How Deep Is Your Defense-in-Depth? Hardening Cybersecurity Network Control Against Adaptive Attackers

Chun Kai Ling[1], **Jakub Černý**[2], **Chin Hui Han**[3], **Christian Kroer**[2], **Garud Iyengar**[2]

[1]National University of Singapore, Singapore, [2]Columbia University, USA, [3]DSO National Labs, Singapore,
chunkail@nus.edu.sg, jakub.cerny@columbia.edu, chuihan@dso.org.sg, christian.kroer@columbia.edu,
garud@ieor.columbia.edu

Abstract

Optimally designing cyber-defenses in a network is a daunting task. In this paper, we study adaptive cyber-attackers, which can modify their attack path in response to any cyber-defense faced during an attack. This problem is formalized as a min-max game played over a network graph. We give examples where adaptive cyber-attackers are more powerful than non-adaptive ones and show that cyber-defenses that do not account for adaptivity can perform arbitrarily worse. We connect the cyber-attacker's optimal strategy with the classical theory of multi-armed bandits, yielding a simple gradient based algorithm to solve the min-max game. Experiments on synthetic settings validate our approach.

1 Introduction

Cyber attacks are increasingly costly, with the global financial impact of cybercrime projected to reach $23 trillion annually by 2027 (US State Department 2024). A mantra in cyber defense is "defense-in-depth", where multiple layers of controls and safeguards are deployed to protect networks and critical assets. Each layer addresses specific threats or vulnerabilities; if one layer is bypassed, others are in place to mitigate risks.

Evaluating the effectiveness of cyber defenses requires understanding the behavior of threat actors and assessing how controls detect and respond to potential attacks. A significant challenge lies in the dynamic nature of cyber operations. For instance, advanced persistent threats (APTs) can adapt to evade existing defenses. In response, defenders may modify their strategies, such as deploying additional firewalls or adjusting network configurations when indicators of compromise are identified. These adjustments, like removing network links or disabling services, aim to deter, disrupt, and defeat attacks while maintaining operational resilience.

Enterprise IT networks. An Enterprise IT network connects an organization's computers, servers, applications, and users to enable communication, collaboration, resource sharing, and supporting business operations. Typical Enterprise IT networks comprise many interconnected systems and devices, forming densely connected graphs. However, in practice, many enterprise services, such as emails and enterprise platforms, operate in parallel and are connected to a few critical components.

One of the critical components in most Enterprise IT networks is the Active Directory (AD) System. The AD system manages IT resources, such as users and computers, identifies users in the network, and automates administrative tasks. Due to the AD system's criticality and connectedness, they are often the prime target for cyber-attacks, as compromising the AD will allow attackers access to most resources. The distance between users and the AD System is often very short and contains many parallel paths through the different enterprise services. Security controls are layered *within* each service. Figure 2 shows possible controls in an email system.

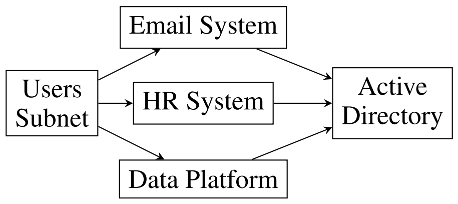


Figure 1: Example of an enterprise IT network.

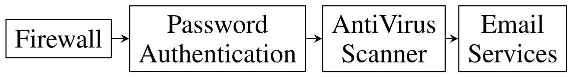


Figure 2: Simplified security controls in an Email System.

Adaptive attackers Prior work on the effectiveness of cybersecurity controls models static and non-adaptive attackers. Such attackers do not account for any changes in the defenses that arise from the attack. Consider Figure 1, and suppose an attacker has failed a password authentication check several times while trying to access the AD via an Email System. These failures have triggered more stringent authentication, impeding future attacks via the Email System. A non-adaptive attacker does not account for such developments and will continue trying to compromise the Email System. However, an adaptive attacker recognizes these elevated defenses and *pivots* to other attack paths such as the HR System or Data Platform.

This paper addresses the following question: *Given that most real-world attackers are adaptive, how should defensive resources be organized*? To this end, we draw upon the successful application of game theory to security and propose a game theoretic model that accounts for the attacker's reactions to the defenses as cyber-attacks unfold.

Contributions. Our contributions include (i) formalizing this problem mathematically as a game between attacker and defender, with the optimal defense allocation as the solution to a min-max problem, (ii) drawing a direct connection between an attacker's optimal policy and the solution to a multi-armed bandit problem and the *Gittins index*, (iii) using the Gittins index to compute the optimal attacker response to any defense efficiently, and (iv) computing the optimal defense policy by projected gradient descent. On the theoretical front, we (v) give examples of where assuming heuristic attacker policy (e.g., greedy) can induce poor defensive allocations, thus necessitating the assumption of an *optimal* adaptive attacker, and (vi) analyze special cases where computing an optimal attack/defense policy is much simpler.

2 Related Work

This paper spans multiple domains, including cybersecurity modeling, game theory, and the decision-making framework of multi-armed bandits and indexible policies.

Cybersecurity, attack graphs, and network interdiction. Attack graphs come in many forms, often tailored by cybersecurity professionals to identify specific weak points in their network infrastructure and develop more effective incident response strategies (Sheyner et al. 2002; Lippmann, Ingols et al. 2005; Strom et al. 2018). They provide a structured representation of the attack process, mapping the various stages from initial reconnaissance to final exploitation. Various models for network security have been proposed. One classic model is that of (stochastic) network interdiction (Cormican, Morton, and Wood 1998; Wang, Noel, and Jajodia 2006), where the defender chooses a subset of edges in the network to remove, subject to some budget constraint. Like us, they assume that the attacker is able to adapt to the altered network. Some sub-variants confer commitment advantages upon the defender (Letchford and Vorobeychik 2013) and some do not (Khouzani, Liu, and Malacaria 2019; Almohri et al. 2015), while others allow the defender to disable vertices instead (Nguyen et al. 2017), or to insert new "honeypot" hosts into the network (Durkota et al. 2015a,b) which can detect and incur penalties for the attacker. Compared to network interdiction, our model does not allow the defender to forcefully remove entire edges or vertices (which drastically affects regular operations) and instead models more realistic controls with internal states that evolve based on the current attack.

