Table of Contents

1

Cyber-attack forecast modeling and complexity reduction using a game-theoretic framework

Malachi Jones, Georgios Kotsalis, and Jeff S. Shamma

Georgia Institute of Technology, Atlanta, GA 30332, USA,
{kye4u, gkotsalis3, shamma}@gatech.edu

Abstract. The security community has placed a significant emphasis on developing tools and techniques to address known security issues. Some examples of this emphasis include security tools such as anti-virus software and Intrusion Detection Systems (IDS). This reactive approach to security is effective against novice adversaries (i.e. script kiddies) because they typically use off-the-shelf tools and popular techniques to conduct their attacks. In contrast, the innovative adversaries often devise novel attack vectors and methodologies that can render reactive measures inadequate. These pioneering adversaries have continually pushed the security frontier forward and motivate a need for proactive security approaches. A proactive approach that we pursue in this research is actionable cyber-attack forecasting. The objectives of actionable cyber-attack forecasting are to learn an attacker's behavioral model, to predict future attacks, and to select appropriate countermeasures. The computational complexity of analyzing attacker models has been an impediment to the realization of reliable cyber-attack forecasting. We address this complexity issue by developing adversary models and corresponding complexity reduction techniques. We then introduce a heuristic for learning behavioral models of potentially deceptive adversaries online. Last, we consider a capture-the-flag problem, formulate the problem as a cyber-security game with asymmetric information, and demonstrate how the models and techniques developed in this paper can be used to forecast a cyber-attack and recommend appropriate countermeasures.

1 Introduction

A reactive mindset is often the status-quo in the security community. Consider popular security products that include intrusion detecion systems (i.e. Snort) and anti-virus software as examples.[1] In the case of anti-virus (AV) software, the typical scenario is the following. First, new malware is developed by cyber hackers and then tested against popular versions of AV software tools.[2] *Testing malware against the AV tools virtually guarantees that new malware will initially go undetected.* Next, the malware infects

* This research was supported by ARO/MURI Project W911NF-09-1-0553.

[1] A consequence of reactive security tools is that once a threat has been detected, damage has been done and costs have been incurred.

[2] Websites such as http://www.virustotal.com can test malware against over 44 different AV tools. Although these sites are intended for security professionals, they can also be used by hackers.

computing devices, and after some period of time has elapsed, which can range from days to years, AV signatures are developed by the AV vendors (i.e Symantec, Kaspersky, and McAfee).[3] This cat-and-mouse pattern then repeats itself and can be observed within and across the security community.

Reactive security methodologies are effective against novice adversaries because these adversaries typically use off-the-shelf tools and implement popular hacking techniques. In contrast, the pioneering adversaries are able to push both the cyber-hacking and the security frontiers forward by developing new malware, designing advanced hacking techniques and methodologies, and challenging security researchers and practitioners to match their ingenuity.[4] In order to combat these adept adversaries and also address advanced persistent threats (APTs), measures and approaches that are forward looking and predictive are needed. This is why we think proactive security is the next frontier in cyber security.

A proactive security approach that we consider in this research is actionable cyber-attack forecasting. The objectives of actionable cyber-attack forecasting are to learn an attacker's behavioral model, to predict future attacks, and to select appropriate countermeasures to prevent future attacks. Impediments that have prevented the realization of reliable cyber forecasting include, but are not limitied to, difficulty in modeling the adversary in an analytical framework and the computational complexity of analyzing the model to forecast attacks. We will address the computational complexity issues in this research.

Previous work to address forecasting challenges, where uncertainty exists about the capabilities of an attacker, include the work of Alpcan et al [1]. In this work, they model the interaction between attackers and an Intrusion Detection System (IDS) using a stochastic (Markov) game. The defender who operates the IDS has uncertainty about the attacker's intent. Tools that include value iteration are used to solve Markov Decision Processes. In [2], You et al. describe how to model cyber-security problems that consider the interactions between an attacker and a defender in a two player, zero-sum game. They illustrate how the Nash and Bayesian Equilibria can be used to predict the behavior of an attacker and to analyze the interactions between attacker and defender. You et al. suggest that linear programs could be used to solve these problems.

Similar to You et al., we also model cyber-security problems as zero-sum games in a Bayesian framework. We introduce computational methods to approximate solutions to these problems. A key feature of these methods is that the solutions can be computed by solving a linear program whose complexity is invariant with respect to the number of stages of the zero-sum game. These methods also have tight lower bounds on their performance that converge asymptotically to optimal with respect to the number of stages of the game.

The outline of this paper is as follows. In Section 2, we will discuss iCTF2010, which is a cyber-security challenge problem that we will later formulate into a strategic

[3] Flame malware that was discovered in May 2012 had been operating in the wild since February 2010.

[4] Malware developers have introduced polymorphic viruses that mutate the machine code of the virus after each execution. This polymorphism feature is designed to defeat AV tools that look for patterns in new viruses that match older versions of the viruses.

game. We will then introduce zero-sum games with asymmetric information and discuss basic concepts and definitions in Section 3.[5] In Section 4, we will develop capture-the-flag (CTF) adversarial models and explore methods and techniques to reduce the complexity of the models. We will then formulate the CTF problem as a security game with asymmetric information in Section 5. Last, we will use simulations to demonstrate how the models and techniques developed in this paper can be used to learn the behavioral model of an adversary, to predict future attacks, and to launch appropriate countermeasures.

2 iCTF2010

2.1 Overview

Security researchers at the University of California Santa Barbara (UCSB) host a live capture-the-flag tournament each year [3]. *We will consider the 2010 version, called iCTF2010 in this research.* There are typically over 900 participants in this tournament, and the participants are hackers from the international community. The purpose of the tournament is to observe and analyze strategies and techniques of real hackers and collect datasets that can be used in security research projects. The iCTF is designed as an abstraction of real-word cyber-security scenarios. For instance consider as a target the security system at Georgia Tech. The various departments, (i.e. ECE, ME, etc.) are subsystems that have to run certain services such as ssh, smtp, citrix, in order to facilitate the computing needs of faculty, staff and students.[6] The attacker's objective is to successfully disrupt critical services based on partial information it receives about the operational state of the system.[7]

The iCTF challenge problem is abstracted as a controlled partially observable stochastic system and each subsystem is modelled as a Markov chain. The attacker however does not have full knowledge of the operational state and is forced to estimate it based on a subset of emitted signals that are correlated with the state transitions. Given those estimates it then chooses an appropriate action that corresponds to the critical services it disrupts. The objective of the attacker is to cause maximal disruption to the overall system.

