Security Games with Layered Defenses: Adaptive Adversaries and Gittins Indices

Chun Kai Ling[1], Jakub Černý[2], Chin Hui Han[3], Garud Iyengar[2], Christian Kroer[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, garud@ieor.columbia.edu,
christian.kroer@columbia.edu

Abstract

Real-world security applications (e.g., cybersecurity) often involve multiple attack paths, each with layers of defenses that an attacker needs to sequentially overcome before a successful attack on the entire system. Each defensive resource changes dynamically in efficacy as the attack unfolds. In this paper, we study the case where attackers are adaptive, potentially switching paths over time in response to these changes with the goal of minimizing the expected time until a successful attack. We formalize this as a min-max game and give examples where adaptive attackers are more powerful than non-adaptive ones. We show that defenses that do not account for adaptivity can perform arbitrarily worse. A connection between the attacker's optimal strategy with the classical theory of multi-armed bandits and the Gittins index is made, yielding a simple gradient based algorithm to solve our proposed min-max game. Experiments on synthetic settings validate our approach.

Code —
https://github.com/lingchunkai/SecGame-LayeredDef

1 Introduction

Cyber attacks are becoming increasingly costly, with global cybercrime projected to reach $23 trillion annually by 2027 (US State Department 2024). A foundational principle in cybersecurity, known as defense-in-depth, advocates deploying multiple layers of safeguards to mitigate risks. Even if one layer is breached, others remain to protect critical assets. This approach is especially important in dynamic environments, where advanced persistent threats (APTs) adapt to evade detection, requiring defenders to continuously revise configurations and defensive measures to maintain operational resilience.

More broadly, many real-world security settings, both digital and physical, involve layered defenses organized across networks of interconnected components. In enterprise IT systems, this may include firewalls, access control policies, and intrusion detection tools. In physical security, airports combine ID checks, metal detectors, and behavioral screening at various stages. Critical infrastructure, such as power grids or water treatment plants, may utilize protection zones with increasing scrutiny. Even online platforms use layered defenses like login rate-limiting, CAPTCHA verification, and two-factor authentication. Across these domains, each control varies in cost and coverage, and defenders must allocate resources strategically. The key challenge is anticipating how attackers, especially adaptive ones, will respond to these layers.

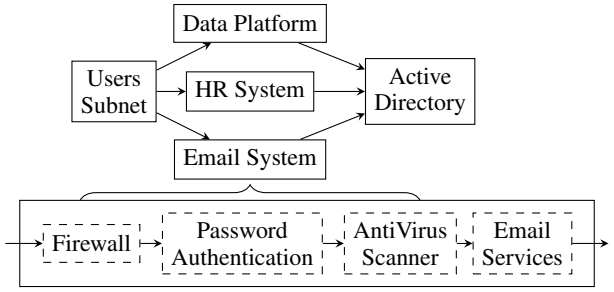

Figure 1: Example of an enterprise IT network with 3 services, one of which is an email system. To access the active directory through the email system, a cyber-attacker has to overcome 3 security controls in sequence: a firewall, password authentication, and an antivirus scanner.

Consider an enterprise IT network as shown in Figure 1, where subsystems such as email, HR, and data platforms rely on a shared authentication service like the Active Directory (AD). Organizations must strategically decide where to deploy their security controls across these interconnected systems, balancing limited resources to maximize protection of critical assets. Different security controls, such as various tiers of firewalls that come with differing costs and protection levels (see the appendix for a comparison), offer distinct trade-offs in effectiveness and expense. For instance, if an attacker repeatedly fails a password check while attempting to access the AD system via the email subsystem (depicted at the bottom), this may trigger stricter authentication protocols, increasing the cost and difficulty of continuing along that path. A non-adaptive attacker would persist with the same attack vector, whereas an adaptive attacker could pivot to an alternative route, such as the HR system. However, many existing models assume attackers are static and non-adaptive, which risks suboptimal defensive resource allocation. In this paper we develop a game-theoretic frame-

work for layered security games and address the following question: *Given that most real-world attackers are adaptive, how should defensive resources be organized*?

Contributions. Our contributions include (i) formalizing this problem mathematically as a two-player zero-sum game, with the optimal defense allocation as the Nash equilibrium, (ii) drawing a 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) devising algorithms for computing the optimal defense policy via first-order methods. 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.

Attack graphs and network interdiction. Attack graphs are often customized by cybersecurity professionals to identify weak points and develop effective incident response strategies (Sheyner et al. 2002; Lippmann, Ingols et al. 2005; Strom et al. 2018). Stochastic network interdiction (Cormican, Morton, and Wood 1998; Wang, Noel, and Jajodia 2006), studies settings where the defender chooses a subset of edges in the network to remove. Some variants also confer commitment advantages upon the defender (Letchford and Vorobeychik 2013; Khouzani, Liu, and Malacaria 2019; Almohri et al. 2015). Nguyen et al. (2017) propose a model allowing defenders to disable vertices, while Durkota et al. (2015a,b) allow "honeypot" hosts which detect and penalize the attacker. Unlike network interdiction, we do not allow removal of entire edges/ vertices, but model realistic controls with internal states that evolve based on the attack.

Multi-armed bandits and indexable policies. Multi-armed bandits (MAB) are a foundational framework tailored at balancing tradeoffs between exploration and exploitation. Fundamental to MAB is the optimality of the Gittins policy, which is to select the arm in the state with the highest Gittins index (Gittins 1979), Extensions include 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 efficiently computing Gittins indices; see, e.g., Chakravorty and Mahajan (2014).