Multi-armed bandits and indexable policies. Our network is abstracted into a collection of parallel chains (see Figures 1 and 2) and can be viewed as a multi-armed bandits and indexible policies lens, making analysis much easier. Multi-armed bandits (MAB) are a well-known foundational framework in decision-making under uncertainty due to their theoretical elegance in balancing tradeoffs between exploration and exploitation. In the MAB framework, a single agent makes decisions sequentially to maximize cumulative rewards by selecting from a set of n options (pulling arms), each with uncertain and potentially stochastic reward distributions. MABs come in many flavors, but the one most relevant to this paper allows for random but known state transitions within each arm, affecting rewards obtained. A common theme in this variant of MAB is the optimality of index policies, which states that it is possible to assign to each state in every arm a rank (independent of all other arms) such that the optimal policy is to pull the arm with the highest index. Such dynamic allocation index is now commonly known as the Gittins index, in honor of Gittins (1979). Since then, a huge number of extensions have been proposed, including branching bandits and arm arrivals (Weiss 1988; Bertsimas, Paschalidis, and Tsitsiklis 1995; Bertsimas and Nino-Mora 1996; Whittle 1981), extensions to jobs with precedence constraints (Glazebrook and Gittins 1981), restless bandits (Whittle 1988), as well as a variant by Dumitriu, Tetali, and Winkler (2003) minimizing the time needed to reach a target. Advances have also been made in computing Gittins indices; see, e.g., Chakravorty and Mahajan (2014).

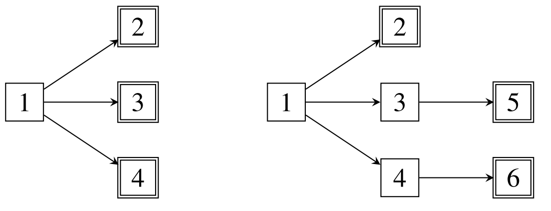
Indexable policies in adversarial settings. Adversarial approaches in multi-armed bandits, often called nonstochastic bandits, focus on settings where reward distributions are influenced by an adversary who dynamically alters them at each timestep (Auer et al. 1995). This formulation differs from our setting, where the defender commits to a stochastic defense strategy, and the attacker faces a rested stochastic environment. While adversarial bandits rarely admit indexable policies, exceptions exist (Xiong and Li 2024). Beyond purely adversarial settings, robust formulations have explored problems where adversarial perturbations interact with stochastic dynamics. For instance, Scully and Harchol-Balter (2018) study job scheduling where job ages are observed with adversarial perturbations. Similarly as in adversarial bandits, attacker can further modify these ages during execution. A variant of the Gittins index achieves near-optimal mean response times. In a related vein, Tan et al. (2018) employ Gittins indices heuristically for controlling a set of pursuers, selecting among n possible pursuing modes against unknown fixed stochastic evader strategies. Another adversarial formulation is studied by Gummadi et al. (2013) who perform a mean-field analysis of multi-armed bandit games where a large population of players interact with a bandit system. The aggregate actions of all players then influence individual reward distributions.

Indexable policies in games. In spirit, our work is closest to two-player game-theoretic models where one player commits to a stochastic strategy, leading to a multi-armed bandit problem for the other player, who responds with an optimal index-based policy. An example is the semi-finite zero-sum hide-and-seek game of Clarkson and Lin (2024), where one player hides randomly among n locations, and the seeker uses a Gittins index policy to execute an optimal search strategy. Similarly, Fudenberg and He (2018) study a two-player general-sum Bayesian game involving a sender and a receiver. The sender, characterized by a type θ, employs a Gittins index policy to select an optimal signal for the receiver who commits to a stochastic strategy.

3 Mathematical Formulation

In this paper, cyberattacks and defenses are modeled as a game defined over graphs. The game is defined by two layers of graphs: (i) there is a single high-level graph, which we call the the network level graph. Vertices in this graph represent IT resources that may potentially be compromised, and edges are potential entry points and controls (if any) between vertices, and (ii) for every edge in the network level graph, we have an edge, or control, level graph. The control-level graph is a Markov chain which represents the evolution of the internal state of each control.

3.1 Network level graph



(a) Single Layered graph (b) General Parallel Chains graph

Figure 3: Examples of network level graphs. Node 1 serves as the entry point v_0, double-squares represent the target.

At a macro-level, we consider a network organized as *parallel chains* of controls. This is represented by a tree $G = (\mathcal{V}, \mathcal{E})$ rooted at $v_0 \in \mathcal{V}$. Examples are shown in Figure 3, with v_0 representing the entry point and leaves representing the target (e.g., the active directory). While the target is represented by multiple leaves (e.g., vertices $2, 3, 4$ in Figure 3a), they refer to the same target (e.g., AD).

Each parallel path represents an IT resource where the target could be reached; for example, a print server could constitute a path leading from access point to target, while an email server constitutes another. Each represents a distinct means to which an attacker may eventually access the target. Edges lying in the path between v_0 and the leaves represent *controls* that an attacker must circumvent to proceed (see Figure 2). Since these controls are placed in series, an attacker must circumvent all of them in a row to gain access to the target. For example, in Figure 3b, the attacker has to circumvent edges $((1, 2))$ or $((1, 3)$ and $(3, 5))$ or $((1, 4)$ and $(4, 6))$ in order to compromise the target.

Associated to each edge $e \in \mathcal{E}$ is the duration required *for each attempt* to traverse an edge, $\ell(e) \in \mathbb{R}_+$, which we call its *length*. We write $\ell \in \mathbb{R}_+^{|\mathcal{E}|}$. In general, we will require $\ell \in \mathcal{L}$, where $\mathcal{L} \subseteq \mathbb{R}_+^{|\mathcal{E}|}$ is some compact, convex set (typically a polyhedron) representing the defender possible strategies (details found in Section 3.4). For convenience, we will suppose $\mathcal{L}$ is the simplex of size $|\mathcal{E}|$; this simplifying assumption will not dramatically affect our algorithms or analysis. Again, for simplicity, we will focus on a class of network-level graphs we call general parallel chain graphs (GPCG, Figure 3b). A GPCG is a graph where all vertices except v_0 have an indegree of 1 and an outdegree of at most 1. Only the root node v_0 has an indegree of 0 and an outdegree > 1. A special case where each chain has only a single vertex we call a Single Layered Graph (SLG, Figure 3a).

3.2 Control state graphs

Cybersecurity applications are complicated by the fact that controls themselves have internal states that evolve upon any interaction with the attacker. For example, a failed attempt at guessing a password can disable attempts for a fixed amount of time; additional attempts increase restrictions to the point where access is blocked entirely.

