

Intrusion Prevention through Optimal Stopping

Kim Hammar †‡ and Rolf Stadler†‡
† Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden
‡ KTH Center for Cyber Defense and Information Security, Sweden
Email: {kimham, stadler}@kth.se
October 30, 2021

***Abstract*—We study automated intrusion prevention using reinforcement learning. Following a novel approach, we formulate the problem of intrusion prevention as an (optimal) multiple stopping problem. This formulation gives us insight into the structure of optimal policies, which we show to have threshold properties. For most practical cases, it is not feasible to obtain an optimal defender policy using dynamic programming. We therefore develop a reinforcement learning approach to approximate an optimal policy. Our method for learning and validating policies includes two systems: a simulation system where defender policies are incrementally learned and an emulation system where statistics are produced that drive simulation runs and where learned policies are evaluated. We show that our approach can produce effective defender policies for a practical IT infrastructure of limited size. Inspection of the learned policies confirms that they exhibit threshold properties.**

***Index Terms*—Network security, automation, optimal stopping, reinforcement learning, Markov Decision Process, MDP, POMDP**

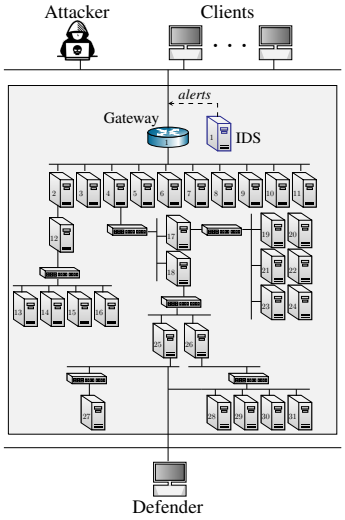

Fig. 1: The IT infrastructure and the actors in the use case.

I. Introduction

An organization's security strategy has traditionally been defined, implemented, and updated by domain experts [1]. Although this approach can provide basic security for an organization's communication and computing infrastructure, a growing concern is that infrastructure update cycles become shorter and attacks increase in sophistication [2], [3]. Consequently, the security requirements become increasingly difficult to meet. To address this challenge, significant efforts have started to automate security processes and functions. Examples of this research include: automated creation of threat models [4]; computation of defender policies using dynamic programming and control theory [5], [6]; computation of exploits and corresponding defenses through evolutionary methods [7]; identification of infrastructure vulnerabilities through attack simulations and threat intelligence [8], [9]; computation of defender policies through game-theoretic methods [10], [11]; and use of machine learning techniques to estimate model parameters and policies [12], [13].

In this paper, we present a novel approach to automatically learn defender policies. We apply this approach to an *intrusion prevention* use case. Here, we use the term "intrusion prevention" as suggested in the literature, e.g. in [1]. It means that a defender prevents an attacker from reaching its goal, rather than preventing it from accessing any part of the infrastructure.

Our use case involves the IT infrastructure of an organization (see Fig. 1). The operator of this infrastructure, which we call the defender, takes measures to protect it against a possible attacker while, at the same time, providing a service to a client population. The infrastructure includes a public gateway through which the clients access the service and which also is open to a possible attacker. The attacker decides when to start an intrusion and then executes a sequence of actions that includes reconnaissance and exploits. Conversely, the defender aims at preventing intrusions and maintaining service to its clients. It monitors the infrastructure and can defend it by taking defensive actions, which can prevent a possible attacker but also incur costs. What makes the task of the defender difficult is the fact that it lacks direct knowledge of the attacker's actions and must infer that an intrusion occurs from monitoring data.

We study the use case within the framework of discrete-time dynamical systems. Specifically, we formulate the problem of finding an optimal defender policy as an *(optimal) multiple stopping problem*. In this formulation, the defender can take a finite number of *stops*. Each stop is associated with a defensive action and the objective is to decide the optimal times when to stop. This approach gives us insight into the structure of optimal defender policies through the theory of dynamic programming and optimal stopping [14]. In particular, we show that an optimal *threshold policy* exists that can be efficiently computed and implemented. Other related research on automated intrusion prevention, in contrast, do not demonstrate structural properties of optimal policies [12],

[15], [16], [17], [18], [19], [20], [21], [22], [23].

Although the optimal stopping problem frequently is used to formulate problems in the fields of finance and communication systems [24], [25], [26], [14], [27], [28], [29], [30], [31], [13], to the best of our knowledge, formulating intrusion prevention as a multiple stopping problem is a novel approach.

Since the defender can access only a set of infrastructure metrics and does not directly observe the attacker, we use a Partially Observed Markov Decision Process (POMDP) to model the multiple stopping problem. To obtain the defender policy, we simulate a long series of POMDP episodes where the defender continuously updates its policy based on outcomes of previous episodes. For updating the policy, we use a state-of-the-art reinforcement learning algorithm. This approach enables us to approximate an optimal policy despite uncertainty about the attacker's behavior.

Our method for learning and validating policies includes building two systems (see Fig. 2). First, we develop an *emulation system* where key functional components of the target infrastructure are replicated. In this system, we run attack scenarios and defender responses. These runs produce system metrics and logs that we use to estimate empirical distributions of infrastructure metrics, which are needed to simulate POMDP episodes. Second, we develop a *simulation system* where POMDP episodes are executed and policies are incrementally learned. Finally, the policies are extracted and evaluated in the emulation system, and can also be implemented in the target infrastructure. In short, the emulation system is used to provide the statistics needed to simulate the POMDP and to evaluate policies, whereas the simulation system is used to learn policies.

We make two contributions with this paper. First, we formulate intrusion prevention as a problem of multiple stopping. This novel formulation allows us a) to derive properties of an optimal defender policy using results from dynamic programming and optimal stopping; and b) to approximate an optimal policy for a non-trivial infrastructure configuration. Second, we present a reinforcement learning approach to obtain policies in an emulated infrastructure. With this approach, we narrow the gap between the evaluation environment and a scenario playing out in a real system. We also address a limitation of many related works, which rely on simulations solely to evaluate policies [12], [7], [15], [17], [18], [32], [16].

This paper is a significant extension of our work published at CNSM 2021 [13]. First, we generalize the formal model of attacker and defender interactions described in [13] to allow multiple defender-actions. Moreover, we study the structure of the optimal solution in the generalized framework and derive important properties of optimal policies. Specifically, we show how threshold properties of an optimal policy generalize from the single stopping case to the multiple stopping case. Second, in [13] we studied the use case through simulation only, in this paper we add an emulation system to collect system metrics and to validate the simulation results.

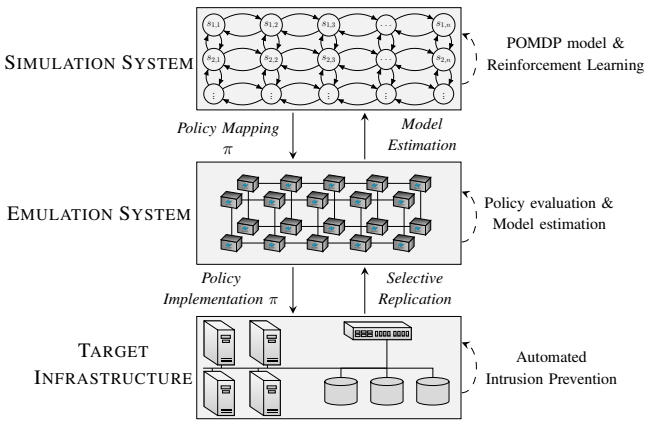

Fig. 2: Our approach for finding and evaluating intrusion prevention policies.

II. The Intrusion Prevention Use Case

We consider an intrusion prevention use case that involves the IT infrastructure of an organization. The operator of this infrastructure, which we call the defender, takes measures to protect it against an attacker while, at the same time, providing a service to a client population (Fig. 1). The infrastructure includes a set of servers that run the service and an intrusion detection system (IDS) that logs events in real-time. Clients access the service through a public gateway, which also is open to the attacker.

We assume that the attacker intrudes into the infrastructure through the gateway, performs reconnaissance, and exploits found vulnerabilities, while the defender continuously monitors the infrastructure through accessing and analyzing IDS statistics and login attempts at the servers. The defender can take a fixed number of defensive actions to prevent the attacker, each of which has a cost. A defensive action is for example to reset user accounts in the infrastructure, which will recover accounts compromised by the attacker. It is assumed that the defender takes the defensive actions in a pre-determined order, starting with the action that has the lowest cost. The final action that the defender can take is to block all external access to the gateway. As a consequence of this action, the service as well as any ongoing intrusion are disrupted.

In deciding when to take defensive actions, the defender has two objectives: (*i*) maintain service to its clients; and (*ii*), keep a possible attacker out of the infrastructure at a low cost. The optimal policy for the defender is to monitor the infrastructure and maintain service until the moment when the attacker enters through the gateway, at which time the attacker must be prevented with the minimal cost by taking defensive actions. The challenge for the defender is to identify the precise time when this moment occurs and select the number of defensive actions that prevents the attacker at the lowest cost.

In this work, we model the attacker as an agent that starts the intrusion at a random point in time and then takes a predefined sequence of actions, which includes reconnaissance to explore the infrastructure and exploits to compromise servers.

We study the use case from the defender's perspective. The evolution of the system state and the actions by the defender are modeled with a discrete-time Partially Observed Markov Decision Process (POMDP). The reward function of

this process encodes the benefit of maintaining service, the cost of defensive actions, and the loss of being intruded. Finding an optimal defender policy thus means maximizing the expected reward.

III. Theoretical Background

This section covers the preliminaries on Markov decision processes, reinforcement learning, and optimal stopping.

A. Markov Decision Processes

A Markov Decision Process (MDP) models the control of a discrete-time dynamical system and is defined by a seven-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{s,s'}, \mathcal{R}^a_{s,s'}, \gamma, \rho_1, T \rangle$ [33], [34]. $\mathcal{S}$ denotes the set of states and $\mathcal{A}$ denotes the set of actions. $\mathcal{P}^a_{s,s'}$ refers to the probability of transitioning from state s to state s' when taking action a (Eq. 1), which has the Markov property $\mathbb{P}\left[s_{t+1}|s_t\right] = \mathbb{P}\left[s_{t+1}|s_1, \ldots, s_t\right]$. Similarly, $\mathcal{R}^a_{s,s'} \in \mathbb{R}$ is the expected reward when taking action a and transitioning from state s to state s' (Eq. 2). Finally, $\gamma \in (0, 1]$ is the discount factor, $\rho_1 : \mathcal{S} \to [0, 1]$ is the initial state distribution, and T is the time horizon.

$$\mathcal{P}^{a_t}_{s_t,s_{t+1}} = \mathbb{P}\left[s_{t+1}|s_t, a_t\right] \tag{1}$$

$$\mathcal{R}^{a_t}_{s_t,s_{t+1}} = \mathbb{E}\left[r_{t+1}|a_t, s_t, s_{t+1}\right] \tag{2}$$

The system evolves in discrete time-steps from $t = 1$ to $t = T$, which constitute one *episode* of the system.

A Partially Observed Markov Decision Process (POMDP) is an extension of an MDP [35], [36]. In contrast to an MDP, in a POMDP the states are not directly observable. A POMDP is defined by a nine-tuple $\mathcal{M}_\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{s,s'}, \mathcal{R}^a_{s,s'}, \gamma, \rho_1, T, \mathcal{O}, \mathcal{Z} \rangle$. The first seven elements define an MDP. $\mathcal{O}$ denotes the set of observations and $\mathcal{Z}(o', s', a) = \mathbb{P}[o'|s', a]$ is the observation function, where $o' \in \mathcal{O}$, $s' \in \mathcal{S}$, and $a \in \mathcal{A}$.