Indexable policies in adversarial settings. In adversarial multi-armed bandits, rewards are influenced by an adversary who alters them every timestep (Auer et al. 1995). While adversarial bandits rarely admit indexable policies, exceptions exist (Xiong and Li 2024). Some formulations study how adversarial perturbations interact with stochastic dynamics. For instance, Scully and Harchol-Balter (2018) study job scheduling with adversarially perturbed job ages and show near optimality of a variant of the Gittins policy. Tan et al. (2018) employ Gittins indices for controlling a set of pursuers, against unknown fixed stochastic evader strategies. Gummadi et al. (2013) perform mean-field analysis of multi-armed bandit games where a population of players interact with a bandit system.

Indexable policies in games. Our work is closest to two-player game-theoretic models where one player commits to a stochastic strategy. Clarkson and Lin (2024) propose a zero-sum hide and seek game where the seeker employs a Gittins index policy to execute an optimal search strategy. Fudenberg and He (2018) study a two-player general-sum Bayesian game where a sender employs a Gittins index policy to select an optimal signal for a receiver.

3 Security Games with Layered Defences

In this paper, an attack graph comprises $n \in \mathbb{Z}_+$ chains, each with access to a single critical component ϕ (e.g., the AD) which is the target of a attacker. The i-th service (e.g., email system) comprises $m_i \in \mathbb{Z}_+$ security controls (e.g., firewalls) layered one after another. The j-th control for the i-th chain is denoted by $e_{i,j}$. We denote the set of all controls by $\mathcal{E}$. A successful attack on the entire system requires the attacker to compromise all of the security controls associated to *at least one chain*. The controls must be compromised sequentially, so $e_{i,j}$ must be compromised before $e_{i,j+1}$. Control $e_{i,j}$ has $q_{i,j}$ distinct states, where each state is indexed by the number of times $k \geq 0$ that a attacker has failed in attempting to circumvent the control; there may be at most $q_{i,j}$ attempts. For example, failed attempts at guessing passwords disable further attempts for a fixed amount of time until a point where access is blocked for that control indefinitely. We call the special case where $m_i = 1$ for all chains a Single Layered Graph (SLG).

An example of a network with 2 chains is shown in Figure 2. A successful attack would require compromising either (i) all 3 controls in the top chain or (ii) both controls in the bottom chain. The target ϕ is identical, so achieving either one of the objectives is sufficient. For instance, a successful attack could occur when the attacker compromises $e_{1,1}, e_{1,2}, e_{2,1}$ and $e_{2,2}$, as this completes the bottom chain.

We denote the *lengths* of each control $\ell \in \mathbb{R}_+^{|\mathcal{E}|}$, such that the time needed by the attacker *for each attempt* to compromise the control e is $\ell(e)$. This ℓ is typically optimized by the defender subject to some budget constraint.

Consider $e = e_{i,j}$, the j-th control of chain i. Each attempt to compromise e incurs a time of $\ell(e)$. We define the probability of success by $p_e(k) \in [0, 1]$, where $k \in \{0, \ldots, q_{i,j}\}$ the number of prior failed attempts on e so far. These p_e are constants that depend on the nature of e and is public knowledge. If the attempt is successful, then e is compromised and (i) if $j = m_i$, i.e., e is the last control in chain i then ϕ is exposed the attacker succeeds in the attack, ending the game or (ii) if $j < m_i$ then the

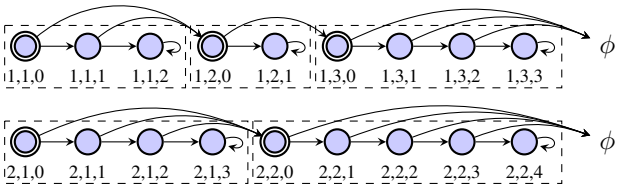

Figure 2: Example of a network with $n = 2$ chains, the first (top) has $m_1 = 3$ controls, the second has $m_2 = 2$. The maximum number of attempts for each control are $q_{1,1} = 2$, $q_{1,2} = 1$, $q_{1,3} = 3$ and $q_{2,1} = 3$, $q_{2,2} = 4$. Controls are represented by dotted boxes containing vertices which represents its possible state; doubled outlines are initial states. Edges leaving a vertex i, j, k are a potential outcome of attacking control j of chain i the $k+1$-th time. Loops represent states where the attacker has been locked out.

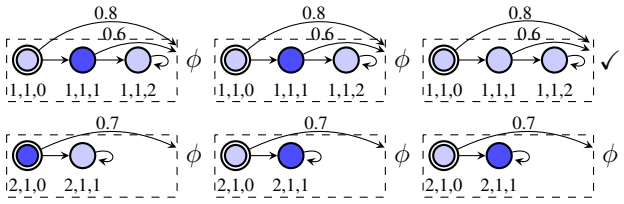

Figure 3: Example of an attack on a SLG with two chains (top and bottom) over for $\zeta = 1, 2, 3$ (left to right), with attacks on chains $1, 2$ and 1 respectively. Probabilities of succeeding are labeled at edges, and current states of each chain are shaded. The attack succeeds at timestep $\zeta = 3$ after compromising chain 1 (indicated by ✓).

next control in the chain $e_{i,j+1}$ is exposed to be compromised (recall that controls in a chain must be compromised in sequence). If the attempt is unsuccessful, then the control changes state and the number of unsuccessful attempts increases. By convention, $p_e(q_{i,j}) = 0$, since these are controls where the attacker has been locked out. There are many possible p_e, each dependent on the type of control in question. For instance, ***constant work*** sets $p_e(k) = 0$ if $k < \kappa_e$ and $p_e(k) = 1$ if $k = \kappa_e$ for some $\kappa_e \in \mathbb{Z}$, ***exponential backoffs*** set $p_e(k) = \hat{\kappa}_e \cdot \kappa_e^k$ for $\hat{\kappa}_e, \kappa_e \in [0, 1]$ and ***beta priors*** have $p_e(k) = \beta_e/(\beta_e + \hat{\beta}_e + k)$ for $\beta_e, \hat{\beta}_e > 0$. Due to space constraints, we defer other examples to the appendix.

