views in real-world networks.

Chapter 5

Cyber-alert Allocation Games

5.1 Problem Domain

While many organizations face the challenge of cyber alert allocation, this chapter highlights 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 [1]. 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

[1] 24th Air Force - AFCYBER: `http://www.24af.af.mil`

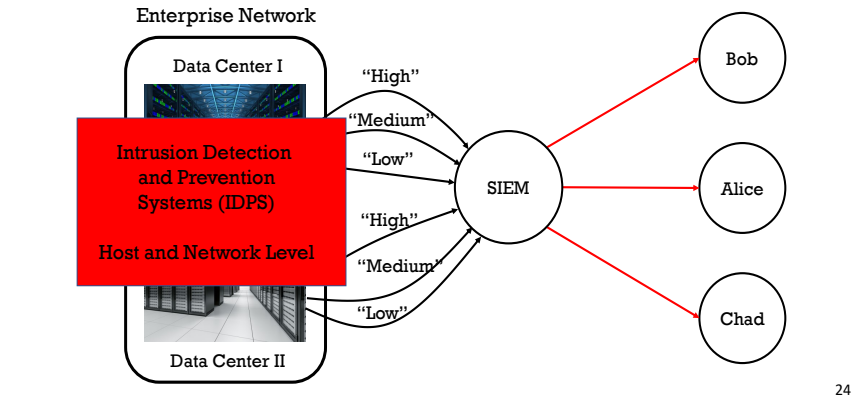


Figure 5.1: To protect against cyber intrusions, enterprise networks deploy Intrusion Detection and Prevention Systems across their network that work at both a host and network level. The alerts generated are given a risk classification and aggregated into a central repository called a SIEM. The network administrator then must determine how to allocate the alerts to analysts for investigation and remediation if necessary.

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 [2]. 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 the central analyst assignment problem covered in this section in the real world.

In this chapter, I will discuss the Cyber-alert Allocation Game (CAG) model which captures the assignment problem a network administrator is faced with in this domain.

[2] 688th Cyberspace Wing: `http://www.24af.af.mil/Units/688th-Cyberspace-Wing`

This model helps prioritize the resolution of alerts by accounting for a strategic adversary and constraints on the cyber analysts investigating the alerts. Next, the CAG is computationally analyzed and techniques are developed to solve for the defender's optimal assignment policy. Finally, experiments are shown which demonstrate the benefit of the game theoretic approach versus ad-hoc assignment policies while the scalability of the algorithms is also analyzed.

5.2 Cyber-alert Allocation Games

The *Cyber-alert Allocation Game* (CAG) is modeled 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. A worst-case assumption is made such that the attacker moves with complete knowledge of the defender's strategy and plays a best-response attack strategy [43]. 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) [88], so the order of the moves is *not* consequential in the model.

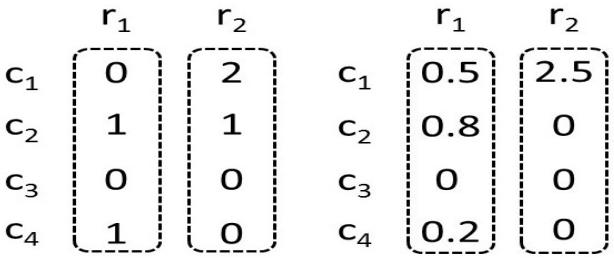
Systems and Alerts: The defender responds to alerts originating from a set of systems $k \in K$. A "system" in this model could represent any level of abstraction, ranging from a specific server to a complete network. IDPS for each system generate alerts of different types, $a \in A$. The alert types correspond to levels of severity (e.g., high, medium, and low), reflecting the likelihood of a malicious event. The combination of the alert type and the origin system is represented 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. It is assumed that both the defender and attacker know the typical value of N_c from historical averages (similar to [33]).

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. These broad classes of attacks are represented as attack methods $m \in M$. For every attack method there is a corresponding probability distribution β_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 the 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, the effectiveness of an analyst against an attack method, representing her expertise, is captured 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



(a) Pure Strategy (b) Marginal Strategy

Figure 5.2: CAG Strategies for the defender.

allocations that satisfy the following constraints; C_a denotes all categories with the alert type a:

$$\sum_{a \in A} \sum_{c \in C_a} T_a^r P_{c,r} \leq 1 \qquad \forall r \in R \tag{5.1}$$

$$\sum_{r \in R} P_{c,r} \leq N_c \qquad \forall c \in C \tag{5.2}$$

$$P_{c,r} \text{ are integers} \tag{5.3}$$

Inequality (5.1) ensures that each analyst is assigned a valid number of alerts, while inequality 5.2 ensures the number of alerts assigned are not more 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 (5.1)) for r_1 is instantiated as follows (the other columns are similar):

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

For c_1 the alert capacity constraint (Inequality (5.2)) we have (the other rows are similar):

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

An example of a pure strategy P is given in Figure 5.2(a). The dashed boxes in Figure 5.2(a) represent the set of variables in the analyst capacity constraints, i.e. constraints of type (5.1). An example marginal strategy is shown in Figure 5.2(b). This drops constraint (5.3), but satisfies constraints (5.1) and (5.2).

Define a mixed strategy $\mathbf{q}$ over pure strategies $P \in \hat{P}$ ($\sum_{P \in \hat{P}} q_P = 1, 0 \leq q_P \leq 1$). From the mixed strategy one 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 $\mathbf{n}$ 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 $\mathbf{n}$ 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 δ is the defender. A CAG is formulated as a zero-sum game, hence the payoffs for the adversary (θ) 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 β_a^m, the defender's utility is:

$$U_s = \sum_{a \in A} \beta_a^m [x_{c,m} * U_{\delta,c}^d + (1 - x_{c,m}) U_{\delta,c}^u] \tag{5.4}$$

Bayesian Game It is possible to extend the CAG game formulation to allow for varying adversary types. The motivation behind this extension is to handle the situation where a defender may be protecting against adversaries that may value targets in the network differently, e.g. a nation-state versus script-kiddie. In this case, denote the set of adversary types as Θ and it is assumed the defender knows a prior z over the chance of encountering the varying adversary types. Define the utilities for adversary type θ as $U_{\delta,c}^d(\theta)$ and $U_{\delta,c}^u(\theta)$. The defender's utility given θ, k, m and β_a^m is then $U_{s,\theta} = \sum_{a \in A} \beta_a^m [x_{c,m} * U_{\delta,c}^d(\theta) + (1 - x_{c,m}) U_{\delta,c}^u(\theta)]$. Although the bayesian formulation can easily be handled, for clarity the rest of the analysis is completed without considering the adversary types.

5.3 Defender's Optimal Strategy

The defender's optimal mixed strategy (maximin strategy) can be computed a linear program, denoted as $MixedStrategyLP$:

$$\max_{\mathbf{n},\mathbf{v}} \quad v \tag{5.5}$$

$$s.t. \quad v \leq U_s \qquad \forall k \in K, \forall m \in M \tag{5.6}$$

$$x_{c,m} = \textstyle\sum_{r \in R} E_m^r \frac{n_{c,r}}{N_c} \qquad \forall c \in C, \forall m \in M \tag{5.7}$$

$$n_{c,r} = \sum_{P \in \hat{P}} q_P P_{c,r} \qquad \forall c \in C, \forall r \in R \tag{5.8}$$

$$\sum_{P \in \hat{P}} q_P = 1, q_P \geq 0 \tag{5.9}$$

This LP requires *exponentially* many pure strategies $P \in \hat{P}$. The objective function in Equation 5.5 maximizes the defender's utility, v. Equation 5.6, which uses Equation (5.4), ensures the adversary selects a best response over all $m \in M$ and $k \in K$. Equation 5.7 calculates the detection probabilities $\mathbf{x}$ from the marginal strategy $\mathbf{n}$, which is computed by Equation 5.8. Equation 5.9 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 [22]. The computational hardness arises from the underlying team formation of applying a group of screening resources to screen incoming passengers. However, in CAGs there are not teams of analysts, the defender only needs 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, in Theorem 5.1 it is shown 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.

Theorem 5.1 *Computing the defender maximin strategy is weakly NP-hard when there is only one resource, and is strongly NP-hard with multiple resources.*

Proof 5.1 *From [85], it is shown that the computational complexity of computing minimax equilibrium is equivalent to that of finding the best response. Here, I show that the best response problem in CSGs is weakly NP-hard even when there is only one resource via a reduction from the Knapsack problem and becomes strongly NP-hard when there are multiple resources via a reduction from the Generalized Assignment Problem (GAP). This, together will the results of [85], yields the claimed conclusion. First, the weak NP-hardness is proven when there is a single resource. In the knapsack problem, we have N items each with a weight w_i and value v_i $\forall i \in N$, and aim to pick items of maximum possible value subject to a weight-capacity budget B.*

Now create a CSG instance with 1 system k_1, 1 attack method m_1, and one resource r_1. Set $E_{m_1}^{r_1} = 1.0$ and $L_{r_1} = 1$. Also, N alert levels are created, thus $|A| = |C| = N$. For each alert level $a \in A$, set $T_{a_i}^{r_1} = w_i/B \leq 1$. Each category $c_i \in C$ also corresponds to a_i since there is only one system. Set $U_{s,c_i}^d = v_i$ and $U_{s,c_i}^u = 0$. Also set $q_{a_1}^{m_1} = \ldots = q_{a_N}^{m_1} = 1/|N|$. Further, set $N_{c_i} = 1$ $\forall i \in N$, i.e., each category has precisely one alert. The adversary only has one choice k_1, m_1. In this constructed instance, the defender's best response is a pure strategy that picks alerts of the maximum value but subject to total time limit constraint. Let $n_i \in \{0, 1\}$ denote whether the resource resolves category c_i, the best response problem then solves the following optimization program:

$$\max_{\mathbf{n}} \quad \sum_{i=1}^{N} n_i v_i \tag{5.10}$$

$$s.t. \quad \sum_{i=1}^{N} n_i \cdot w_i / B \leq 1 \tag{5.11}$$

$$n_i \in \{0, 1\} \qquad \forall i = 1, ..., N \tag{5.12}$$

It is easy to see that this is precisely the Knapsack problem described above, yielding the weak NP-hardness. To prove that the problem is strongly NP-hard with multiple resources, the reduction is from the following Generalized Assignment Problem (GAP), a well-known NP-hard problem: given R machines and A jobs, assign job a to machine r which costs T_a^r time units and achieves utility E_a^r; machine R has a limit of 1 time unit. The goal is to assign these jobs to machines to maximize the total utility subject to each machine's time capacity. It is easy to verify that this corresponds to the best response problem of a particular CSG as follows: one system, R resources, alert set A; $|A|$ attack methods with method m_a triggering alert a with probability 1. Trivial details are omitted here.

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 5.1. This result is discussed further in Section 5.4.

