Intrusion Prevention Through Optimal Stopping

Kim Hammar, *Graduate Student Member, IEEE*, and Rolf Stadler, *Senior Member, IEEE*

***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 threshold policy. We introduce T-SPSA, an efficient reinforcement learning algorithm that learns threshold policies through stochastic approximation. We show that T-SPSA outperforms state-of-the-art algorithms for our use case. Our overall 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 this approach can produce effective defender policies for a practical IT infrastructure.**

***Index Terms*—Network security, automation, optimal stopping, reinforcement learning, Markov decision process, MDP, POMDP.**

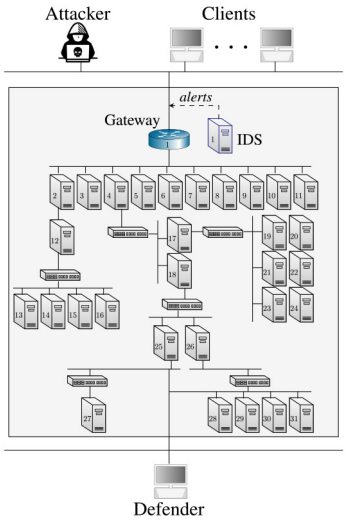

Fig. 1. The IT infrastructure and the actors in the use case.

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 frameworks and the process of obtaining effective security policies. 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].

Manuscript received 30 October 2021; revised 1 April 2022; accepted 16 May 2022. Date of publication 20 May 2022; date of current version 12 October 2022. 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 associate editor coordinating the review of this article and approving it for publication was C. Fung. *(Corresponding author: Kim Hammar.)*

The authors are with the Division of Network and Systems Engineering and the KTH Center for Cyber Defense and Information Security, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden (e-mail: kimham@kth.se; stadler@kth.se).

Digital Object Identifier 10.1109/TNSM.2022.3176781

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], [15]. In particular, we show that an optimal *multi-threshold policy* exists that can be efficiently computed and implemented.

The structure of optimal policies in dynamical systems is a well studied area [16], [17]. However, it has not been considered in prior research on automated intrusion prevention [12], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]. Further, although the optimal stopping problem frequently is used to formulate problems in the fields of finance and communication systems [13], [30], [31], [32], [33], [34], [35], [36], [37], [38], 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. An optimal policy for a POMDP can be obtained through two main methods: dynamic programming and reinforcement learning. In our case, dynamic programming is not feasible due to the size of the POMDP [39]. Therefore, we use a reinforcement learning approach to obtain the defender policy. We simulate a long series of POMDP episodes whereby the defender continuously updates its policy based on outcomes of previous episodes. To update the policy, we introduce T-SPSA, a reinforcement learning algorithm that exploits the threshold structure of optimal policies. We show that T-SPSA efficiently learns a near-optimal policy despite the high complexity of computing optimal policies for general POMDPs [39].

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 possibly implemented in the target infrastructure (see Fig. 2). 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. (A video demonstration of our software framework that implements the emulation and simulation systems is available in [40].)

We make three 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

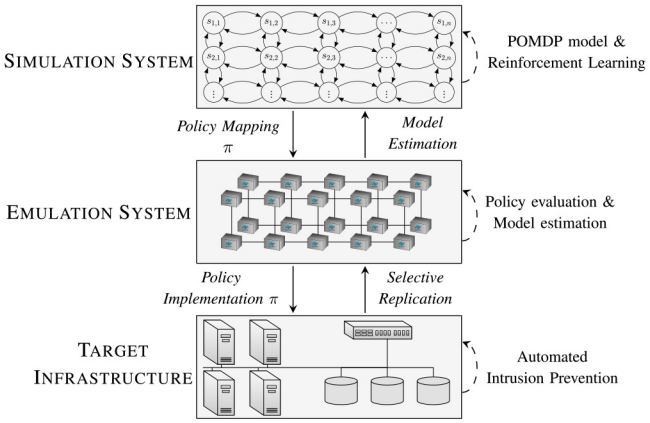

Fig. 2. Our approach for finding and evaluating intrusion prevention policies.

of many related works, which rely on simulations solely to evaluate policies [7], [12], [18], [19], [20], [21], [41]. Third, we present T-SPSA, an efficient reinforcement learning algorithm that exploits the threshold structure of optimal policies and outperforms state-of-the-art algorithms for our use case.