Thus, each chain $i \in [n]$ and its constituent controls can be collectively represented as a single Markov chain (Figure 2) with states $\mathcal{M}_i = \{(j, k) | j \in [m_i], k \in \{0, \ldots, q_{i,j}\}\}$, containing states of the form (j, k) where j represents the uncompromised control with the smallest index, and $k \geq 0$ is the number of attempts the control has endured. Terminal states are those $(j, k) \in \mathcal{M}_i$ where $q_{i,j} = k$.

Monotonicity. Sometimes, $p_e(k)$ is *monotonic* in k. For example, constant work is monotonically non-decreasing, while exponential backoff is monotonically non-increasing. The former (latter) means that subverting a control gets easier (harder) to compromise with more attempts. Most real-world controls are monotonic non-increasing, thus we focus our efforts on them for the rest of this paper.

Remark 1. We assumed the attacker may only attempt to compromise $e_{i,j}$ $q_{i,j}$ times before being locked out. We find this modeling assumption reasonable especially when p_e is monotonically decreasing such that past a point $p_e(k) \approx 0$.

3.1 Attacker Policies

An attacker acts sequentially at discrete timestep $\zeta = 1, \ldots,$ each associated with some *time* $t(\zeta)$. Note the distinction between timestep ζ (which *indexes* actions of the attacker) and time $t(\zeta)$ (which is continuous and represents physical time passed at the ζ-th timestep). For every $\zeta \geq 1$ the attacker chooses a chain $i \in [n]$ to compromise; doing so will attack the (unique) control $e_{i,j}$ in chain i that is exposed, assuming that control is not in a terminal state. This incurs time $\ell(e)$, and if successful either ends the game (if $j = m_i$) or exposes control $e_{i,k+1}$ (if $j < m_i$). The total time spent in the first ζ steps inclusive is $t(\zeta)$. The run ends just *after* the final control e_{i,m_i} in some chain i is compromised. Let this random completion time be $t^{\odot}$. Given a discount rate $\lambda > 0$, the attacker then obtains $\exp(-\lambda \cdot t^{\odot})$ utility. Note that $t^{\odot}$ could be ∞, in which case the attacker obtains 0 reward.

Optimal attacker policies. Given some fixed $\ell(e)$, the attacker seeks to end the game such that the time-discounted reward $\mathbb{E}_{t^{\odot}}(\exp(-\lambda \cdot t^{\odot}))$ 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 (Attacker pivoting)**.** Consider the SLG in Figure 3 where $\ell(e) = 1$. In the optimal strategy, the attacker attempts to attack the top chain first due to the higher initial success probability ($0.8 > 0.7$). If it fails, the attacker reevaluates and assesses that the bottom chain now offers a higher success probability of $0.7 > 0.6$. Thus, it pivots to the bottom chain. We give thorough explanations and other examples in the appendix.

Finding a optimal attacker strategy against fixed ℓ can be naively formulated as a Markov Decision Process (MDP) with exponentially sized state space $\mathcal{M}_1 \times \ldots, \mathcal{M}_n$, enabling intractible but theoretically simple solvers like value or policy iteration (Puterman 2014). We denote the value of any policy π by $V_\ell^\pi = \mathbb{E}_{t^{\odot}}[\exp(-\lambda \cdot t^{\odot}) | \ell, \pi]$, i.e., the expected utility under π and ℓ. From standard MDP theory, at least one optimal policy $\pi_\ell^* = \operatorname{argmin}_\pi V_\ell^\pi$ is *deterministic and Markovian*; we denote its value denoted by V_ℓ^*.

3.2 Defender Policies

The defender selects the lengths $\ell \in \mathcal{L} \subseteq \mathbb{R}_+^{|\mathcal{E}|}$. Here $\mathcal{L}$ is some convex set that accounts for defender budget or performance constraints. For example, one may choose a huge $\ell(e)$ when verifying a password, but in turn deteriorate performance for legitimate users. A reasonable $\mathcal{L}$ is the nonempty polyhedron $\{\ell \in \mathbb{R}_+^{|\mathcal{E}|} | A\ell + c \geq d\}$. For this paper, we assume $\mathcal{L} = \Delta_{|\mathcal{E}|} = \{\ell \in \mathbb{R}_+^{|\mathcal{E}|} | 1^T \ell = 1\}$, the $|\mathcal{E}|$-simplex. This simplifying assumption can be easily relaxed.

For fixed ℓ, the attacker's optimal response yields an expected discounted utility of V_ℓ^*. The defender's goal is to

select ℓ to minimize V_ℓ^*, equivalent to the min-max problem

$$\min_{\ell \in \mathcal{L}} \max_{\pi} V_\ell^\pi = \min_{\ell \in \mathcal{L}} \max_{\pi} \mathbb{E}_{t^\odot}[\exp(-\lambda \cdot t^\odot) | \ell, \pi]. \quad (1)$$

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

Remark 2. If we allow the attacker to play a *random* policy (of which there are a finitely many), 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 $\ell \in \mathcal{L}$. We focus on computing optimal defender policies, which yield one side of the corresponding Nash equilibrium. Computing the attacker's randomized equilibrium strategies is of future interest, but it requires more involved algorithms due to the exponential number of deterministic policies.