Proposition 5.1 *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.*

Proof 5.2 *When all* T_a^r *for a given analyst* r *are equal, constraint:* $\sum_{a \in A} \sum c \in C_a T_a^r n_{c,r} \leq 1$ *can be converted into* $\sum_{a \in A} \sum c \in C_a n_{c,r} \leq \frac{1}{T_a^r}$. *WLOG,* $\frac{1}{T_a^r} \rightarrow \lfloor \frac{1}{T_a^r} \rfloor$ *as any marginal assignment over* $\lfloor \frac{1}{T_a^r} \rfloor$ *will not be implementable. This happens as all pure strategies have integer assignments. Hence, for all constraints of type* (5.16) *a new constraint* $\sum c \in C_a n_{c,r} \leq \lfloor \frac{1}{T_a^r} \rfloor$ *is introduced. This new set of analyst capacity constraints forms a hierarchy* H_2 *and therefore, the set of constraints form a bihierarchy. The defender's optimal marginal strategy* $\mathbf{n}$ *can be found by solving the MSLP.*

5.3.1 Defender's Optimal Marginal Strategy

In the security games literature, two approaches are commonly used to handle scale-up: marginal strategies [43, 50] and column generation [37]. A marginal strategy based approach is adopted which finds the defender's marginal strategy $\mathbf{n}$ and does not need to explicitly enumerate the exponential number of pure strategies. A relaxed version of LP (5.5)∼(5.9) in given in LP (5.13)∼(5.17). LP (5.13)∼(5.17) is similar to LP (5.5)∼(5.9) except that equations (5.8) and (5.9) are replaced with equations (5.16) and (5.17) to model the relaxed marginal space. Recall that marginal strategies satisfy constraints (5.1)∼(5.2) (which lead to Equations 5.16 and 5.17) but drop constraint

(5.3). The optimal marginal strategy $\mathbf{n}$ for the defender can then be found by solving the following $MarginalStrategyLP$[3] (MSLP):

$$\max_{\mathbf{n},\mathbf{v}} \quad v \tag{5.13}$$

$$s.t. v \leq U_s \qquad \forall k \in K, \forall m \in M \tag{5.14}$$

$$x_{c,m} = \sum_{r \in R} E_m^r \frac{n_{c,r}}{N_c} \qquad \forall c \in C, \forall m \in M \tag{5.15}$$

$$\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1 \qquad \forall r \in R \tag{5.16}$$

$$\sum_{r \in R} n_{c,r} \leq N_c, \quad n_{c,r} \geq 0 \qquad \forall r \in R, \forall c \in C \tag{5.17}$$

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, one 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, the defender's optimal implementable marginal strategy can be efficiently computed using the MSLP.

Theorem 5.2 *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.*

[3]Note for the bayesian formulation, the objective function would be $\sum_{\theta \in \Theta} z_\theta v_\theta$ while the second constraint becomes $v_{s,\theta} \leq U_{s,\theta} \quad \forall k \in K, \forall m \in M, \forall \theta \in \Theta$

Proof 5.3 *Let Q be the polytope defined by constraints 5.16 and 5.17. Notice that since $N_c \geq \sum_{r \in R} \frac{1}{T_a^r}$, constraint 5.17 is trivially satisfied, where $\sum_{r \in R} n_{c,r} \leq N_c$. Now, because 5.16 is independent across $r \in R$, Q can be written as $Q = Q_1 \times Q_2 \times \ldots \times Q_{r_{|R|}}$, where $Q_r = \{n_{c,r} | \sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1, n_{c,r} \geq 0\}$.*

To show any feasible marginal strategy $\mathbf{n}$ *from the MSLP has a valid mixed strategy* $\mathbf{q}$ *it needs to be shown that the extreme points of Q belong to $\Delta_{\hat{P}}$. Using a result from [20] it is known that $\mathbf{n} \in Q$ is an extreme point* iff $\mathbf{n}_r \in Q$ *is an extreme point of Q_r, $\forall r \in R$. Hence, it only needs to be shown that the extreme points of $Q_r \in \Delta_{\hat{P}}$.*

Take an arbitrary point extreme point of Q_r, then $|C|$ linearly independent constraints must be active. Since, $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} = 1$, $|C|$-1 of the $n_{c,r} \geq 0$ constraints must be active meaning $|C|$-1 entries of $\mathbf{n}_{c,r} = 0$ for a given analyst r. Hence, $n_{c,r} > 0$ for only one entry and given $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} = 1$ implies that $n_{c,r} = w_a$. As this is integer, this point is also a pure strategy. Therefore, $Q_r \in \Delta_{\hat{P}}$ and by extension $Q \in \Delta_{\hat{P}}$. In a similar way, one can argue the opposite direction. If $MarginalStrategyLP$ has a valid solution, then a corresponding mixed strategy can be found.

The intuition behind Theorem 5.2 is that when the $T_a^r = \frac{1}{w_a}$ and $w_a \in \mathbb{Z}^+$, the extreme points of the defender's strategy space become integer. This can be seen from the maximum number of alerts each resource is able to resolve. Whenever, $T_a^r = \frac{1}{w_a}$ the number of alerts of a given type a resource can solve will be w_a, which corresponds to an integer assignment. Hence, the defender's marginal strategy space is the same as the defender's mixed strategy space when these conditions are true and the MSLP returns the optimal marginal strategy for the defender.

5.4 CAG Algorithmic Approach

The problem of non-implementability of marginals in security games has been studied in previous research [50, 22], but the non-implementability arose because of spatio-temporal resource constraints and constraints from combining resources into teams. For CDGs, non-implementability arises from the presence of the T_a^r coefficients (an example is discussed later). In this section, an algorithm is presented that takes the initial constraints on a CAG and converts them to ensure the implementability of the marginal strategy. To that end, [23] 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 5.16 and 5.17, 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 5.17, $\{\{c_1, r_1\}, \{c_1, r_2\}\}$ 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 5.17.

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.

5.4.1 Constraint Conversion

The T_a^r coefficients admits possibly non-implementable marginal strategies to be returned. For instance, in Figure 5.2(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 (5.1) and (5.3), force the relevant pure strategy $P_{c_1,r_2} \leq \lfloor 1/0.4 \rfloor = 2$. The column constraints can be converted, 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 5.16) is replaced with $|A|$ constraints:

$$\sum_{c \in C_a} n_{c,r} \leq L_r^{C_a} \tag{5.18}$$

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

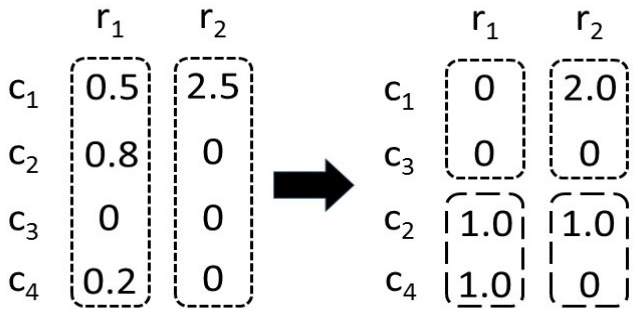


Figure 5.3: Conversion of Column Constraints on CAG

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 This example refers to the example CAG where the marginal strategy is given in Figure 5.3. The column constraints must be converted to a hierarchy. This conversion is shown for r_1 (as r_2 is converted in the same manner). Initially, for r_1 we have the following constraint:

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

The T_a^r coefficients can be removed by grouping together all $n_{c,r}$ which share T_a^r and introducing two new constraints like (5.18). This leads to two new constraints:

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

These new constraints are shown for r_1 in Figure 5.3 on the right of the arrow. Next, the $L_r^{C_a}$ variables must be set. 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, the constraint structure is now a bihierarchy. The original marginals shown in Figure 5.3 do not satisfy these new constraints; but solving the MSLP with these additional constraints in H_2 is guaranteed to give an implementable marginal.

Rounding T_a^r Values In the conversion process, a hierarchy H_2 is created on the column constraints by introducing $|A|$ $L_r^{C_a}$ values for each resource. The conversion process then allows for combinatorially many configurations of the $L_r^{C_a}$ values which satisfy the original capacity constraints for a resource, i.e. Constraint (5.16). To alleviate this search, an algorithm could take advantage of Theorem 5.2 and round each T_a^r to the nearest $\frac{1}{w_a}$ value which is greater than T_a^r where $w_a \in \mathbb{Z}^+$. The marginal strategy $\mathbf{n}$ returned for this modified CAG is then guaranteed to be implementable. However, as is shown next this can lead to a $\frac{1}{2}$ loss for the defender in the worst case.

Counter Example Consider a CAG with one system $K = \{k_1\}$, two alert levels $A = \{a_1, a_2\}$, and one analyst $r = \{r_1\}$. There are two alert categories $C = \{c_1, c_2\}$, where $c_1 = (k_1, a_1)$ and $c_2 = (k_1, a_2)$. For the alert categories we have $N_{c_1} = 1$ and $N_{c_2} = 1$. For r_1, assume $T_{a_1}^{r_1} = 0.5 + \epsilon$ and $T_{a_2}^{r_1} = 0.5 - \epsilon$. If one rounds the T_a^r values up to the nearest $\frac{1}{w_a}$ we would have the following: $T_{a_1}^{r_1} = 1$ and $T_{a_2}^{r_1} = 0.5$.

Now assume the adversary has one attack method m_1 with $E_{m_1}^{r_1} = 1$ and where $\beta_{a_1}^{m_1} = 0.5 + \epsilon$ and $\beta_{a_2}^{m_1} = 0.5 - \epsilon$. Assume $U_{\delta,c_1}^d = U_{\delta,c_2}^d$ and $U_{\delta,c_1}^u = U_{\delta,c_2}^u$ where $U_{\delta,c_1}^d > U_{\delta,c_1}^u \geq 0$. The adversary has one choice and hence, chooses to use m_1 to attack system k_1. The modified CAG would then assign the alert in c_1 to r_1 and receive a utility

of $v^{'} = (0.5 - \epsilon)U_{\delta}^{u} + (0.5 + \epsilon)U_{\delta}^{d}$. In the unmodified CAG, however, the defender would be able to assign both alerts to r_1 and therefore, achieve a utility $v^* = U_{\delta}^{d}$. In this case, the worst possible loss from the modification of the CAG happens when $U_{\delta}^{u} = 0$. This results in the following:

$$\frac{v^{'}}{v^*} = \frac{(0.5 - \epsilon)U_{\delta}^{u} + (0.5 + \epsilon)U_{\delta}^{d}}{U_{\delta}^{d}} \leq \frac{(0.5 + \epsilon)U_{\delta}^{d}}{U_{\delta}^{d}}$$

In the worst case, $v^{'} = (0.5 + \epsilon)v^*$. Hence, rounding the T_a^r values means the defender can lose up to $\frac{1}{2}$ of the optimal utility. This amount of loss is not acceptable in cyber security domains which have highly sensitive targets and therefore, algorithms must be devised which provide better solutions that mitigate this loss.

5.4.2 Branch-and-Bound Search

So far, it has seen that a marginal strategy $\mathbf{n}$ for a CAG output from the MSLP may be non-implementable. The 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, one may need to search through the combinatorially many ways to convert the constraints of CAG to a bihierarchy. Previous work [22] 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 a novel branch-and-bound search is proposed: 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, are the original constraints (13) and (14); running MSLP potentially yields a non-implementable marginal strategy $\mathbf{n}$. Then branches are created from this root, where at each level in the tree, new constraints are added 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, the capacity constraint of some analysts have been substituted, and for these, we have constraints of type (5.18), but for others, we still have constraint (5.16). This set of constraints does not form a hierarchy H_2 as T_a^r coefficients are present in some analyst constraints. Still, each intermediate node gives an upper bound on the defender's utility v which is stated in Proposition 5.2, as each conversion from (5.16) to (5.18) introduces new constraints on the defender's strategy space.