Such interactions are modeled by the control state graph. The control state graph describes the probability of successfully circumventing a control given by edge $e \in \mathcal{E}$. This probability is governed by e's internal control state, which changes with each attempt via a discrete-state-discrete-time Markov chain with a state space $\mathcal{S}_e$ and transition function $T_e(s, s') \in [0, 1]$, where T_e specifies the probability that a transition occurs from state s to s'. An edge's state $s(e) \in \mathcal{S}_e$ reflects the extent to which an attacker has attempted to circumvent the controls for that edge, and includes a special state ϕ, which indicates that the control for e has been fully circumvented. If e has a state ϕ, then the vertex $e[1]$ is compromised. Note that a vertex v is compromised if *any* edge e where $e[1] = v$ is compromised.

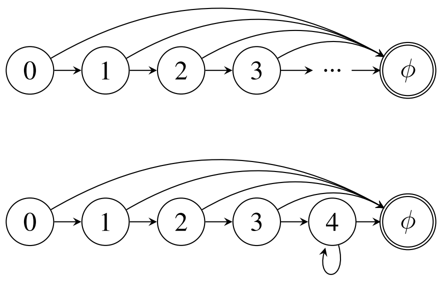


Figure 4: Example of control state graphs. (Left) graph with infinite $\mathcal{S}_e$ and (right) graph with finite $\mathcal{S}_e$ and $i_{\text{max}} = 4$. In the latter, if $p_e(4, \phi) = 0$, then the control is "stuck" at state 4.

For this paper, we restrict ourselves to chain-like state structures where $\mathcal{S}_e = \{0, 1, \dots, \phi\}$ and $T_e(i, \phi) + T_e(i, i + 1) = 1$.[1] In chain-like structures, the control state represents the number of attempts that have been made to cross an edge. There are only two transitions: *success*, which brings $s(e)$ from i to ϕ, or *failure*, which brings one from state i to $i + 1$ (Figure 4). Specifying a chain structure requires specifying $T_e(i, \phi)$ for all $i \geq 1$. Some reasonable transition functions are:

[1]In general, any Markov chain would work just as well (i.e., the general idea behind our formulation and algorithms should still hold). However, this more general formulation significantly increases computational costs and needlessly complicates the model.

- **Constant probability of success.** $T_e(i, \phi) = k_e$, for some constant $k_e \in [0, 1]$. E.g., cracking a time-based one-time-pin. The pin is reset periodically.
- **Constant work.** $T_e(i, \phi) = 0$ if $i < k_e$ and $T_e(i, \phi) = 1$ if $i = k_e$ for some $k_e \in \mathbb{Z}$. E.g., doing a port scan for unsecure ports and using a known exploit.
- **Increasing probability of success.** $T_e(i, \phi) = 1/(k_e - i)$ for some $k_e \in \mathbb{Z}_+$. E.g., cracking static password hashes. The probability of cracking improves over time as the search space of possible keys reduces.
- **Exponential backoffs.** $T_e(i, \phi) = \hat{k}_e \cdot k_e^i$ for $\hat{k}_e, k_e \in [0, 1]$. The probability of success decreases exponentially with the number of failed attempts, and occurs when the control throttles the rate of attempts with each failure.
- **Success probabilities via beta priors.** $T_e(i, \phi) = \alpha_e/(\alpha_e + \beta_e + i)$ for $\alpha_e, \beta_e > 0$. Here, the success probability is governed by beta prior and its posterior is updated based on the number of failures so far.

Other chain-like transitions could be conceived for different types of controls. Also note that the form of T_e may differ for each edge, i.e., a network could mix and match between different types of controls.

Finite control state graphs. To simplify analysis and implementation, we allow for finite approximations. For example, under exponential backoffs, when k_e is sufficiently small the probability of circumventing the control after a failed attempts becomes practically 0. We thus assume that past a particular state $i_{\max}$, we have $T_e(i, \phi) = T_e(i_{\max}, \phi)$. This then reduces to a finite control state graph. This is illustrated in Figure 4 for $i_{\max} = 4$. Note that if $T_e(i_{\max}, \phi) = 0$ the attacker is can be locked out from this control entirely if the final once state $i_{\max}$ is reached.

Monotonicity. Sometimes, T_e is *monotonic* in i, i.e., T_e is non-increasing or non-decreasing. For example, the constant work and hash collision examples are monotonically non-decreasing, while exponential backoff and beta priors are monotonically non-increasing (constant probabilities satisfy both). The former means that subverting a control gets easier with more attempts, while the latter models the opposite.

3.3 Attacker policy

The attacker makes decisions sequentially at discrete times depending on the lengths ℓ, starting from time $t = 0$. At any time t, every edge $e \in \mathcal{E}$ maintains its internal state $s_t(e) \in \mathcal{S}_e$. Remember that $s_t(e)$ reflects the extent to which the attacker has attempted to circumvent that e's control, inclusive of a special state ϕ indicating that all controls for that edge have been circumvented. For instance, the set of internal states for edges with controls involving exponential backoff would be $\{0, 1, \ldots, \phi\}$, i.e., the number of failed attempts. We denote by $\Phi_t \subseteq \mathcal{E}$ the set of edges e where $s_t(e) = \phi$. At the start of any decision point at time t, the attacker chooses an edge $e = (v_i, v_j)$ to attempt to compromise, such that there exist $e' = (v_k, v_i) \in \Phi_t$ and $e \notin \Phi_t$. The next decision point is at $t + \ell(e)$, where $s_{t+\ell(e)}(e)$ evolves according to some transition function $T_e(s_t(e))$, and $s_{t+\ell(e)}(e') = s_t(e')$ for all $e' \in \mathcal{E}\backslash\{e\}$. We denote the discrete times that actions are taken by t_i, starting from $t_1 = 0$.

The run ends just *after* the some edge $e = (v_i, w)$ is traversed, where w is a leaf, after the i-th action is completed at t_{i+1}. We denote this completion time as t^*. Concretely, this is the point where there exists some path $e_1, e_2, \ldots, e_m$ from v_0 to w, where $s(e_1) = s(e_2) = \cdots = s(e_m) = \phi$, i.e., a successful attack chain from the network's entry point to target is formed. Given a discount rate $\lambda > 0$, the attacker then obtains $\exp(-\lambda \cdot t^*)$ utility.

Optimal attacker policies. Given some fixed $\ell(e)$, the attacker seeks to end the game such that the time-discounted reward $\mathbb{E}_{t^*}(\exp(-\lambda \cdot t^*))$ is maximized. Here, the expectation is taken over any randomness in state transitions in any edges, as well as any randomness in the attacker policy.