In the proceeding section, we will present a formal description of our version of the iCTF2010 capture-the-flag problem. We have made some modifications to the original iCTF2010 problem and will present these modifications as well. *The original CTF formulation captures aspects of real-world cyber-security problems that include a dynamic security system whose behavior is at least partially observable and an adversary that can potentially learn and predict the system's behavior.* Our modifications incorporate

[5] This game theoretic framework will be used later in the paper to model the capture-the-flag (CTF) problem as a security game.

[6] An operational state of each subsystem in this example can be defined as the services that are not offline due to maintenance. Maintenance could consist of security patch updates and/or server upgrades that effect a particular service.

[7] It will be assumed that each subsystem hosts its own services and is therefore not impacted by service disruptions in other subsystems.

an additional characteristic of real-world cyber-security problems. This characteristic is a defender who initially has some uncertainy about the capabilities and behavior of an attacker, but who can potentially learn an attacker's capabilities and predict its behavior by using previous observations.

2.2 Model description

2.3 Target System

The target system is abstracted as a discrete-time finite state, finite output Hidden Markov Model (HMM). We refer to the target system as T composed of N subsystems $T_{(1)}, T_{(2)}, ..., T_{(N)}$. The operational states of subsystem T_i are denoted by

$$\mathbb{A}_{(i)} = \{a_{(i)_1}, a_{(i)_2} \ldots, a_{(i)_{n_i}}\}.$$

The state space of the target system is

$$\mathbb{A} = \mathbb{A}_{(1)} \times \mathbb{A}_{(2)} \times \ldots \times \mathbb{A}_{(N)}.$$

The set of observation signals of subsystem T_i is denoted by

$$\mathbb{B}_{(i)} = \{b_{(i)_1}, b_{(i)_2} \ldots, b_{(i)_{m_i}}\}.$$

The observation signal of the attacker at every instant is an unordered N-tuple of output symbols generated by the subsystems. Given the observation signal $\bar{B} = \{\bar{b}_1, \ldots, \bar{b}_N\}$ at some instant, the attacker knows that there there exists an ordering of its elements, say $(\bar{b}_{\sigma(1)}, \ldots, \bar{b}_{\sigma(N)})$, where $\sigma : \{1, \ldots, N\} \to \{1, \ldots, N\}$ is a bijection, such that

$$(\bar{b}_{\sigma(1)}, \ldots, \bar{b}_{\sigma(N)}) \in \mathbb{B}_{(1)} \times \mathbb{B}_{(2)} \times \ldots \times \mathbb{B}_{(N)}.$$

He may not know that particular ordering, but can always perform some probabilistic inference given the HMM abstraction. The output space of the target system is denoted by $\mathbb{B}$. After relabelling of the states and outputs we denote them by $\mathbb{A} = \{a_1, \ldots, a_n\}$, and $\mathbb{B} = \{b_1, \ldots, b_m\}$, where $n = \prod_{i=1}^{N} n_i$ and m is the number of unordered N-tuples of output symbols generated by the subsystems.

It is assumed that the state transitions of each subsystem are independent from each other. Let $\{(X_t, Y_t)\}_{t \in \mathbb{Z}_+}$ denote the state and output process of the given HMM, then the statistical description of the model is given by an initial distribution vector π, where

$$\pi_i = \mathbf{Pr}[X_t = a_i], \;\; i \in \{1, \ldots, n\},$$

and a set of m transition matrices $\{M[y_1], \ldots, M[y_m]\}$ where

$$M[y_k]_{ij} = \mathbf{Pr}[Y_{t+1} = b_k, X_{t+1} = a_i \mid a_t = x_j],$$

where $k \in \{1, \ldots, m\}, i, j \in \{1, \ldots, n\}$.

2.4 Attacker

The action set of the attacker is denoted by

$$\mathbb{S} = \mathbb{S}_{(1)} \times \ldots \times \mathbb{S}_{(N)}.$$

It corresponds to the set of services he disrupts in each subsystem. Associated with state action pairs is a payoff structure, $r_{\mu,\lambda} : \mathbb{X} \times \mathbb{S} \rightarrow \mathbb{R}$. This payoff structure reflects whether an attacker based on his/her information chose to disrupt a service that is relevant to the current operational state of the system. The subscript μ reflects the skill level of the attacker and λ is associated with the resources that the defender allocates to the system. The notion of probing [8] is relevant to the attacker's problem. The payoff incurred at each time step provides the attacker with additional information in estimating the current operational state. As such the attacker is faced with the problem of leveraging short term payoff versus obtaining more accurate information about the current state that will prove beneficial in the long run.

2.5 Defender

The defender's objective in iCTF is to minimize the cost of an attack by an adversary on the target system. Since an attacker's goal is to target critical services, the defender can allocate resources to protect these critical services from being disrupted. Let λ_j represent the amount of resources that the defender allocates to protecting service s_j. The likelihood of a successful attack on s_j decreases as λ_j increases. Deciding on how to allocate resources among the services can be challenging for the defender because of its uncertainty about an attacker's type. We discuss this in detail in Section 5.2 and offer an approach to address the issue.

3 Asymmetric Information Games

3.1 Overview

In a repeated zero-sum game, two players (defender and attacker) repeatedly play the same zero-sum game over several stages. We assume that while both players can observe the actions of the other, only the attacker knows the specific opponent he is playing against. Although the defender has uncertainty on the type of attacker it faces, it has a probability distribution of attacker types and can use its observation of the attacker's actions during game play to eventually learn the attacker's type. The dilemma faced by an attacker is how should he trade off the short-term reward by exploiting his private information versus long-term consequences resulting from revelation of his type.

Classic work by Aumann and Maschler [4] derives a recursive formula for the value of the game, which quantifies the exploitation tradeoff, and also derives the optimal policy for the attacker. Using Aumann and Maschlers' model for explicit computations of optimal policies is prohibitive for games with multiple stages. In [5], this computational

[8] The idea of optimal probing is an interesting topic that we would like to consider in future research.

issue is addressed by introducing methods to compute suboptimal strategies by solving linear programs whose complexity is constant with respect to the number of stages. The methods from [5] are discussed in Section 5.2.

3.2 Game Setup

Game Play Two players repeatedly play a zero-sum matrix game over l stages. The attacker is the row player and maximizer, and the defender is the column player and minimizer. There are a finite set $\mathbb{K}$ of possible attacker types that the defender can face. A specific attacker is chosen from this set to play against the defender. Let S be the set of pure strategies of an attacker and similarly define J to be the set of pure strategies of the defender. The payoff matrix for an attacker of type k will be denote as $M^k \in \mathbb{R}^{|S|\times|J|}$. Before the initial stage $m = 1$, nature selects an attacker type according to a probability distribution $p \in \Delta(K)$, which is common knowledge. The outcome of this selection is not revealed to the defender. Once selected, the attacker's type remains fixed over all stages of the game.