Heuristic attackers and their suboptimality An alternative to an optimal attacker is one utilizing heuristics. Three heuristics come to mind: unfortunately, as Theorem 3 states, all of them are highly suboptimal and should not be used to inform the optimal defender strategy. The following are some examples. (i) The ***constant*** heuristic attacker sticks to a single chain throughout and never pivots. (ii) A ***greedy*** attacker picks the control 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 $p_e(k) \cdot \exp(-\lambda \cdot \ell(e))$. (iii) The ***greedy-probability*** heuristic chooses the adjacent control most easily crossed, i.e., maximize $p_e(k)$.

Theorem 3. *Constant and greedy heuristics are arbitrarily suboptimal attacker strategies. Furthermore, an optimal ℓ against greedy attackers can be arbitrarily exploitable.*

4 Computing Optimal Defender Policies

Given that V_ℓ^* is convex in ℓ, the simplest approach for finding the optimal ℓ is by treating it as a convex minimization problem over V_ℓ^*. Recall from (1) that this is the minimization over a convex function defined over the simplex $\mathcal{L}$.

4.1 First Order Methods and Regret Matching

Solving (1) can be done via *first order methods* (FOM), which require oracle access to (or an unbiased estimator of) (i) the objective V_ℓ^* and (ii) the subgradient $g(\ell)$, where $g(\ell) \in \partial V_\ell^*$ and ∂V_ℓ^* is the subdifferential. First order methods come in many forms, including projected subgradient descent and generalizations such as mirror descent (Nemirovskij and Yudin 1983), as well as a plethora of online learning algorithms such as online mirror descent, follow-the-regularized leader, and regret matching (Warmuth, Jagota et al. 1997; Gordon 1999; Zinkevich 2003). Note that certain first order methods require smoothness to varying degrees, e.g. the Frank-Wolfe algorithm requires Lipschitz gradients; which is not true in our setting. It is noteworthy that these methods admit noisy/stochastic variants of subgradients, enabling a range of sampling based optimizers. For this paper, we focus on the regret matching algorithm (RM) by Hart and Mas-Colell (2000). The pseudocode is outlined in Algorithm 1. 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 $\mathcal{O}(1/\sqrt{T})$. While RM does not necessarily have the optimal dependence on instance parameters, it has the advantage of being parameter free, unlike the other algorithms we discussed, which requires one to select a suitable learning rate or schedule. Furthermore, it is computationally simple. For these reasons, and its general numerical performance, RM and its variants have been used extensively in game solving (Bowling et al. 2015; Brown and Sandholm 2018, 2019). We focus on the basic RM algorithm, which works well with sampling. However, future work could explore variants such as RM+ (Tammelin et al. 2015) or predictive RM+ (Farina, Kroer, and Sandholm 2021).

Algorithm 1: Regret Matching for optimizing ℓ

$\ell^{(1)} \leftarrow 1 \cdot \frac{1}{|\mathcal{E}|}$ {Lengths}
$y^{(1)} = 0 \cdot \frac{1}{|\mathcal{E}|}$ {Regret}
for $T = 1, \ldots, T_{\text{max}}$ **do**
  $\pi^*_{\ell^{(T)}} \leftarrow$ AttackerOptimalStrategy$(\ell^{(T)})$
  $g^{(T)} \leftarrow$ Subgradient$(\ell^{(T)}, \pi^*_{\ell^{(T)}})$
  $y^{(T+1)} \leftarrow y^{(T)} + 1\langle g^{(T)}, \ell^{(T)} \rangle - g^{(T)}$ {Update regrets}
  if $y^{(T+1)}$ is not all ≤ 0 **then**
    $\ell^{(T+1)} \leftarrow \max\left(y^{(T)}, 0\right) / \sum_e \max\left(y^{(T)}(e), 0\right)$
    {Set $\ell^{(T+1)} \propto$ nonnegative regrets in $y^{(T+1)}$}
  else
    $\ell^{(T+1)} \leftarrow 1 \cdot \frac{1}{|\mathcal{E}|}$ {Set $\ell^{(T+1)}$ to be uniform}
  end if
end for
return $\sum_{T=1}^{T_{\text{max}}} \ell^{(T)} / T_{\text{max}}$ {Return average strategy}

All FOMs will require access to V_ℓ^* and $g(\ell)$ for $\ell \in \mathcal{L}$. This requires us to compute, or at least estimate the best response $\pi_\ell^* = \text{argmax}_\pi V_\ell^\pi$ of the attacker to the defender's allocation. In this paper, we first compute π_ℓ^*, then use it to obtain V_ℓ^* and $g(\ell)$. Performing these three steps *efficiently* is a key technical contribution of the paper.