Example 1 (Transition functions and attacker pivoting)**.** Consider a single-layer network graph with two target nodes, 2 and 3, connected to the start node via two network-level edges, as shown in Figure 5. Assume that both edges have a length of 1, but the individual edge control state graphs differ: the longer (upper) edge contains two intermediate states, while the shorter (lower) edge contains one intermediate state. We will show the difference between constant work and exponential backoff transition functions. Assume that in the constant work function, the longer edge requires 1 try to cross, while the shorter edge requires a constant work of 2 attempts. In this case, $a = 1$, $b = 1$, and $c = 0$. The optimal policy of the attacker is to choose the edge with the smaller constant work, resulting in the longer edge being selected without pivoting. With exponential backoff transitions, the dynamics change. Let the longer edge transition function have $\hat{k} = 0.8$ and $k = 0.75$, corresponding to $a = 0.8$ and $b = 0.6$. For the shorter edge, let $\hat{k} = 0.7$ and $k = 1$, giving $c = 0.7$. The attacker rationally attempts the longer edge first due to the higher initial success probability ($0.8 > 0.7$). If the attempt succeeds, the game ends immediately. If it fails, however, the attacker reevaluates the remaining probabilities: the longer edge now offers a success probability of 0.6, while the shorter edge has 0.7. The attacker then pivots to the shorter edge, maximizing the probability of ending the game sooner. Larger networks can exhibit even more complex pivoting, as shown in Figure 6.

MDP formulation to compute optimal attacker policies. The problem of finding an optimal policy can be formulated as a Markov Decision Process (MDP). We omit a formal mathematical MDP formulation since the details are lengthy but not illuminating and instead focus on portions relevant to our proposed algorithm.

Let there be n chains (i.e., v_0 has n children). At a high level, the k-th chain can be seen as a Markov chain with state space $\mathcal{C}_k$, comprising elements of the form (i, j), where i is an edge in that chain that has not yet been compromised, and j is how many times control i (in that chain) has been attempted. The "global" state over the entire MDP is simply the Cartesian product $\mathcal{C} = \prod_{k \in [n]} \mathcal{C}_k$ over the states of each chain, while the action would be to choose a single chain and make its corresponding Markov chain advance take a single

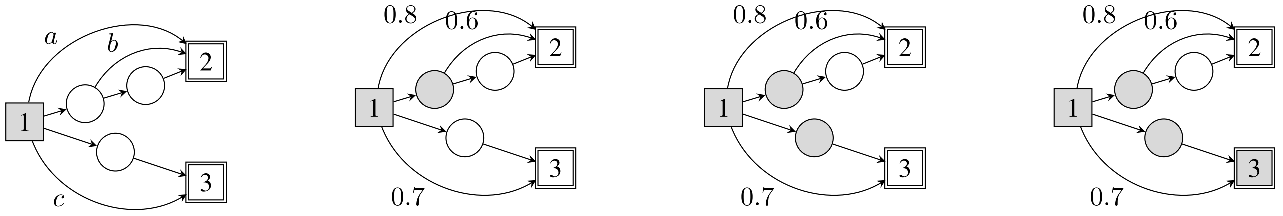


Figure 5: (Left-most) Example of a single layered graph with two network-level edges (upper edge $(1,2)$ and lower edge $(1,3)$) corresponding to control state graphs with 2 and 1 intermediate states, respectively. The transition functions are given by probabilities a, b, and c. (Next 3 figures) Path to compromise the network in the exponential backoff transition model when attempts to traverse edges fail. Compromised nodes are shaded.

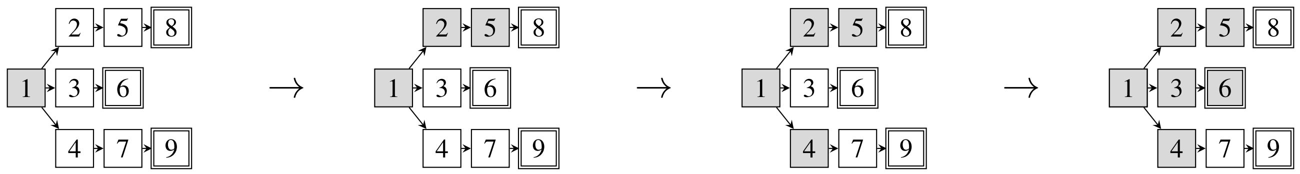


Figure 6: Example in a larger network-level graph, showcasing a more complex pivoting strategy of the attacker when attempts to cross edges fail. Control state graphs of individual edges are omitted. Compromised nodes depicted shaded.

transition. Therefore, a network with n chains, m controls (edges) in each chain each having $|\mathcal{S}| = q$ states has $(mq)^n$ states and n actions. When the global state is such that one chains has reached a target, it receives a payoff of 1 and the game ends (by entering a global terminal state). To account for lengths, we allow for state dependent discounting of rewards, i.e., when some control e is attempted to be traversed, it incurs a discount of $\exp(-\lambda \cdot \ell(e))$. We can avoid relying on state dependent discounting by introducing a probability of $1 - \exp(-\lambda \cdot \ell(e))$ of transitioning to the global terminal state at every transition. The optimal attacker strategy against fixed ℓ is the solution to this MDP. A deterministic, stationary (time independent) policy $\pi : \mathcal{S} \to \mathcal{E}$ assigns each state to a choice of chain $[n]$.

The key implication is that there exists a *deterministic, stationary* attacker solution with a finite optimal value (Puterman 2014). Furthermore, there are a finite number of deterministic stationary policies π to choose from. We denote such an optimal strategy by π^*_ℓ. We will also denote the values of such policies by $V^\pi_\ell = \mathbb{E}_{t^*}[\exp(-\lambda \cdot t^*)|\lambda, \pi]$ and V^*_ℓ. We omit the subscript of ℓ when its dependence is clear from the context.

If $\mathcal{S}_e$ are finite and $\ell(e) > 0$ for all $e \in \mathcal{E}$, then the MDP itself is finite with strict discounting. Thus, its solution may be theoretically found via textbook methods such as value or policy iteration (Puterman 2014). Solving this MDP directly is however intractable, since the number of states is exponential in the number of chains. Later, we will exploit a simple but important observation that MDPs specified by GPCGs in our setting can be made tractable by resorting to *index policies* (Section 4.1).

Remark 1. We stress that in the MDP formulation, the attacker is *not* traversing G by moving from vertex to vertex, and the state space $\mathcal{S}$ is *not* the set of vertices in G.