The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s_t) = \mathbb{P}[s_t|h_t]$ and $\mathcal{B}$ is the unit $(|\mathcal{S}| - 1)$-simplex [37]. It is a sufficient statistic of the state s_t based on the history h_t of the initial state distribution, the actions, and the observations: $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$. By defining the state at time t to be the belief state b_t, a POMDP can be formulated as a continuous-state MDP: $\mathcal{M} = \langle \mathcal{B}, \mathcal{A}, \mathcal{P}^a_{b,b'}, \mathcal{R}^a_{b,b'}, \gamma, \rho_1, T \rangle$.

The belief state can be computed recursively as follows [36]:

$$b_{t+1}(s_{t+1}) = C\mathcal{Z}(o_{t+1}, s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} \mathcal{P}^{a_t}_{s_t s_{t+1}} b_t(s_t) \tag{3}$$

where $C = 1/\sum_{s_{t+1} \in S} \mathcal{Z}(o_{t+1}, s_{t+1}, a_t) \sum_{s_t \in S} \mathcal{P}^{a_t}_{s_t,s_{t+1}} b_t(s_t)$ is a normalizing factor independent of s_{t+1} to make b_{t+1} sum to 1.

B. The Reinforcement Learning Problem

Reinforcement learning deals with the problem of choosing a sequence of actions for a sequentially observed state variable to maximize a reward function [38], [39]. It can be modeled with an MDP if the state space is observable, or with a POMDP if the state space is not fully observable.

In the context of an MDP, a *policy* is defined as a function that maps states to actions $\pi : \mathcal{S} \to \mathcal{A}$. In the case of a POMDP, a policy is defined as a function that maps histories to actions $\pi : \mathcal{H} \to \mathcal{A}$, or, alternatively, as a function that maps belief states to actions: $\pi : \mathcal{B} \to \mathcal{A}$. In both cases, an optimal policy π^* is a policy that maximizes the expected discounted cumulative reward over the time horizon T:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} r_t \right] \tag{4}$$

where Π is the policy space, γ is the discount factor, r_t is the reward at time t, and $\mathbb{E}_\pi$ denotes the expectation under π.

The Bellman equations relate any optimal policy π^* to the two value functions $V^* : \mathcal{S} \to \mathbb{R}$ and $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces of an MDP [40]:

$$V^*(s_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}\big[r_{t+1} + \gamma V^*(s_{t+1}) | s_t, a_t\big] \tag{5}$$

$$Q^*(s_t, a_t) = \mathbb{E}\big[r_{t+1} + \gamma V^*(s_{t+1}) | s_t, a_t\big] \tag{6}$$

$$\pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} Q^*(s_t, a_t) \tag{7}$$

Here, $V^*(s_t)$ and $Q^*(s_t, a_t)$ denote the expected cumulative discounted reward under π^* for each state and state-action pair, respectively. In the case of a POMDP, the Bellman equations contain b_t instead of s_t. Solving the Bellman equations (Eqs. 5-6) means computing the value functions from which an optimal policy can be obtained (Eq. 7).

Two principal methods are used for finding an optimal policy in a MDP or POMDP: dynamic programming and reinforcement learning.

First, the dynamic programming method (e.g. value iteration [41], [34]) assumes complete knowledge of the seven-tuple MDP or the nine-tuple POMDP and obtains an optimal policy by solving the Bellman equations iteratively (Eq. 7), with polynomial time-complexity per iteration for MDPs and PSPACE-complete time-complexity for POMDPs [42].

Second, the reinforcement learning method computes or approximates an optimal policy without complete knowledge of the transition probabilities or observation probabilities of the MDP or POMDP. Three classes of reinforcement learning algorithms exist: *value-based algorithms*, which approximate solutions to the Bellman equations (e.g. Q-learning [43]); *policy-based algorithms*, which directly search through policy space using gradient-based methods (e.g. Proximal Policy Optimization (PPO) [44]); and *model-based algorithms*, which learn the transition or observation probabilities of the MDP or POMDP (e.g. Dyna-Q [39]). The three algorithm types can also be combined, e.g. through *actor-critic* algorithms, which are mixtures of value-based and policy-based algorithms [39]. In contrast to dynamic programming algorithms, reinforcement learning algorithms generally have no guarantees to converge to an optimal policy except for the tabular case [45], [46].

C. The Markovian Optimal Stopping Problem

Optimal stopping is a classical problem domain with a well-developed theory [47], [48], [49], [50], [41], [51], [52], [34]. Example use cases for this theory include: asset selling [41],

change detection [30], machine replacement [36], hypothesis testing [47], gambling [50], selling decisions [28], queue management [29], advertisement scheduling [14], and the secretary problem [34], [26].

Many variants of the optimal stopping problem have been studied. For example, discrete-time and continuous-time problems, finite horizon and infinite horizon problems, problems with fully observed state spaces and partially observed state spaces, Markovian and non-Markovian problems, and single-stop and multi-stop problems. Consequently, different solution methods for these variants have been developed. The most commonly used methods are the *martingale approach* [49], [50] and the *Markovian approach* [48], [41], [34], [51], [52].

In this paper, we investigate the multiple stopping problem with a *maximum* of L stops, a finite time horizon T, discrete-time progression, and the Markov property. We use the Markovian solution approach and model the problem as a POMDP, where the system state evolves as a discrete-time Markov process $(s_t)_{t=1}^{T}$ that is partially observed.

At each time-step t of the decision process, two actions are available: "stop" (S) and "continue" (C). The *stop* action with l stops remaining yields a reward $\mathcal{R}^{S}_{s,s',l}$ and if only one of the L stops remain, the process terminates. In the case of the *continue* action, the decision process proceeds to the next time-step and yields a reward $\mathcal{R}^{C}_{s,s',l}$.

The *stopping time* with l stops remaining is a random variable τ_l that is dependent on $s_1, \ldots, s_{\tau_l}$ and independent of $s_{\tau_l+1}, \ldots s_T$ [49]:

$$\tau_l = \min\{t : t > \tau_{l+1}, a_t = S\}, \quad l \in 1, .., L, \ \tau_{L+1} = 0 \quad (8)$$

If the lth stop is not used, $\tau_l = T$.

The objective is to find a stopping policy $\pi^*(s_t) \rightarrow \{S, C\}$ that maximizes the expected discounted cumulative reward of the stopping times $\tau_L, \tau_{L-1}, \ldots, \tau_1$:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\Bigg[\sum_{t=1}^{\tau_L - 1} \gamma^{t-1}\mathcal{R}^{C}_{s_t,s_{t+1},L} + \gamma^{\tau_L - 1}\mathcal{R}^{S}_{s_{\tau_L},s_{\tau_L+1},L} + \ldots + \sum_{t=\tau_2+1}^{\tau_1 - 1} \gamma^{t-1}\mathcal{R}^{C}_{s_t,s_{t+1},1} + \gamma^{\tau_1 - 1}\mathcal{R}^{S}_{s_{\tau_1},s_{\tau_1+1},1}\Bigg] \quad (9)$$

Due to the Markov property, any policy that satisfies Eq. 9 also satisfies the Bellman equation (Eq. 7), which in the partially observed case is:

$$\pi^*_l(b) = \arg\max_{\{S,C\}} \quad (10)$$
$$\Bigg[\underbrace{\mathbb{E}_l\left[\mathcal{R}^{S}_{b,b^o_S,l} + \gamma V^*_{l-1}(b^o_S)\right]}_{\text{stop } (S)}, \underbrace{\mathbb{E}_l\left[\mathcal{R}^{C}_{b,b^o_C,l} + \gamma V^*_l(b^o_C)\right]}_{\text{continue } (C)}\Bigg] \forall b \in \mathcal{B}$$

where π_l is the stopping policy with l stops remaining, $\mathbb{E}_l$ denotes the expectation with l stops remaining, b is the belief state, V^*_l is the value function with l stops remaining, b^o_S and b^o_C can be computed using Eq. 3, and $\mathcal{R}^{a}_{b,b',l}$ is the expected reward of action $a \in \{S, C\}$ in belief state b when transitioning to belief state b' with l stops remaining.

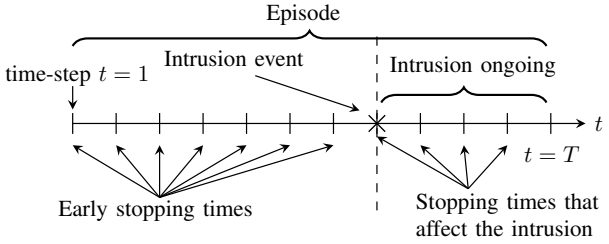

Fig. 3: Optimal multiple stopping formulation of intrusion prevention; the horizontal axis represents time; T is the time horizon; the episode length is $T - 1$; the dashed line shows the intrusion start time; the optimal policy is to prevent the attacker at the time of intrusion with the fewest stops.

IV. Formalizing The Intrusion Prevention Use Case and Our Reinforcement Learning Approach

We first present a formal model of the use case described in Section II and then we introduce our solution method. Specifically, we first define a POMDP model of the intrusion prevention use case. Then, we describe our reinforcement learning approach to approximate an optimal defender policy. Lastly, we apply the theory of dynamic programming and optimal stopping to obtain structural results of an optimal policy.

A. A POMDP Model of the Intrusion Prevention Use Case

We formulate the intrusion prevention use case as a multiple stopping problem where an intrusion starts at a random time and each stop is associated with a defensive action (Fig. 3). We model this problem as a POMDP.

1) Actions $\mathcal{A}$: The defender has two actions: "stop" (S) and "continue" (C). The action space is thus $\mathcal{A} = \{S, C\}$. We encode S with 1 and C with 0 to simplify the formal description below.

A *maximum* of L stop actions can be performed during an episode. The number of stops that the defender must execute to prevent an intrusion is $L - l^A$. Both $L \geq 1$ and $l^A \in \{0, 1, \ldots, L\}$ are predefined parameters of our use case, where l^A depends on the attacker.

2) States $\mathcal{S}$ and Initial State Distribution ρ_1: The system state $s_t \in \{0, 1\}$ is zero if no intrusion is occurring and $s_t = 1$ if an intrusion is ongoing. In the initial state, no intrusion is occurring and $s_1 = 0$. Hence, the initial state distribution is the degenerate distribution $\rho_1(0) = 1$. Further, we introduce a terminal state $\emptyset \in \mathcal{S}$, which is reached after the defender takes the final stop action or after the attacker completes an intrusion (see below).

3) Transition Probabilities $\mathcal{P}^{a}_{s,s'}$: We model the start of an intrusion by a Bernoulli process $(Q_t)_{t=1}^{T}$, where $Q_t \sim Ber(p = 0.2)$ is a Bernoulli random variable. The time of the first occurrence of $Q_t = 1$ is the start time of the intrusion I_t, which thus is geometrically distributed, i.e., $I_t \sim Ge(p = 0.2)$ (Fig. 4).

An intrusion lasts for a finite number of time-steps, denoted by a constant T_{int}, which is a predefined parameter of our use case and depends on the attacker. This implies that an intrusion has ended if $t \geq I_t + T_{int}$.

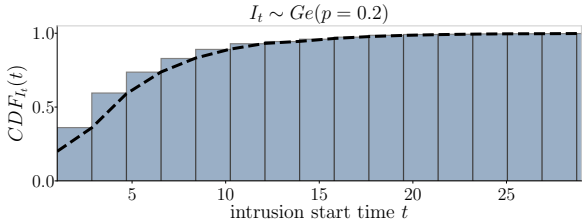

Fig. 4: The cumulative distribution function (CDF) of the intrusion start time I_t.