4.2 Optimal Attacker Policies via Gittins Indices

As it turns out, the optimal policy π_ℓ^* for a fixed ℓ can be computed efficiently by appealing to the *Gittins policy and index*. Essentially, the exponential-sized MDP may be sidestepped by isolating each chain i *independently* of the rest and computing the Gittins index $\alpha(i, j, k)$ for every chain state in $c = (j, k) \in \mathcal{M}_i$. Then, the celebrated theorem of Gittins (Gittins 1979) states that the optimal attacker strategy is to select the chain corresponding to the chain with the highest index, $\text{argmax}_{i \in [n]} \alpha(i, j, k)$, where (j, k) is the current chain state of chain i. We call this the *Gittins policy*. Crucially, $\alpha(i, j, k)$ is solely a function of chain i, independently of other chains. We have provided a more thorough review of the Gittins index in the appendix. In the special case of SLGs, we apply standard results (Gittins, Glazebrook, and Weber 2011) to obtain the special case

Theorem 4. *Suppose G is a single-layered graph. Then we have: (i) if all controls have $p_e(k)$ monotonically non-increasing in k, then the greedy-probability attacker policy (Section 3.2) is optimal, and (ii) if $p_e(k)$ is monotonically non-decreasing, then a constant chain (non-pivoting) attacker policy (Section 3.2) is optimal.*

Remark 5. The literature on Gittins indices is typically presented as the multi-armed bandit problem, where arms correspond to Markov Reward Processes. We use the term *chain* to emphasize that ours is a special case where controls are layered one after the other; this leads to computational gains.

Remark 6. The Gittins policy maximizes discounted *cumulative* rewards, unlike our setting where successful attacks end games. A standard result from queuing theory guarantees equivalent optimal policies in both settings (Gittins, Glazebrook, and Weber 2011). See the appendix for details.

4.3 Efficiently Computing Gittins Indices

For the i-th chain, off-the-shelf methods for computing Gittins indices exactly require asymptotically cubic time in the number of states, i.e., $\mathcal{O}(|\mathcal{M}_i|^3)$. Our main theorem in this section shows that the Gittins indices can be computed much more efficiently by exploiting the structure of each chain.

Theorem 7. *If chain i comprises only monotonically non-increasing controls, $\alpha(i,j,k)$ for all $(j,k) \in \mathcal{M}_i$ can be computed in $\mathcal{O}(m_i \cdot |\mathcal{M}_i|)$ time and $\mathcal{O}(|\mathcal{M}_i|)$ space.*

Theorem 7 implies that our computation is essentially optimal in space and much faster than the $\mathcal{O}(|\mathcal{M}_i^3|)$ obtained by off-the-shelf methods. In practice, m_i's are small relative to n and $q_{i,j}$ (i.e., fewer controls, but large $q_{i,j}$), further favoring our method. The method is based on dynamic programming. Due to space constraints, we will only outline the high level ideas in the main text. We restrict ourselves to monotonically non-increasing controls, this is fairly common in most controls in cybersecurity.

The Gittins index $\alpha(i,j,k)$ for any state is a quantitative measure of how "good" it is to be in the state (i,j,k), while *accounting for future rewards*. Given this intuition, we have the following theorems regarding relative orders of Gittins indices for any fixed chain i.

Theorem 8. *For all $j \in [m_i]$, we have $\alpha(i,j,q_{i,j}) = 0$.*

Theorem 9. *If control j is monotonic non-increasing then for all $k \in \{0,\ldots,q_{i,j}-1\}$, $\alpha(i,j,k) \geq \alpha(i,j,k+1)$.*

Theorem 10. *For all $j \in [m_i - 1]$, $k \in \{0,\ldots,q_{i,j}\}$, we have $\alpha(i,j,k) \leq \alpha(i,j+1,0)$.*

Theorem 8 states that there is no value in spending time on an attack chain that has been "locked out". Theorem 9 says that for monotonic controls, having fewer failed attempts is always desirable. For example, in Figure 2, we have $0 = \alpha(1,1,2) \leq \alpha(1,1,1) \leq \alpha(1,1,0)$. Theorem 10 tells us that it is always preferred to have crossed a control rather than not. This makes sense intuitively, since we are closer to the target ϕ. In our example, we have $\alpha(1,3,0) \geq \alpha(1,2,0) \geq \alpha(1,1,0)$. By applying Theorem 9, we have that $\alpha(1,3,0) \geq \alpha(1,j,k)$ for all $j < 3$.

Fix a chain i. We will compute Gittins indices of its states in descending order, based on the largest-remaining-index method (Varaiya, Walrand, and Buyukkoc 1985; Chakravorty and Mahajan 2014). Their key idea is to iteratively add states corresponding to the next highest Gittins index into a *continuation set* $\mathcal{C}_i$, gradually populating it until the Gittins indices of all states in $\mathcal{M}_i$ are computed. For ease of exposition, we will assume there are no ties. The continuation set at iteration z is given by $\mathcal{C}_i^{(z)}$, with $\mathcal{C}_i^{(0)} = \{\}$. Note that z refers to the iteration number in our algorithm that computes Gittins indices and is separate from t and ζ. Each iteration, we consider a *Markov Reward Process* (MRP) that follows the transition probabilities p_e, but terminates when entering a non-continuation state. The reward upon reaching ϕ is 1, discounted by the time taken (possibly ∞). The candidate Gittins index for state (i,j,k) is the ratio of the *expected discounted reward* $d_{i,j,k}^{(z)}$ to *expected discounted time* $b_{i,j,k}^{(z)}$ when the MRP begins at state (i,j,k). The state (i,j^*,k^*) with the highest Gittins index is assigned its Gittins index $\alpha(i,j^*,k^*)$ based on this ratio and added to $\mathcal{C}_i^{(z)}$ to obtain $\mathcal{C}_i^{(z+1)}$. The process repeats until all nonterminal states are in the continuation set.