3.4 Defender Policies

The defender selects some $\ell \in \mathcal{L} \subseteq \mathbb{R}^{|\mathcal{E}|}_+$, a vector containing the length of each edge. Here $\mathcal{L}$ is some convex set that accounts for budget or other performance constraints on the part of the defender. For example, one may choose a huge $\ell(e)$ when verifying a password. However, an extremely high $\ell(e)$ will dramatically deteriorate performance for legitimate users. For instance, a reasonable $\mathcal{L}$ could be the (nonempty) polyhedron given by $\{\ell \in \mathbb{R}^{|\mathcal{E}|}_+ | A\ell + c \geq d\}$. For the rest of the paper and our experiments, we assume $\mathcal{L} = \Delta_{|\mathcal{E}|} = \{\ell \in \mathbb{R}^{|\mathcal{E}|}_+ | 1^T\ell = 1\}$, the simplex of size $|\mathcal{E}|$.

For any fixed ℓ, the attacker's optimal play yields an expected discounted utility of V^*_ℓ. The defender's objective is select ℓ to minimize V^*_ℓ, i.e., it solves the min-max problem

$$\min_{\ell \in \mathcal{L}} V^*_\ell = \min_{\ell \in \mathcal{L}} \max_\pi V^\pi_\lambda = \min_{\ell \in \mathcal{L}} \max_\pi \mathbb{E}_{t^*}[\exp(-\lambda \cdot t^*)|\ell, \pi].$$

First observe that V^*_ℓ is finite, since $\exp(-\lambda \cdot t^*) \leq 1$. Second, it is also convex in ℓ because it is the pointwise maximum over a set of convex functions (and $\mathbb{E}_{t^*}[\exp(-\lambda \cdot t^*)|\ell, \pi]$ is convex in λ). Since $\mathcal{L}$ is compact, a minimum exists and is attainable.

Remark 2. Interestingly, if we allow the attacker to play a *random* policy (of which there are a finite of for finite $\mathcal{S}_e$), then (i) the value of this min-max problem remains the same, and (ii) the minimax theorem holds (v. Neumann 1928; Sion 1958), implying that a random policy performs at least as well for *any* choice of $\ell \in \mathcal{L}$. Computing such a Nash equilibrium is beyond the scope of this paper.

3.5 Heuristic attackers and their suboptimality

An alternative to an optimal (and complicated) attacker is one utilizing heuristics. Three heuristics come to mind:

- **Constant.** An attacker chooses and sticks to a single chain throughout, never pivoting to another chain.
- **Greedy.** A greedy attacker looks at the set of adjacent controls to be circumvented and picks the one that has the highest discounted probability of being circumvented in the next attempt, i.e., at time t for a edge e with control state $s_t(e) = i$, $T_e(i, \phi) \cdot \exp(-\lambda \cdot \ell(e))$. The myopic attacker makes the most "immediate progress", irrespective of how future controls lead to the target.
- **Greedy probability.** Choose the adjacent control most likely crossed in the next attempt, (i.e., max $T_e(i, \phi)$).

These attacker heuristics are highly (in fact, arbitrarily) suboptimal compared to the optimal policy, as shown below.

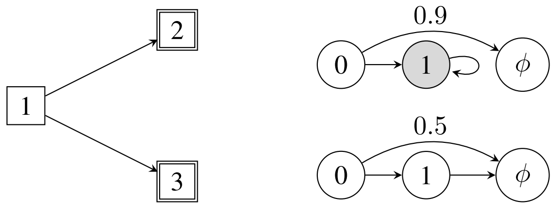


Figure 7: Example of a graph where a constant player may fail to reach a target entirely by not pivoting. (Left) A network level representation. (Right) Control state graphs of the top and bottom edges.

Example 2 (Suboptimality of constant heuristic)**.** Figure 7 depicts a network with two network-level edges. Assume the $\ell(e) = 1$ and $\lambda = 1$. At the beginning, the upper edge is preferred by a constant attacker due to its higher utility of $0.9 \exp(-1) + (1 - 0.9) \exp(-\infty) \approx 0.33$, compared to the lower edge's utility of $0.5 \exp(-1) + (1 - 0.5) \exp(-2) \approx 0.25$. Thus, it selects the upper edge. However, if the first attempt fails, the state of the upper edge moves to the shaded absorbing state and the constant attacker becomes permanently stuck. In contrast, after such an initial failure an optimal attacker pivots to the lower edge.

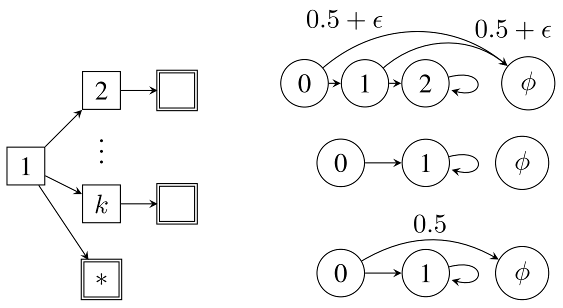


Figure 8: Graph where a greedy player traverses $n-1$ chains before pivoting to an optimal chain. (Left) A network level representation. (Right) Control state graphs of [top] edges $(1, 2), \ldots, (1, n)$, [middle] edges $(2, \text{target}), \ldots, (n, \text{target})$, and [bottom] edge $(1, *)$.

Example 3 (Suboptimality of greedy heuristic)**.** Now consider a network with k chains in Figure 8. The right side of the figure shows individual control state graphs for the first and second layer in the first $k-1$ chains, and the edge connecting the last chain, respectively. Assume again $\ell(e) = 1$ and $\lambda = 1$. In this network, the first $k-1$ chains lead to targets that are unreachable due to absorbing states in the second-layer control state graphs. However, a greedy attacker does not account for the future. For $\epsilon \in (0, 0.5)$, the first $k-1$ chains are strictly preferred by a greedy attacker over the bottom chain. Hence, a greedy attacker attempts the first $k-1$ chains first, spending at least $k-1$ timesteps to do so before pivoting to the bottom chain. In contrast, an optimal attacker disregards the first $k-1$ chains and immediately focuses on the bottom chain. The expected discounted reward of a greedy attacker is less than $0.5 \cdot \exp(-k)$, exponentially worse than the optimal of $0.5 \cdot \exp(-1)$.

Examples 2 and 3 show how heuristic attackers may be highly suboptimal against a fixed ℓ. One may still hope that such heuristic attackers could help in approximating the optimal defender ℓ. Unfortunately, this turns out to be untrue.

Example 4 (Exploitability of optimal ℓ allocation against greedy attacker)**.** Consider again Example 3 and a greedy attacker, except ℓ is not fixed. A defender can "trick" a greedy attacker into wasting time to attempt the first $k-1$ edges. To achieve this, it increases the length of each edge in the first $k-1$ chains by $1 + \ln(0.5 + \epsilon) - \ln(0.5)$, just enough to make the greedy attacker indifferent between the upper and bottom chains (or more precisely, to prefer the upper chains slightly). At this point, increasing $\ell(e)$ for the first $k-1$ edges increases the total expected time by at least $1 + 1 \cdot 0.5 = 1.5$, compared to 1 in the bottom edge. In other words, the optimal defender policy is to increase ℓ over the first layer of edges in the upper $k-1$ chains uniformly until attacker indifference. Further increasing the lengths of these edges would require simultaneously increasing the length of the bottom edge to prevent the attacker from pivoting.