Strategies Mixed strategies correspond to distributions over pure strategies. Let $x_m^k \in \Delta(S)$ denote the mixed strategy of an attacker of type k at stage m. In repeated play, this strategy can be a function of the actions of both players during stages $1, ..., N$. Likewise, let $y_m \in \Delta(J)$ denote the mixed strategy of the defender at stage m, which again can depend on player actions over stages m=1,...,N. Let $x_m = \{x_m^1, ..., x_m^K\}$ denote the collection of mixed strategies for all attacker types and for all states at stage m, and $x = \{x_1, ..., x_N\}$ denote mixed strategies over all states and stages. Likewise, let $y = \{y_1, .., y_N\}$ denote the defender's mixed strategies over all stages.

Payoffs Let $\gamma_m^p(x, y)$=$\sum_{k\in K} p^k x_m^k M^k y_m$ denote the expected payoff for the pair of mixed strategies (x, y) at stage m. The payoff for the n-stage game is then defined as

$$\bar{\gamma}_n^p(x, y) = \frac{1}{n} \sum_{m=1}^{n} \gamma_m^p(x, y) \tag{1}$$

.

3.3 Concepts and Definitions

Beliefs Since the defender is not informed of the attacker's type k, it can build beliefs on the type. These beliefs are a function of the initial distribution p of attacker types and the observed moves of an attacker. An attacker must therefore carefully consider its actions at each stage as they could potentially reveal its type to the defender. In order to get a worse case estimate of how much information an attacker transmits about its type through its actions, it models the defender as a Bayesian player and assumes that the defender knows its mixed strategy. The updated belief p^+ is computed as

$$p^+(p, x, s) = \frac{p^k x^k(s)}{\bar{x}(p, s)} \tag{2}$$

where $\bar{x}(p,s) := \sum_{k\in K} p^k x^k(s)$ and $x^k(s)$ is the probability that an attacker of type k plays pure action s.

Non-revealing strategies Revealing information is defined as an attacker selecting a mixed strategy that is dependent on his type k. From (2), it follows that a mixed strategy x_m at stage m does not change the current beliefs of the defender if $x_m^k = x_m^{k'}\ \forall\ k, k'$. As a consequences, an attacker who plays as if it is oblivious of its type, ensures that its opponents beliefs about its type do not change.[9]

An optimal non-revealing strategy can be computed by solving

$$u(p) = \max_{x\in \text{NR}} \min_{y} \sum p^k x^k M^k y, \tag{3}$$

where NR $= \{x_m \mid x_m^k = x_m^{k'}\ \ \forall k, k' \in K\}$ is the set of non-revealing strategies [11]. By playing an optimal non-revealing strategy at each stage of the game, an attacker can guarantee a game payoff of $u(p)$.[10]

Definition 1. *Let Cav$[u(p)]$ denote the point-wise smallest concave function g on $\Delta(K)$ satisfying $g(p) \geq u(p)\ \forall p \in \Delta(K)$*

Short-term vs. Long-term Payoffs The dynamic programming recursive formula

$$v_{n+1}(p) = \max_{x_1} \min_{y_1} \left[\frac{1}{n+1} \sum_{k\in K} p^k x_1^k M^k y_1 + n \sum_{s\in S} \bar{x}_s v_n\Big(p^+(p, x_1, s)\Big) \right], \tag{4}$$

introduced by Aumann and Maschler [4], characterizes the value of repeated zero-sum games with asymmetric information. Note that n is a non-negative integer. When $n = 0$, the problem reduces to

$$v_1(p) = \max_{x_1} \min_{y_1} \sum_{k\in K} p^k x_1^k M^k y_1, \tag{5}$$

which is the value of the 1-shot zero-sum game.

A key interpretation of this formulation is that it also serves as a model of the trade-off between short-term gains and the long-term informational advantage. For each decision x_1 of an attacker , the model evaluates the payoff for the current stage, which is represented by the expression $\sum_{k\in K} p^k x_1^k M^k y_1$, and the long-term cost for decision x_1, which is represented by $n \sum_{s\in S} \bar{x}_s v_n\Big(p^+(p, x_1, s)\Big)$.

It is worth pointing out that the computational complexity of finding the optimal decision x_1 can be attributed to the cost of calculating the long-term payoff. Since the long-term payoff is a recursive optimization problem that grows with respect to the

[9] In stochastic games, it is possible for the defender's beliefs about an attacker's type to change even if an attacker plays as if it is oblivious of its type.

[10] In [5], this idea of non-revelation was exploited to reduce the complexity of Aumann and Maschler's formulation.

game length, it can be difficult to find optimal strategies for games of arbitrary length. This difficulty is because the number of decision variables in the recursive optimization problem grows exponentially with respect to the game length. A revised formulation

$$\hat{v}_n(p) = \max_{x_1} \min_{y_1} \left[\frac{1}{n} \sum p^k x_1^k M^k y_1 + (n-1) \sum_{s \in S} \bar{x}_s u\Big(p^+(p, x_1, s)\Big) \right] \tag{6}$$

was introduced in [5] to address a complexity issue of the recursive formulation of the value of the game. In this formulation, it is assumed that the informed player uses optimal non-revealing strategies (i.e. $u(p)$) for all future stages. Therefore the cost-to-go function $v_n(p)$ in (4) can be expressed as $u(p)$ in (6). As a consequence of the non-revealing assumption, the computational complexity remains constant with respect to the number of stages of the game.

Theorem 1. *[5] A perpetual policy improvement strategy can be computed by solving a linear program online at each stage of the game, and the computational complexity of the linear program is constant with respect to the number of stages of the game.*

In [5], lower bounds on $\hat{v}_n(p)$ were established. It was also shown that the lower bounds were tight, and it was proved that $\hat{v}_n(p)$ has asymptotic convergence to optimality with respect to the number of stages n.

Theorem 2. *[5] One-time policy improvement and perpetual policy improvement achieve Cav[u(p)] and the optimality bounds are*

$$Cav[u(p)] \leq \hat{v}_n(p) \leq v_n(p) \leq Cav[u(p)] + \frac{C}{\sqrt{n}} \sum_{k \in K} \sqrt{p^k(1-p^k)} \tag{7}$$

We will use the $\hat{v}_n(p)$ formulation in Section 5.2 to reduce the computational complexity of attacker models.

4 Attacker Modeling and Complexity Reduction

A basic adversarial model should addresses the following questions about a specific adversary:

1. What are its skills? **(Skillset/Capabilites)**
   Since each critical service may require specific technical skills to disrupt, what is the probability that an attacker can successfully disrupt service s_j?
2. What is its intent? **(Intent)**
   Is the ultimate goal of an adversary to prevent the success of the security system under consideration, or just to create general disruption?
3. How patient is the adversary? **(Patience)**
   Is the attacker greedy, or is it patient and willing to forgo an immediate gain in order to maximize its long-term payoff?
4. How does it build beliefs about the system's current state? **(Beliefs)**
   Is computing the system's belief function computationally prohibitive? If so, what technique does the adversary use to approximate the belief function?