We conclude this section with remarks about the context of this research and the practical relevance of the results in this paper. The objective of our line of research is to construct a mathematical and conceptual framework, validated by an experimental environment, that produces defender policies for realistic scenarios through self-learning. We are engaged in a program with high potential reward that will need many years of investigation. This paper provides an important result and milestone in this program.

From a practical point of view, the main question the paper answers is this: at which points in time should a defender take defensive actions given periodic but limited observational data? The paper proposes a fundamental framework to study this question. We show theoretically and experimentally that the optimal action times can be obtained through thresholds that the framework predicts and which can be efficiently implemented in a real system.

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. A defensive action is for example to revoke user certificates in the infrastructure, which will recover user accounts compromised by the attacker. It is assumed that the defender

takes the defensive actions in a predetermined order. 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. 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 by taking defensive actions. The challenge for the defender is to identify the precise time when this moment occurs.

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 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 $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^{a_t}_{s_t,s_{t+1}}, \mathcal{R}^{a_t}_{s_t,s_{t+1}}, \gamma, \rho_1, T \rangle$ [14], [16]. $\mathcal{S}$ denotes the set of states and $\mathcal{A}$ denotes the set of actions. $\mathcal{P}^{a_t}_{s_t,s_{t+1}}$ refers to the probability of transitioning from state s_t to state s_{t+1} when taking action a_t (Eq. (1)), which has the Markov property $\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t]$. Similarly, $\mathcal{R}^{a_t}_{s_t,s_{t+1}} \in \mathbb{R}$ is the expected reward when taking action a_t and transitioning from state s_t to state s_{t+1} (Eq. (2)), which is bounded, i.e., $|\mathcal{R}^{a_t}_{s_t,s_{t+1}}| \leq M < \infty$ for some $M \in \mathbb{R}$. If $\mathcal{P}^{a_t}_{s_t,s_{t+1}}$ and $\mathcal{R}^{a_t}_{s_t,s_{t+1}}$ are independent of the time-step t, the MDP is said to be *stationary* and if $\mathcal{S}$ and $\mathcal{A}$ are finite, the MDP is said to be *finite*. 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.

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

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

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 [17], [42]. In contrast to an MDP, in a POMDP the states are not directly observable. A POMDP is defined by a nine-tuple $\mathcal{M}_\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^{a_t}_{s_t,s_{t+1}}, \mathcal{R}^{a_t}_{s_t,s_{t+1}}, \gamma, \rho_1, T, \mathcal{O}, \mathcal{Z} \rangle$. The first seven elements define an MDP. $\mathcal{O}$ denotes the set of observations and $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t) = \mathbb{P}[o_{t+1}|s_{t+1}, a_t]$ is the observation function, where $o_{t+1} \in \mathcal{O}$, $s_{t+1} \in \mathcal{S}$, and $a_t \in \mathcal{A}$. If $\mathcal{O}$, $\mathcal{S}$, and $\mathcal{A}$ are finite, the POMDP is said to be finite.

The belief state $b_t \in \mathcal{B}$ is defined as $b_t(s) = \mathbb{P}[s_t = s|h_t]$ for all $s \in \mathcal{S}$. b_t 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: $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t) \in \mathcal{H}$. The belief space $\mathcal{B} = \Delta(\mathcal{S})$ is the unit $(|\mathcal{S}| - 1)$-simplex [43], [44], where $\Delta(\mathcal{S})$ denotes the set of probability distributions over $\mathcal{S}$. By defining the state at time t to be the belief state b_t, a POMDP can be formulated as a continuous-state MDP: $\mathcal{M} = \langle \mathcal{B}, \mathcal{A}, \mathcal{P}^{a_t}_{b_t,b_{t+1}}, \mathcal{R}^{a_t}_{b_t,b_{t+1}}, \gamma, \rho_1, T \rangle$.