We define the transition probabilities $\mathcal{P}^{a_t}_{s_t,s_{t+1}} = \mathbb{P}_{l_t}[s_{t+1}|s_t, a_t]$ as follows, where $\mathbb{P}_{l_t}$ denotes the probability with l_t stops remaining:

$$\mathbb{P}_1[\emptyset|\cdot, 1] = \mathbb{P}_{l_t}[\emptyset|\emptyset, \cdot] = 1 \tag{11}$$

$$\mathbb{P}_{l_t}[\emptyset|\cdot, \cdot] = 1 \quad \text{if } t \geq I_t + T_{int} \tag{12}$$

$$\mathbb{P}_{l_t}[0|0, a_t] = 1 - p\mathbb{1}_{l_t - a_t > l^A} \tag{13}$$

$$\mathbb{P}_{l_t}[1|0, a_t] = p \quad \text{if } l_t - a_t > l^A \tag{14}$$

$$\mathbb{P}_{l_t}[1|1, a_t] = 1 \quad \text{if } l_t - a_t > l^A \tag{15}$$

$$\mathbb{P}_{l_t}[0|1, 1] = 1 \quad \text{if } l_t - 1 = l^A \tag{16}$$

All other transitions have probability 0.

Eqs. 11-12 define the transition probabilities to the terminal state $\emptyset$. The terminal state is reached in two cases: when the *final* stop action S is taken (Eq. 11), and when the intrusion ends (Eq. 12). If Eqs. 11-12 are not applicable, i.e., if the system does not reach the terminal state, then the transition probabilities when taking action S ($a = 1$) or C ($a = 0$) are defined by Eqs. 13-16.

Eq. 13 captures the case where no intrusion occurs and $s_{t+1} = s_t = 0$ ($\mathbb{1}$ is the indicator function); Eq. 14 specifies the case when the intrusion starts where $s_t = 0$ and $s_{t+1} = 1$; Eq. 15 describes the case where an intrusion is in progress and $s_{t+1} = s_t = 1$; and Eq. 16 captures the case where an ongoing intrusion is prevented by taking the stop action S ($a = 1$).

With this definition of the transition probabilities, the evolution of the system can be understood using the state transition diagram in Fig. 5.

4) Time Horizon $T_\emptyset$: The time horizon $T_\emptyset$ is a random variable that indicates the time t when the terminal state $\emptyset$ is reached. Since both $\mathbb{E}[I_t]$ and $\mathbb{E}[T_{int}]$ are finite, it follows that $\mathbb{E}_\pi[T_\emptyset] < \infty$ for any policy π. (Remark: it is also possible to define $T = \infty$ and let $\emptyset$ be an infinitely absorbing state.)

5) Observations $\mathcal{O}$: The defender has a partial view of the system and if $s_t \neq \emptyset$ it observes $o_t = (l_t, \Delta x_t, \Delta y_t, \Delta z_t)$. Here, $l_t \in \{1, 2, \ldots, L\}$ is the number of stops remaining and Δx_t, Δy_t, and Δz_t denote the number of severe IDS alerts, warning IDS alerts, and login attempts generated during time-step t, respectively. If the system is in the terminal state, the defender observes $o_{T_\emptyset} = \emptyset$.

6) Observation Function $\mathcal{Z}(o', s', a)$: We assume that the number of IDS alerts and login attempts generated during one time-step are random variables $X \sim f_X$, $Y \sim f_Y$, $Z \sim f_Z$ that depend on the state and the time. Consequently, the probability that Δx severe alerts, Δy warning alerts, and Δz login attempts occur during time-step t can be expressed as $f_{XYZ}(\Delta x, \Delta y, \Delta z|s_t, I_t, t)$.

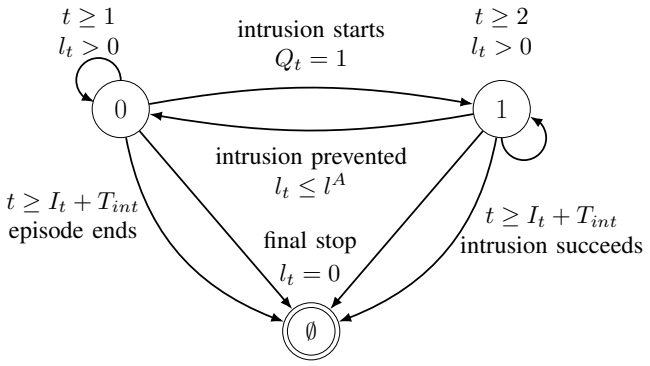

Fig. 5: State transition diagram of the POMDP: each circle represents a state; an arrow represents a state transition; a label indicates the event that triggers the state transition; an episode starts in state $s_1 = 0$ with $l_1 = L$.

We define $\mathcal{Z}(o', s', a) = \mathbb{P}[o'|s', a]$ as follows:

$$\mathcal{Z}\big((l_t, \Delta x, \Delta y, \Delta z), s, \cdot\big) = f_{XYZ}(\Delta x, \Delta y, \Delta z|s_t, I_t, t) \tag{17}$$

$$\mathcal{Z}\big(\emptyset, \emptyset, \cdot\big) = 1 \tag{18}$$

7) Reward Function $\mathcal{R}^a_{s,s',l}$: The objective of the intrusion prevention use case is to maintain service on the infrastructure while, at the same time, preventing a possible intrusion at minimal cost. Therefore, we define the reward function to give the maximal reward if the defender maintains service until the intrusion starts and then takes the fewest number of stop actions to prevent the intrusion (i.e. $L - l^A$ stops).

The reward per time-step $\mathcal{R}^{a_t}_{s_t,l_t} = \mathcal{R}^{a_t}_{s_t,s_{t+1},l_t}$ is parameterized by the reward that the defender receives for stopping an intrusion ($R_{st} = 200$), the reward for maintaining service ($R_{sla} = 10$), and the loss of being intruded ($R_{int} = -100$):

$$\mathcal{R}_\emptyset = 0 \tag{19}$$

$$\mathcal{R}^S_{s_t,l_t} = -\frac{100}{2^{l_t - 1}} \tag{20}$$

$$\mathcal{R}^C_{s_t,l_t} = \frac{R_{sla}}{2^{L - l_t}} \tag{21}$$

$$\mathcal{R}^{T_\emptyset} = \mathbb{1}_{I_t < \tau_{l^A} < T_\emptyset} R_{st} + \mathbb{1}_{I_t < \tau_{l^A}}(\tau_{l^A} - I_t - 1) R_{int} \tag{22}$$

Eq. 19 states that the reward in the terminal state is zero. Eq. 20 indicates that each stop incurs an increasing cost and Eq. 21 states that the defender receives a positive reward for maintaining service, which is decaying. Lastly, Eq. 22 states that the defender receives a reward when transitioning to the terminal state if the intrusion was prevented, i.e., if $\mathbb{1}_{I_t < \tau_{l^A} < T_\emptyset}$. (Note that there is no gain of additional stop actions after the intrusion has been prevented.) It also receives a loss proportional to the length of the intrusion: $\tau_{l^A} - I_t - 1$. Here, τ_{l^A} is the time when the intrusion was prevented ($\tau_{l^A} = T_\emptyset$ if the intrusion was not prevented) and I_t is the start time of the intrusion ($I_t = T_\emptyset$ if the intrusion did not start).

8) Policy Space Π_θ and Objective Function J: We consider the space of policies Π_θ where a policy $\pi_\theta \in \Pi_\theta$ is parameterized by a vector $\theta \in \mathbb{R}^d$. An optimal policy $\pi^*_\theta \in \Pi_\theta$ maximizes the expected discounted cumulative reward over the horizon $T_\emptyset$:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\emptyset - 1} \gamma^{t-1}\mathcal{R}^{a_t}_{s_t} + \gamma^{T_\emptyset - 1}\mathcal{R}^{T_\emptyset}\right] \tag{23}$$

$$\pi^*_\theta = \arg\max_{\pi_\theta \in \Pi_\theta} J(\theta) \tag{24}$$

We set the discount factor to be $\gamma = 1$. (The objective in Eq. 23 is bounded when $\gamma = 1$ since $\mathbb{E}_{\pi_\theta}[T_\emptyset]$ is finite for any policy $\pi_\theta \in \Pi_\theta$.)

Eq. 23 defines an optimization problem which reflects the objective of our use case. In the following section, we describe our approach for solving this problem using reinforcement learning.

B. Our Reinforcement Learning Approach

In our use case, the defender does not know the full POMDP model. For instance, the transition probabilities, the number of stops required to prevent an intrusion, and the observation function are unknown to the defender. Therefore, we cannot use dynamic programming to solve the optimization problem in Eq. 24 without further assumptions. Instead, we use a reinforcement learning approach to approximate an optimal policy without having access to the full POMDP model. Specifically, we use the state-of-the-art reinforcement learning algorithm PPO [44] to learn a policy $\pi_\theta : \mathcal{H} \to \mathcal{A}$ that maximizes the objective function in Eq. 23.

The input to the policy π_θ at time $t \in \{1, 2, \ldots, T_\emptyset - 1\}$ is the summarized history of observations $\hat{h}_t = (l_t, x_t, y_t, z_t, t)$, where x_t, y_t, z_t are the accumulated counters of the observations $o_i = (l_i, \Delta x_i, \Delta y_i, \Delta z_i)$ for $i = 1, \ldots, t$. That is, $x_t = \sum_{i=1}^{t} \Delta x_i$, $y_t = \sum_{i=1}^{t} \Delta y_i$, and $z_t = \sum_{i=1}^{t} \Delta z_i$.

PPO implements the *policy gradient* method and uses stochastic gradient ascent with the following gradient [44]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\underbrace{\nabla_\theta \log \pi_\theta(a_t|\hat{h}_t)}_{\text{actor}} \underbrace{A^{\pi_\theta}(\hat{h}_t, a_t)}_{\text{critic}}\right] \tag{25}$$

where $A^{\pi_\theta}(\hat{h}_t, a_t) = Q^{\pi_\theta}(\hat{h}_t, a_t) - V^{\pi_\theta}(\hat{h}_t)$ is the so-called *advantage function* [53]. We implement π_θ with a deep neural network that takes as input the summarized history $\hat{h}_t$ and produces as output a discrete conditional probability distribution $\pi_\theta(a_t|\hat{h}_t)$ that is computed with the softmax function. The neural network structure of π_θ follows an actor-critic architecture and computes a second output (the critic) that estimates the value function $V^{\pi_\theta}_\theta(\hat{h}_t)$, which in turn allows to estimate $A^{\pi_\theta}(\hat{h}_t, a_t)$ in Eq. 25 using the *generalized advantage estimator* $\hat{A}^{\pi_\theta}_{GAE}$ [53].

The neural network is trained through simulation of the POMDP as follows. First, we run a given number of episodes. We then use the episode outcomes and trajectories to: (*i*) estimate the prediction error of the critic: $\mathbb{E}_{\pi_\theta}\left[\left(V^{\pi_\theta}_\theta(\hat{h}_t) - V^{\pi_\theta}(\hat{h}_t)\right)^2\right]$; and (*ii*) estimate the expectation of the gradient in Eq. 25. Then, we use the critic error, the estimated gradient, and the PPO algorithm [44] with the Adam optimizer [54] to update the value function (the critic) and the policy (the actor). This process of running episodes and updating the value function and the policy continues until the policy has sufficiently converged.

We made the code of the reinforcement learning algorithm available [55] and give the hyperparameters in Appendix C.

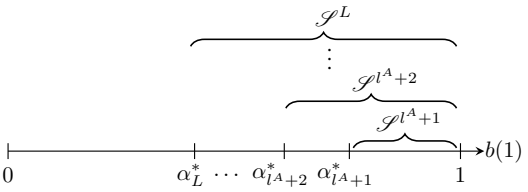

Fig. 6: Illustration of Theorem 1: there exist $L - l^A$ thresholds $\alpha^*_{l^A+1} \geq \alpha^*_{l^A+2} \ldots, \geq \alpha^*_L \in \mathcal{B}$ and an optimal threshold policy π^*_l that satisfies Eq. 26-27.

C. Threshold Properties of an Optimal Policy

A policy that solves the multiple stopping problem is a solution to Eqs. 23-24. We know from the theory of dynamic programming that this policy satisfies the Bellman equation formulated in terms of the belief state (Eq. 10) [36], [37].