Proposition 5.2 *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.*

Proof 5.4 *In an intermediate node there are two types of column constraints present: (1) $\sum_{a \in A} \sum_{c \in C_a} T_a^r n_{c,r} \leq 1$ and (2) $\sum_{a \in A} n_{c,r} \leq L_r^{C_a}$. At the next level of the tree*

a constraint of the first type is replaced with constraints of the second type. These new constraints restrict the defender's marginal strategy space and hence, the defender's utility v will either stay the same or decrease.

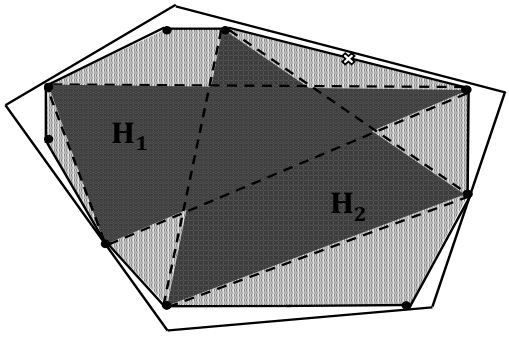
A leaf in the search tree has column constraints only of the form: $\sum_{a \in A} n_{c,r} \leq L_r^{C_a}$. Hence, they form a hierarchy H_2 as all $n_{c,r}$ have unitary coefficients and an integer upper bound. At a leaf, the MSLP can be solved with the resulting bihierarchical constraints to find a lower bound on the defender's utility v. Combining this with Proposition 5.2 gives the components needed for a branch-and-bound search tree which returns the optimal bihierarchy for the defender.

5.4.2.1 Heuristic Search

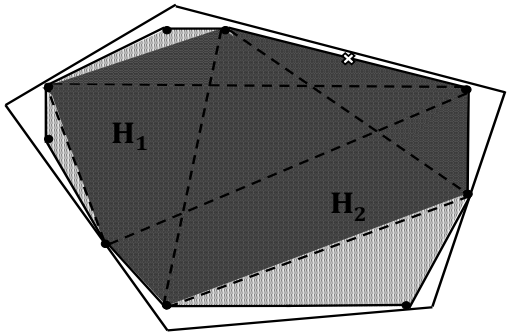
The full branch-and-bound procedure struggles with large CAG. To find good bihierarchies, one can take advantage of the optimal marginal strategy $\mathbf{n}^*$ returned from $MSLP$ at an intermediate node to reduce the amount of branching done. The intuition for this strategy, is that the optimal bihierarchy either contains, or is near, $\mathbf{n}^*$. For example, in the conversion done in Figure 5.3, the $L_r^{C_a}$ values can be set close to $\mathbf{n}$. 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, the following rules are used 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.



(a) Individual Bihierarchies



(b) Convex Hull

Figure 5.4: Geometric view of the defender's strategy space.

5.4.2.2 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. In Figure 5.4 a geometric representation of the defender's strategy space is shown. Individual points represent the defender's pure strategies and the region contained in the convex hull of these points is the defender's mixed strategy space, while the outer region represents the defender's relaxed marginal strategy space. Figure 5.4(a) shows how individual bihierarchies capture portions of the defender's mixed strategy space represented by the shaded regions enclosed by the dashed lines. Figure 5.4(b) shows that by taking the convex hull of the two bihierarchies H_1 and H_2 the size of the defender's strategy space can be increased without generating any new bihierarchies. Note, as each bihierarchy is implementable, the convex hull will also be implementable [22].

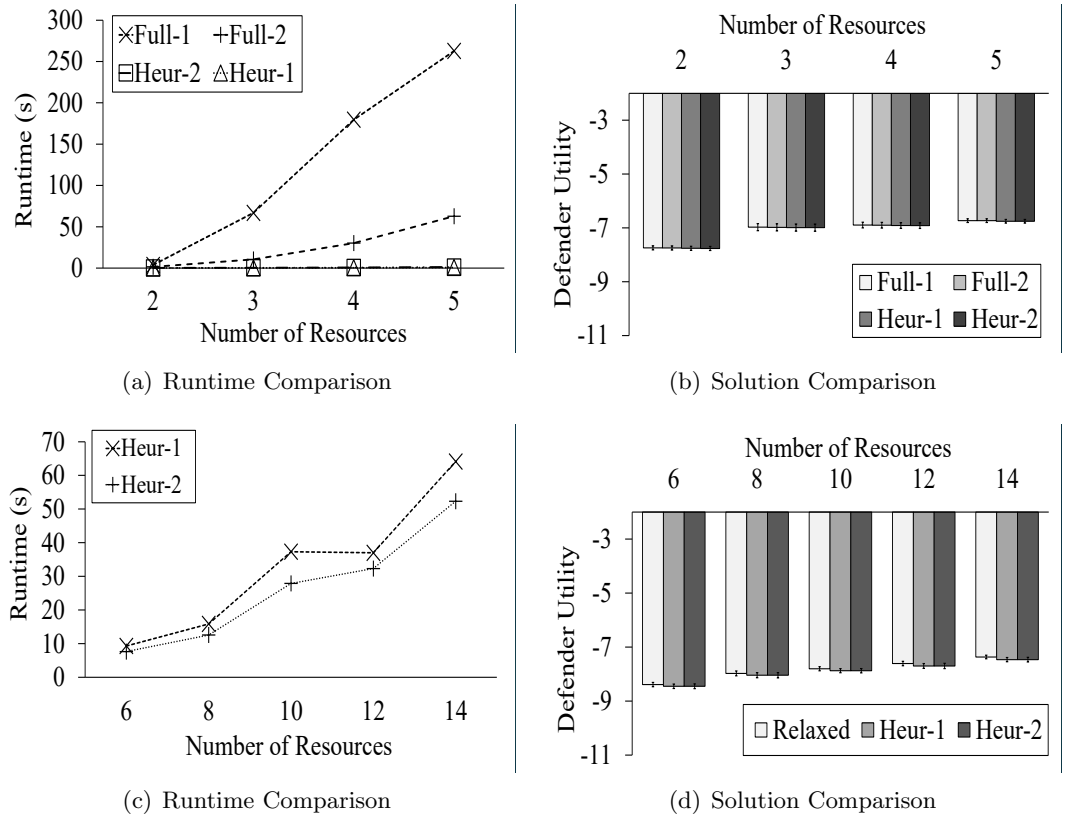
To take the convex hull, first notice each bihierarchy H_i is a set of linear constraints and can be written as $D_i n \leq b_i$ for matrix D_i and vector b_i. Hence, by definition $\mathbf{n}(H_i) = \{\mathbf{n} | D_i \mathbf{n} \leq b_i\}$. Using a result from [11] that represents the convex hull using linear constraints, one can write: $conv(\mathbf{n}(H_1), \ldots, \mathbf{n}(H_l)) = \{\mathbf{n} | \sum_i \mathbf{n}_i, D_i \mathbf{n}_i \leq \lambda_i b_i, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$. This allows for the convex hull of the bihierarchies to be computed efficiently using an LP similar to MSLP.

In terms of the convex hull there are 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, the convex hull of the top two bihierarchies is taken which gives a utility $v^{'}$ to the defender.

Bihierarchies are added to the convex hull while the utility v' returned increases by at least some ϵ, and stops otherwise.

5.5 Evaluation

The CAG model and solution algorithms are evaluated with experiments inspired by the operations of the AFCYBER. The game payoffs are set to be zero-sum, i.e. $U^u_{\delta,c} = -U^u_{\theta,c}$, and the defender's payoffs are randomly generated with $U^u_{\delta,c}$ uniformly distributed in $[-1, -10]$. The rest of the game payoffs, $U^d_{\delta,c}$ and $U^d_{\theta,c}$, are set to be zero. Each experiment is averaged over 30 randomly generated game instances.



(a) Runtime Comparison

(b) Solution Comparison

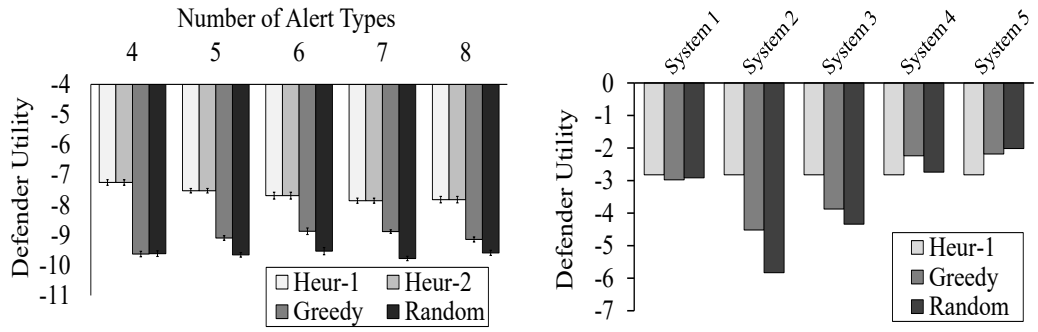
(c) Runtime Comparison

(d) Solution Comparison

Figure 5.5: Experimental Results for CAG instances.

5.5.1 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, the performance of the full branch-and-bound search (Full) is compared to the heuristic search (Heur). For this experiment two variations of the full search are tested: Full-1 which uses the full convex hull and Full-2 which uses the iterative convex hull. The Heuristic search the same two variations are tested, labeled as Heur-1 and Heur-2. Each instance has 20 systems, 3 attack methods, and 3 alert types.



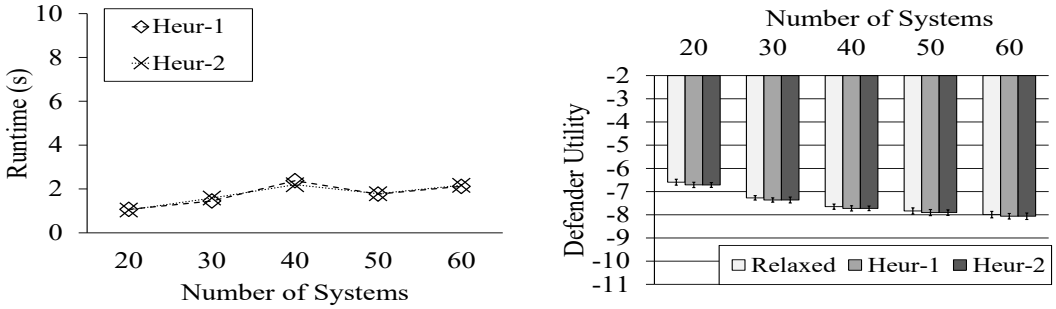
(a) Runtime Comparison

(b) Solution Comparison

Figure 5.6: Allocation Approach Comparison.

In Figure 5.5(a) the number of resources are varied on the x-axis and the runtime in seconds is shown 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 5.5(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 the heuristic significantly improves runtime without sacrificing solution quality.