5. How does it make decisions based on its state estimates? **(Strategies)**
   Given an estimate of the system, will an adversary disrupt critical services of the most likely operational state or disrupt services that maximize its expected payoff?

In the adversarial models we develop in this section, we will make assumptions that allow us to convey the main ideas of attacker modeling and complexity reduction techniques in a clear and accessible manner. These assumptions serve as intermediate steps that allow us to explore the prominent issues in modeling an adversary and also to consider possible approaches to address those issues. We will assume the following: 1) Available actions of the attacker and the probability distribution of attacker skill types are public knowledge. 2) Intents of the attacker are zero-sum. 3) The Attacker is greedy 4) Consider the worst case with respect to the attacker's computational ability (i.e. only prohibitive for the defender to computer the belief function of the system).

4.1 Capabilities

An attacker's type will be defined as his skill level at disrupting a set of services. The skill level will be represented as a vector, where the jth component of the vector represents the attackers ability to disrupt service j. The values of the skill vector are in the range between 0 and 1, where 1 is expert skill and 0 is no skill at disrupting service s_j.

4.2 Intent

The ultimate objective of the adversary may be unknown. Although many cyber hackers aim to profit from their attacks, other hacker groups such as Annonymous employ denial of service attacks to make political statements and seek publicity. Therefore, an approach that we use to address the uncertainty of the adversaries objective is to consider the worst case with respect to the defender. In particular, we model the problem as zero-sum (i.e. a reward α for the attacker is a corresponding cost α to the defender). This zero-sum assumption allows us to make performance guarantees on security policies.

4.3 Patience

One can model an adversary as having a discount factor λ on its future payoff. A discount factor of $\lambda \approx 0$ would then indicate a greedy adversary that heavily discounts the future, while $\lambda \approx 1$ would be indicative of an adversary that is a long-term player that heavily discounts the present. An alternative interpretation of the discount factor is the patience of the adversary. Modeling a patient adversary introduces a complication that is absent in the greedy models we use. This complication is probing.[11] Since a patient attacker may be willing to defer an immediate reward, it can consider choosing actions that may provide it will a better estimate of the current state of the security system. This improved estimate along with the adversaries knowledge of the HMM can then be used to make a better prediction of the future behavior of the system. The question that follows is when should it probe and when should it attack. This is an interesting question that we would like to consider in future research.

[11] We briefly discussed probing in Section 2.4.

4.4 Beliefs & Strategies

An adversaries decision to disrsupt a particular service s_j can be dependent on his ability to disrupt service s_j (i.e. his skill set), his payoff for disrupting service s_j, and his beliefs about the current operational state of each subsystem T_i. Computing the belief function of the current state of the system T is prohibitive. Consider the following example as an illustration. Suppose that a system T' is composed of N subsystems that each have 10 operational states. The size of the state space of the system T' is the product of the individual subsystems and is equal to 10^N, and the beliefs are probabilities of state combinations of the subsystems, e.g. $\Delta(10^N)$. We assume the worst case about the adversary, which is that he can compute the belief function of the system, while the defender may only be able to compute an estimate. To address the computational challenges of the defender, we introduce two techniques, quasi-beliefs and belief compression, that can be used to calculate estimates of the belief function of the system T.

Quasi-beliefs The main idea of quasi-beliefs (QB) is the following. Instead of computing the true beliefs of system T (e.g. $\Delta(10^N)$), we consider computing an estimate of the beliefs of each subsystem T_i independently of the other subsystems T_{-i}. An issue that arrises with computing independent beliefs of each subsystem is signal assignment. *Recall that the attacker only observes the collection of signals emitted from the subsystems. Therefore, he has uncertainty about the mapping between each signal and the subsystem that emitted the signal.* We will first introduce a method described in Algorithm 1 that provides us with an estimate of the likely mapping between signals and subsystems.

Algorithm 1 Signal assignment

1: **procedure** SIGNALASSIGN
2: **initialize matrix** P_S
3: **while** size$(P_S) > 0$ **do**
4: find the maximum element of P_S
5: let position of max element be denoted as (i^*, m^*)
6: assign signal y_{m^*} to subsystem T_{i^*}
7: remove row i^* and column m^* from matrix P_S
8: **end while**
9: **end procedure**

Note that for the matrix P_S at step 2 of Algorithm 1, each column corresponds to a signal y_m, each row corresponds to a subsystem T_i, and element (i, m) represents the probability that signal y_m was emitted from subsystem T_i. Also note that if there are more than one maximum element at step 4, we break the tie by randomly selecting a maximum element. We then use this signal assignement method in the quasi-belief greedy (QBG) algorithm (Algorithm 2) that is described below.

Algorithm 2 QBG Strategy

1: **procedure** QBGSTRATEGY
2: start with set of individual subsystem beliefs
3: **run** procedure SIGNALASSIGN
4: update individual beliefs using assignment
5: ***attack services with highest expected reward***
6: renormalize beliefs given success/failure of attack
7: **end procedure**

Belief Compression The point of departure is the statistical description of the HMM abstraction of the target system. Let $\mathbb{A} = \{a_1, \ldots, a_n\}$, $\mathbb{B} = \{b_1, \ldots, b_m\}$ denote the state and output space respectively. The statistics of the joint state and output process $\{(X_t, Y_t)\}_{t\in\mathbb{Z}_+}$ are encoded by the initial distribution vector π, where

$$\pi_i = \mathbf{Pr}[X_t = a_i],\ i \in \{1, \ldots, n\},$$

and a set of m transition matrices $\{M[y_1], \ldots, M[y_m]\}$ where

$$M[y_k]_{ij} = \mathbf{Pr}[Y_{t+1} = b_k, X_{t+1} = a_i \mid X_t = a_j],$$

for $k \in \{1, \ldots, m\}$ and $i, j \in \{1, \ldots, n\}$. Let $\mathbb{B}^*$ denote the set of all emitted finite strings of observation signals including the empty sting $\varnothing$. Let $v = v_k \ldots v_1$ stand for a string of length k. Let $1_n \in \mathbb{R}^n$ denote the vector whose entries are all 1. Introduce the function

$$p : \mathbb{B}^* \times \mathbb{R}^n_+ \to [0, 1],$$

where

$$p[(v, \pi)] = 1_n^T\ M[v_k]\ \ldots\ M[v_1]\pi.$$

The function p is referred to as the probability function. It is used to compute the probability of observing a particular under initial distribution π, i.e.