The belief state b_t is defined as $b_t(s_t) = \mathbb{P}[s_t|h_t]$ (see Section III-A). As the state space of the POMDP is $\mathcal{S} = \{0, 1, \emptyset\}$ (see Fig. 5), b_t is a probability vector with two components: $b_t(0) = \mathbb{P}[s_t = 0|h_t]$ and $b_t(1) = \mathbb{P}[s_t = 1|h_t]$, where $t = 1, \ldots T_\emptyset - 1$. Further, since $b_t(0) = 1 - b_t(1)$, the belief state is determined by $b_t(1)$ and the *belief space* $\mathcal{B}$ can be described by the unit interval, i.e. $\mathcal{B} = [0, 1]$.

We partition $\mathcal{B}$ into two non-overlapping sets—the stopping set $\mathscr{S}^l = \{b(1) \in [0, 1] : \pi^*_l\big(b(1)\big) = S\}$, which contains the belief states where it is optimal to *stop*, and the continuation set $\mathscr{C}^l = \{b(1) \in [0, 1] : \pi^*_l\big(b(1)\big) = C\}$, which contains the belief states where it is optimal to *continue*. The number of stops remaining, l, ranges from 1 to L.

Applying the theory developed in [36], [27], [14], we obtain the following structural result for an optimal policy.

Theorem 1. *Given the POMDP in Section IV-A, let L denote the maximum number of stop actions, $L - l^A$ the minimum number of stops to prevent an intrusion, f_{XYZ} the distribution of the observations, $b(1)$ the belief state, $\mathscr{S}^l$ the stopping set, and $\mathscr{C}^l$ the continuation set. The following holds:*

(A) $\mathscr{S}^{l-1} \subseteq \mathscr{S}^l$ for $l = 2, \ldots L$.

(B) If $L - l^A = 1$, there exists $\alpha^* \in [0, 1]$ and an optimal policy π^*_l that satisfies:

$$\pi^*_L(b(1)) = S \iff b(1) \geq \alpha^* \tag{26}$$

(C) If $L - l^A \geq 1$ and $f_{XYZ|s}$ is totally positive of order 2 (i.e., TP2), there exist $L - l^A$ values $\alpha^*_{l^A+1} \geq \alpha^*_{l^A+2} \geq \ldots \geq \alpha^*_L \in [0, 1]$ and an optimal policy π^*_l that satisfies:

$$\pi^*_l(b(1)) = S \iff b(1) \geq \alpha^*_l, l \in l^A + 1, \ldots, L \tag{27}$$

Proof. See Appendix A. □

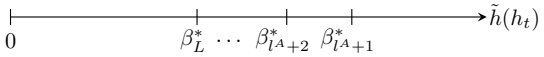

Fig. 7: Illustration of Corollary 1: there exist $L-l^A$ thresholds $\beta^*_{l^A+1} \geq \beta^*_{l^A+2} \ldots, \geq \beta^*_L \in \mathbb{R}$ and an optimal threshold policy π^*_l that satisfies Eq. 28.

Theorem 1.A states that the stopping sets have a nested structure. This means that if it is optimal to stop when $b(1)$ has a certain value while $l-1$ stops remain, then it is also optimal to stop for the same value when l or more stops remain.

Theorem 1.B and Theorem 1.C state that there exist an optimal policy with threshold properties (see Fig. 6). If $L - l^A \geq 1$, an additional condition applies: the probability matrix of $f_{XYZ|s}$ must be TP2 (all second order minors must be non-negative) [36, Definition 10.2.1, pp. 223][56]. This condition is satisfied for example if $f_{XYZ|s}$ is stochastically monotone in s.

Since the statements in Theorem 1 are expressed in terms of belief, the theorem cannot be applied to learn defender policies (see Section IV-B). However, if we assume a relationship between observations and beliefs, the theorem lets us deduce the existence of an optimal threshold policy that can be learned through reinforcement learning.

We assume a real-valued function $\tilde{h} : h_t \mapsto \mathbb{R}$ on the history of observations h_t. (In our case, $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$.) We further assume a monotonically increasing function $g : \tilde{h}(h_t) \mapsto b_t(1)$, which maps $\tilde{h}(h_t)$ onto $\mathcal{B}$. Then, the following corollary holds.

Corollary 1. *If $f_{XYZ|s}$ has property TP2 and $g : \tilde{h}(h_t) \mapsto b_t(1)$ is monotonically increasing. Then, there exist $L - l^A$ values $\beta^*_{l^A+1} \geq \beta^*_{l^A+2} \geq \ldots \geq \beta^*_L \in \mathbb{R}$ and an optimal policy π^*_l that satisfies:*

$$\pi^*_l(h_t) = S \iff \tilde{h}(h_t) \geq \beta^*_l, \quad l \in l^A + 1, \ldots, L \tag{28}$$

Proof. See Appendix B □

Corollary 1 implies the existence of an optimal threshold policy that is expressed in terms of $\tilde{h}(h_t)$ instead of $b(1)$ (see Fig. 7). Note that $\tilde{h}(h_t)$ can be computed from observations of the defender. In Section VI we examine whether the policies learned with our reinforcement learning approach have these threshold properties when $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$.

Knowing that there exists optimal policies with special structure has two benefits. First, insight into the structure of optimal policies often leads to a concise formulation and efficient implementation of the policies [34], [11]. This is obvious in the case of threshold policies. Second, many times the complexity of computing or learning an optimal policy can be reduced by exploiting structural properties [36], [29]. For example, Corollary 1 suggests that the dimensionality of the parameter vector θ of the policy π_θ can be decreased to L.

V. Emulating the Target Infrastructure to Instantiate the Simulation and to Evaluate the Learned Policies

To simulate episodes of the POMDP we must know the distributions of alerts and login attempts conditioned on the system state. We estimate these distributions using measurements from the emulation system shown in Fig. 2. Moreover, to evaluate the performance of policies learned in the simulation system, we run episodes in the emulation system by executing actions of an emulated attacker and having the defender execute stop actions at times given by the learned policies.

Client	*Functions*	*Application servers*
1	HTTP, SSH, SNMP, ICMP	N_2, N_3, N_{10}, N_{12}
2	IRC, PostgreSQL, SNMP	$N_{31}, N_{13}, N_{14}, N_{15}, N_{16}$
3	FTP, DNS, Telnet	N_{10}, N_{22}, N_4

TABLE 1: Emulated client population; each client interacts with application servers using a set of network functions.

l_t	*Action*	*Command in the Emulation*
3	Reset users	`deluser <username>`
2	Blacklist IPs	`iptables -A INPUT -s <ip> -j DROP`
1	Block gateway	`iptables -A INPUT -i eth0 -j DROP`

TABLE 2: Defender stop commands in the emulation.

A. Emulating the Target Infrastructure

The emulation system executes on a cluster of machines that runs a virtualization layer provided by Docker [57] containers and virtual links. The configuration of the emulated infrastructure is given by the topology in Fig. 1 and the configuration in Appendix D. The system emulates the clients, the attacker, the defender, as well as 31 physical components of the target infrastructure (e.g application servers and the gateway). Physical entities are emulated and software functions are executed in Docker containers of the emulation system. The software functions replicate important components of the target infrastructure, such as, web servers, databases, and an IDS.

The *client population* is emulated by three client processes that interact with the application servers through functions and protocols listed in Table 1.

The emulation evolves in time-steps. During each step, the defender and the attacker can perform one action each. The *defender* executes either a continue action or a stop action. The continue action has no effect on the progression of the emulation but the stop action has. We have implemented $L = 3$ stop actions which are listed in Table 2. The *first* stop resets all user accounts and recovers accounts compromised by the attacker. The *second* and *third* stops update the firewall configuration of the gateway. Specifically, the *second* stop adds a rule to the firewall that drops incoming traffic from IP addresses that have been flagged by the IDS and the *third* stop blocks *all* incoming traffic.

We have implemented three *attacker profiles*: NoviceAttacker, ExperiencedAttacker, and ExpertAttacker, all of which execute the sequence of actions listed in Table 3, where I_t is the start time of the intrusion. The actions consist of reconnaissance commands and exploits. During each time-step, one action is executed. The three attackers differ in

Time-steps t	NOVICEATTACKER	EXPERIENCEDATTACKER	EXPERTATTACKER
1-$I_t \sim Ge(0.2)$	(Intrusion has not started)	(Intrusion has not started)	(Intrusion has not started)
$I_t + 1$-$I_t + 6$	RECON$_1$, brute-force attacks (SSH,Telnet,FTP) on N_2, N_4, N_{10}, login(N_2, N_4, N_{10}), backdoor(N_2, N_4, N_{10})	RECON$_2$, CVE-2017-7494 exploit on N_4, brute-force attack (SSH) on N_2, login(N_2, N_4), backdoor(N_2, N_4), RECON$_2$	RECON$_3$, CVE-2017-7494 exploit on N_4, login(N_4), backdoor(N_4)
$I_t + 7$-$I_t + 10$	RECON$_1$, CVE-2014-6271 on N_{17}, login(N_{17}), backdoor(N_{17})	CVE-2014-6271 on N_{17}, login(N_{17}) backdoor(N_{17}), SSH brute-force attack on N_{12}	RECON$_3$, SQL Injection on N_{18} login(N_{18}), backdoor(N_{18}),
$I_t + 11$-$I_t + 14$	SSH brute-force attack on N_{12}, login(N_{12}) CVE-2010-0426 exploit on N_{12}, RECON$_1$	login(N_{12}), CVE-2010-0426 exploit on N_{12}, RECON$_2$, SQL Injection on N_{18}	RECON$_3$, CVE-2015-1427 on N_{25} login(N_{25}), backdoor(N_{25}),
$I_t + 15$-$I_t + 16$		login(N_{18}), backdoor(N_{18})	RECON$_3$, CVE-2017-7494 exploit on N_{27} login(N_{27}), backdoor(N_{27})
$I_t + 17$-$I_t + 19$		RECON$_2$, CVE-2015-1427 on N_{25}, login(N_{25})	

TABLE 3: Attacker actions in the emulation.

Attacker	T_{int}	$L - l^A$	Reconnaissance
NOVICEATTACKER	14	1	TCP/UDP scan
EXPERIENCEDATTACKER	19	2	ICMP ping scan
EXPERTATTACKER	16	3	ICMP ping scan

TABLE 4: Intrusion lengths T_{int}, number of stops required to prevent the attacker $L - l^A$, and reconnaissance commands of the attacker profiles.

the reconnaissance command that they use, the length of the intrusion T_{int}, and the number of stops $L - l^A$ required to prevent the attack (see Table 4).

NOVICEATTACKER uses brute-force attacks to exploit password vulnerabilities (e.g. SSH dictionary attacks) and uses a TCP/UDP port scan for reconnaissance. The attack is prevented if the defender takes a stop action and resets the user accounts.

EXPERIENCEDATTACKER uses a ping scan for reconnaissance and performs both brute-force attacks and more sophisticated attacks, such as a command injection attack (e.g. CVE-2014-6271). The attack is prevented if the defender takes two stop actions and blacklists IP addresses that have been flagged by the IDS in addition to resetting the user accounts.

Lastly, EXPERTATTACKER only targets vulnerabilities that can be exploited without brute-force methods and thus generates less network traffic, for example remote execution vulnerabilities, such as, CVE-2017-7494. The attacker uses a ping scan for reconnaissance like EXPERIENCEDATTACKER. The attack is prevented if the defender executes three stop actions and blocks the gateway.

Since the ping-scan generates fewer IDS alerts than the TCP/UDP port scan, the reconnaissance actions of EXPERIENCEDATTACKER and EXPERTATTACKER are harder to detect than those of NOVICEATTACKER.

B. Estimating the Distributions of Alerts and Login Attempts

In this section, we describe how we collect data from the emulation system and estimate the distributions of alerts and login attempts.