This loop runs for $\mathcal{O}(|\mathcal{M}_i|)$ iterations, each performed in $\mathcal{O}(m_i)$ time by exploiting Theorem 9, which permits us to compare at most m_i candidate states as opposed to all states not in $\mathcal{C}_i^{(z)}$. The candidate set is $\{\mathfrak{C}_{i,j}^{(z)} | j \in [m_i]\}$ where $\mathfrak{C}_{i,j}^{(z)}$ is the smallest k' such that (i,j,k') does not belong to the continuation set at iteration z. The next trick is to utilize dynamic programming to compute the desired ratio for each state in the candidate set in $\mathcal{O}(1)$ time each iteration; this is done by maintaining the expected discounted reward and time locally for each of the m_i controls as well as some efficient bookkeeping. Refer to the appendix for details.

4.4 Efficiently Computing Subgradients

Once π_ℓ^* is found, the second half of Algorithm 1 finds subgradients $g(\ell)$. We give two approaches for computing $g(\ell)$. The first uses simulated trajectories via π_ℓ^* to empirically obtain noisy/approximate subgradients. This method is extremely straightforward with π_ℓ^*, but is far slower and entails a tradeoff between the accuracy of $g(\ell)$ and sampling time.

The second method and our main contribution is to once again exploit the structure of our attack chains to compute *exact* subgradients in polynomial (in n, m and q) time. While many orders of magnitude faster than trajectory sampling, it is more challenging to describe and implement. Thus, we limit ourselves to a proof sketch in the main text.

Suppose have already computed all the Gittins indices $\alpha(i,j,k)$. We rank $\alpha(i,j,k)$ in decreasing order, breaking ties arbitrarily (e.g., by lexicographic order). We denote by $\sigma(w)$ the (i,j,k) tuple with the w-th highest $\alpha(\sigma(w))$, i.e., $\alpha(\sigma(i)) \geq \alpha(\sigma(i+1))$ for all $i \geq 1$ and $\alpha(i) = \alpha(j) \implies i = j$. We think of attack trajectories as commencing in *stages*. Consider the example shown in Figure 2. The plot in Figure 4 shows the Gittins index of the chain with the highest Gittins index during part of a single trajectory (note that this is across all chains) and the running minimum of the

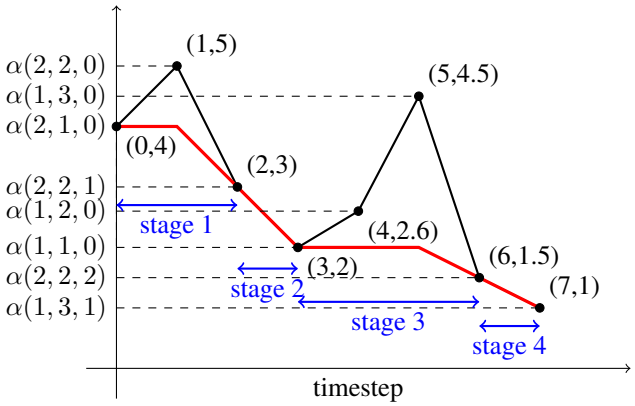

Figure 4: Gittins index evolution over timesteps. Black plots show the highest Gittins index, red plots the cumulative minimum. Coordinates show timestep ζ and α's respectively.

Gittins index of the chain attacked. Every time attacking a chain fails and brings the system to a new minimum Gittins index, the stage ends and a new one begins. Every trajectory can be broken down into a finite number of such stages.

Consider a different interpretation where instead of starting a new stage when the highest Gittins index is higher than the prevailing minimum, we segment the entire attack into $w_{\max} = \sum_{i=1}^{n} \sum_{j}^{m_i} q_{i,j} + 1$ such stages. In the w-th stage, all states in each chain which have the w-th highest Gittins indices, i.e., $\alpha(i,j,k) \geq \alpha(\sigma(w))$, are in the continuation set. The Markov chain is ran based on the Gittins policy until (i) some chain reaches the target, or (ii) *every* chain is in some stopping state. In case (i), the process ends, a discounted reward is collected and recorded (weighted by its probability). In case (ii), we will increase w by 1 and add $\sigma(w) = (i_w, j_w, k_w)$ into the continuation state. If the current state in chain i_w is not (i_w, j_w, k_w), then we increase w by 1 and continue the process until the state of some chain is the one added to the continuation set. When the current state in chain i_w is (i_w, j_w, k_w), then we freeze all other chains and keep pulling chain i_w (causing transitions to new states) until i_w either (a) reaches the target or (b) enters a new state (i_w, j'_w, k'_w) not in the continuation set (note that j'_w and k'_w are random). At this point, every chain is at a stopping state and we can consider increasing w again to "trigger" additional changes, perhaps in another chain. This artificial process can be applied for *any* trajectory and always yields $w_{\max}$ stages, though some could last 0 time steps.

Our key insight is the following. Suppose we are at the end of stage w and no chain has reached ϕ yet, i.e., the attack is still ongoing. All of the chains are in some stopping (non-continuation) state, i.e., their Gittins index is less than $\alpha(\sigma(w))$. Crucially, while the possible number of *joint-states* is exponentially large in n, the distribution at the end of stage w *follows a product distribution*. Since none of the chains have reached ϕ, this is equivalent to running each of the chains *independently* until they enter a stopping state, This fact helps us to decompose expected utilities (and gradients) up to stage w into product forms as well.