$$p[(v, \pi)] = \sum_{i\in\{1,\ldots,n\}} \mathbf{Pr}[Y_k = v_k, \ldots, Y_1 = v_1 \mid X_o = a_i]\ \pi_i.$$

The value $p[(v, \pi)]$ is computed recursively using the rule

$$p[(v, \pi)] = 1_n^T\ H_k,$$

where $H_t = M[v_t]\ H_{t-1}$, $t \in \{1, \ldots, k\}$, and $H_0 = \pi$. Consider also the functions

$$p_{co} : \mathbb{Y} \times \mathbb{Y}^* \times \mathbb{R}^n_+ \to [0, 1], \quad p_{cs} : \mathbb{X} \times \mathbb{Y}^* \times \mathbb{R}^n_+ \to [0, 1],$$

referred to as the conditional output probability and conditional state probability function. The value $p_{co}[(b, v, \pi)]$ corresponds to the conditional probability of emitting the signal b given that the signal v has been observed under the initial distribution π, i.e.

$$p_{co}[(b, v, \pi)] = \sum_{i\in\{1,\ldots,n\}} \mathbf{Pr}[Y_{k+1} = b \mid Y_k = v_k, \ldots, Y_1 = v_1, X_0 = a_i]\ \pi_i.$$

Similarly the value $p_{cs}[(a, v, \pi)]$ corresponds to the conditional probability of being at state a given that the signal v has been observed under the initial distribution π, i.e.

$$p_{cs}[(a, v, \pi)] = \sum_{i \in \{1,\ldots,n\}} \mathbf{Pr}[X_k = a \mid Y_k = v_k, \ldots, Y_1 = v_1, X_0 = a_i]\ \pi_i.$$

The belief function is

$$\mathit{\Pi} : \mathbb{Y}^* \times \mathbb{R}_+^n \to \mathbb{R}_+^n,$$

where

$$\mathit{\Pi}[v, \pi]_i = p_{cs}[(a_i, v, \pi)], \quad i \in \{1, \ldots, n\}.$$

The value of the belief function is computed recursively.

$$\mathit{\Pi}[v, \pi] = \frac{H_k}{1_n^T\ H_k}.$$

At every time step the attacker chooses an action to maximize his/hers instantaneous expected reward. For $s \in \mathbb{S}$ let $g[s] \in \mathbb{R}^n$ where $g[s]_i = r[s, a_i]$, $i \in \{1, \ldots, n\}$. Having observed the signal v and following a greedy strategy the attacker is faced with the optimization problem

$$\max_{s \in \mathbb{S}} < g[s], \mathit{\Pi}[v, \pi] > .$$

The notion of belief compression is associated with projecting the dynamics of the given HMM onto a lower dimensional manifold. Let $\hat{n} < n$, $V \in \mathbb{R}^{n \times \hat{n}}$, $U \in \mathbb{R}^{\hat{n} \times n}$ with $U\ V = I_{\hat{n}}$, so that $V\ U$ is a projection matrix. The parameters of a reduced complexity model are given by

$$\hat{c}^T = 1_n^T\ V, \quad \hat{b} = U\ \pi,$$
$$\hat{A}[y] = U\ M[y]\ V, \quad y \in \mathbb{Y}.$$

Using the reduced complexity model one can determine a greedy strategy while performing the relevant calculations on a $\hat{n}$ dimensional space with obvious computational and storage advantages. In particular consider the function

$$\hat{p} : \mathbb{Y}^* \times \mathbb{R}^n \to \mathbb{R},$$

where

$$\hat{p}[(v, \hat{b})] = \hat{c}^T\ \hat{A}[v_k]\ \ldots\ \hat{A}[v_1]\hat{b}.$$

The value $\hat{p}[(v, \hat{b})]$ is computed recursively using the rule

$$\hat{p}[(v, \hat{b})] = \hat{c}^T\ \hat{H}_k,$$

where $\hat{H}_t = \hat{A}[v_t]\ \hat{H}_{t-1}$, $t \in \{1, \ldots, k\}$, and $\hat{H}_0 = \hat{b}$. The function $\hat{p}$ is a low complexity surrogate for the probability function of the given HMM. Similarly consider the function

$$\hat{\mathit{\Pi}} : \mathbb{Y}^* \times \mathbb{R}^{\hat{n}} \to \mathbb{R}^{\hat{n}},$$

where

$$\hat{\Pi}[v,\hat{b}] = \frac{\hat{H}_k}{1_n^T\ \hat{H}_k}.$$

Let $\hat{g}[s] = V^T g[s]$, when employing the low complexity model the attacker is faced with the optimization problem

$$\max_{s\in\mathbb{S}} < \hat{g}[s], \hat{\Pi}[v,\pi] > .$$

In order to compute the compression matrix U and dilation matrix V we will employ the balanced truncation algorithm developed for HMM's in [7]. The reduction method is based on stable numerical linear algebra tools employing the singular value decomposition and is accompanied by an a priori bound to the approximation error. In other words it leverages the favourable features of Hankel norm based reduction techniques for linear time invariant systems.

First one solves linear algebraic equations to obtain "gramian like" quantities $W_c, Wo \in \mathbb{R}^{n\times n}$ where $W_c, W_o \succeq 0$,

$$W_o = \sum_{y\in\mathbb{Y}} M[y]^T\ W_o\ M[y] + 1_n^T\ 1_n,\ W_c = \sum_{y\in\mathbb{Y}} M[y]\ W_c\ M[y]^T + \pi\ \pi^T. \qquad (8)$$

Denote by L_o, L_c the Cholesky factors of $W_o = L_o^T\ L_o$ and $W_c = L_c\ L_c^T$ and consider the SVD of $L_c^T\ L_o^T$,

$$L_c^T\ L_o^T = \left[\ \Psi^{(1)}\ \Psi^{(2)}\ \right] \begin{bmatrix} \Sigma^{(1)} & 0 \\ 0 & \Sigma^{(2)} \end{bmatrix} \begin{bmatrix} \Xi^{(1)^T} \\ \Xi^{(2)^T} \end{bmatrix}.$$