1) At the end of every time-step, the emulation system collects the metrics Δx, Δy, Δz, which contain the alerts and login attempts that occurred during the time-step. The metrics are obtained by parsing the outputs of the commands in Table 5. For the evaluation reported in this paper we collected measurements from 21000 time-steps of 30 seconds each.

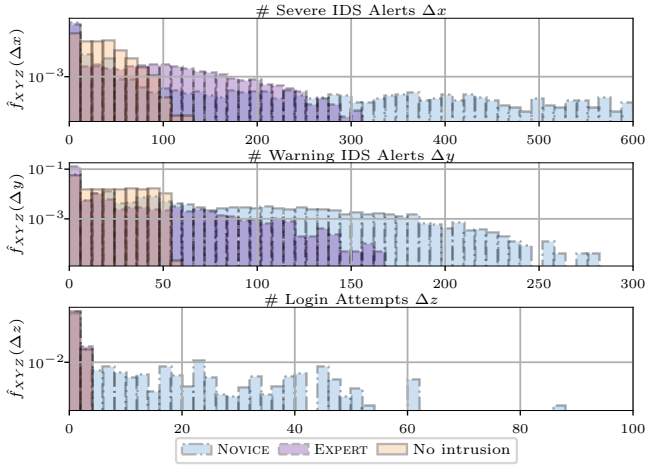

Fig. 8: Empirical distributions of severe IDS alerts Δx (top row), warning IDS alerts Δy (middle row), and login attempts Δz (bottom row) generated during time-steps of intrusions by different attackers as well as during time-steps when no intrusion occurs.

Metric	*Command in the Emulation*
Login attempts	`cat /var/log/auth.log`
IDS Alerts	`cat /var/snort/alert.csv`

TABLE 5: Commands to extract observation data.

2) From the collected measurements, we compute the empirical distribution $\hat{f}_{XYZ}$ as estimate of the corresponding distribution f_{XYZ} in the target infrastructure. For each triple (s_t, I_t, t), we obtain the conditional distribution $\hat{f}_{XYZ|s_t,I_t,t}$.

Fig. 8 shows some of the empirical distributions. The distributions related to EXPERIENCEDATTACKER are omitted for better readability. The estimated distributions from EXPERTATTACKER and EXPERIENCEDATTACKER mostly overlap with the distributions obtained when no intrusion occurs. However, a clear difference between the distributions obtained during an intrusion of NOVICEATTACKER and the distributions when no intrusion occurs can be observed.

C. Simulating an Episode of the POMDP

During a simulation of the POMDP, the system state evolves according to the dynamics described in Section IV, and the

observations evolve according to the estimated distribution $\hat{f}_{XYZ}$. In the initial state, no intrusion occurs. During an episode, an intrusion normally occurs at a random start time. It is also possible that the defender performs $L - l^A$ stops before the intrusion would start, in which case no intrusion starts.

A simulated episode evolves as follows. The episode starts in state $s_1 = 0$ and $l_1 = L$. During each time-step, the simulation system samples an action from the defender policy $a_t \sim \pi_\theta(\cdot|\hat{h}_t)$. If the action is stop ($a_t = 1$) and $l_t = 1$, the episode ends. Otherwise, the number of remaining stop actions is decremented: $l_{t+1} = l_t - a_t$. Further, if $s_t = 1$ (i.e. an intrusion occurs), the intrusion is prevented if $l_{t+1} \leq l^A$, otherwise the system executes an attacker action following Table 3. It then samples $\Delta x_t, \Delta y_t, \Delta z_t$ from the empirical distribution $\hat{f}_{XYZ}$, increments the counters x_t, y_t, z_t, and updates $\hat{h}_{t+1} = (l_{t+1}, x_{t+1}, y_{t+1}, z_{t+1}, t+1)$ using the procedure described in Section IV-B. (The activities of the clients are not simulated but are captured by $\hat{f}_{XYZ}$.) The simulation then computes the reward of the defender using the reward function given in Section IV. The sequence of time-steps continues until either the defender performs the final stop or the intrusion completes, after which the episode ends.

D. Emulating an Episode of the POMDP

Just like a simulated episode, an emulated episode starts with the same initial conditions, evolves in discrete time-steps, and experiences an intrusion event at a random time. However, an episode in the emulation system differs from an episode in the simulation system in the following ways. First, attacker and defender actions in the emulation system include computing and networking functions with real side-effects in the emulation environment (see Table 2 and Table 3). Further, the defender observations in the emulation system are not sampled but are obtained through reading log files and metrics of the emulated infrastructure using the commands in Table 5. Lastly, the emulated client population performs requests to the emulated application servers just like on a real infrastructure (see Section V-A). Due to these differences, running an episode in the emulation system takes much longer time than running a similar episode in the simulation system.

VI. Learning Intrusion Prevention Policies for the Target Infrastructure

Our approach for finding effective defender policies includes (1) extensive simulation of POMDP episodes in the simulation system to learn the policies; and (2), evaluation of the learned policies through running POMDP episodes in the emulation system. This section describes our evaluation results.

The environment for training policies and running simulations is a Tesla P100 GPU. The hyperparameters for the training algorithm are listed in Appendix C. The emulated infrastructure is deployed on a server with a 24-core Intel Xeon Gold 2.10GHz CPU and 768 GB RAM. We have made available the code of our simulation system, as well as the measurement traces used to estimate the observation distributions of the POMDP, which can be used by others to extend and validate our results [55].

A. Evaluation Process

We train three defender policies against NoviceAttacker, ExperiencedAttacker and ExpertAttacker until convergence. For each attacker we run $750'000$ training episodes to estimate an optimal defender policy using the method described in Section IV-B. After each batch of $10'000$ training episodes we evaluate the current defender policy. To evaluate the policy, we run evaluation episodes and compute various performance metrics. Specifically, we run 500 evaluation episodes in the simulation system and 5 evaluation episodes in the emulation system.

The $750'000$ training episodes and the evaluation described above constitute one *training run*. We run five training runs with different random seeds. A single training run takes about 48 hours of processing time on a P100 GPU to perform the simulations and the policy-training, as well as around 12 hours for evaluating the policies in the emulation system.

We compare the learned policies with two baselines. The first baseline is obtained from a policy that prescribes the stop action when the first IDS alert occurs, i.e., when $(\Delta x + \Delta y) \geq 1$ for the first time. The second baseline is obtained from an ideal policy which presumes knowledge of the exact intrusion time.

B. Learning Policies for a Single Stop

We first examine the scenario where the defender takes a single stop action (i.e. $L = 1$ and $l^A = 0$ for all attackers), which involves blocking all outside access to the gateway.

Fig. 9 shows the performance of the learned policies against the three attacker types. The red curves represent the results from the simulation system and the blue curves show the results from the emulation system. The orange curves give the performance of the baseline policy that mandates a stop action at the first IDS alert. The dashed black curves give the performance of the baseline policy that assumes knowledge of the exact intrusion time.

An analysis of the graphs in Fig. 9 leads us to the following conclusions. We observe that the learning curves converge quickly to constant mean values for all attackers and across all investigated performance metrics. After approximately $100'000$ training episodes the curves have sufficiently converged. From this we conclude that the learned policies have converged as well.

Second, we observe that the converged values of the learning curves are close to the dashed black curves, which give an upper bound to any optimal policy. In addition, we see that the empirical probability of preventing an intrusion of the learned policies is close to 1 (middle column of Fig. 9) and that the empirical probability of stopping before the intrusion starts is close to 0 (second rightmost column of Fig. 9). This suggests that the learned policies are close to optimal. We also observe that all learned policies do significantly better than the baseline that stops on the first IDS alert (leftmost column in Fig. 9).

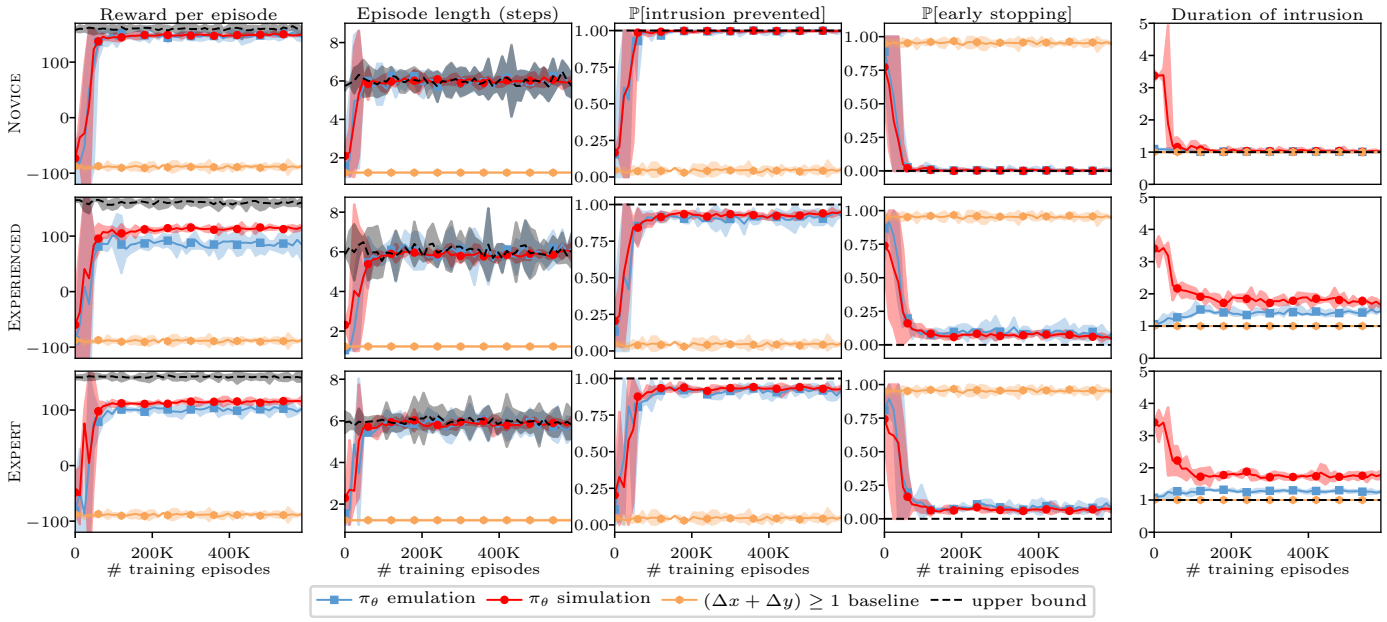

Fig. 9: Learning curves obtained during training; the defender takes a single stop ($L = 1$); red curves show simulation results and blue curves show emulation results; the rows from top to bottom relate to: NOVICEATTACKER, EXPERIENCEDATTACKER, and EXPERTATTACKER; the columns from left to right show episodic reward, episode length, empirical prevention probability, empirical early stopping probability, and the time between the start of intrusion and the stop action; all curves show the mean and 95% confidence interval for five training runs with different random seeds.

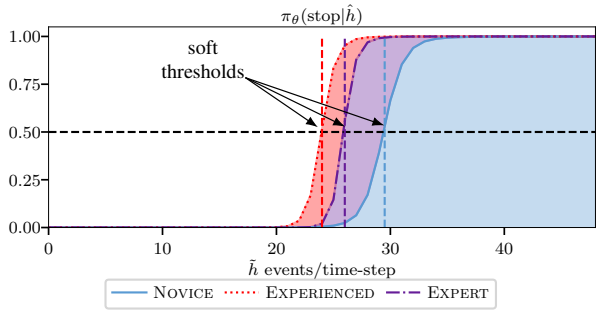

Fig. 10: Probability of the stop action by the learned policies in function of $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$ when $L = 1$; the probability is computed using $\pi_\theta(\text{stop}|\hat{h}_t)$ where $\hat{h}_t$ is obtained from $\hat{f}_{XYZ}$.