The belief state can be computed recursively as follows [17]:

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

where the denominator is independent of s_{t+1} and makes 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 [45], [46]. This problem 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 $\pi : \{1, \ldots, T\} \times \mathcal{S} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the set of probability distributions over $\mathcal{A}$. In the case of a POMDP, a policy is defined as a function $\pi : H \to \Delta(\mathcal{A})$, or, alternatively, as a function $\pi : \{1, \ldots, T\} \times \mathcal{B} \to \Delta(\mathcal{A})$. In both cases, a policy is called *stationary* if it is independent of the time-step t given the current state or belief state.

An optimal policy π^* is a policy that maximizes the expected discounted cumulative reward over the time horizon:

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

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.

Optimal *deterministic* policies exist for finite MDPs and POMDPs with bounded rewards and either finite horizons or infinite horizons with $\gamma \in [0, 1)$ [16], [17]. If the MDPs or POMDPs also are stationary and the horizons are either random or infinite with $\gamma \in [0, 1)$, optimal *stationary* policies exist [16], [17].

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 [47]:

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

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

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

$V^*(s_t)$ and $Q^*(s_t, a_t)$ denote the expected cumulative discounted reward under π^* for each state and state-action pair, respectively. Solving Eqs. (5), (6) means computing the value

functions from which an optimal policy can be obtained (Eq. (7)). In the case of a POMDP, the Bellman equations contain b_t instead of s_t and $V^*(b_t)$ is piecewise linear and convex [48].

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

First, the dynamic programming method (e.g., value iteration [16], [47], [49]) 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 [39].

Second, the reinforcement learning method computes or approximates an optimal policy without requiring 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 [50]); *policy-based algorithms*, which directly search through policy space using gradient-based methods (e.g., Proximal Policy Optimization (PPO) [51]); and *model-based algorithms*, which learn the transition or observation probabilities of the MDP or POMDP (e.g., Dyna-Q [46]). The three algorithm types can also be combined, e.g., through *actor-critic* algorithms, which are mixtures of value-based and policy-based algorithms [46]. In contrast to dynamic programming algorithms, reinforcement learning algorithms generally have no guarantees to converge to an optimal policy except for the tabular case [52], [53].

C. The Markovian Optimal Stopping Problem

Optimal stopping is a classical problem domain with a well-developed theory [15], [16], [49], [54], [55], [56], [57], [58], [59]. Example use cases for this theory include: asset selling [49], change detection [37], machine replacement [17], hypothesis testing [15], gambling [56], selling decisions [35], queue management [36], advertisement scheduling [33], industrial control [60], and the secretary problem [16], [32].

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 and partially observed state spaces, problems with finite and infinite 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* [55], [56], [61] and the *Markovian approach* [16], [49], [54], [57], [58].

In this paper, we investigate the multiple stopping problem with L stops, a finite time horizon T, discrete-time progression, bounded rewards, a finite state space, 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 $(s_{t,l})_{t=1}^{T}$ that is partially observed and depends on the number of stops remaining $l \in \{1, \ldots, L\}$.

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 $\mathcal{R}^{S}_{s_t,s_{t+1},l_t}$ and if only one of the L stops remain, the process terminates. In the case of a *continue* action or a non-final stop action a_t, the decision process transitions to the next state according to the transition probabilities $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ and yields a reward $\mathcal{R}^{a_t}_{s_t,s_{t+1},l_t}$.

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 $s_{\tau_l+1}, \ldots s_T$ [55]:

$$\tau_l = \inf\{t : t > \tau_{l+1}, a_t = S\}, l \in 1, \ldots, L, \ \tau_{L+1} = 0 \tag{8}$$

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

$$\pi^*_l \in \arg\max_{\pi_l} \mathbb{E}_{\pi_l}\Bigg[\sum_{t=1}^{\tau_L-1} \gamma^{t-1}\mathcal{R}^{C}_{s_t,s_{t+1},L} + \gamma^{\tau_L-1}\mathcal{R}^{S}_{s_{\tau_L},s_{\tau_L+1},L} + \cdots + \sum_{t=\tau_2+1}^{\tau_1-1} \gamma^{t-1}\mathcal{R}^{C}_{s_t,s_{t+1},1} + \gamma^{\tau_1-1}\mathcal{R}^{S}_{s_{\tau_1},s_{\tau_1+1},1}\Bigg] \tag{9}$$

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:

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

for all $b \in \mathcal{B}$, 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 $\mathcal{R}^{a}_{b,b^o_a,l}$ is the expected reward of action $a \in \{S, C\}$ in belief state b_t when observing o with l stops remaining.