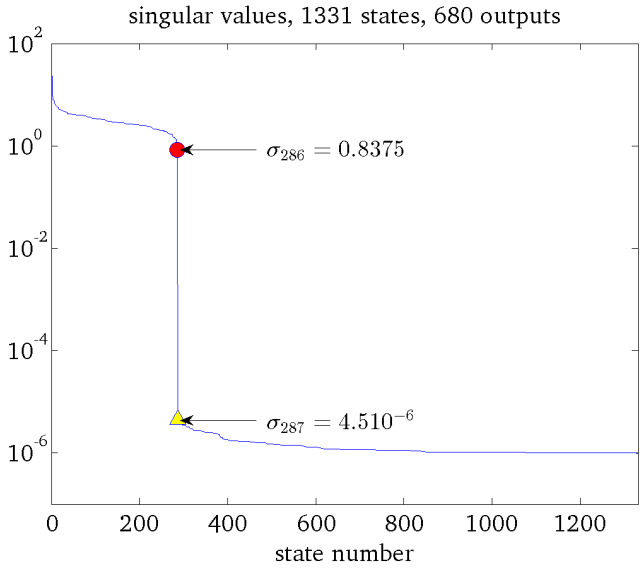
where $\Sigma^{(1)} = \mathrm{diag}[\sigma_1,\ldots,\sigma_{\hat{n}}]$, $\Sigma^{(2)} = \mathrm{diag}[\sigma_{\hat{n}+1},\ldots,\sigma_n]$ and $\sigma_1 \geq \ldots \geq \sigma_{\hat{n}} > \sigma_{\hat{n}+1} \geq \ldots \geq \sigma_n > 0$. In the above notation

$$V = L_c[\psi_1^{(1)}\frac{1}{\sqrt{\sigma_1}},\ldots,\psi_{\hat{n}}^{(1)}\frac{1}{\sqrt{\sigma_{\hat{n}}}}],\ U = \begin{bmatrix} \frac{1}{\sqrt{\sigma_1}}\xi_1^{(1)^T} \\ \vdots \\ \frac{1}{\sqrt{\sigma_{\hat{n}}}}\xi_{\hat{n}}^{(1)^T} \end{bmatrix} L_o.$$

The following error bound controls the approximation of the given probability function:

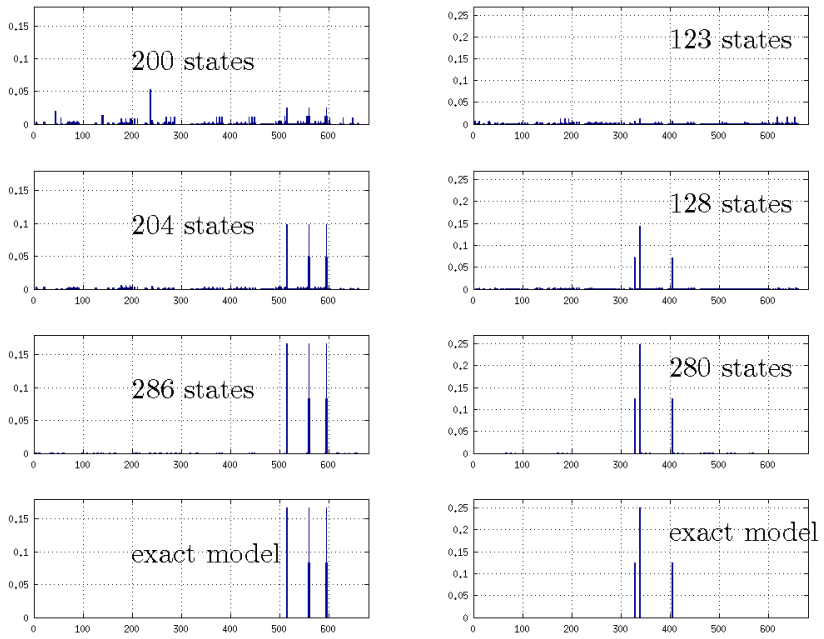
$$\sqrt{\sum_{v\in\mathbb{Y}^*} (p[v,\pi] - \hat{p}[v,\hat{b}])^2} \leq 2(\sigma_{\hat{n}+1} + ... + \sigma_n).$$

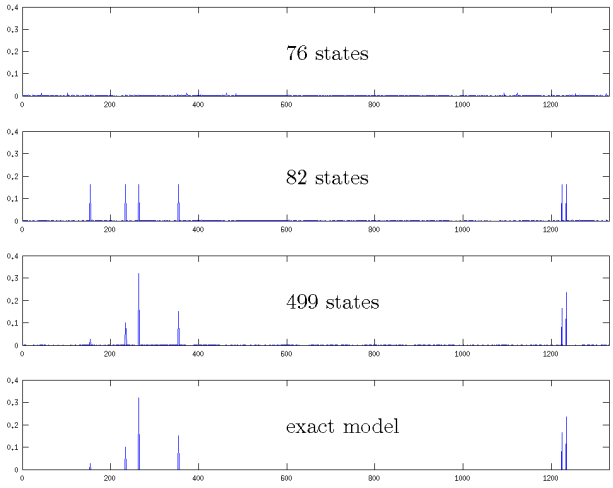
The algorithm is demonstrated on a target system comprised from 3 subsystems used in the iCTF, with $|\mathbb{A}| = 1331$ and $|\mathbb{B}| = 680$. The following figure depicts the singular values $\sigma_1,\ldots,\sigma_{1331}$ that control the error bound. There is a clear cut-off behavior indicating that a choice of a reduced complexity model with 286 states is appropriate.



Next it is demonstrated that the reduced complexity model delivers a very accurate approximation to the conditional output probability function. The following figure depicts the vector $(p_{co}[b_1, v_k \dots v_1, \pi], \dots, p_{co}[b_{680}, v_k \dots v_1, \pi])$ conditioned on two trajectories $v_k \dots v_1$ of length 1000. The bottom row corresponds to the exact model and the rows above it correspond to approximations computed using the reduced order model. In both cases a reduced order model of at most 286 states approximates this conditional probability within 0.1%.

One can use the reduced order model also to compute an approximation to the belief function of the system and subsequently solve the attackers greedy optimization problem. The next figure depicts how a reduced order model of 499 states delivers a very accurate approximation to the belief function within 0.1% error. The belief function was computed for a trajectory $v_k \dots v_1$ of length 1000. The bottom row corresponds to the exact model and the rows above it correspond to approximations computed using the reduced order model.


200 states
123 states
204 states
128 states
286 states
280 states
exact model
exact model



For the same trajectory greedy optimization using the reduced order model let to the same choice of action in 96% of the instances.

5 CTF Security Game Formulation

In this section, we will formulate iCTF as a security game with asymmetric information. Recall that these games were discussed extensively in Section 3. In our formulation of iCTF, we assume that only the attacker knows its opponents type. Therefore, the information asymmetry of the iCTF security game is on the side of the attacker. We will proceed by first discussing the game play. Next, we will discuss the strategies available to the players. Last, we will cover prominent issues that each player must consider in this security game and also discuss approaches to address those issues.

5.1 Game Play

One-shot game formulation The one-shot game formulation of iCTF consists of two players, an attacker and the defender. An attacker of type k is selected by nature before the start of the game from a known distribution p of attacker types.[12] The attacker's objective in this game is to maximize its reward for attacking the security system T that was defined in Section 2.3.[13] The attacker maximizes its reward by disrupting critical services that are needed by the individual subsystems T_i of T. The attacker knows the specific services that are critical for each state of the subsystem. However it has uncertainty on the current state of each subsystem T_i.