(a) Runtime Comparison

(b) Solution Comparison

Figure 5.7: Scaling Number of Systems

5.5.2 Solving large CAG

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

In Figure 5.5(c) the runtime results are shown 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 the number of analysts is increased from 6 to 14. In Figure 5.5(d) the solution quality results are shown with the number of analysts on the x-axis and the defender's expected utility on the y-axis. The solution quality is compared 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 game theoretic approaches scale to large CAG without sacrificing much solution quality.

In the next experiment, the number of systems that have to be protected are varied. For this experiment the defender has 5 cyber experts to assign. Figure 5.7 shows the runtime and solution quality results. In Figure 5.7(a) the number of systems are varied on the x-axis and the runtime in seconds is shown on the y-axis. For instance, for 50 systems Heur-1 takes an average of 1.78 seconds to finish running. This graph shows Heur-1 and Heur-2 show no issues in scaling to a larger number of systems. In Figure 5.7(b) the x-axis shows the number of systems while the y-axis gives the defender's expected utility. The solution quality is again compared to the MSLP solution. In all cases, the heuristic approaches suffer only a small loss in defender expected utility compared to the MSLP value. As can be seen from the results, the heuristic approaches scale to CAG with a larger number of systems without sacrificing much in the way of solution quality.

5.5.3 Allocation Approach

The last experiment aim to show that the game theoretic apporach for CAGs outperform approaches used in practice. In addition to the heuristic approach, a greedy approach which investigates the highest priority alerts from the most critical bases first and a random approach for the allocation are used for comparison. The parameters for this experiment are 20 systems, 5 attack methods, and 10 analysts. In Figure 5.6(a) the solution quality results are shown with the number of alerts on the x-axis and defender's utility on the y-axis. 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 5.6(b), a solution comparison for a specific CAG instance is shown. This graph gives intuition for why the game theoretic approach performs so well. The greedy and random approaches tend to overprotect some systems (system 4) while leaving others without adequate protection (system 2).

5.6 Chapter Summary

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

Chapter 6

General-sum Threat Screening Games

6.1 Problem Domain

Extending the CAG and TSG models to general-sum domains is important for real-world applicability of these models as security domains commonly have non-symmetric payoffs between defenders and adversaries. In this chapter, I investigate Bayesian general-sum Threat Screening Games (TSG) and present an approach to solve these games which has additional applicability to the CAG model presented in Chapter 5. It is suggested for the reader to familiarize themselves of the details of the TSG model which is given in the background section. To highlight the usefullness of modeling general-sum payoffs, consider that for an adversary attacking an airport he may receive a positive payoff even when getting caught as he gains notoriety for his attack. The defender, however, would not receive any payoff given that she is only able to stop the attack and no damage occurs. In this chapter, I present an approach that handles the computational complexities

arising from solving TSGs with non zero-sum payoffs which leverages a hierarchical decomposition method for handling the adversary action tree and uses MGA [22] to compute approximate marginal strategies for the defender.

6.2 Approach

While it has been shown that Bayesian Stackelberg games are hard to solve [27], it is interesting to observe that Bayesian general-sum TSGs are hard to solve even in the marginal strategy space as is shown next. This result shows that finding the optimal marginal strategy n for Bayesian general-sum TSGs is fundamentally more complex than the zero-sum case which can be solved in polynomial time as an LP. Thus, it is not surprising that the solution approaches used in [22] are not directly applicable to the general-sum case.

Theorem 6.1 *Finding the optimal solution in Bayesian general-sum TSGs is NP-hard even in the relaxed marginal strategy space.*

Proof 6.1 *The reduction is given from the knapsack problem to the TSG problem. Assume n items with weights w_i and value v_i with a sack of capacity K. Wlog, assume w_i and K are integers. Construct a game with n adversary types $|\Theta| = n$. Each type of adversary has two flights to board: f_0, f_1. Thus, $C = \{(\theta_i, f_j) \mid 0 \leq i \leq n, j \in \{0, 1\}\}$. Choose the other parameters of the game as follows: two resources r_1,r_2 with capacities $L_{r_1} = K$ and $L_{r_2} = \infty$. There have two teams $t_1 = \{r_1\}$ and $t_2 = \{r_2\}$. There is only one attack method m_1. For this game $E^{t_1}_{m_1} = 1$ and $E^{t_2}_{m_1} = 0$, i.e., t_2 is not effective at*

all at detecting m_1. *The number of screenees for each screenee category* $N_{(\theta_i,f_0)} = w_i$ *and* $N_{(\theta_i,f_1)} = 1$ *for all* $\theta_i \in \Theta$. *Each type* θ_i *occurs with probability* $\frac{v_i}{\sum_{\theta_i} v_i}$.

The utilities for the screener are $U^d_{s,(\theta_i,f_0)} = 0, U^u_{s,(\theta_i,f_0)} = 0, \forall\ \theta_i$ *and* $U^d_{s,(\theta_i,f_1)} = 2, U^u_{s,(\theta_i,f_1)} = 1, \forall\ \theta_i$. *Thus, the screener strictly prefers the adversary of every type to choose* f_1. *The utilities for the adversary are set as follows:* $U^d_{\Theta,(\theta_i,f_0)} = 1, U^u_{\Theta,(\theta_i,f_0)} = 2, \forall\ \theta_i$ *and* $U^d_{\Theta,(\theta_i,f_1)} = 0, U^u_{\Theta,(\theta_i,f_1)} = 1, \forall\ \theta_i$. *Thus, the adversary will choose* (θ_i, f_1) *only when the probability of detection* $x_{(\theta_i,f_0)} = 1, x_{(\theta_i,f_1)} = 0$ *(breaking ties in favor of the screener). This happens only when all of the screenees* $N_{(\theta_i,f_0)} = w_i$ *are screened by* t_1. *Thus, we have from the capacity constraints that* $\sum_{\theta_i \text{ chooses } f_1} w_i \leq K$. *Therefore, for the choice of* f_1 *by adversary of type* θ_i, *the screener earns* $\frac{v_i}{\sum_{\theta_i} v_i}$ *and otherwise the screener earns* 0. *Given, the optimization problem maximizes this utility:* $\sum_{\theta_i \text{ chooses } f_1} \frac{v_i}{\sum_{\theta_i} v_i}$, *the optimization provides a solution for the knapsack problem.*

The resulting TSG instance from the knapsack problem has two flights, two resources, two teams, and as many adversary types as the number of items n *in the knapsack problem. The screenee types assigned to* t_1 *gives the knapsack solution. Therefore, the reduction from the knapsack problem is overall polynomial in the number of items* n.

The optimal marginal strategy for a Bayesian general-sum TSG can be obtained by solving the mixed integer linear program *MarginalStrategyMILP*, provided below.

$$\max_{\mathbf{n},\mathbf{s},\mathbf{x},\mathbf{a}} \quad]quad \sum_{\theta \in \Theta} z_\theta s_\theta \tag{6.1}$$

$$s.t. \quad s_\theta - U_s(a_{c,m}^{\theta,w}) \leq (1 - a_{c,m}^{\theta,w}) \cdot Z \qquad \forall \theta, w, c, m \tag{6.2}$$

$$0 \leq k_\theta - U_\theta(a_{c,m}^{\theta,w}) \leq (1 - a_{c,m}^{\theta,w}) \cdot Z \qquad \forall \theta, w, c, m \tag{6.3}$$

$$x_{c,m}^w = \sum_{t \in T} E_m^t \frac{n_{c,t}^w}{N_c^w} \qquad \forall w, c, m \tag{6.4}$$

$$\sum_{t \in T} I_r^t \sum_{c \in C} n_{c,t}^w \leq L_r^w \qquad \forall w, r \tag{6.5}$$

$$\sum_{t \in T} n_{c,t}^w = N_c^w \qquad \forall w, c \tag{6.6}$$

$$n_{c,t}^w \geq 0 \qquad \forall w, c, t \tag{6.7}$$

$$a_{c,m}^{\theta,w} \in \{0, 1\} \qquad \forall \theta, w, c, m \tag{6.8}$$

$$\sum_{w,c,m} a_{c,m}^{\theta,w} = 1 \qquad \forall \theta \tag{6.9}$$

Equations (6.8) and (6.9) force each adversary type θ to choose a pure strategy. Equation (6.1) is the objective function which maximizes the screener expected utility as a weighted summation of screener expected utility against adversary type θ, s_θ, multiplied by the probability of encountering adversary type θ, z_θ. Equation 6.2 defines the screener's expected payoff against each adversary type, contingent on the choice of pure strategies by the adversary types. The constraint places an upper bound of $U_s(a_{c,m}^{\theta,w})$ on s_θ, but only if $a_{c,m}^{\theta,w} = 1$, as Z denotes an arbitrarily large constant. For all other pure strategies of adversary type θ, the RHS is arbitrarily large. Similarly, Equation (6.3) places an upper bound of $U_\theta(a_{c,m}^{\theta,w})$ on the utility of adversary type θ, k_θ, for $a_{c,m}^{\theta,w} = 1$. Additionally,

Equation (6.3) also lower bounds k_θ by the largest $U_\theta(a_{c,m}^{\theta,w})$ over all $a_{c,m}^{\theta,w}$. Taken together these upper and lower bounds ensure that the pure strategy selected by adversary type θ is a best response to the screener marginal strategy $\mathbf{n}$. Equations (6.5) and (6.7) requires that $\mathbf{n}$ satisfies the resource type capacity constraints and screenee category assignment constraints.

Even though *MarginalStrategyMILP* represents a NP-hard problem, solving it does not necessarily produce an implementable marginal screener strategy, i.e., the marginal strategy may not map to a probability distribution over pure strategies. The issue of implementability was addressed in [22] for TSGs by introducing the Marginal Guided Algorithm (MGA), which uses the (potentially non-implementable) marginal strategy to additionally restrict the constraints of a TSG to obtain a provably implementable marginal strategy though it can possibly lose solution quality in the process.

6.3 GATE: Solving Bayesian General-Sum TSG

Despite operating in the marginal screener strategy space, solving the *MarginalStrategyMILP* is computationally expensive due to the presence of the integer variables that encode the adversary strategy space. Therefore, to solve Bayesian general-sum TSGs a clever algorithmic is needed to explore the adversary strategy space. An example of the adversary strategy space for two adversary types and two actions per type is shown in Figure 6.1(a). The leaf nodes in this tree represent all possible joint pure strategy combinations for the two adversary types.