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 apply the theory of dynamic programming and optimal stopping to obtain structural results of an optimal defender policy. Lastly, we describe our reinforcement learning approach to approximate 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.

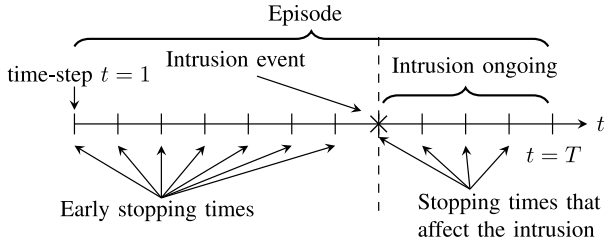

Fig. 3. Optimal multiple stopping formulation of intrusion prevention; the horizontal axis represents time; T is the time horizon; the episode length is $T - 1$; the dashed line shows the intrusion start time; the optimal policy is to prevent the attacker at the time of intrusion.

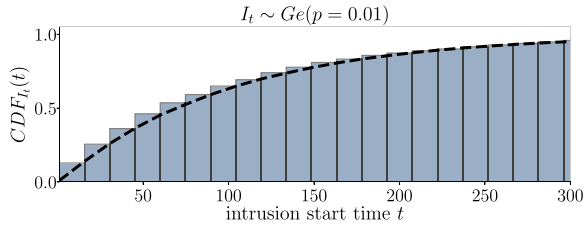

Fig. 4. The cumulative distribution function (CDF) of the intrusion start time I_t.

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.

The number of stops that the defender must execute to prevent an intrusion is $L \geq 1$, which is a predefined parameter of our use case.

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 an intrusion is prevented (see below). The state space is thus $\mathcal{S} = \{0, 1, \emptyset\}$.

3) Observations $\mathcal{O}$: The defender has a partial view of the system. If $s_t \neq \emptyset$, the defender observes $o_t = (l_t, \Delta x_t, \Delta y_t, \Delta z_t)$, where $l_t \in \{1, 2, \ldots, L\}$ is the number of stops remaining and (Δx_t, Δy_t, Δz_t) are bounded counters that 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 $o_T = \emptyset$. Hence, the observation space is $\mathcal{O} = \{0, \ldots, \Delta x_{max}\} \times \{0, \ldots, \Delta y_{max}\} \times \{0, \ldots, \Delta z_{max}\} \cup \emptyset$.

4) Transition Probabilities $\mathcal{P}^{a_t}_{s_t, s_{t+1}, l_t}$: We model the start of an intrusion by a Bernoulli process $(Q_t)_{t=1}^{T}$, where $Q_t \sim Ber(p = 0.01)$ 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., $I_t \sim Ge(p = 0.01)$ (Fig. 4).

We define the time-homogeneous transition probabilities $\mathcal{P}^{a_t}_{s_t, s_{t+1}, l_t} = \mathbb{P}_{l_t}[s_{t+1} | s_t, a_t]$ as follows:

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

$$\mathbb{P}_{l_t}[0 | 0, a_t] = 1 - p \qquad \text{if } l_t - a_t > 0 \tag{12}$$

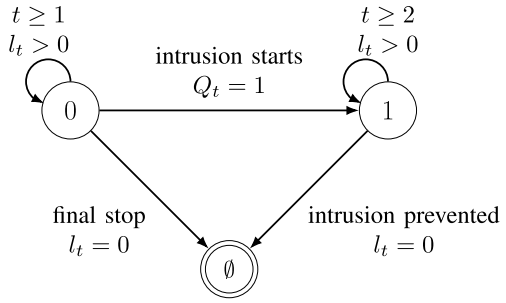

Fig. 5. State transition diagram of the POMDP: each circle represents a state; an arrow represents a state transition; a label indicates the event that triggers the state transition; an episode starts in state $s_1 = 0$ with $l_1 = L$.

$$\mathbb{P}_{l_t}[1 | 0, a_t] = p \qquad \text{if } l_t - a_t > 0 \tag{13}$$

$$\mathbb{P}_{l_t}[1 | 1, a_t] = 1 \qquad \text{if } l_t - a_t > 0 \tag{14}$$

where $\mathbb{P}_{l_t}$ denotes the probability with l_t stops remaining. All other state transitions occur with probability 0.