Third, although the learned policies, as expected, perform better in the simulation system than in the emulation system, we are encouraged by the fact that the curves of the emulation system are close to those of the simulation system.

We also note from Fig. 9 that the learned policies do best against NOVICEATTACKER and less well against EXPERIENCEDATTACKER and EXPERTATTACKER. For instance, the learned policies against EXPERIENCEDATTACKER and EXPERTATTACKER are more likely to stop before an intrusion has started (second rightmost column of Fig. 9). This indicates that NOVICEATTACKER is easier to detect for the defender as its actions create more IDS alerts than those of the other attackers, as pointed out in Section V-A.

Lastly, Fig. 10 illustrates some of the structural properties of the learned policies. The y-axis shows the probability of the stop action S and the x-axis shows the values of $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$. The curves are constructed as follows. For a given value of $\tilde{h}(h_t)$, we determine the tuple $\hat{h}_t = (\Delta x_t, \Delta y_t, \Delta z_t, t)$ as the maximum likelihood of the empirical distribution $\hat{f}_{XYZ}$ given $\tilde{h}(h_t)$. Then, applying the learned policy $\pi_\theta(S|\hat{h})$ gives us the probability of the stopping action S, which is shown on the curve.

We see that the learned policies mandates a stop action with high probability if $\tilde{h}(h_t)$ exceeds three different thresholds, namely $\tilde{h}(h_t) \geq 24$ against EXPERIENCEDATTACKER, $\tilde{h}(h_t) \geq 25$ against EXPERTATTACKER, and $\tilde{h}(h_t) \geq 30$ against NOVICEATTACKER. From this we conclude that the learned policies can be expressed through thresholds, which is consistent with the properties of an optimal policy under the assumptions stated in Corollary 1. That the learned policy against NOVICEATTACKER is associated with the highest threshold is anticipated as NOVICEATTACKER generates more alerts than the other attackers. However, it is somewhat surprising that the learned threshold against EXPERTATTACKER is higher than the threshold learned against EXPERIENCEDATTACKER, indicating that an intrusion by EXPERTATTACKER generates more IDS alerts and login attempts.

C. Learning Policies for Multiple Stops

Here we study the scenario where the defender is allowed to take at most $L = 3$ stop actions, each of which implements a defensive action (see Table 2 for the list of stop actions and Table 4 for the number of stops, $L - l^A$, required to prevent each attacker).

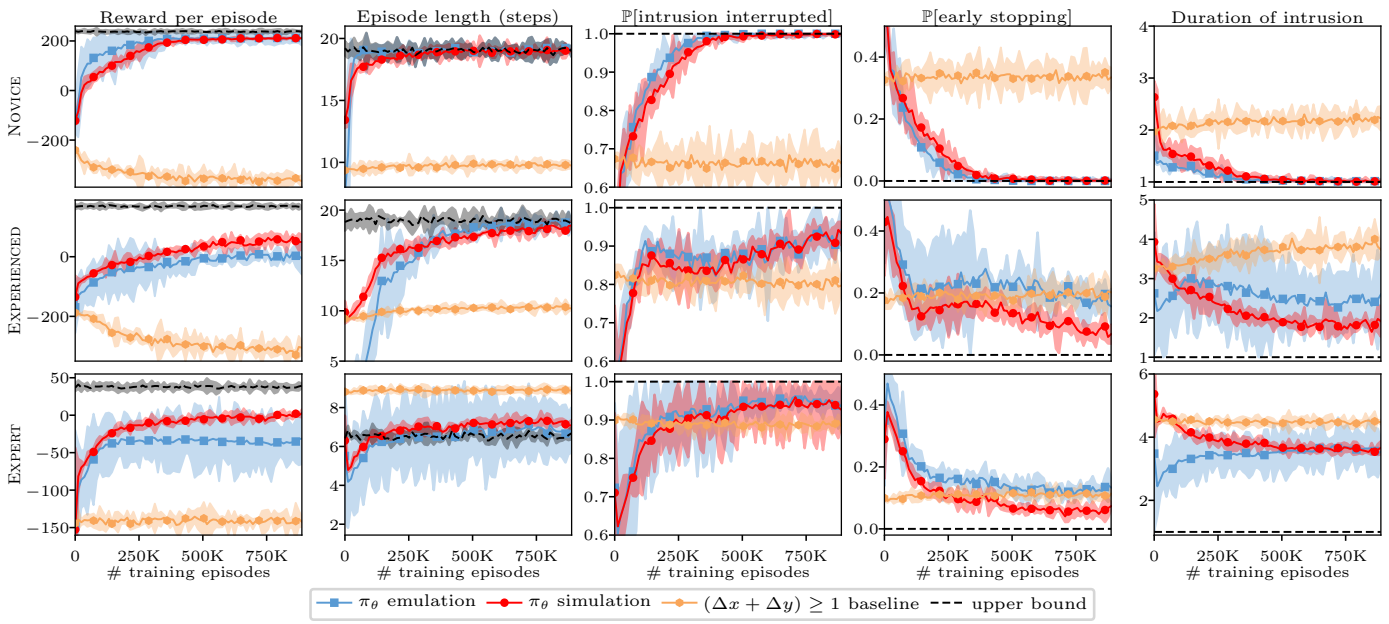

Fig. 11: Learning curves obtained during training; the defender takes at most three stops ($L = 3$); red curves show simulation results and blue curves show emulation results; the rows from top to bottom relate to: NoviceAttacker, ExperiencedAttacker, and ExpertAttacker; the columns from left to right show performance metrics: episodic reward, episode length, empirical prevention probability, empirical early stopping probability, and the time between the start of intrusion and the $L - l^A$th stop action; the curves show the mean and 95% confidence interval for five training runs with different random seeds.

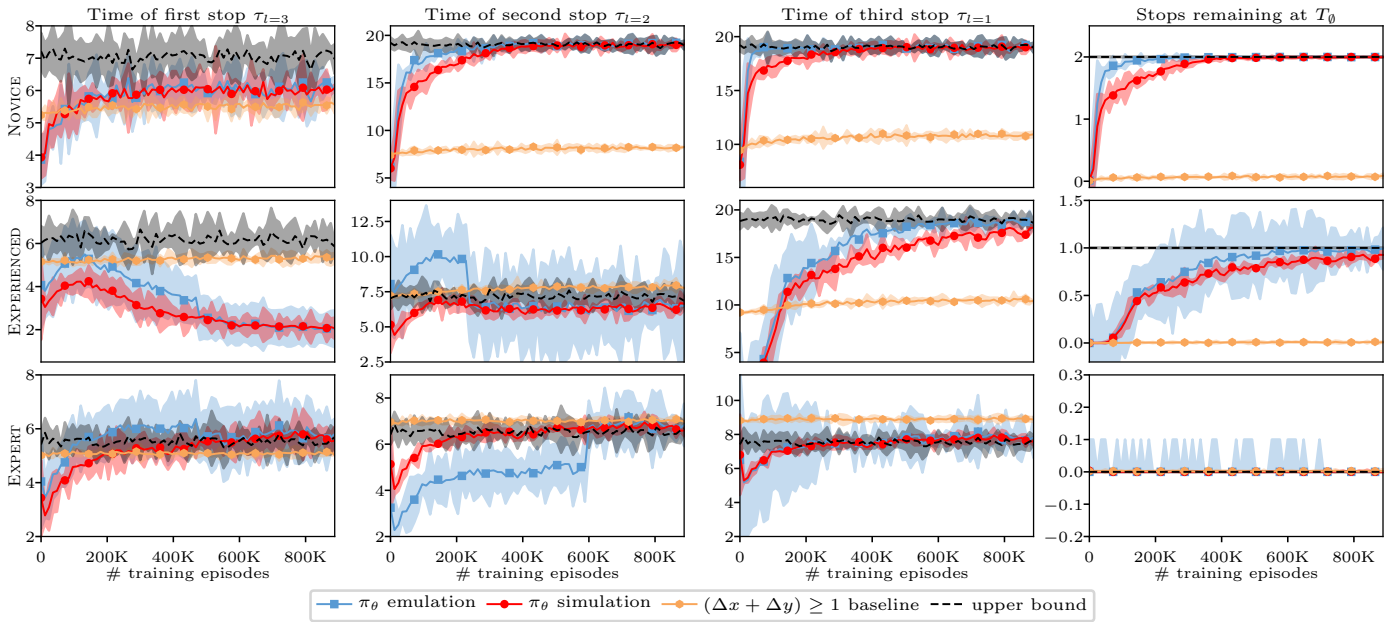

Fig. 12: Learning curves showing the stopping times during training ($L = 3$); the stopping time of unused stops are set to the episode length $T_\emptyset$; red curves show simulation results and blue curves show emulation results; the rows from top to bottom relate to: NoviceAttacker, ExperiencedAttacker, and ExpertAttacker; the columns from left to right show: the time-step of the first, second, and third stop, as well as the number of stops remaining when the episode ends; the curves show the mean and 95% confidence interval for five training runs with different random seeds.

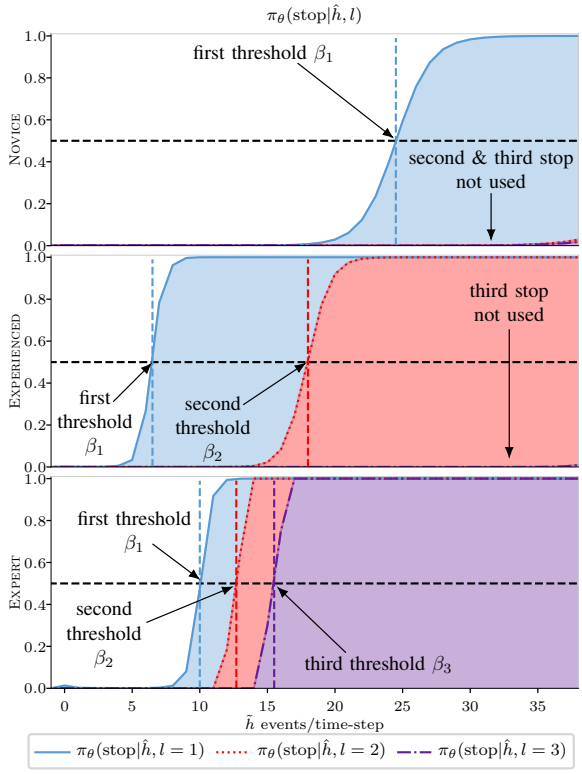

Fig. 13: Probability of the stop action by the learned policies in function of $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$ when $L = 3$; the probability is computed using $\pi_\theta(\text{stop}|\hat{h}, l)$ where $\hat{h}$ is obtained from $\hat{f}_{XYZ}$ and l indicates the number of stops remaining.

Figs. 11-12 show the performance of the learned policies against the three attacker types. The red curves represent the results from the simulation system and the blue curves show the results from the emulation system. The orange curves give the performance of the baseline that stops at the first IDS alert and the dashed black curves give the performance of the baseline that assumes knowledge of the exact intrusion time.

Similar to the evaluation in Section VI-B, we observe in Fig. 11 that the learning curves converge to constant mean values for all attackers and across all investigated performance metrics. After approximately $750'000$ training episodes the curves have sufficiently converged, indicating that the learned policies have converged as well.

Second, we observe, again similar to the evaluation in Section VI-B, that the converged values of the learning curves are close to the dashed black curves, which give an upper bound to any optimal policy. Furthermore, we see that the empirical probability of preventing an intrusion of the learned policies is close to 1 (middle column of Fig. 11) and that the empirical probability of stopping before the intrusion starts is close to 0 (second rightmost column of Fig. 11). This suggests that the learned policies are close to optimal. We also observe that all learned policies do significantly better than the baseline that stops on the first IDS alert (leftmost column in Fig. 11).

Third, we notice that, although the learned policies do slightly better in the simulation system than in the emulation system, the learning curves are similar. Finally, we also note that the confidence intervals of the learning curves in the emulation system are wider than that those of the learning curves in the simulation system, indicating that the performance in the emulation system is more dependent on random factors.