However, an optimal attacker would never traverse any edges in the upper chains since it knows there is no chance of circumventing the downstream control. Thus, such an allocation of ℓ is highly inefficient, spending less than $1/k$ of the budget on the only edge that matters (the bottom one).

4 Computing Optimal Policies

Given that V^*_ℓ is convex in ℓ, the simplest approach towards finding the optimal ℓ is projected subgradient descent. To do this, we require some subgradient of $\max_\pi V^\pi_\ell$, which can be done by appealing to Danskin's theorem (Bertsekas 1997), giving $g = \nabla_\ell V^*_\ell$. In particular, our implementation using the regret matching (Algorithm 1) method by Hart and Mas-Colell (2000) and is straightforward. At each iteration j, we compute the optimal attacker strategy; with this on hand, we obtain g via stochastic samples of $\nabla_\ell V^*_\ell$ via sampling, and finally, we update $\ell^{(j+1)}$ using the sampled g.

4.1 Optimal Attacker Policies via Gittins Indices

As it turns out, the optimal policy π^*_ℓ for fixed ℓ can be efficiently computed by appealing to the *Gittins policy and*

Algorithm 1: Regret Matching for optimizing ℓ

```
$\ell^{(1)} \leftarrow 1 \cdot \frac{1}{|\mathcal{E}|}$,                         {Initialize lengths}
$y^{(1)} = 0 \cdot \frac{1}{|\mathcal{E}|}$                          {Initialize regret}
for $j = 1, \ldots, j_{\max}$ do
    $\pi^{(j)} \leftarrow$ AttackerOptimalStrategy($\ell^{(j)}$)
    $g^{(j)} \leftarrow$ Subgradient($\ell^{(j)}, \pi^*_{\ell^{(j)}}$)
    $y^{(t+1)} \leftarrow y^{(t)} + 1\langle g^{(j)}, \ell^{(j)}\rangle - g^{(t)}$    {Update regrets}
    if $y^{(t+1)}$ is not all $\leq 0$ then
        $\ell^{(j+1)} \leftarrow \max\left(y^{(j)}, 0\right) / \sum_e \max\left(y^{(j)}(e), 0\right)$
    else
        $\ell^{(j+1)} \leftarrow 1 \cdot \frac{1}{|\mathcal{E}|}$            {Set $\ell^{(j+1)}$ to be uniform}
    end if
end for
return $\sum_j^{j_{\max}} \ell^{(j)} / j_{\max}$           {Return average strategy}
```

index. Recall the MDP formulation of the attacker's optimal policy (Section 3.3) and the state representation for the k-th chain given by $\mathcal{C}_k$. Essentially, the exponential-sized MDP may be sidestepped by isolating each chain *independently* of the rest and computing the Gittins index $\alpha_k(i,j)$ for every chain state in $c = (i,j) \in \mathcal{C}_k$. Then, the celebrated theorem of Gittins (Gittins, Glazebrook, and Weber 2011) states that the optimal attacker strategy is to select the control corresponding to the chain with the highest index, $\text{argmax}_{k \in [n]} \alpha_k(i_k, j_k)$, where i_k, j_k is the current chain state of chain k. The crucial takeaway is that α_k can be computed solely as a function of i_k, j_k, independently of the other states of other chains. Yet, the indices $\alpha_k(\cdot,\cdot)$'s are compared across chains when determining the π^*. The Gittins index for a given (i_k, j_k) is given by:

$$\alpha_k(i_k, j_k) = \max_{\tau > 0} \frac{\mathbb{E}[R \cdot \exp(-\lambda \cdot \tau)]}{\mathbb{E}[\int_0^\tau \exp(-\lambda \cdot t) dt]}, \tag{1}$$

where τ is a strictly positive *stopping time* and R is an indicator random variable denoting whether the target was reached. The numerator in (1) is the expected discounted reward under the optimal stopping policy (possibly 0), and the denominator is the expected discounted time. In the special case of SLGs (Figure 3a), the following theorem follows.

Theorem 3 (Gittins, Glazebrook, and Weber (2011))**.** *Suppose G is a single-layered graph. Then we have: (i) if all controls have $T_e(\cdot, j)$ monotonically non-increasing in j, then the greedy-probability attacker policy (Section 3.5) is optimal, and (ii) if $T_e(\cdot, j)$ is monotonically non-decreasing, then the constant chain (non-pivoting) attacker policy (Section 3.5) is optimal.*

Remark 4. The Gittins policy has many variants, the most common of which optimizes discounted *cumulative* rewards. Unlike our setting, the game ends once any leaf is compromised. Thankfully, a standard result from queuing theory guarantees equivalent optimal policies in both settings (Gittins, Glazebrook, and Weber 2011).

Computing the Gittins Index. For the k-th chain, standard methods for computing Gittins index require cubic time in the number of states ($|\mathcal{C}_k|$). However, additional speedups may be enjoyed because each control itself is a chain.

Theorem 5. *Consider a chain comprising m monotonically non-increasing controls, each with control graphs with q states. The Gittins index for all mq elements of $\mathcal{C}_k$ can be computed in $\mathcal{O}(m^2 q)$ time.*

Proof. Our algorithm follows standard methods to computing Gittins indices (Gittins, Glazebrook, and Weber 2011) based on state elimination.[2] Our algorithm computes Gittins index for each of the $|\mathcal{C}_k| = mq$ elements in decreasing order. In iteration z, we look at all $\mathcal{C}_k - z + 1$ un-added elements in $\mathcal{C}_k$ and find the element with the next highest Gittins index. However, because of monotonicity, we only need to check m, not mq of them. In addition, because of the chain structure, finding this next highest element only requires $\mathcal{O}(m)$ amortized time. ☐

4.2 Performing gradient descent

Once π^*_ℓ is found, the second half of Algorithm 1 finds approximate subgradients and takes a gradient step. In this paper, we adopt the regret matching algorithm of Hart and Mas-Colell (2000). The uniform average of the iterates generated by RM is guaranteed to converge to the optimum, in the sense that the duality gap decreases at a rate of $\mathcal{O}(1/\sqrt{T})$ The main technical challenge is to compute V^*_ℓ and its subgradient with respect to ℓ. We are unaware of a procedure to obtaining exact subgradients of V^*_ℓ [3]. Thus, we rely on stochastic subgradients by running $Z \geq 1$ simulations under π^* and taking averages of the subgradients with respect to ℓ of every simulation.