Eq. (11) defines the transition probabilities to the terminal state $\emptyset$. The terminal state is reached when the *final* ($l_t = 1$) stop action S ($a_t = 1$) is taken. If Eq. (11) is not applicable, i.e., if the system does not reach the terminal state, then the transition probabilities when taking action S ($a_t = 1$) or C ($a_t = 0$) are defined by Eqs. (12)–(14).

Eq. (12) captures the case where no intrusion occurs and $s_{t+1} = s_t = 0$; Eq. (13) specifies the case when the intrusion starts where $s_t = 0$ and $s_{t+1} = 1$; and Eq. (14) describes the case where an intrusion is in progress and $s_{t+1} = s_t = 1$.

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

5) Observation Function $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t)$: We assume that the number of IDS alerts and login attempts generated during one time-step are discrete random variables $X \sim f_X$, $Y \sim f_Y$, $Z \sim f_Z$ that depend on the state. Consequently, the probability that Δx severe alerts, Δy warning alerts, and Δz login attempts occur during time-step t can be expressed as $f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t)$.

We define the stationary observation function $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t) = \mathbb{P}[o_{t+1} | s_{t+1}, a_t]$ as follows:

$$\mathcal{Z}((l_t, \Delta x, \Delta y, \Delta z), s_t, \cdot) = f_{XYZ}(\Delta x, \Delta y, \Delta z | s_t) \tag{15}$$

$$\mathcal{Z}(\emptyset, \emptyset, \cdot) = 1. \tag{16}$$

6) Reward Function $\mathcal{R}^{a_t}_{s_t, l_t}$: The objective of the intrusion prevention use case is to maintain service on the infrastructure while, at the same time, preventing a possible intrusion. Therefore, we define the reward function to give the maximal reward if the defender maintains service until the intrusion starts and then prevents the intrusion by taking L stop actions.

The reward per time-step $\mathcal{R}^{a_t}_{s_t, l_t}$ is parameterized by the reward that the defender receives for stopping an intrusion ($R_{st} = 50$), the reward for maintaining service ($R_{sla} = 1$), and the loss of being intruded ($R_{int} = -10$):

$$\mathcal{R}_{\emptyset, 0} = 0 \tag{17}$$

$$\mathcal{R}^{S}_{s_t, l_t} = s_t R_{st} / 4 l_t \;\; s_t \in \{0, 1\} \tag{18}$$

$$\mathcal{R}^{C}_{s_t,l_t} = R_{sla} + s_t R_{int}/L \; s_t \in \{0,1\} \tag{19}$$

Eq. (17) states that the reward in the terminal state is zero. Eq. (18) indicates that each stop incurs a cost by interrupting service and possibly a reward if it affects an ongoing intrusion. Lastly, Eq. (19) states that the defender receives a positive reward for maintaining service and a loss for each time-step that it is under intrusion. (Remark: the reward function can equivalently be stated to give a cumulative reward upon transitioning to the terminal state and zero reward otherwise [16].)

7) 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. It follows from Eqs. (11)–(14) that $\mathbb{E}_{\pi_l}[T_\emptyset] < \infty$ for any policy π_l that is guaranteed to use L stops as $t \to \infty$. Further, since the expected time of intrusion $\mathbb{E}[I_t]$ is finite and the continue reward is negative when $t > I_t$ (Eqs. (17)–(19)), the optimal stopping times $\tau_1^*, \ldots, \tau_L^*$ exist. (Remark: it is also possible to define $T_\emptyset = \infty$ and let $\emptyset$ be an infinitely absorbing state.)

8) Policy Space Π_l and Objective Function J: As the POMDP is stationary and the time horizon $T_\emptyset$ is not predetermined, it is sufficient to consider stationary policies. Further, since the POMDP is finite, an optimal deterministic policy exists [16], [17]. Despite this, we consider stochastic policies to enable smooth optimization. Specifically, we consider the space of stationary stochastic policies Π_l where $\pi_l \in \Pi_l$ is a policy $\pi_l : \mathcal{B} \to \Delta(\mathcal{A})$, which depends on $l \in \{1, \ldots, L\}$.