Next, Fig. 12 shows learning curves related to the stop actions and the stopping times. (If fewer than L stops are used in an episode, the stopping times of the unused stops are set to the length of the episode $T_\emptyset$.) In the rightmost column of Fig. 12, we observe that the learned policies use one stop against NoviceAttacker, two stops against ExperiencedAttacker, and three stops against ExpertAttacker. This shows that the learned policies use the number of stops required to prevent each attacker, $L - l^A$, as listed in Table 4 and encoded in the reward function (Section IV-A).

Lastly, Fig. 13 illustrates some of the structural properties of the learned policies for the case $L = 3$. Again, the discussion is related to that of the scenario where $L = 1$ (Section VI-B). As expected, we find that the learned policies have threshold properties. Specifically, we observe that the learned policies implement $L - l^A$ thresholds. That is, the policy learned against NoviceAttacker uses a single threshold, the policy learned against ExperiencedAttacker uses two thresholds, and the policy learned against ExpertAttacker uses three thresholds. Furthermore, the thresholds are increasing with each stop, as the bottom graph of Fig. 13 shows. These properties of the learned policies are consistent with those of the optimal policy expressed in Corollary 1 and Fig. 7.

VII. Related Work

The problem of automatically finding security policies has been studied using concepts and methods from different fields, most notably: statistics (see example [30]), control theory (see survey [6]), game theory (see textbook [10]), artificial intelligence (see survey [58]), dynamic programming (see example [5]), reinforcement learning (see survey [59]), evolutionary methods (see example [7]), and attack graphs (see example [60]).

Many recent results of automating security policies have been obtained using reinforcement learning methods. In particular, a large number of studies have focused on intrusion prevention use cases [12], [13], [15], [16], [17], [31], [18], [19], [20], [32], [21], [22], [23]. Other security problems addressed with reinforcement learning include: defense against jamming attacks [61], detection of distributed denial of service (DDoS) attacks [62], defense against advanced persistent threats (APTs) [63], botnet detection [64], and defense against topology attacks [65].

This paper differs from the works referenced above in two ways. First, we formulate the intrusion prevention problem as a multiple stopping problem ([31] and [13] use a similar formulation but with a single stop). The other works formulate the problem as a general MDP, POMDP, or Markov game. The advantage of our approach is that we obtain structural properties of optimal policies, which have practical benefits (see Section IV-C). A drawback of our approach, however,

is that the current stopping formulation restricts the defender to a predefined sequence of actions. We plan to address this drawback in future work.

Second, our method to find effective policies for intrusion prevention includes using an emulation system in addition to a simulation system. All of the above referenced works rely on simulation studies only (except [20], which uses an emulation based on Mininet [66]). The advantage of our method is that the parameters of the simulation system are determined by measurements from the emulation system instead of being based on assumptions. Further, the learned policies are evaluated in the emulation system, not in the simulation system. As a consequence, the evaluation results give higher confidence of the obtained policies' performance in the target infrastructure than what simulation results give.

Finally, several research initiatives focus on creating platforms for developing security solutions based on reinforcement learning. Some of these platforms include both simulation and emulation components ([67], [68]) and some are simulation only ([18], [12]). For the work presented in this paper, we have developed our own simulation system and emulation system. In contrast to the referenced works, our systems have been built to investigate the specific use case of intrusion prevention. At this point, our focus has not been to design a generic platform to experiment with reinforcement learning for different use cases.

VIII. Conclusion and Future Work

In this paper, we proposed a novel formulation of the intrusion prevention problem based on the theory of optimal stopping. This formulation allowed us to derive that a threshold policy based on infrastructure metrics is optimal, which has several practical benefits

To find and evaluate policies, we used a reinforcement learning method that includes a simulation system and an emulation system. In contrast to a simulation-only approach, our method produces policies that can be executed in a target infrastructure.

Through extensive evaluations, we showed that our approach can produce effective defender policies for a practical configuration of an IT infrastructure of limited size (Figs. 9-13). Further, when inspecting the learned policies, we discovered that they exhibit threshold properties that are consistent with an optimal policy (Theorem 1 and Corollary 1).

We plan to extend this work in three directions. First, the formulation of the intrusion prevention use case as a multiple stopping problem provides us with effective policies for *when* to take defender actions, but it does not tell us *which* actions to take. As a result, our defender model is restricted to a predefined sequence of defender actions. We plan to extend the defender model to remove this restriction. One possible approach is to learn two policies independently, a policy that decides when to take a defensive action and another policy that decides which action to take. Second, as pointed out in Section IV-C, the existence of an optimal threshold policy can be exploited to improve the sample efficiency of the reinforcement learning algorithm. We intend to develop a reinforcement learning algorithm that takes advantage of this property. Third, in the current paper, the attacker policy is static. We plan to extend the model to include a dynamic attacker, which will improve the robustness of the learned defender policies. Such an extension requires a game-theoretic formulation of the intrusion prevention use case.

IX. Acknowledgments

This research has been supported in part by the Swedish armed forces and was conducted at KTH Center for Cyber Defense and Information Security (CDIS). The authors would like to thank Pontus Johnson for his useful input to this research, and Forough Shahab Samani and Xiaoxuan Wang, and Jakob Nyberg for their constructive comments on a draft of this paper.

Appendix

A. Proof of Theorem 1.

Given the POMDP introduced in Section IV-A, let L denote the maximum number of stop actions, $L - l^A$ the minimum number of stops to prevent an intrusion, f_{XYZ} the observation distribution, $\mathcal{B} = [0, 1]$ the belief space (see Section IV-C), $b(1)$ the belief state, $\mathscr{S}^l$ the stopping set, and $\mathscr{C}^l$ the continuation set. We further assume a real-valued function $\tilde{h} : h_t \mapsto \mathbb{R}$ on the history of observations h_t and a monotonically increasing function $g : \tilde{h}(h_t) \mapsto b_t(1)$, which maps $\tilde{h}(h_t)$ onto $\mathcal{B}$. (In our case, $\tilde{h}(h_t) = \Delta x_t + \Delta y_t + \Delta z_t$.)

The main idea behind the proof of Theorem 1 is to show that the stopping sets $\mathscr{S}^l$ have the form $\mathscr{S}^l = [\alpha^*_l, 1] \subseteq \mathcal{B}$ for $l = l^A + 1, \ldots, L$. Towards this goal, we state the following three lemmas.

Lemma 1. *During a POMDP episode, an optimal policy π^* prescribes $L - l^A$ stop actions.*

Proof. The proof follows directly from the definition of the reward function (see Eqs. 19-22). □

Lemma 2. *If $L - l^A = 1$, $\mathscr{S}^{l^A+1}$ is a convex subset of $\mathcal{B}$.*

Proof. The proof can be found in [13, pp. 10, Lemma 3] and in [36, pp. 258, Theorem 12.2.1]. □

Lemma 3. *If $L - l^A \geq 1$, $f_{XYZ|s}$ is TP2, $\mathcal{P}^C_{ss'}$ is TP2, and $\mathcal{R}^a_{s,l}$ is decreasing in s, then the stopping sets $\mathscr{S}^l, l = l^A + 1, \ldots, L$ are connected.*

Proof. The proof can be found in [14, pp. 389, Theorem 1.A,1.B]. □

Proof of Theorem 1.A. The proof has originally been published in [27, Propositions 4.5-4.8, pp. 437-441]. It is also available in a more accessible form in [14, Theorem 1.C, Theorem 8, pp. 389-397]. □

Proof of Theorem 1.B. The proof follows the chain of reasoning in [36, Corollary 12.2.2, pp. 258].

Using Lemma 2, we know that the stopping set $\mathscr{S}^{l^A+1}$ is a convex subset of $\mathcal{B} = [0, 1]$. That is, it has the form $[\alpha^*, \beta^*]$ where $0 \leq \alpha^* \leq \beta^* \leq 1$. We show that $\beta^* = 1$.

Parameters	*Values*
γ, lr α, batch, # layers, # neurons, clip ϵ	1, 10^{-5}, $12 \cdot 10^3 t$, 4, 128, 0.2
$X_{max}, Y_{max}, Z_{max}$, GAE λ, ent-coef, activation	$6 \cdot 10^2$, $3 \cdot 10^2$, 10^2, 0.95, 10^{-4}, ReLU

TABLE 6: Hyperparameters of the learning algorithm.

If $b(1) = 1$, the Bellman equation (Eq. 10) states that:

$$\pi^*_{l^A+1}(1) = \arg\max_{\{S,C\}} \Bigg[\underbrace{200 - 100/2^{l^A} + V^*_{l^A}(0)}_{a=S}, \underbrace{-100 + 10/2^{L-l^A-1} + \sum_{o \in \mathcal{O}} \mathcal{Z}(o,1,C) V^*_{l^A+1}\big(b^o_C(1)\big)}_{a=C} \Bigg] \quad (29)$$

As $L - l^A = 1$, it follows from Lemma 1 that an optimal policy prescribes one stop action during a POMDP episode. Hence, $V^*_{l^A}(0) = (T_\emptyset - t)10/2^{L-l^A}$. Moreover, since $s = 1$ is an absorbing state until the stop is made or the intrusion completes, $b^o_C(1) = 1$ for all $o \in \mathcal{O} \setminus \emptyset$. This follows from the definition of b^o_C (Eq. 3). Then, since $V^*_{l^A+1}(1) \leq 200 - 100/2^{l^A} + V^*_{l^A}(0)$ we get:

$$\pi^*_{l^A+1}(1) = \arg\max_{\{S,C\}} \Bigg[\underbrace{200 - 100/2^{l^A} + (T_\emptyset - t)10/2^{L-l^A}}_{a=S}, \underbrace{-100 + 10/2^{L-l^A-1} + V^*_{l^A+1}(1)}_{a=C} \Bigg] = S \quad (30)$$

This means that $\beta^* = 1$ and therefore $\mathscr{S}^{l^A+1} = [\alpha^*, 1]$. □

Proof of Theorem 1.C. Since $\mathcal{B} = [0, 1]$, the transition probabilities $\mathcal{P}^C_{ss'}$ are TP2 (see Eqs. 11-16), and the reward function $\mathcal{R}^a_{s,l}$ is decreasing in s (see Eqs. 19-22), it follows from Lemma 3 that $\mathscr{S}^l$ is a convex subset of $[0, 1]$ for $l = l^A + 1, \ldots, L$. Further, from Theorem 1.B we know that $\mathscr{S}^{l^A+1} = [\alpha^*_{l^A+1}, 1]$. Then, because $\mathscr{S}^l \subseteq \mathscr{S}^{l+1}$ for $l = l^A + 1, \ldots, L - 1$ (Theorem 1.A), we conclude that $\mathscr{S}^l = [\alpha^*_l, 1]$ and that $\alpha^*_l \leq \alpha^*_{l+1}$ for $l = l^A+1, \ldots, L-1$. □

B. Proof of Corollary 1.

From Theorem 1.C we know that there exist $L-l^A$ thresholds $\alpha^*_{l^A+1} \geq \alpha^*_{l+A+2} \geq \ldots \alpha^*_L$ and an optimal policy π^*_l such that $\pi^*_l(b(1)) = S \iff b(1) \geq \alpha^*_l$ for $l \in l^A + 1, \ldots, L$.

By assumption, there is a monotonically increasing function $g : \tilde{h}(h_t) \mapsto b_t(1)$. It follows that a minimum value $\beta^*_l = \inf\{\tilde{h}(h_t) | g(\tilde{h}(h_t)) \geq \alpha^*_l\}$ exist for $l = l^A + 1, \ldots, L$. From this we conclude that it is optimal to stop when $\tilde{h}(h_t) = \beta^*_l$. Since g is monotonically increasing it also follows that for any value $\tilde{h}(h_t) < \beta^*_l$ it is optimal to continue, i.e. $\pi^*_l(h_t) = C \iff \tilde{h}(h_t) < \beta_l$. Similarly, for any value $\tilde{h}(h_t) > \beta^*_l$, it is optimal to stop, i.e. $\pi^*_l(h_t) = S \iff \tilde{h}(h_t) \geq \beta_l$ for $l \in l^A + 1, \ldots, L$. □

C. Hyperparameters

The hyperparameters of the reinforcement learning algorithm are listed in Table 6 and were decided based on a grid search in parameter space (Fig. 14).