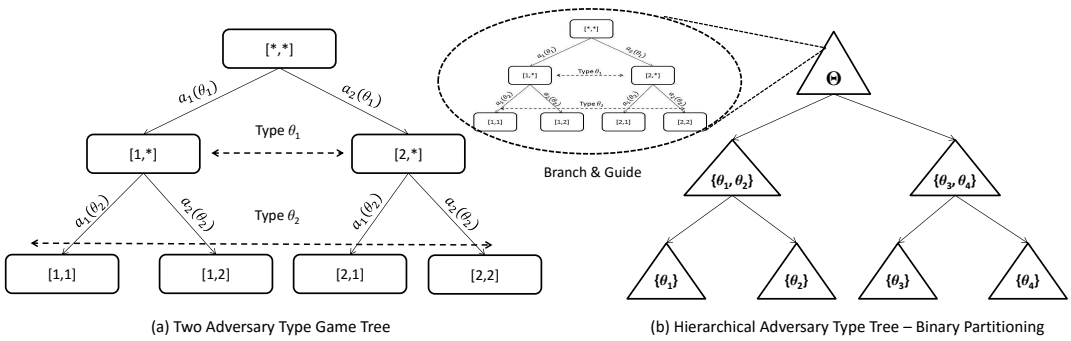


Figure 6.1: Bayesian Adversary Strategy Space

As mentioned previously a standard algorithmic approach that could be used to solve Bayesian general-sum TSGs is Multiple LPs [27]. This technique exploits the fact that, by fixing the joint adversary pure strategy, the underlying optimization problem is converted from a MILP to an LP. Thus, an LP can be solved for each leaf node in the adversary strategy tree to obtain the best marginal screener strategy which induces that joint adversary pure strategy as a best response. The marginal strategy with the highest screener utility is returned as the solution for the TSG. However, the number of joint adversary pure strategies is exponential in the number of adversary types Θ and thus Multiple LPs is not scalable for large problem instances. Therefore, the General-sum Algorithm for Threat screening game Equilibria (GATE) is proposed and in this section an intuitive, high level description is provided. In Section 6.4 a detailed algorithm is given as the experimental results show GATE does not scale leading to the need for heuristics to further speed up computation.

6.3.1 Hierarchical Type Trees

At a high level GATE seeks to exploit the structure of TSGs and reduce the number of joint adversary pure strategies that need to be evaluated. GATE achieves this reduction by building off intuition from the HBSA algorithm [39], which involves constructing a hierarchical type tree. Such a tree decomposes the game with each node in the tree corresponding to a restricted game over a subset of adversary types. The idea is to solve these smaller, restricted games to efficiently obtain (1) infeasibility information to eliminate large sets of joint adversary pure strategies, and (2) utility upper bound information that can be used to terminate the evaluation of joint adversary pure strategies.

GATE operates on restricted TSGs, where $TSG(\Theta^{'})$ is defined to be a TSG with a subset of adversary types $\Theta^{'} \subset \Theta$. It is important to note that, despite not including all adversary types, $TSG(\Theta^{'})$ does not ignore the screenee categories associated with adversary types $\Theta^{\setminus}\Theta^{'}$. Indeed, $TSG(\Theta^{'})$ must still satisfy the constraint that all screenees in each category must be assigned to a screening team type, i.e., Equation 6.6. By continuing to enforce these constraints, the upper bounds generated will be tighter as the screener cannot focus all the screening resources on just a subset of screenee categories, helping to improve the ability of GATE to prune out joint adversary pure strategies.

The subsets of adversary types are decomposed such that each level in the hierarchical type tree forms a partition over Θ satisfying $\Theta_i^{'} \cap \Theta_j^{'} = \emptyset, \forall i, \forall j, i \neq j$ as well as $\cup_i \Theta_i^{'} = \Theta$. Additionally, the set of adversary types in each parent node is the union of the sets of adversary types of all of its children, with the root node of the hierarchical type tree corresponding to the full problem. Figure 6.1(b) shows an example of a hierarchical tree

structure with full binary partitioning for a game with four adversary types. The root node is the parent to two restricted games with two adversary types, each of which is a parent to two restricted games for individual adversary types.

The evaluation of the hierarchical tree starts at the leaf nodes and works up the tree such that all child nodes are evaluated before the parent nodes are evaluated. Every node is processed by evaluating the pure strategies of the restricted game and propagating up only the feasible pure strategies (i.e., pure strategies inducible as an adversary best response). [39] proved that if a pure strategy $a_{\theta'}$ can never be a best response for adversary type θ' in a restricted game $TSG(\Theta')$ with $\Theta' = \{\theta'\}$ then any joint pure strategy containing $a_{\theta'}$ can never be a best response in any $TSG(\Theta'')$ with $\Theta' \subset \Theta''$. Thus, at a given node, it is only necessary to consider joint pure strategies in the cross product of the sets of feasible pure strategies passed up from the child nodes.

For each pure strategy to be propagated, the corresponding utility with respect to the restricted game is also passed up. [39] also proved that it is possible to upper bound the screener utility for joint adversary pure strategy a_Θ in $TSG(\Theta)$ by $\sum_{\theta \in \Theta} z_\theta \beta(a_\theta)$ where z_θ is the normalized probability of adversary type θ in $TSG(\Theta')$ and $\beta(a_\theta)$ is the upper bound on the screener utility for adversary pure strategy a_θ in the restricted game $TSG(\{\theta\})$. These upper bounds can be used to determine the order to evaluate joint pure strategies as well as when it no longer necessary to evaluate joint pure strategies. This propagation of pure strategies and upper bounds continues until the root node is solved to obtain the best solution for the game.

Example Consider a game with four adversary types $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$, like in Figure 6.1(b), where each adversary type has two actions available, $a^1_{\theta_i}, a^2_{\theta_i}$. The full

game is broken down into the restricted games and the leaf nodes in the hierarchical tree are solved first. Suppose that after all of the leaf nodes are solved we get the following action sets for the adversaries: $\theta_1 = \{a^1_{\theta_1}\}$, $\theta_2 = \{a^1_{\theta_2}, a^1_{\theta_2}\}$, $\theta_3 = \{a^1_{\theta_3}\}$, $\theta_4 = \{a^1_{\theta_4}\}$ (as the other actions are found to be infeasible). Then, in the node $\{\theta_1, \theta_2\}$ we evaluate $[a^1_{\theta_1}, a^2_{\theta_2}]$ and $[a^1_{\theta_1}, a^1_{\theta_2}]$ while for node $\{\theta_3, \theta_4\}$ we evaluate $[a^2_{\theta_3}, a^1_{\theta_4}]$. Now, at the root node (Θ) one only has to evaluate two joint adversary pure strategies (i.e., $[a^1_{\theta_1}, a^1_{\theta_2}, a^2_{\theta_3}, a^1_{\theta_4}]$ and $[a^1_{\theta_1}, a^2_{\theta_2}, a^2_{\theta_3}, a^1_{\theta_4}]$) instead of 16 (cross product of all adversary type strategy sets) in order to find the optimal strategy for the game.

6.3.2 Advantages of GATE

As mentioned earlier, security games techniques do not apply to TSGs directly, and thus, HBSA is not well suited for this problem. In particular, HBSA utilizes a Branch-and-Price framework [14] which requires running column generation (given the large number of defender strategies in complex domains) to evaluate every single adversary joint pure strategy in the hierarchical type tree. While Branch-and-Price is a general approach frequently used for Bayesian Stackelberg games, column generation has been shown to be incapable of scaling for large-scale TSGs even in the zero-sum case due to the massive number of screener pure strategies [22]. Thus, having to repeatedly run column generation for the Bayesian general-sum TSGs is a non-starter.

To efficiently evaluate the joint adversary pure strategies in the nodes of the hierarchical tree, Branch-and-Guideis introduced, which combines branch-and-bound search with MGA to simultaneously mitigate the challenges of both scalability and implementability when solving Bayesian general-sum TSGs. Branch-and-Guide may be run at the root

node of the hierarchical type tree, so that given a large of joint adversary pure strategies, a large portion of them can be pruned using upper-bounds, speeding up the computation. Furthermore, Branch-and-Guide exploits the fact that for a fixed joint adversary pure strategy, an implementable marginal screener strategy can be obtained quickly (even if it not necessarily optimal) and thus can avoid having to rely on column generation.

An example of the adversary strategy tree explored in Branch-and-Guide is shown in Figure 6.2, with the size and ordering of the tree based on the feasible joint adversary pure strategies and corresponding upper bounds propagated up by the child nodes. Branches to the left fix the joint adversary pure strategy, converting *MarginalStrategyMILP* into a linear program which can be solved efficiently. However, the resulting marginal strategy may not be implementable and thus MGA is run on the marginal while ensuring that the selected joint adversary pure strategy is still a best response. This two-step process produces an implementable marginal strategy that gives a lower bound on the overall solution quality. Branches to the right represent the upper bound on the screener utility for the next best joint adversary pure strategy which is calculated using the solution quality information passed up from the child nodes. If the screener utility for the best solution found thus far is higher than the upper bound than the next best joint adversary pure strategy, then the execution of Branch-and-Guide can be terminated without exploring the remaining joint adversary pure strategies.

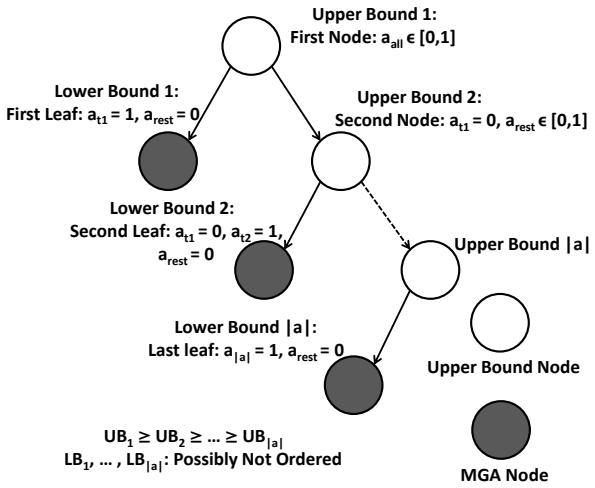


Figure 6.2: Branch-and-Guide Tree

6.4 Scaling Up GATE

While GATE incorporates state-of-the-art techniques to solve *MarginalStrategyMILP*, it fails to scale up to real world problem sizes for TSGs (see comparison in Evaluation). Thus, intuitive heuristics are employed that further narrows down the search space for GATE, thereby enabling up to 10X run time improvement with only 5-10% solution quality loss. There are two distinct steps in GATE where additional heuristics can help: the processing step at the leafs of the hierarchical type tree and the processing step at the intermediate and root nodes. In this section GATE-H (GATE with heuristics) is presented first formally and then the heuristics used to speed up the computation are described.

6.4.1 GATE-H: GATE with Heuristics

GATE-H solves TSGs efficiently by limiting both the number of adversary pure strategies passed up the hierarchical adversary type tree from restricted games and by limiting the

number of adversary strategies evaluated in the individual nodes. Each node in the hierarchical tree is solved using Algorithm 3, beginning at the leaf nodes. The feasible

Algorithm 3: $\mathsf{GATE-H-NODE}(\Theta, \mathsf{A}^{\mathsf{i}}_{\Theta}, \mathsf{B}^{\mathsf{i}}, \mathsf{U_s}, \mathsf{U}_{\Theta}, \mathsf{K})$

```
//A^i_Θ: Pruned feasible pure strategy set for all adversary types
1. A'' := all-Joint-Adversary-Pure-Strategies()
2. B'(a_Θ) := getBound(a_Θ, B^i) ∀a_Θ ∈ ∏_Θ A^i_Θ
3. sort(A'', B'(a_Θ)) //sort a_Θ in descending order of B'(a_Θ)
4. a_Θ := [A^1_Θ(1), A^2_Θ(1), ..., A^|Θ|_Θ(1)]
5. r*, r' = −∞ //Save best and iterative solutions
repeat
    6. (feasible, n, r) := MGA(a_Θ)
       if feasible then
           7. A' := A' ∪ a_Θ
           if r > r* then
               8a. r* := r
               8b. n* := n
           9. B'(a_Θ) := r
       else
           10. A'' := A''\a_Θ
    11. Every K iterations
       if r' = r* ≠ −∞ then
           12. break
       else
           13. r' := r*
    14. a_Θ := getNextStrategy(a_Θ, r*, A^i_Θ, B^i)
until a_Θ = NULL
return (n*, r*, A', B')
```

adversary pure strategy set, denoted as $A^{'}$, is passed up to the parent nodes as each child is solved. Notice that not all strategies need be evaluated at a given node for the computation to terminate as either the Branch-and-Guide heuristic or K cutoff heuristic, both introduced later in this section, can end the computation early.