Remark 6. In SLGs, subgradients of V^*_ℓ can be easily be obtained without sampling, because any control successfully passed ends the game. Computing this expectation does not require "branching" and can be done in $\mathcal{O}(nmq)$ time.

Remark 7. An earlier version of this paper utilized online mirror descent/Hedge, which has the practical downside of requiring an appropriately chosen step size that cannot be too large. This was the source of a significant amount of numerical instability which was avoided by employing regret matching instead. We have also implemented a newer variant that computes *exact* V^{π_ℓ} and its (sub)gradients in polynomial time. It's implementation is beyond the scope of this paper, albeit significantly faster than most choices of Z. At any rate, we observed that with sufficiently many samples (e.g., $Z \geq 1000$), the resultant ℓ is virtually the same.

5 Experiments

For simplicity, we consider chain graphs of n parallel chains each with $m = 3$ controls and monotonic controls (Section 3.2). We consider two types of controls: those using exponential backoffs and beta priors. We randomly sample $\hat{k}_e, \hat{k}_e, \alpha_e, \beta_e$ uniformly in $[0.2, 0.8]$. For each case, we fix the class of controls with randomized hyperparameters,

[2]A full, formal proof would require delving into the details of prior work. Hence, we provide only a sketch.

[3]Very recent follow-up work presents a algorithm for doing so

n	λ	GIvGI	GRvGR	GRvGI	$\frac{\text{GIvGI}}{\text{GRvGI}}$	$\frac{\text{GRvGR}}{\text{GRvGI}}$
3	0.5	4.0e-1	3.4e-1	4.9e-1	0.82	0.69
3	1.0	2.6e-1	2.1e-1	3.9e-1	0.67	0.54
3	2.0	1.5e-1	1.4e-1	2.3e-1	0.65	0.61
3	5.0	3.4e-2	2.9e-2	4.9e-2	0.69	0.59
3	10.0	4.7e-3	4.0e-3	8.1e-3	0.58	0.49
5	0.5	5.9e-1	4.8e-1	7.1e-1	0.83	0.68
5	1.0	4.4e-1	2.9e-1	6.1e-1	0.72	0.48
5	2.0	2.7e-1	1.5e-1	5.1e-1	0.53	0.29
5	5.0	9.1e-2	8.1e-2	2.0e-1	0.45	0.40
5	10.0	2.3e-2	3.4e-2	5.7e-2	0.40	0.60
8	0.5	7.2e-1	5.4e-1	8.7e-1	0.83	0.62
8	1.0	5.7e-1	3.4e-1	8.2e-1	0.70	0.41
8	2.0	3.9e-1	1.6e-1	7.0e-1	0.56	0.23
8	5.0	1.7e-1	1.1e-1	2.9e-1	0.59	0.38
8	10.0	6.0e-2	7.7e-2	1.6e-1	0.38	0.48

Table 1: Results for controls utilizing exponential backoffs.

n	λ	GIvGI	GRvGR	GRvGI	$\frac{\text{GIvGI}}{\text{GRvGI}}$	$\frac{\text{GRvGR}}{\text{GRvGI}}$
3	0.5	6.0e-1	3.4e-1	7.7e-1	0.78	0.44
3	1.0	4.2e-1	2.1e-1	6.4e-1	0.66	0.33
3	2.0	2.2e-1	1.4e-1	3.8e-1	0.58	0.37
3	5.0	5.0e-2	2.9e-2	8.7e-2	0.57	0.33
3	10.0	6.2e-3	4.0e-3	1.0e-2	0.62	0.40
5	0.5	7.3e-1	4.8e-1	9.3e-1	0.78	0.52
5	1.0	5.6e-1	2.9e-1	8.7e-1	0.64	0.33
5	2.0	3.6e-1	1.5e-1	7.4e-1	0.49	0.20
5	5.0	1.3e-1	8.1e-2	3.3e-1	0.39	0.25
5	10.0	3.1e-2	3.4e-2	6.7e-2	0.46	0.51
8	0.5	8.1e-1	5.4e-1	9.9e-1	0.82	0.55
8	1.0	6.8e-1	3.4e-1	9.7e-1	0.70	0.35
8	2.0	4.9e-1	1.6e-1	8.9e-1	0,55	0.18
8	5.0	2.3e-1	1.1e-1	5.7e-1	0.40	0.19
8	10.0	8.5e-2	7.7e-2	2.1e-1	0.40	0.37

Table 2: Results for controls utilizing beta priors.

namely n and λ. For each hyperparameter, we ran 5 experimental runs and reported averages. The experiments were implemented in Python and run on a Macbook Pro M1 with 16GB of RAM (memory was not a limiting factor). We run Algorithm 1 for 2000 iterations each and used the average ℓ returned by Algorithm 1 for evaluation of V^{π_ℓ}. [4]

Quantitative Results. The results are reported in Tables 1 and 2 for varying values of n and λ. The columns of the form "XvY" report the average rewards obtained by the attacker. The first term X refers to the attacker model that the defender optimized ℓ against (which could depend on ℓ, e.g., the policy induced by the Gittins index), while the second term Y refers to the opponent faced at test time. The column containing $\frac{\text{GIvGI}}{\text{GRvGI}}$ is the (multiplicative) suboptimality of the defender. This tells us how much worse it gets when the defender assumes the attacker is greedy, but the actual attacker is optimal. This ratio should be strictly lower than 1, and highlights the importance of accounting for the attacker's adaptive behavior. The column containing $\frac{\text{GRvGR}}{\text{GRvGI}}$ is the misevaluation that can occur. The numerator is the payoff the defender thinks they will get against the greedy attacker, while the denominator is what they actually get when the attacker turns out to be optimal. If this number is very low, then the defender would be misled into thinking that it has much stronger defences than in reality.

We perform some sanity checks. For a fixed n, as λ increases, both [GIvGI] and [GRvGR] decrease. This is expected, as the reward at the end is more highly discounted. Conversely, if we fix λ and vary n, we see that [GIvGI] and [GRvGR] increase. This is again unsurprising: as n increases, there are more attack paths to pivot to, forcing the defender to spread its defenses thinly. Next, we observe that $\frac{\text{GRvGR}}{\text{GRvGI}}$ was consistently less than 1 for all experiments. In other words, a defender who optimizes (and tests against) a greedy attacker will be significantly overconfident when facing off against an optimal adaptive attacker.