n	λ	GI/GI	GI*/GI	GR/GR	GR/GI	$\frac{\text{GI/GI}}{\text{GR/GI}}$	$\frac{\text{GR/GR}}{\text{GR/GI}}$	Tm-S	Tm-D
3	0.5	4.0e-1	4.0e-1	3.4e-1	4.9e-1	0.82	0.69	585	8.16
3	1.0	2.6e-1	2.7e-1	2.1e-1	3.9e-1	0.67	0.54	586	8.55
3	2.0	1.5e-1	1.5e-1	1.4e-1	2.3e-1	0.65	0.61	587	8.61
3	5.0	3.4e-2	3.4e-2	2.9e-2	4.9e-2	0.69	0.59	571	8.39
3	10.0	4.7e-3	4.6e-3	4.0e-3	8.1e-3	0.58	0.49	598	5.55
5	0.5	5.9e-1	5.9e-1	4.8e-1	7.1e-1	0.83	0.68	720	10.6
5	1.0	4.4e-1	4.4e-1	2.9e-1	6.1e-1	0.72	0.48	726	10.9
5	2.0	2.7e-1	2.7e-1	1.5e-1	5.1e-1	0.53	0.29	721	11.1
5	5.0	9.1e-2	9.1e-2	8.1e-2	2.0e-1	0.45	0.40	692	11.8
5	10.0	2.3e-2	2.3e-2	3.4e-2	5.7e-2	0.40	0.60	721	9.12
8	0.5	7.2e-1	7.2e-1	5.4e-1	8.7e-1	0.83	0.62	896	18.4
8	1.0	5.7e-1	5.7e-1	3.4e-1	8.2e-1	0.70	0.41	913	18.8
8	2.0	3.9e-1	3.9e-1	1.6e-1	7.0e-1	0.56	0.23	910	18.7
8	5.0	1.7e-1	1.6e-1	1.1e-1	2.9e-1	0.59	0.38	872	19.7
8	10.0	6.0e-2	5.9e-2	7.7e-2	1.6e-1	0.38	0.48	861	19.1

Table 1: Results for controls utilizing exponential backoffs. The X/Y column report average rewards under different defender and attacker models, $\frac{\text{GI/GI}}{\text{GR/GI}}$ and $\frac{\text{GR/GR}}{\text{GR/GI}}$ quantify the suboptimality and misevaluation of the defender, while Tm-S and Tm-D is the wall-clock time in seconds used to optimize ℓ (against an optimal attacker playing the Gittins policy) under stochastic and deterministic gradients. More details are found in Section 5.

This decomposition turns out to be extremely useful, since it allows us to analyze iterates over w while maintaining and reasoning about *marginal* distributions over chains when performing dynamic programming over w from 1 to $w_{\max}$. The precise details of the updates are significantly trickier and deferred to the appendix. The overall time complexity for computing V_ℓ^* is $\mathcal{O}(m_{\max} \cdot w_{\max})$, and for subgradients g, $\mathcal{O}(n \cdot m_{\max} \cdot w_{\max})$, where $m_{\max} = \max_{i \in [n]} m_i$.

5 Experimental Evaluation

For simplicity, we consider chain graphs of n parallel chains each with $m_1 = m_2 = \cdots = m_n = 3$ monotonic non-decreasing controls. We experiment with two types of controls: those using exponential backoffs and beta priors. In both cases, we lock the attacker out after 4 failed trials, i.e., $q_{i,j} = 4$ and $p_e(k) = 0$ for all $k \geq 4$. We randomly sample the parameters required for exponential backoffs and beta priors $\hat{\kappa}_e, \hat{\kappa}_e, \beta_e, \hat{\beta}_e$ uniformly in $[0.2, 0.8]$.

For each case, we fix the class of controls and vary n and λ. We benchmarked our approach (abbreviated GI) against a greedy baseline (abbreviated GR), where we ran Algorithm 1 but where instead of computing V_ℓ^* and its subgradient exactly, we instead set π to be a simple greedy heuristic (see Section 3.2) and optimized ℓ via stochastic subgradients of V_ℓ^π. Since policies based on the greedy heuristic are not optimal, the argument following (1) does not hold and hence V_ℓ^π may not be convex in ℓ, i.e., we are not necessarily finding the optimal ℓ against the greedy heuristic. Nonetheless, this is a good baseline for comparison.

For each test, we ran 5 experiments over different parameters and report averages. The results and runtime for exponential backoffs with varying n and λ are reported in Table 1, the rest are deferred to the appendix. The tables contain:

- Columns "X/Y" reports average rewards for the attacker. The first term X refers to the attacker model that the defender optimized ℓ against (possibly dependent on ℓ, e.g., the policy induced by the Gittins index). The second term Y refers to the opponent faced at test time. For example the column under [GR/GI] means that we obtained ℓ that was trained against an opponent adopting the greedy policy but tested it against the Gittins (i.e., optimal) attacker. For both evaluation and optimization, we report the performance when using exact subgradient methods. [GI*/GI] is a special case where we ran RM for 20000 iterations rather than 2000; we used exact subgradient methods here.
- $\frac{\text{GI/GI}}{\text{GR/GI}}$ is the (multiplicative) suboptimality of the defender. It tells us how much worse it gets when the defender assumes the attacker is greedy, but the actual attacker is optimal. It is strictly lower than 1 and highlights the importance of accounting for the attacker's adaptive behavior.
- $\frac{\text{GR/GR}}{\text{GR/GI}}$ is the misevaluation that can occur, i.e., the ratio of the payoff the defender thinks they will get against the greedy attacker to what they actually get when the attacker turns out to be optimal. Low values suggest that the defender is misled into overestimating its own defenses.