When solving a given node in GATE-H the adversary set (Θ) is given and the screener's and adversary's utilities (U_s and U_Θ, respectively), with the feasible strategies

(A^i_Θ) and bound information (B^i) is acquired from the children of that node as shown in Algorithm 3. In the case of solving a leaf node in the tree all of that adversary type's strategies are enumerated with bound information given by solving an upper bound LP, described in Section 6.4.2. After constructing the joint adversary pure strategy set (Line 1), the set is ordered by their upper bound values (Line 3) and each strategy is evaluated one by one (main loop starts after Line 5). Heuristics are used inside of this loop, leading to GATE-H.

Algorithm 4: getNextStrategy($a_\Theta, r^*, A^i_\Theta, B^i$)

for $i = |\Theta|$ to 1 **Step**-1 **do**
  j := index-of(a_Θ,A^i_Θ)
  //Set each adversary type strategy equal to left most leaf
  $a^i_\Theta := [A^1_\Theta(1), ..., A^{|\Theta|-1}_\Theta(1), A^{|\Theta|}_\Theta(j+1)]$
  if $r^* <$ getBound(a_Θ, B) **then**
    return a_Θ
return NULL

The first of the two heuristics used is the Branch-and-Guide approach (Line 14 - getNextStrategy()) which ends the computation early if the value of the current best strategy is greater than the next highest upper bound. The second heuristic, discussed in more detail in Section 6.3, is the K cutoff (Line 11) which ends the computation early if the current best solution is not improving after K iterations.

Algorithm 4 describes the *getNextStrategy* function in detail. Essentially the function builds the next strategy to be evaluated by iterating through all of the adversary types and grabbing the highest valued strategy in their respective pure strategy lists.

6.4.2 Tuning Leaf Node Computation

At the leaf nodes in the hierarchical type tree the restricted game for each adversary type is solved. For GATE to be exact, all feasible adversary strategies from each leaf node must be returned. Returning a portion of the feasible pure strategies gives a heuristic approach, which may run well in practice but is not guaranteed to be optimal. Nonetheless, in some cases even for optimality it might suffice for us to return only some promising strategies. Below a special condition is highlighted under which it is optimal to just consider one adversary strategy at each leaf in the hierarchical tree.

Lemma 6.1 *Let $\mathbf{n}_\theta$ represent the optimal allocation against type θ at the leaf. Let $\mathbf{n}_\theta[C_\theta]$ be the part of the allocation that is assigned to screenees in screenee category C_θ. If the strategy $\mathbf{n}$ formed by putting together all $\mathbf{n}_\theta[C_\theta]$: $\mathbf{n} = \sum_\theta \mathbf{n}_\theta[C_\theta]$ is feasible then $\mathbf{n}$ is the optimal defender strategy and the single adversary best response for each single type is the adversary best response in the overall game.*

Proof 6.2 (Proof Sketch) *First, note that the strategy $\mathbf{n}$ achieves the payoff $\sum_\theta z_\theta s^*_\theta$, where s^*_θ is the defender utility in the restricted game with just the type θ. Also, clearly s^*_θ is an upper bound on the defender utility for the restricted game. By the result from [39], the upper bound on the defender utility is $\sum_\theta z_\theta s^*_\theta$ which is achieved by $\mathbf{n}$.*

Branch-and-Guide can be used as described earlier to return fewer promising adversary strategies, but the question that arises when using Branch-and-Guide at the leaf nodes is how to compute the upper bounds for the nodes on the right of the tree (Recall for non-leaf nodes this upper bound is computed from the upper bounds that are propagated up from each child). One approach to compute this upper bound is to adapt

the ORIGAMI [43] approach; ORIGAMI is the fastest technique for solving non-Bayesian general-sum security games without resource scheduling constraints. The underlying idea is to solve an LP to minimize the utility of the adversary by inducing the largest possible attack set (set of targets that are equally and most attractive for the attacker) and [43] shows this provides the optimal defender utility in games without scheduling constraints. However, even for a TSG with a single time window and a single adversary type, ORIGAMI may not provide the optimal solution.

Therefore, ORIGAMI cannot be applied directly to find upper bounds on the adversary pure strategies at the leaf nodes. Thus, *UpperBoundLP* is provided, shown below, to calculate the upper bound at the leaf nodes in the hierarchical tree.

$$\min_{\mathbf{n},\mathbf{q},\mathbf{s},\mathbf{x}} \quad k_{\theta'} \tag{6.10}$$

$$s.t. \quad k_{\theta'} \geq x^w_{c,m} U^d_{\theta',c} + (1 - x^w_{c,m}) U^u_{\theta',c} \;\; \forall w, \forall c, \forall m \tag{6.11}$$

$$x^w_{c,m} = \textstyle\sum_{t \in T} E^t_m \frac{n^w_{c,t}}{N^w_c} \qquad \forall w, \forall c, \forall m \tag{6.12}$$

$$\textstyle\sum_{t \in T} I^t_r \sum_{c \in C} n^w_{c,t} \leq L^w_r \qquad \forall r, \forall w \tag{6.13}$$

$$\textstyle\sum_{t \in T} n^w_{c,t} \leq N^w_c, \quad n^w_{c,t} \geq 0 \qquad \forall c, \forall w \tag{6.14}$$

The objective function (6.10) minimizes the attacker's utility for the adversary type θ'. Equation (6.11) enforces that the adversary payoff be the maximal payoff for the adversary given a marginal strategy $\mathbf{n}$. Equations (6.12)-(6.14) enforce the resource constraints from the original game. This LP uses a slightly modified formulation when enforcing the

capacity constraints. Namely, replace Equation (6.5) with (6.14), i.e., the constraint that every passenger in a given category c for a time window w must be screened is relaxed.

Theorem 6.2 *UpperBoundLP provides an upper bound on the screener utility $d_{\theta'}$ for a non-Bayesian TSG.*

Proof 6.3 *By relaxing constraint* (6.5) *to constraint 6.14, the attack set of the adversary can only be expanded from the original formulation. This happens as Equation 6.14 allows for an adversary to not be screened which can only increase their utility for targets, thus possibly increasing their attack set. Since the adversary breaks ties in the screener's favor this can only increase the screener's possible utility.*

The above approach serves two purposes: it enormously reduces the set of strategies that are sent up to the parent even if all the adversary strategies are evaluated in the Branch-and-Guide tree, as the attack set is much smaller than the set of all feasible strategies. The running time is also reduced, given the small attack set and the efficient LP to obtain this attack set and upper bounds.

6.4.3 Tuning Non-leaf Node Computation

Section 6.4.2 focused on reducing the number of adversary pure strategies returned from the leaf nodes. However, another very important area to prune the search space is at the interior nodes in the binary tree. One way to approach this problem is using Branch-and-Guide in these nodes and stopping the search once the current best solution is better than the next highest upper bound. This can possibly provide significant speed-ups in terms of the computation speed of GATE but it is not quite enough. Unfortunately,

in the experiments it turned out that most of the joint adversary pure strategies were evaluated in the interior nodes by MGA as the stopping condition for Branch-and-Guide was almost never met.

Thus, inspiration is taken from column generation and security games literature [76] where column generation is stopped if the solution quality does not change much with new columns being added. This heuristic almost always provides a very good approximation. This same principle is adopted so that when evaluating adversary strategies in the Branch-and-Guide approach if the current solution quality does not change over the next K strategies then the computation can be stopped and the current solution is declared as the final solution with the adversary strategies evaluated so far propagated to the parent. This approach can also be adopted at the root. Here K is a parameter that can be varied, but $K = 30$ is used for the experiments as it seems to work the best for this problem. This approach serves two purposes: (1) it reduces the run time at each intermediate node and the root node and (2) it reduces the number of adversary pure strategies propagated up the tree.

GATE-H then allows us to solve large-scale Bayesian general-sum TSGs. Further, empirically these heuristics maintain high solution quality while decreasing the runtime by an order of magnitude.

6.5 Evaluation

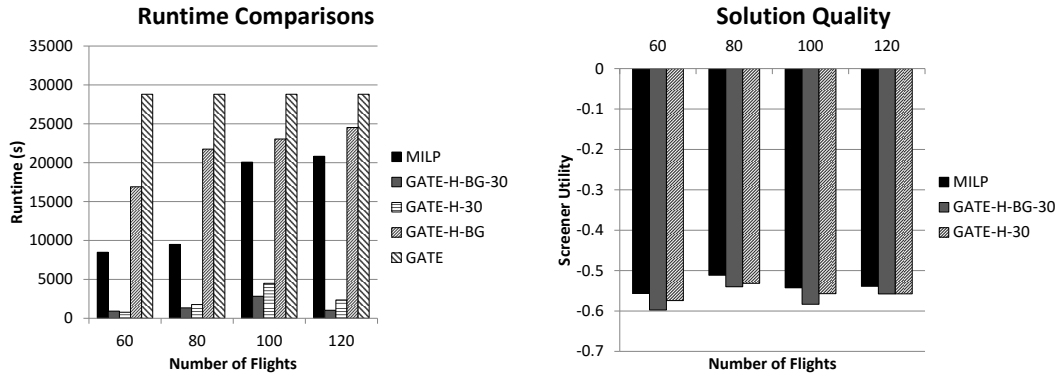
GATE-H is evaluated using synthetic examples from the passenger screening domain as real world data is not available. The LPs and MILPs are solved using CPLEX 12.5

with the barrier method, as this was found to work the best, on USC's HPC Linux cluster limited to one Hewlett-Packard SL230 node with 2 processors. The adversary and screener payoffs are generated uniformly at random with $U_a^u \in [2, 11]$ and $U_s^u \in [-1, -10]$. For both the adversary and the screener $U^d = 0$. The default values for the experiments are 6 adversary risk levels, 5 screening resource types, 10 screening teams, 2 attack methods and 3 time windows unless otherwise specified. For all experiments the capacity resource constraints also remain constant. All results are averaged over 20 randomly generated game instances.