An *optimal* policy $\pi_l^* \in \Pi_l$ maximizes the expected discounted cumulative reward over the time horizon $T_\emptyset$:

$$J(\pi_l) = \mathbb{E}_{\pi_l}\left[\sum_{t=1}^{T_\emptyset} \gamma^{t-1} \mathcal{R}^{a_t}_{s_t,l_t}\right] \tag{20}$$

$$\pi_l^* \in \arg\max_{\pi_l \in \Pi_l} J(\pi_l) \tag{21}$$

We set the discount factor to be $\gamma = 1$. (The objective in Eq. (20) is upper bounded when $\gamma = 1$ since $\mathbb{E}_{\pi_l}[T_\emptyset]$ is finite for any policy $\pi_l \in \Pi_l$ that is guaranteed to use L stops as $t \to \infty$, which is true for any optimal policy (see Lemma 1 in Appendix A).)

Eq. (20) defines an optimization problem which reflects the objective of our use case. In the following section, we state structural properties of an optimal policy that solves this problem.

B. Threshold Properties of an Optimal Policy

A policy that solves the multiple stopping problem is a solution to Eqs. (20), (21). We know from the theory of dynamic programming that this policy satisfies the Bellman equation formulated in terms of the belief state (Eq. (10)) [17], [43].

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., $\mathcal{B} = [0, 1]$.

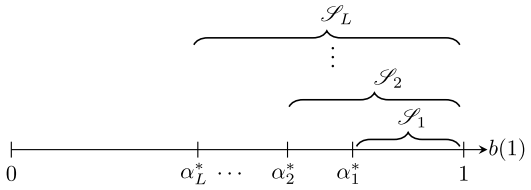

Fig. 6. Illustration of Theorem 1: there exist L thresholds $\alpha_1^* \geq \alpha_2^*, \ldots, \geq \alpha_L^* \in \mathcal{B}$ and an optimal threshold policy π_l^* that satisfies Eqs. (22)-(24).

We partition $\mathcal{B}$ into two sets—the stopping set $\mathscr{S}_l = \{b(1) \in [0,1] : \pi_l^*(b(1)) = S\}$, which contains the belief states where it is optimal to *stop*, and the continuation set $\mathscr{C}_l = \{b(1) \in [0,1] : \pi_l^*(b(1)) = C\}$, 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 [17], [33], [34], we obtain the following structural result for an optimal policy.

Theorem 1: Given the POMDP in Section IV-A, let L denote the number of stop actions, $f_{XYZ|s}$ the conditional 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)

$$\mathscr{S}_{l-1} \subseteq \mathscr{S}_l \quad l \in \{1, \ldots L\} \tag{22}$$

(B) If $L = 1$, there exists a value $\alpha^* \in [0,1]$ and an optimal policy π_L^* that satisfies:

$$\pi_L^*(b(1)) = S \iff b(1) \geq \alpha^* \tag{23}$$

(C) If $L \geq 1$ and $f_{XYZ|s}$ is totally positive of order 2 (i.e., TP2), there exist L values $\alpha_1^* \geq \alpha_2^* \geq \cdots \geq \alpha_L^* \in [0,1]$ and an optimal policy π_l^* that satisfies:

$$\pi_l^*(b(1)) = S \iff b(1) \geq \alpha_l^* \quad l \in \{1, \ldots, L\}. \tag{24}$$

Proof: See Appendix A. ∎

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 $L \geq 1$, an additional condition applies: the probability matrix of $f_{XYZ|s}$ must be TP2 (all second order minors must be non-negative) [17, Definition 10.2.1, p. 223], [62]. This condition is satisfied for example if $f_{XYZ|s}$ is stochastically monotone in s.

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 [11], [16]. This is obvious in the case of threshold policies. Second, the complexity of computing or learning an optimal policy can be reduced by exploiting structural properties [17], [36]. In the following section, we describe a reinforcement learning algorithm that exploits the structural result in Theorem 1.

C. Our Reinforcement Learning Algorithm: T-SPSA

Theorem 1 states that under given assumptions and given $L \geq 1$ stop actions, there exists an optimal policy which uses L thresholds $\alpha_1^* \geq \alpha_2^*, \cdots, \geq \alpha_L^* \in [0,1]$. We present an algorithm, which we call T-SPSA, that computes these thresholds through reinforcement learning.