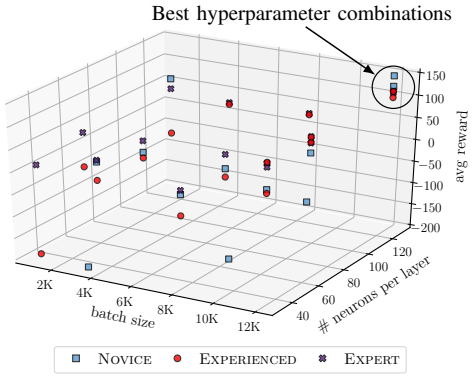

Fig. 14: Average reward after $40'000$ policy updates when tuning the batch size (in time-steps) and the number of neurons per layer; $L = 1$; the other hyperparameters in Table 6 were kept static.

ID (s)	*OS:Services:Exploitable Vulnerabilities*
N_1	Ubuntu20:Snort(community ruleset v2.9.17.1),SSH:-
N_2	Ubuntu20:SSH,HTTP Erl-Pengine,DNS:SSH-pw
N_4	Ubuntu20:HTTP Flask,Telnet,SSH:Telnet-pw
N_{10}	Ubuntu20:FTP,MongoDB,SMTP,Tomcat,TS3,SSH:FTP-pw
N_{12}	Jessie:TS3,Tomcat,SSH:CVE-2010-0426,SSH-pw
N_{17}	Wheezy:Apache2,SNMP,SSH:CVE-2014-6271
N_{18}	Deb9.2:IRC,Apache2,SSH:SQL Injection
N_{22}	Jessie:PROFTPD,SSH,Apache2,SNMP:CVE-2015-3306
N_{23}	Jessie:Apache2,SMTP,SSH:CVE-2016-10033
N_{24}	Jessie:SSH:CVE-2015-5602,SSH-pw
N_{25}	Jessie: Elasticsearch,Apache2,SSH,SNMP:CVE-2015-1427
N_{27}	Jessie:Samba,NTP,SSH:CVE-2017-7494
N_3,N_{11},N_5-N_9	Ubuntu20:SSH,SNMP,PostgreSQL,NTP:-
N_{13-16},N_{19-21},N_{26},N_{28-31}	Ubuntu20:NTP, IRC, SNMP, SSH, PostgreSQL:-

TABLE 7: Configuration of the target infrastructure (Fig. 1).

D. Configuration of the Infrastructure in Fig. 1

The configuration of the target infrastructure (Fig. 1) is available in Table 7.

References

[1] A. Fuchsberger, "Intrusion detection systems and intrusion prevention systems," *Inf. Secur. Tech. Rep.*, vol. 10, no. 3, p. 134–139, Jan. 2005.

[2] K.-K. R. Choo, "The cyber threat landscape: Challenges and future research directions," *Comput. Secur.*, vol. 30, no. 8, p. 719–731, 2011.

[3] E. Zouave, M. Bruce, K. Colde, M. Jaitner, I. Rodhe, and T. Gustafsson, "Artificially intelligent cyberattacks," 2020, ,Swedish Defence Research Agency.

[4] P. Johnson, R. Lagerström, and M. Ekstedt, "A meta language for threat modeling and attack simulations," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ser. ARES 2018, New York, NY, USA, 2018.

[5] M. Rasouli, E. Miehling, and D. Teneketzis, "A supervisory control approach to dynamic cyber-security," in *Decision and Game Theory for Security*. Cham: Springer International Publishing, 2014, pp. 99–117.

[6] E. Miehling, M. Rasouli, and D. Teneketzis, *Control-Theoretic Approaches to Cyber-Security*. Cham: Springer International Publishing, 2019, pp. 12–28.

[7] R. Bronfman-Nadas, N. Zincir-Heywood, and J. T. Jacobs, "An artificial arms race: Could it improve mobile malware detectors?" in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, 2018.

[8] N. Wagner, C. c. Şahin, M. Winterrose, J. Riordan, J. Pena, D. Hanson, and W. W. Streilein, "Towards automated cyber decision support: A case study on network segmentation for security," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–10.

[9] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information*

Sharing and Collaborative Security, ser. WISCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 49–56.
[10] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed. USA: Cambridge University Press, 2010.
[11] S. Sarıtaş, E. Shereen, H. Sandberg, and G. Dán, "Adversarial attacks on continuous authentication security: A dynamic game approach," in *Decision and Game Theory for Security*, Cham, 2019, pp. 439–458.
[12] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and Self-Play," in *International Conference on Network and Service Management (CNSM 2020)*, Izmir, Turkey, 2020.
[13] ——, "Learning intrusion prevention policies through optimal stopping," in *International Conference on Network and Service Management (CNSM 2021)*, Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.07160.pdf.
[14] V. Krishnamurthy, A. Aprem, and S. Bhatt, "Multiple stopping time pomdps: Structural results & application in interactive advertising on social media," *Automatica*, vol. 95, pp. 385–398, 2018.
[15] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Drugan, and M. Wiering, "Adversarial reinforcement learning in a cyber security simulation," in *ICAART*, 2017.
[16] J. Schwartz, H. Kurniawati, and E. El-Mahassni, "Pomdp + information-decay: Incorporating defender's behaviour in autonomous penetration testing," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, no. 1, pp. 235–243, Jun. 2020.
[17] F. M. Zennaro and L. Erdodi, "Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular q-learning," *CoRR*, 2020, https://arxiv.org/abs/2005.12632.
[18] W. Blum, "Gamifying machine learning for stronger security and ai models," 2019, https://www.microsoft.com/security/blog/2021/04/08/gamifying-machine-learning-for-stronger-security-and-ai-models/.
[19] A. Ridley, "Machine learning for autonomous cyber defense," 2018, the Next Wave, Vol 22, No.1 2018.
[20] J. Gabirondo-López, J. Egaña, J. Miguel-Alonso, and R. Orduna Urrutia, "Towards autonomous defense of sdn networks using muzero based intelligent agents," *IEEE Access*, vol. 9, pp. 107 184–107 199, 2021.
[21] K. Tran, A. Akella, M. Standen, J. Kim, D. Bowman, T. Richer, and C.-T. Lin, "Deep hierarchical reinforcement agents for automated penetration testing," 2021, https://arxiv.org/abs/2109.06449.
[22] R. Gangupantulu, T. Cody, P. Park, A. Rahman, L. Eisenbeiser, D. Radke, and R. Clark, "Using cyber terrain in reinforcement learning for penetration testing," 2021, https://arxiv.org/abs/2108.07124.
[23] Z. Hu, M. Zhu, and P. Liu, "Adaptive cyber defense against multi-stage attacks using learning-based pomdp," *ACM Trans. Priv. Secur.*, vol. 24, no. 1, Nov. 2020.
[24] G. Sofronov, J. Keith, and D. Kroese, "An optimal sequential procedure for a buying-selling problem with independent observations," *Journal of Applied Probability*, vol. 43, no. 2, pp. 454–462, 2006.
[25] R. Carmona and N. Touzi, "Optimal multiple stopping and valuation of swing options," *Mathematical Finance*, vol. 18, pp. 239–268, Apr. 2008.
[26] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions," in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '05, 2005.
[27] T. Nakai, "The problem of optimal stopping in a partially observable markov chain," *Journal of Optimization Theory and Applications*, vol. 45, no. 3, pp. 425–442, Mar 1985.
[28] J. du Toit and G. Peskir, "Selling a stock at the ultimate maximum," *The Annals of Applied Probability*, vol. 19, no. 3, Jun 2009.
[29] A. Roy, V. S. Borkar, A. Karandikar, and P. Chaporkar, "Online reinforcement learning of optimal threshold policies for markov decision processes," *CoRR*, 2019, http://arxiv.org/abs/1912.10325.
[30] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blažek, and H. Kim, "Detection of intrusions in information systems by sequential change-point methods," *Statistical Methodology*, vol. 3, no. 3, 2006.
[31] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5174–5185, 2019.
[32] M. Zhu, Z. Hu, and P. Liu, "Reinforcement learning algorithms for adaptive cyber defense against heartbleed," in *Proceedings of the First ACM Workshop on Moving Target Defense*, ser. MTD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 51–58.
[33] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.
[34] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. USA: John Wiley and Sons, Inc., 1994.
[35] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.
[36] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.
[37] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," USA, 1996.
[38] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Belmont, MA: Athena Scientific, 1996.
[39] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[40] R. Bellman, *Dynamic Programming*. Dover Publications, 1957.
[41] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005, vol. I.
[42] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Math. Oper. Res.*, vol. 12, p. 441–450, Aug. 1987.
[43] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, 1989.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017, http://arxiv.org/abs/1707.06347.
[45] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, vol. 6, 1994.
[46] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400 – 407, 1951.
[47] A. Wald, *Sequential Analysis*. Wiley and Sons, New York, 1947.
[48] A. N. Shirayev, *Optimal Stopping Rules*. Springer-Verlag Berlin, 2007, reprint of russian edition from 1969.
[49] G. Peskir and A. Shiryaev, *Optimal stopping and free-boundary problems*, ser. Lectures in mathematics (ETH Zürich). Springer, 2006.
[50] Y. Chow, H. Robbins, and D. Siegmund, "Great expectations: The theory of optimal stopping," 1971.
[51] S. M. Ross, *Introduction to Stochastic Dynamic Programming: Probability and Mathematical*. USA: Academic Press, Inc., 1983.
[52] J. Bather, *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. USA: John Wiley and Sons, Inc., 2000.
[53] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, international Conference for Learning Representations, San Diego.
[55] K. Hammar and R. Stadler, "gym-optimal-intrusion-response," 2021, https://github.com/Limmen/gym-optimal-intrusion-response.
[56] S. Karlin, "Total positivity, absorption probabilities and applications," *Transactions of the American Mathematical Society*, vol. 111, 1964.
[57] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.
[58] N. Dhir, H. Hoeltgebaum, N. Adams, M. Briers, A. Burke, and P. Jones, "Prospective artificial intelligence approaches for active cyber defence," *CoRR*, vol. abs/2104.09981, 2021, https://arxiv.org/abs/2104.09981.
[59] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *CoRR*, 2019, http://arxiv.org/abs/1906.05799.
[60] E. Miehling, M. Rasouli, and D. Teneketzis, "A pomdp approach to the dynamic defense of large-scale cyber networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, 2018.
[61] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "Afrl: Adaptive federated reinforcement learning for intelligent jamming defense in fanet," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 244–258, 2020.
[62] K. Malialis and D. Kudenko, "Multiagent router throttling: Decentralized coordinated response against ddos attacks," in *IAAI*, 2013.
[63] B. Ning and L. Xiao, "Defense against advanced persistent threats in smart grids: A reinforcement learning approach," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 8598–8603.
[64] M. Alauthman, N. Aslam, M. Alkasassbeh, S. Khan, A. al Qerem, and K. Choo, "An efficient reinforcement learning-based botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, 2020.
[65] J. Yan, H. He, X. Zhong, and Y. Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 200–210, 2017.
[66] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010.
[67] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "Cyborg: A gym for the development of autonomous cyber agents," *CoRR*, vol. abs/2108.09118, 2021.
[68] A. Molina-Markham, C. Miniter, B. Powell, and A. Ridley, "Network environment design for autonomous cyberdefense," *CoRR*, vol. abs/2103.07583, 2021.