[12] Attackers differ in their ability to disrupt services, and an attacker's type is defined by its skill vector.

[13] Conversely, the defender's objective is to minimize its costs incurred by the attacker.

An attacker can improve its estimate of each subsystem's state by computing the belief function of the system T. Since computing the belief function is computationally prohibitive for systems with large state spaces, the defender can approximate the true belief function by considering a state estimation technique discussed in Section 4.4. We assume that the defender uses the quasi-beliefs in this paper, although it could also use an alternative technique (i.e. belief compression). A particular greedy strategy that an attacker can consider is to disrupt services that maximizes the expected payoff for an attacker of its type; we will refer to this strategy as the honest strategy. Alternatively, an attacker can disrupt services that maximizes the payoffs of an attacker of another type; we will refer to this strategy as the dishonest strategy. It may seem irrational for an attacker to select a dishonest strategy in the one-shot game because the honest strategy gives him the better expected payoff. However, we will see in the next section that choosing a dishonest strategy can not only be rational, it can be optimal.

The defender has finite resources that it can allocate to maintain the availability of critical services. Dedicating more resources to a particular service s_j decreases the likelihood that service s_j will be disrupted in the event that an attacker targets that service. At the start of iCTF, it selects how these resources are allocated, and the resource allocation decision remains fixed until completion of iCTF. This resource allocation decision can be a function of the defender's belief about an attacker's type. In particular, it can chose to do a best response allocation with respect to an attacker's type or best respond to an attacker's type with some probability.

Repeated game formulation We assumed in the one-shot formulation that once a QBG policy is chosen by the attacker and a resource allocation technique is chosen by the defender at the beginning of the game, the selections by both players remain fixed throughout the duration of iCTF. In the repeated game formulation, however, both players can reevaluate their decisions at specific intervals. We will refer to these intervals as stages. At each stage, the security system S is reset to its initial state. What isn't reset is each player's knowledge about the other player's past actions. In fact, each player's knowledge changes over time, affects its beliefs, and introduces a dynamic aspect to this security game.

5.2 Player Concerns

As we have discussed in the previous section, each player's knowledge is dynamic and changes in time. An important objective of the attacker is to control the beliefs of the defender about its type because knowledge of the attacker's type can allow the defender to make decisions that cost the attacker. Controlling the defender's beliefs often involves deceptive play by the attacker.[14] Because of the potential for deception, it can be difficult for the defender to learn the attacker's true type. In the preceding sections, we will discuss approaches for addressing each players concerns.

[14] An attacker of type k can elect to select a QBG strategy of an attacker of a different type. (i.e. $QBG_{k'}$ where $k \neq k'$)

Attacker Recall that in Section 3.3, we discussed a formulation introduced by Aumann and Maschler that allows us to model an attacker that optimally controls the beliefs of the defender and therefore attains the optimal game payoff. As part of the discussion, we mentioned complexity issues that arose with using this formulation when considering games with multiple stages. To address this complexity issue, we will use policy-improvement strategies presented in [5] to approximate the strategy selection of an optimal attacker. These policy-improvement strategies have error bounds on their performance with respect to optimal strategies, and the performance of the policy improvemement methods converge asymptotically to optimal with respect to n, the number of stages in the game. We will first discuss the one-time policy-improvement method and then preceed to discuss the perpetual policy improvement method.

In one-time policy improvement, an attacker strategizes for the *first stage* of the iCTF game while assuming that it will play in a non-revealing manner in all future stages. Perpetual policy improvement is an extension of one-time policy improvement. In the perpetual policy improvement method, the attacker strategizes for the current stage while assuming a non-revealing strategy in all future stages at *every stage*. The perpetual policy improvement method involves solving an LP online, and the computational complexity of the LP is constant with respect tho the number of stages of the game. Below is the perpetual policy improvement algorithm.

Algorithm 3 Perpetual policy improvement

procedure PERPPOLICYIMPROVE
  initialize: set $p_1 = p$
  for $m = 1 \rightarrow N$ **do** ▷ Note that m is the current stage
    compute $\hat{x}_m$ by solving one-time policy improvement LP with p_m
    select a move s for attacker type k using mixed strategy $\hat{x}_m^k$
    update beliefs vector (i.e.$p_{m+1} = p^+(p, \hat{x}, s)$)
  end for
end procedure

Defender Learning an attacker's true type can be challenging because it can play in a deceptive manner [8]. If the defender knew the mixed strategy x^k for each type of attacker k, then learning an attacker's type would be straightforward because the defender could follow the standard Bayesian update approach. Unfortunately, in the actual play of the game, the defender does not know each attackers mixed strategy as this information is private. Another approach that the defender can consider is solving a linear program to compute an optimal defensive strategy.[15] However, the complexity of the LP is exponential with respect to n, the number of stages of the game. We will introduce a payoff-based heuristic for learning an attacker's type that is computationally tractable for arbitrary n and only depends on the information that the defender has. This information is namely the history of the attacker's actions.

The main idea behind the payoff-based heuristic is as follows. The defender's belief of an attacker of type k will be correlated with the actual game payoff of the attacker.

[15] Ponssard and Sorin [13] showed that zero-sum repeated games of incomplete information can be formulated as linear programing problems to compute optimal strategies.

After each stage, the defender keeps track of what the overall game payoff would be for each type of attacker. The game payoff for an attacker of type k at stage n, given history h_n will be denoted $\tilde{\gamma}_n^k(h_n)$. The game payoff $\tilde{\gamma}_n^k(h_n)$ for an attacker of type k will be compared to the best possible payoff that a type k attacker can achieve. We will denote the best possible payoff by $|M^k|$, where $|M^k| := \max_{i,j} M_{i,j}^k$.[16] Let

$$\xi^k(h_n) = \frac{\tilde{\gamma}_n^k(h_n)}{|M^k|} \tag{9}$$

be a measure of the likelihood that an attacker is of type k given history h_n. The belief update procedure is then

$$\tilde{p}_{n+1}^k(h_n) = \tilde{p}_n^k \frac{\xi^k(h_n)}{\bar{\xi}(h_n)} \tag{10}$$

where $\bar{\xi}(h_n) = \sum_{k \in K} \tilde{p}_n^k \xi^k(h_n)$. To compute a best response strategy $\tilde{y}^*$ for the defender given the approximate belief $\tilde{p}_n$ at stage m, solve the optimization equation

$$\tilde{y}_m^* = \arg\min_{y_m} \max_{x_m} \sum_{k \in K} \tilde{p}_m^k x_m^k M^k y_m. \tag{11}$$

6 Simulation

6.1 Game Setup

As usual, this game consists of two players, an attacker and a defender. In this example, we assume that their are two types of attackers (i.e. type I and type II) and each attacker is uniquely defined by its skill vector. The probability distrubution of attacker types is uniform (i.e. $p^k = \frac{1}{2}$ for $k = 1, 2$), and there are two stages in this game. Matrix payoffs for the attacker types are

	BR_1	BR_2
QBG_1	23	375
QBG_2	−92	69

Type I

	BR_1	BR_2
QBG_1	−6	−28
QBG_2	128	−20

Type II

(12)

Note that an attacker of type I has the option of playing as his type by selecting QBG_I or playing deceptively by selecting QBG_{II}. Similarly, a type II attacker can opt to play either honestly or deceptively. The defenders available actions are to play a best-response resource-allocation strategy for a specific attacker type (i.e BR_I or BR_{II}.