We parameterize π_l with a vector $\theta \in \mathbb{R}^L$. The component θ_l of θ relates to the threshold with $l \in \{1, \ldots L\}$ stops remaining. T-SPSA updates θ through stochastic gradient ascent with the following gradient [63]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{\theta,l}} \left[\sum_{t=1}^{T_\emptyset} \nabla_\theta \log \pi_{\theta,l}(a_t|b_t(1)) \sum_{\tau=t}^{T_\emptyset} \mathcal{R}_{b_\tau, l_t}^{a_\tau} \right] \quad (25)$$

To ensure differentiability, we define $\pi_{\theta,l}$ to be a smooth stochastic policy that approximates a threshold policy:

$$\pi_{\theta,l}(S|b(1)) = \left(1 + \left(\frac{b(1)(1-\sigma(\theta_l))}{\sigma(\theta_l)(1-b(1))}\right)^{-20}\right)^{-1} \quad (26)$$

where $\sigma(\cdot)$ is the sigmoid function and $\sigma(\theta_1)$, $\sigma(\theta_2)$, ..., $\sigma(\theta_L) \in [0,1]$ are the L thresholds.

We learn the threshold vector θ through simulation of the POMDP as follows. First, we initialize $\theta_{(1)} \in \mathbb{R}^L$ randomly. Second, for each iteration $n \in \{1, 2, \ldots\}$ of T-SPSA, we perturb $\theta_{(n)}$ to obtain $\theta_{(n)} + c_n \Delta_n$ and $\theta_{(n)} - c_n \Delta_n$, where $c_n \in \mathbb{R}$ and $\Delta_n \in \mathbb{R}^L$. Then, we run two POMDP episodes where the defender takes actions according to the two perturbed threshold vectors (Eq. (26)). We then use the obtained episode outcomes $\hat{J}(\theta_{(n)} + c_n\Delta_n)$ and $\hat{J}(\theta_{(n)} - c_n\Delta_n)$ to estimate the gradient in Eq. (25) using the Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimator [64]:

$$\left(\hat{\nabla}_{\theta_{(n)}} J(\theta_{(n)})\right)_i = \frac{\hat{J}(\theta_{(n)} + c_n\Delta_n) - \hat{J}(\theta_{(n)} - c_n\Delta_n)}{2c_n(\Delta_n)_i} \quad (27)$$

where $i \in \{1, \ldots, L\}$ is the component index of the gradient, $c_n = \frac{c}{n^\lambda}$ is the perturbation size and c and λ are hyperparameters.

The perturbation vector Δ_n is defined as:

$$(\Delta_n)_i = \begin{cases} +1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases} \quad (28)$$

Next, we use the estimated gradient and the stochastic approximation algorithm [53] to update the vector of thresholds to maximize $J(\theta)$ (Eq. (20)):

$$\theta_{(n+1)} = \theta_{(n)} + a_n \hat{\nabla}_{\theta_{(n)}} J\left(\theta_{(n)}\right) \quad (29)$$

where $a_n = \frac{a}{(n+A)^\epsilon}$ is the step size and A and ϵ are hyperparameters [65].

This process of running two episodes and updating the threshold vector continues until it has sufficiently converged. The described algorithm, T-SPSA, converges to a *local* maximum of $J(\theta)$ with probability one under standard conditions [64]. For this reason, we run the algorithm several times with different initial conditions. We list the pseudocode of T-SPSA in Appendix D and give its hyperparameters in Appendix B. Our Python implementation of T-SPSA is available at: [66].

TABLE I
EMULATED CLIENT POPULATION; EACH CLIENT INTERACTS WITH APPLICATION SERVERS USING A SET OF NETWORK FUNCTIONS

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

V. EMULATING THE TARGET INFRASTRUCTURE TO INSTANTIATE THE SIMULATION AND TO EVALUATE THE LEARNED POLICIES

To simulate episodes of the POMDP and to compute the belief state 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.

A. Emulating the Target Infrastructure

The emulation system executes on a cluster of machines that runs a virtualization layer provided by Docker [67] containers and virtual links. It implements network isolation and traffic shaping on the containers using network namespaces and the NetEm module in the Linux kernel [68]. Resource constraints of the containers, e.g., CPU and memory constraints, are enforced using cgroups.