6.5.1 Scaling Up and Solution Quality

The first experiment tests the scalability of each approach and provides solution quality information for GATE-H relative to the *MarginalStrategyMILP*. This experiment provides information about the trade-off between runtime and solution quality for the heuristic algorithm. The four different variations of GATE that were tested are: (1) GATE which evaluates all adversary pure strategies in each of the restricted games and only uses Branch-and-Guide at the root, (2) GATE-H-BG which uses the Branch-and-Guide heuristic in all nodes, (3) GATE-H-K which uses the $K = 30$ cutoff in all nodes and (4) GATE-H-BG-K which uses both Branch-and-Guide and the K cutoff in all nodes. In Figure 6.3(a) the runtime results are shown for all of the algorithms. On the x-axis the number of flights are varied from 60 to 120 in increments of 20. On the y-axis is the runtime in seconds. For example, for 80 flights *MarginalStrategyMILP* takes almost 10,000 seconds to finish. GATE did not finish in any of the instances showing it cannot scale to

large TSG instances. GATE-H-BG also does not perform well as it fails to beat the average runtime of the MILP over all of the instances. As can be seen GATE-H-BG-30 and GATE-H-30 significantly reduce the runtime with an average of a 10 fold improvement over all cases and GATE-H-BG-30 providing a 20 fold speed up at 120 flights.



(a) Runtime Comparison

(b) Solution Comparison

Figure 6.3: In Figure 6.3(a) is a runtime comparison between the MILP and GATE. In Figure 6.3(b) is a solution quality comparison of the MILP and GATE.

In Figure 6.3(b) the solution quality of the MILP is compared with GATE-H-BG-30 and GATE-H-30. GATE and GATE-H-BG are not included as they did not finish in a majority of instances making it difficult to compare the solution quality. On the x-axis the number of flights are varied from 60 to 120 in increments of 20. On the y-axis is the screener's utility. For instance, for 60 flights GATE-H-BG-30 returns an average screener utility of -0.5974. As the graph shows the average solution quality loss over these game instances is always less than .0411 for both GATE-H-BG-30 and GATE-H-30 compared to the MILP. These results show that both GATE-H-BG-30 and GATE-H-30 provide good approximations for large scale TSGs.

The next experiment aimed to test the ability of GATE-H-BG-30 and GATE-H-30 to scale up to much larger TSG instances. The results are shown in Figure 6.4. On the x-axis

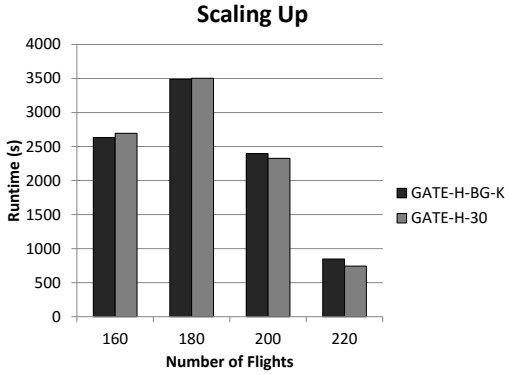


Figure 6.4: Scaling Up to Larger TSG Instances

the number of flights is increased from 160 to 220. On the y-axis the average runtime in seconds is shown to solve each of the TSG instances. For example, GATE-H-BG-30 took on average 2,396 seconds to solve a game with 200 flights. An interesting trend here is that the runtime peaks at 180 flights and starts to decrease afterward. This trend could be related to the resource saturation problem as seen in other security games [38], where the observation is that resource optimization is easiest when the resources available are comparatively small or equal to the number of targets and becomes difficult when this is not the case.

6.5.2 Moving Towards Zero Sum

The last experiment aimed to test what happens to the solution quality of GATE-H-BG-30 and GATE-H-30 as the game payoffs move toward zero-sum. For this experiment, 40 flights are used and there is one time window as the MILP does not scale in these instances. An r-coefficient is varied from 0 to -0.9 in increments of -0.1, where $r = 0$ means there is no correlation between the attacker's and screener's payoffs and $r = -1$ means the game is zero-sum (Note: -1 is not tested as there is a specialized algorithm to deal with

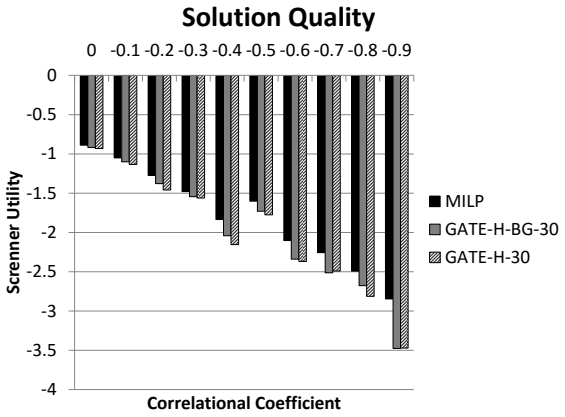


Figure 6.5: Solution Quality - Moving to Zero Sum

that case where the techniques in this chapter are not useful). In previous experiments a correlation between the adversary's and screener's payoffs is not explicitly set. On the x-axis is the r-coefficient and the y-axis shows the screener's utility. For example, when $r = 0$ the *MarginalStrategyMILP* returns a screener utility -0.889, GATE-H-BG-30 returns a screener utility -0.917 and GATE-H-30 returns a solution quality of -0.931. In this experiment an interesting trend appears. As the game moves toward zero-sum payoffs the relative performance of both GATE-H-BG-30 and GATE-H-30 progressively worsens. However, until the game payoffs are nearly zero-sum ($r = -0.9$) both GATE-H variations do have a solution quality loss greater than 11.385%. This experiment again shows that both GATE-H-BG-30 and GATE-H-30 provide good approximations in general-sum TSGs. (A careful reader might notice in Figures 6.3(b) and 6.3(b) that there are slight differences in solution quality between GATE-H-BG-30 and GATE-H-30, however these are not statistically significant.)

6.6 Chapter Summary

The TSG model provides an extensible and adaptable model for game-theoretic screening in the real world. It improves upon previous models in security games that fail to capture important properties of the screening domain, e.g., the presence of non-player screenees in the game and complex team capacity constraints. The model also improves on work done on threat screening, such as screening stadium patrons [68], cargo container screening [6], and screening airport passengers [58, 57].

Previous work done on TSGs [22] focused on the bayesian zero-sum case and in this chapter zero-sum TSGs are extended to the Bayesian general-sum case. Four contributions are provided to accomplish this task: (1) the GATE algorithm which efficiently solves large scale Bayesian general-sum TSGs, (2) the Branch-and-Guide approach which combines branch-and-bound search and MGA in order to efficiently solve nodes in the hierarchical tree, (3) heuristics that speed up the computation of GATE, and (4) experimental evaluation of GATE showing the scalability of the game theoretic algorithm.

Using the aforementioned contributions this chapter presents a practical approach for solving Bayesian general-sum TSGs that scales up to problem sizes encountered in the real world. Thus, with this research I hope to increase the applicability of TSGs by providing techniques for solving large scale Bayesian general-sum TSGs.

Chapter 7

Conclusion

7.1 Contributions

Protecting an organization's cyber assets from malicious actors is one of the most significant and pressing challenges for cybersecurity in the coming years. Previous research completed in cybersecurity has presented numerous approaches for resolving and mitigating problems the defender faces throughout the phases of the Cyber Kill Chain. As alluded to earlier, however, these approaches do not adequately consider the adversarial component of cybersecurity which is crucial in devising strong protection strategies against motivated and intelligent adversaries. My thesis explores the use of game theory in the cybersecurity domain and resolves deficiencies in previous game theoretic approaches for security domains to handle the complex challenges in allocation problems faced by a network administrator. Crucially, my thesis advances the state of the art in game theoretic approaches and algorithms for deceiving cyber adversaries and prioritizing cyber alert resolution which provides a network administrator with more tools to thwart cyber adversaries.

First, to deceive cyber adversaries I introduce the novel Cyber Deception Game model that provides a basis for strategic deception of an intelligent and motivated adversary conducting reconnaissance on an enterprise network. This model gives an additional tool to the network administrator which allows them to thwart an adversary in the initial phase - the reconnaissance phase - of the Cyber Kill Chain and increases the difficulty of successfully breaching the defender's network. To solve this problem, I show how to formulate the adversary's knowledge acquisition phase in a game framework. This gives us a way to then strategically interact with an adversary attempting to hack our network, and most importantly, provides a way to strategically deceive the adversary. I take (1) a robust approach to solve this problem by making the assumption that that we are facing a powerful adversary, e.g., a nation-state, who has distributional information about the defender's deceptive response scheme and (2) I consider a naive adversary, e.g., a script kiddie, with a fixed set of preferences over the set of possible responses from systems on the network. This initial formulation provides valuable insight into future work which will investigate the interesting problem of varied adversarial knowledge states of the defender's network.

Second, for prioritizing alerts raised from IDS across a defender's network I provide the novel Cyber-alert Allocation Game (CAG) model. This model gives a framework for how to randomize the resolution of alerts by cyber analysts and makes it more difficult for a stealthy adversary's attack to go undetected. To achieve this advance, I solve fundamental problems in the field of game theory for threat screening. An important previous model, the Threat Screening Game (TSG), introduced a zero-sum Stackelberg game formulation to solve a problem where a defender must allocate a set of "screening"

resources to detect a possible attack coming into a secure area that attempts to pose as a regular, i.e., "non-adversarial", object. The CAG model makes two important advances by considering screening resources with heterogeneous screening times for the incoming objects and allows for an attack by the adversary to appear as probability distributions over alert types. The CDG framework provides a network administrator with a useful tool in impeding cyber adversaries and protecting from massive breaches of sensitive user information.

Lastly, I extend the applicability of the CAG framework to non zero-sum domains with the introduction of the GATE algorithm. This work additionally applies to physical security domains as it significantly improves the scalability of computing optimal SSE equilibria in Bayesian general-sum TSGs. The GATE algorithm leverages the Marginal Guided Algorithm (MGA) given in [22] which solves for an optimal marginal strategy for the defender in Bayesian zero-sum TSGs. From here, I use a branch and bound algorithmic approach and a hierarchical decomposition adversary type tree to prune a large portion of the adversary's strategy space and solve larger-scale Bayesian general-sum TSGs. Importantly, GATE allows for the application of Bayesian general-sum TSG to problems with non-symmetric payoffs between the defender and an adversary which applies to a wide array of screening domains. Looking to future work, it is important to solve other difficult challenges arising for varying screening domains and problems in cybersecurity along with other threat screening domains, and my work provides significant results to broaden the applicability of the current state-of-the-art.

7.2 Future Work and Directions

One of the most interesting areas for future work beyond my thesis lies in the application of the deceptive techniques to several other domains in cybersecurity. In this thesis, I explored the use of deception for thwarting adversaries in the reconnaissance phase of a cyber attack. These ideas, however, could be extended to conceal the true features of a network to adversaries who may have already compromised systems in the defender's enterprise network. In this situation, the adversary would leverage the compromised nodes to complete further recon of the defender's network to identify machines which may be hidden from the outside. Deception here then could be used to alter the network views [26] which are observed through network reconnaissance conducted from the compromised nodes. In this way, the defender increases the uncertainty throughout all phases of an adversary's attack from recon to moving throughout the defender's network to identify important systems to compromise and subnets in the enterprise network.

In the future, it will also be important to better quantify the advantage gained by the defender utilizing deceptive algorithms and techniques. This can be achieved through the development of a test-bed network scenario where different deceptive algorithms can be tested against human players. To this end, there has been some initial ground work on developing a network scenario with a realistic network recon experience using the Cyber-VAN test-bed [25]. This environment makes it easier as well to study how an adversary plays the game of network reconnaisance. With this information, future research can learn strategies employed by cyber adversaries during reconnaissance and gives better

insight into quantifying the informational gain for adversaries from network recon and attacks.