Technical details and experimental setup. The experiments were implemented in Python and run on a Macbook Pro M1 with 16GB of RAM (memory was not a limiting factor). Other than [GI*/GI], we ran Algorithm 1 for 2000 iterations each. We performed checks to (i) ensure the correct computation of Gittins index (by comparing solutions with the full MDP formulation solved via value iteration (Puterman 2014)) , (ii) verify that empirical averages of V_ℓ^* and those computed using our exact algorithm agree, and (iii) verify that our gradients match Euler's forward method when used to approximate subgradients of V_ℓ^* in ℓ.

General trends and sanity checks. For a fixed n, as λ increases, both [GI/GI] and [GR/GR] decrease. This is expected, as the reward at the end is more highly discounted with larger λ. Conversely, if we fix λ and vary n, we see that [GI/GI] and [GR/GR] increase. Again, this is not surprising. As n increases, there are more attack paths to pivot to, forcing the defender to spread its defensive resources thinly. Next, we observe that both $\frac{\text{GI/GI}}{\text{GR/GI}}$ and $\frac{\text{GR/GR}}{\text{GR/GI}}$ were consistently less than 1 for all experiments. This is just as expected. The former is an illustration of the fact the Gittins policy is indeed optimal, while the latter implies a defender who optimizes (and tests against) a greedy attacker will be significantly overconfident when facing off against an optimal adaptive attacker. This is expected: the optimal attacker strategy may substantially differ from the greedy heuristic.

Exploitability of heuristic-attacker models. In all cases, $\frac{\text{GI/GI}}{\text{GR/GI}}$ is lower than 1. This implies optimizing against an optimal attacker is more effective than the greedy heuristic when facing an optimal attacker. This supports our claim that a defender should optimize based on the true min-max formulation rather than rely on heuristic attackers. This difference is also quite significant, with the optimal attacker obtaining almost 2-times the amount of reward when comparing a heuristic-based to a minimax-optimal defender.

Computational costs. We also see that Tm-D is 1 or almost 2 orders of magnitude smaller than Tm-S. This is a reflection of how important exact gradient methods are. In all instances, we see that [GI*/GI] did not perform better than [GI/GI]. This suggests that just 2000 iterations is enough to converge. We also see that for similar sized networks, beta controls (see table in appendix) can be solved several times faster than the case with exponential backoff controls (Table 1). We believe this is because with exponential backoff controls, the probabilities of success drop drastically with each failure. This means that longer trajectories are more likely, which makes sampling based approaches relatively slowly. This hypothesis is supported by the observation that rewards (in the "X/Y" columns) are generally higher for beta controls. Interestingly, the reverse trend seems true for exact subgradient methods, on average beta controls are a little slower. We cannot explain this trend but this appears to reflect the fact that the exact subgradient method is not directly dependent on average lengths of trajectories.

Scalability. In terms of computational costs, our approach appears to scale gracefully with n. For example, we tested our exact subgradient approach on $n = 100$ and did not encounter any issues, both in terms of memory consumption and time. Use of Gittins indices proves to be crucial, since the naive MDP formulation has roughly 15^{100} states. We note that with very high values of m, e.g., $m = 500$, the expected time for a successful attack can be very large since long sequences of controls have to be compromised. This causes numerical issues in practice, since $V_\ell^* = \mathbb{E}_{t^\odot}[\exp(-\lambda \cdot t^\odot)|\ell, \pi^*] \approx 0$ since $t^\odot$ is large when m_i is large. We hypothesize that this problem may be alleviated by working instead in log space.

Robustness to model misspecification. In practice, one does not necessarily know p_e, the exact probabilities of failure and success as a function of the number of previous failed attempts at circumventing a control and defensive allocation ℓ. Estimating these probabilities is arguably one of the most difficult tasks in modeling cybersecurity, and is either obtained by expert knowledge or by learning from examples. In both cases, such probabilities may suffer form non-trivial errors or noise. To examine the impact on our method, we conducted some abalation studies; details and results are given in the appendix. Overall, we observe that the error to small model misspecification is minor (for instance, these values are much smaller for $\frac{\text{GI/GI}}{\text{GR/GI}}$), suggesting that our approach is somewhat robust. More sophisticated sensitivity analysis is left for future work.

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.

Acknowledgements

This research was supported by the Office of Naval Research award N00014-23-1-2374. Christian Kroer was additionally supported by the Office of Naval Research award N00014-22-1-2530, and the National Science Foundation awards IIS-2147361 and IIS-2238960. Chun Kai Ling was supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2025) and by the National University of Singapore, under the Start-Up Grant Scheme. This project was started when Chun Kai Ling was a postdoctoral research scientist at Columbia University.

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.

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.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science*, 347(6218): 145–149.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374): 418–424.

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890.

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.

Farina, G.; Kroer, C.; and Sandholm, T. 2021. Faster game solving via predictive blackwell approachability: Connecting regret matching and mirror descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5363–5371.

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.

Gordon, G. J. 1999. Regret bounds for prediction problems. In *Proceedings of the twelfth annual conference on Computational learning theory*, 29–40.

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.

Nemirovskij, A. S.; and Yudin, D. B. 1983. Problem complexity and method efficiency in optimization.

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.

Tammelin, O.; Burch, N.; Johanson, M.; and Bowling, M. 2015. Solving heads-up limit texas hold'em. In *Twenty-fourth international joint conference on artificial intelligence*.

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.

Varaiya, P.; Walrand, J.; and Buyukkoc, C. 1985. Extensions of the multiarmed bandit problem: The discounted case. *IEEE Transactions on Automatic Control*, 30(5): 426–439.

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

Warmuth, M. K.; Jagota, A. K.; et al. 1997. Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence. In *Electronic proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*, volume 326. Citeseer.

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*.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, 928–936.