[16] Without loss of generality, we will assume in this section that each matrix M^k is scaled with values ranging from 0 to 1.

6.2 Discussion

We will discuss the performance of four attacker strategies in this section. These strategies are dominant strategy, non-revealing strategy, one-time policy improvement, and perpetual policly improvement. In the one-shot game, the optimal strategy for an attacker is to behave as its type by selecting his domiant strategy. However, for games where $n > 1$, this can be a suboptimal strategy because it can reveal the attackers true type to the defender and cost the attacker the informational advantage. Specifically, in the two stage game, the attacker can achieve a better payoff by selecting the perpetual policy improvement strategy. For games where N is large, the dominant strategy has the worst performance out of the four strategies and the policy improvement strategies have the best performance.

Two-stage game An optimal non-revealing strategy requires the attacker, regardless of his type, to play as a type I attacker with probability $.70$ and to play as a type II attacker with probability $.30$ at each stage. This strategy rewards the attacker with a payoff of 18 and has the worst performance of the four strategies in the two-stage game.[17] One-time policy improvement performs better by guranteeing an expected payoff of 27. The two strategies differ conceptually only at the first stage as how he plays when using a one-time policy improvement strategy is dependent on his type. A type I attacker plays deceptively at stage one with probability $.92$, while a type II attacker plays honestly with probability 1. At stage two, an attacker plays as a type II attacker with probability $.96$, which is independent of his type. An attacker that chooses to use his domimant strategy, which requires him to play honestly, at each stage of the game yields the attacker an expected payoff of 39, which outperforms the two previously mentioned strategies. Perpetual policy improvement yields the attacker the highest reward, 53, of the four strategies in consideration. At the first stage of perpetual policy improvement, an attacker plays the same way it would have played had it chosen one-time policy improvement. However, the key difference is at the second stage. Instead of playing non-revealing as with the former strategy, the attacker behaves as his type in the second stage of perpetual policy improvement.

***N*-stage game** We discussed the performance of the four strategies in the two-stage case in the previous section and will now examine their performance as N grows large. The expected payoff for the dominant strategy converges asymptotically to 2 and has the worst performance of the four strategies for large N. This is because the defender can readily learn the attacker's type as the attacker does not play deceptively. The defender can then use this knowledge to select a resource allocation scheme that is a best response to his type. An optimal non-revealing strategy performs better than the dominant strategy because the defender is unable to learn any additional information about the attacker after observing his action at each stage. As a consequence, the defender has uncertainty on which resource alloction scheme will perform best against the attacker. An attacker who chooses this strategy can therefore guarantee a payoff of 18 at every stage. An immediate consequrence of this guarantee is that an attacker can achieve a

[17] There are games where playing non-revealing is optimal for all n.

game payoff of 18 for games of any length. Policy improvement methods have the best performance of the four strategies for large N. Both methods have identical behavior and converge asymptotically to optimal and yields a payoff of 23. At stage one of the policy improvement methods, the attacker behaves deceptively with some probability that is type dependent. For all stages thereafter, the attacker plays a non-revealing strategy that is independent of its type.

References

1. T. Alpcan and T. Basar, "An intrusion detection game with limited observations," in *Proc. of the 43rd IEEE Conference on Decision and Control*, 2004.
2. X. You and Z. Shiyong, "A kind of network security behavior model based on game theory," *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003.
3. A. Doupé, M. Egele, B. Caillat, G. Stringhini, G. Yakin, A. Zand, L. Cavedon, and G. Vigna, "Hit 'em where it hurts: a live security exercise on cyber situational awareness," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 51–61. [Online].
4. R. J. Aumann and M. Maschler, *Repeated Games with Incomplete Information*. MIT Press, 1995.
5. M. Jones and J. S. Shamma, "Policy improvement for repeated zero-sum games with asymmetric information," in *51st IEEE Conference on Decision and Control*, Dec 2012.
6. V. C. Domansky and V. L. Kreps, "Eventually revealing repeated games of incomplete information," *International Journal of Game Theory*, vol. 23, pp. 89–109, 1994.
7. G. Kotsalis and A. Megretski and M. Dahleh, "Balanced Truncation for a Class of Stochastic Jump Linear Systems and Model Reduction of Hidden Markov Models," *IEEE Transactions on Automatic Control*, vol. 53, 2008.
8. M. Heur, "Optimal strategies for the uninformed player," *International Journal of Game Theory*, vol. 20, pp. 33–51.
9. S. Zamir, "On the relation between finitely and infinitely repeated games with incomplete information," *International Journal of Game Theory*, vol. 23, pp. 179–198.
10. A. Gilpin and T. Sandholm, "Solving two-person zero-sum repeated games of incomplete information," *International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 2, pp. 903–910, 2008.
11. S. Zamir, "Repeated games of incomplete information: Zero-sum," *Handbook of Game Theory*, vol. 1, pp. 109–154, 1999.
12. R. Aumann, *Mixed and behavior strategies in infinite extensive games*. Princeton University, 1961.
13. J. Ponssard and S. Sorin, "The l-p formulation of finite zero-sum games with incomplete information," *International Journal of Game Theory*, vol. 9, pp. 99–105, 1999.
14. D. Blackwell, "An analog of the minimax theorem for vector payoffs." *Pacific Journal of Mathematics*, vol. 1956, no. 1, pp. 1–8, 1956.
15. Y. Freund and R. E. Schapire, "Game theory, on-line prediction and boosting," in *Proceedings of the ninth annual conference on Computational learning theory*, ser. COLT '96. New York, NY, USA: ACM, 1996, pp. 325–332. [Online]. Available: http://doi.acm.org/10.1145/238061.238163
16. D. Rosenberg, E. Solan, and N. Vieille, "Stochastic games with a single controller and incomplete information," Northwestern University, Center for Mathematical Studies in Economics and Management Science, Discussion Papers 1346, May 2002.