Exploitability of heuristic-attacker models. We observe that in all cases $\frac{\text{GIvGI}}{\text{GRvGI}}$ is lower than 1. This implies optimizing against an optimal attacker is more effective than the greedy heuristic. This supports our claim that a defender should optimize based on the true min-max formulation rather than to rely on heuristic attackers.

Computational costs. We observe that computational costs scale roughly linearly with n, primarily because Gittins indices are computed independently for each of the n chains. This is practically significant, as the naive MDP formulation has exponentially many states. More investigation is required as other factors are involved, e.g., sampling time required for stochastic gradients, arguably more costly than Gittins index computation. Our stochastic subgradient method took no more than 1000 seconds, while experiments for beta priors (Table 2) were around 5 times faster.

Qualitative observations. In general, more ℓ is allocated to the controls at the front, as these must be circumvented before the later controls. In some rare cases, stronger controls are placed at the end of the chain, particularly when T_e is such that the attacker must attempt to cross multiple times.

6 Conclusion

In this paper, we studied the effect of adaptive cyber-attackers and the impact that adaptivity has on optimal defender policies. Our min-max formulation gives the optimal defense against an optimal adaptive attacker. Our proposed algorithm based on running gradient descent on the optimal Gittins policy yields promising results superior to those based on heuristic-based attackers. Future work includes scaling up and stabilizing our gradient method and extending the network graph beyond parallel chain graphs.

[4]In an earlier version, we sampled 100000 playthroughs to obtain empirical averages of V^{π_ℓ}; an updated method allowed us to efficiently and exactly compute this without resorting to sampling (Remark 7). There is essentially no quantitative difference in evaluation, though the exact method is much faster.

References

Almohri, H. M.; Watson, L. T.; Yao, D.; and Ou, X. 2015. Security optimization of dynamic networks with probabilistic graph modeling and linear programming. *IEEE Transactions on Dependable and Secure Computing*, 13(4): 474–487.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th annual foundations of computer science*, 322–331. IEEE.

Bertsekas, D. P. 1997. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334.

Bertsimas, D.; and Nino-Mora, J. 1996. Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21(2): 257–306.

Bertsimas, D.; Paschalidis, I. C.; and Tsitsiklis, J. N. 1995. Branching bandits and Klimov's problem: Achievable region and side constraints. *IEEE Transactions on Automatic Control*, 40(12): 2063–2075.

Chakravorty, J.; and Mahajan, A. 2014. Multi-Armed Bandits, Gittins Index, and its Calculation. *Methods and applications of statistics in clinical trials: Planning, analysis, and inferential methods*, 2: 416–435.

Clarkson, J.; and Lin, K. Y. 2024. Computing optimal strategies for a search game in discrete locations. *INFORMS Journal on Computing*.

Cormican, K. J.; Morton, D. P.; and Wood, R. K. 1998. Stochastic network interdiction. *Operations Research*, 46(2): 184–197.

Dumitriu, I.; Tetali, P.; and Winkler, P. 2003. On playing golf with two balls. *SIAM Journal on Discrete Mathematics*, 16(4): 604–615.

Durkota, K.; Lisý, V.; Bošanský, B.; and Kiekintveld, C. 2015a. Approximate solutions for attack graph games with imperfect information. In *Decision and Game Theory for Security: 6th International Conference, GameSec 2015, London, UK, November 4-5, 2015, Proceedings 6*, 228–249. Springer.

Durkota, K.; Lisý, V.; Bošanský, B.; and Kiekintveld, C. 2015b. Optimal network security hardening using attack graph games. In *Proceedings of IJCAI*, 7–14.

Fudenberg, D.; and He, K. 2018. Learning and type compatibility in signaling games. *Econometrica*, 86(4): 1215–1255.

Gittins, J.; Glazebrook, K.; and Weber, R. 2011. *Multi-armed bandit allocation indices*. John Wiley & Sons.

Gittins, J. C. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2): 148–164.

Glazebrook, K. D.; and Gittins, J. 1981. On single-machine scheduling with precedence relations and linear or discounted costs. *Operations Research*, 29(1): 161–173.

Gummadi, R.; Johari, R.; Schmit, S.; and Yu, J. Y. 2013. Mean field analysis of multi-armed bandit games. *Available at SSRN 2045842*.

Hart, S.; and Mas-Colell, A. 2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5): 1127–1150.

Khouzani, M.; Liu, Z.; and Malacaria, P. 2019. Scalable min-max multi-objective cyber-security optimisation over probabilistic attack graphs. *European Journal of Operational Research*, 278(3): 894–903.

Letchford, J.; and Vorobeychik, Y. 2013. Optimal interdiction of attack plans. In *AAMAS*, 199–206. Citeseer.

Lippmann, R. P.; Ingols, K. W.; et al. 2005. An annotated review of past papers on attack graphs.

Nguyen, T. H.; Wright, M.; Wellman, M. P.; and Baveja, S. 2017. Multi-stage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis. In *Proceedings of the 2017 Workshop on Moving Target Defense*, 87–97.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Scully, Z.; and Harchol-Balter, M. 2018. SOAP bubbles: Robust scheduling under adversarial noise. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 144–154. IEEE.

Sheyner, O.; Haines, J.; Jha, S.; Lippmann, R.; and Wing, J. M. 2002. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, 273–284. IEEE.

Sion, M. 1958. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1): 171–176.

Strom, B. E.; Applebaum, A.; Miller, D. P.; Nickels, K. C.; Pennington, A. G.; and Thomas, C. B. 2018. Mitre att&ck: Design and philosophy. In *Technical report*. The MITRE Corporation.

Tan, C.; Xu, C.; Yang, L.; and Wong, W. S. 2018. Gittins index based control policy for a class of pursuit-evasion problems. *IET Control Theory & Applications*, 12(1): 110–118.

US State Department. 2024. United States International Cyberspace & Digital Policy Strategy. *U.S. Department of State*.

v. Neumann, J. 1928. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1): 295–320.

Wang, L.; Noel, S.; and Jajodia, S. 2006. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18): 3812–3824.

Weiss, G. 1988. Branching bandit processes. *Probability in the Engineering and Informational Sciences*, 2(3): 269–278.

Whittle, P. 1981. Arm-acquiring bandits. *The Annals of Probability*, 9(2): 284–292.

Whittle, P. 1988. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A): 287–298.

Xiong, G.; and Li, J. 2024. Provably Efficient Reinforcement Learning for Adversarial Restless Multi-Armed Bandits with Unknown Transitions and Bandit Feedback. In *Forty-first International Conference on Machine Learning*.