Beyond network defense, another interesting area of application lies in applying deceptive principles to mitigate social engineering attacks which represents an area of significant importance for cybersecurity defenses. Organizations that suffer from targeted social engineering attacks could leverage the use of deception to make it difficult for an adversary to ascertain the true employees within an organization. This allows a defender to recognize when a particular department, e.g., financial department, is under attack from an adversary's campaign and mount a better defense to protect against breaches by sending out timely warnings to its employees.

My work on cyber alert allocation is a crucial first step in applying game theory to real world cyber security settings, but there remain significant challenges which need to be addressed in future work. Firstly, I assume the time to resolve an alert is known exactly, but in the real world there is uncertainty for how long it would take to resolve an alert. Second, the CAG model assumes that attacks show up as known alert categories, but it is possible that in the real-world some attacks may show up as "unknown" categories. The question then is how to assign these alerts to analysts given we do not know which expert may have an expertise in dealing with this type of attack. Lastly, in CAG's there is not an overflow of alerts from one time period to the next. Overflow of passengers for screening has been explored in the context of TSGs [56], but in the context of cybersecurity, overflow also includes the possibility of unexpected alerts flooding the system along with the expected alerts. In the real-world resolving alerts in a timely manner is crucially important to limit the possible damage from a network intrusion.

Bibliography

[1] Massimiliano Albanese, Ermanno Battista, and Sushil Jajodia. A deception based approach for defeating os and service fingerprinting. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 317–325. IEEE, 2015.

[2] Massimiliano Albanese, Ermanno Battista, and Sushil Jajodia. Deceiving attackers by creating a virtual attack surface. In *Cyber Deception*, pages 169–201. Springer, 2016.

[3] Mohammed H Almeshekah and Eugene H Spafford. Planning and integrating deception into computer security defenses. In *Proceedings of the 2014 Workshop on New Security Paradigms Workshop*, pages 127–138. ACM, 2014.

[4] Mohammed H Almeshekah and Eugene H Spafford. Cyber security deception. In *Cyber Deception*, pages 25–52. Springer, 2016.

[5] Tansu Alpcan and Tamer Başar. *Network security: A decision and game-theoretic approach*. Cambridge University Press, 2010.

[6] Saket Anand, David Madigan, Richard Mammone, Saumitr Pathak, and Fred Roberts. Experimental analysis of sequential decision making algorithms for port of entry inspection procedures. In *Intelligence and Security Informatics*, pages 319–330. Springer, 2006.

[7] James P Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[8] Michael J Assante and Robert M Lee. The industrial control system cyber kill chain. *SANS Institute InfoSec Reading Room*, 1, 2015.

[9] Patrice Auffret. Sinfp, unification of active and passive operating system fingerprinting. *Journal in computer virology*, 6(3):197–205, 2010.

[10] Erik B Bajalinov. *Linear-Fractional Programming Theory, Methods, Applications and Software*, volume 84. Springer Science & Business Media, 2013.

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

[12] Daniel Barbara, Julia Couto, Sushil Jajodia, and Ningning Wu. An architecture for anomaly detection. In *Applications of Data Mining in Computer Security*, pages 63–76. Springer, 2002.

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

[14] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[15] Nicola Basilico and Nicola Gatti. Automated abstractions for patrolling security games. In *AAAI*, 2011.

[16] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184:78–123, 2012.

[17] Jay Beale, Renaud Deraison, Haroon Meer, Roelof Temmingh, and Charl Van Der Walt. *Nessus network auditing*. Syngress Publishing, 2004.

[18] David Barroso Berrueta. A practical approach for defeating nmap os- fingerprinting. *Retrieved March*, 12:2009, 2003.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] Paxson V Bro. A system for detecting network intruders in real-time. In *Proc. 7th USENIX Security Symposium*, 1998.

[22] 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.

[23] 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.

[24] Thomas E Carroll and Daniel Grosu. A game theoretic investigation of deception in network security. *Security and Communication Networks*, 4(10):1162–1172, 2011.

[25] Ritu Chadha, Thomas Bowen, Cho-Yu J Chiang, Yitzchak M Gottlieb, Alex Poylisher, Angello Sapello, Constantin Serban, Shridatt Sugrim, Gary Walther, Lisa M Marvel, et al. Cybervan: A cyber security virtual assured network testbed. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 1125–1130. IEEE, 2016.

[26] Cho-Yu J Chiang, Yitzchak M Gottlieb, Shridatt James Sugrim, Ritu Chadha, Constantin Serban, Alex Poylisher, Lisa M Marvel, and Jonathan Santos. Acyds: An adaptive cyber deception system. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 800–805. IEEE, 2016.

[27] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

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

[29] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.

[30] Roberto Di Pietro and Luigi V Mancini. *Intrusion detection systems*, volume 38. Springer Science & Business Media, 2008.

[31] 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.

[32] Karel Durkota, Viliam Lisỳ, Branislav Bosanskỳ, and Christopher Kiekintveld. Optimal network security hardening using attack graph games. In *IJCAI*, pages 526–532, 2015.

[33] 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.

[34] William B Haskell, Debarun Kar, Fei Fang, Milind Tambe, Sam Cheung, and Elizabeth Denicola. Robust protection of fisheries with compass. In *AAAI*, pages 2978–2983, 2014.

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

[36] 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.

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

[38] Manish Jain, Kevin Leyton-Brown, and Milind Tambe. The deployment-to-saturation ratio in security games. *Target*, 1(5):5, 2012.

[39] Manish Jain, Milind Tambe, and Christopher Kiekintveld. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[40] Sushil Jajodia, Noseong Park, Fabio Pierazzi, Andrea Pugliese, Edoardo Serra, Gerardo I Simari, and VS Subrahmanian. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security*, 12(11):2532–2544, 2017.

[41] Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 207–214. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[42] Rob Joyce. Disrupting nation state hackers. San Francisco, CA, 2016. USENIX Association.

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

[44] Christopher Kiekintveld, Viliam Lisỳ, and Radek Píbil. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer, 2015.

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

[46] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Security games with multiple attacker resources. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 273, 2011.

[47] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 2011.

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

[49] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon D Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI*, pages 958–964, 2015.

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

[51] Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L Isbell. Computing optimal strategies to commit to in stochastic games. In *AAAI*, 2012.

[52] Kong-wei Lye and Jeannette M Wing. Game strategies in network security. *International Journal of Information Security*, 4(1-2):71–86, 2005.

[53] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[54] Mandiant. Apt1: Exposing one of chinas cyberespionage units, 2013.

[55] Lockheed Martin. Cyber kill chain®. *URL: http://cyber. lockheedmartin. com/hubfs/Gaining_the_Advantage_Cyber_Kill _Chain. pdf*, 2014.

[56] Sara Marie Mc Carthy, Phebe Vayanos, and Milind Tambe. Staying ahead of the game: adaptive robust optimization for dynamic allocation of threat screening resources. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3770–3776. AAAI Press, 2017.

[57] Laura A McLay, Adrian J Lee, and Sheldon H Jacobson. Risk-based policies for airport security checkpoint screening. *Transportation science*, 44(3):333–349, 2010.

[58] Xiaofeng Nie, Rajan Batta, Colin G Drury, and Li Lin. Passenger grouping with risk levels in an airport security system. *European Journal of Operational Research*, 194(2):574–584, 2009.

[59] NIST. *National Vulnerability Database*, 2017. `https://nvd.nist.gov/`.

[60] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[61] 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.

[62] Jeffrey Pawlick and Quanyan Zhu. Deception by design: evidence-based signaling games for network defense. *arXiv preprint arXiv:1503.05458*, 2015.

[63] Radek Pıbil, Viliam Lisỳ, Christopher Kiekintveld, Branislav Bošanskỳ, and Michal Pechoucek. Game theoretic model of strategic honeypot selection in computer networks. *Decision and Game Theory for Security*, 7638:201–220, 2012.

[64] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[65] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards: game theoretic security allocation on a national scale. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 37–44. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[66] Niels Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop, Hamburg, Germany*, volume 2, page 4, 2003.

[67] Antonia Rana. What is amap and how does it fingerprint applications. *SANS Institute*, 2014.

[68] Brian C Ricks, Brian Nakamura, Alper Almaz, Robert DeMarco, Cindy Hui, Paul Kantor, Alisa Matlin, Christie Nelson, Holly Powell, Fred Roberts, et al. Modeling the impact of patron screening at an nfl stadium. In *IIE Annual Conference. Proceedings*, page 3086. Institute of Industrial Engineers-Publisher, 2014.

[69] 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.

[70] Rahul Savani and Bernhard Von Stengel. Exponentially many steps for finding a nash equilibrium in a bimatrix game. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 258–267. IEEE, 2004.

[71] Aaron Schlenker, Matthew Brown, Arunesh Sinha, Milind Tambe, and Ruta Mehta. Get me to my gate on time: Efficiently solving general-sum bayesian threat screening games. In *ECAI*, pages 1476–1484, 2016.

[72] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Milind Tambe, Phebe Vayanos, Fei Fang, Long Tran-Thanh, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. In *International Conference on Autonomous Agents and Multiagent Systems*, 2018.

[73] Aaron Schlenker, Haifeng Xu, Mina Guirguis, Chris Kiekintveld, Arunesh Sinha, Milind Tambe, Solomon Sonya, Darryl Balderas, and Noah Dunstatter. Dont bury your head in warnings: a game-theoretic approach for intelligent allocation of cyber-security alerts. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 381–387. AAAI Press, 2017.

[74] Edoardo Serra, Sushil Jajodia, Andrea Pugliese, Antonino Rullo, and VS Subrahmanian. Pareto-optimal adversarial defense of enterprise systems. *ACM Transactions on Information and System Security (TISSEC)*, 17(3):11, 2015.

[75] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 13–20. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[76] Eric Shieh, Albert Xin Jiang, Amulya Yadav, Pradeep Varakantham, and Milind Tambe. Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *European Conference on Artificial Intelligence (ECAI)*, 2014.

[77] 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.

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

[79] VS Subrahmanian, Michael Ovelgonne, Tudor Dumitras, and B Aditya Prakash. *The Global Cyber-Vulnerability Report*. Springer, 2015.

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

[81] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[82] Bernhard Von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Technical report, Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

[83] Xiaowen Wang, Cen Song, and Jun Zhuang. Simulating a multi-stage screening network: A queueing theory and game theory application. In *Game Theoretic Analysis of Congestion, Safety and Security*, pages 55–80. Springer, 2015.

[84] 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.

[85] Haifeng Xu. The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. EC '16, New York, NY, USA, 2016. ACM.

[86] Fyodor Yarochkin, Meder Kydyraliev, and Ofir Arkin. Xprobe project, 2014.

[87] Zhengyu Yin, Albert Xin Jiang, Matthew Paul Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012.

[88] 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.

[89] Dajun Yue, Gonzalo Guillén-Gosálbez, and Fengqi You. Global optimization of large-scale mixed-integer linear fractional programming problems: A reformulation-linearization method and process scheduling applications. *AIChE Journal*, 59(11):4255–4272, 2013.

[90] Michal Zalewski. p0f v3 (version 3.08 b), 2014.

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