The configuration of the emulated infrastructure is given by the topology in Fig. 1 and the configuration in Appendix C. 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.

We emulate internal connections between servers in the infrastructure as full-duplex loss-less connections with bit capacities of 1000 Mbit/s in both directions and emulate external connections between the gateway and the client population and the attacker as full-duplex connections with bit capacities of 100 Mbit/s with 0.1% packet loss in normal operation and random bursts of 1% packet loss.

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

The emulation evolves in time-steps of length 30s. 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 II. The *first* stop revokes all user certificates and recovers user accounts compromised by the attacker. The *second* and *third* stops update

TABLE II
DEFENDER STOP COMMANDS IN THE EMULATION

$L - l_t$	*Action*	*Command in the Emulation*
0	Revoke user certificates	`openssl ca -revoke <certificates>`
1	Blacklist IPs	`iptables -A INPUT -s <ip> -j DROP`
2	Block gateway	`iptables -A INPUT -i eth0 -j DROP`

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 III, 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 the reconnaissance command that they use and the number of stops L required to prevent the attack (see Table IV).

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 revokes the user certificates.

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 revoking the user certificates.

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. For the evaluation reported in this paper we collected measurements from 21000 time-steps of 30 seconds each.

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 state s_t, we obtain the conditional distribution $\hat{f}_{XYZ|s_t}$.

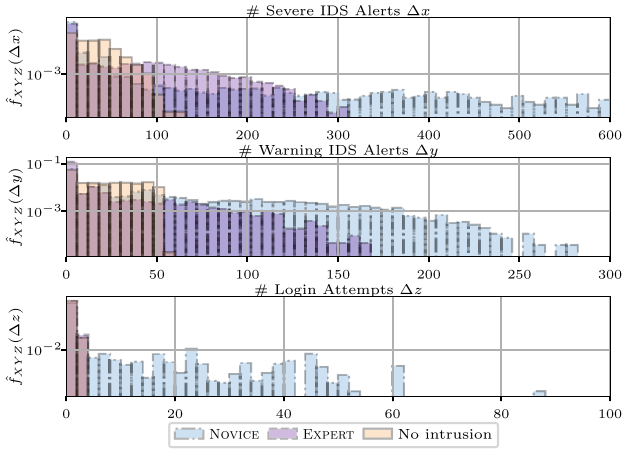

Fig. 7. Empirical distributions of severe IDS alerts Δx (top row), warning IDS alerts Δy (middle row), and login attempts Δz (bottom row) generated during time-steps of intrusions by different attackers as well as during time-steps when no intrusion occurs.

Fig. 7 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. From these empirical distributions, we note that the assumption that the observation distribution is TP2 in Theorem 1.C is reasonable.

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 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 $a_t \sim \pi_{\theta,l}(\cdot|b_t)$. If the action is stop ($a_t = 1$) and $l_t = 1$, the episode ends. Otherwise, the number of remaining stop actions is updated: $l_{t+1} = l_t - a_t$. Further, if an intrusion is in progress, the system executes an attacker action following Table III. It then updates the state $s_t \rightarrow s_{t+1}$ and samples $\Delta x_{t+1}, \Delta y_{t+1}, \Delta z_{t+1}$ from the empirical distribution $\hat{f}_{XYZ|s_{t+1}}$. (The activities of the clients are not simulated but are captured by $\hat{f}_{XYZ}$.) The simulation then computes the belief b_{t+1} using Eq. (3) and computes the defender reward r_{t+1} using Eqs. (17)–(19). (Note that the exact reward can be computed during training and evaluation of policies but not when the policies are deployed in the target infrastructure as it depends on the hidden state.) The sequence of time-steps continues until the defender performs the final stop, after which the episode ends. If the attacker sequence in Table III

TABLE III
ATTACKER ACTIONS IN THE EMULATION

Time-steps t	NOVICEATTACKER	EXPERIENCEDATTACKER	EXPERTATTACKER
1-$I_t \sim Ge(0.01)$	(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}), RECON_3, CVE-2015-1427 on N_{25}
$I_t + 11$-$I_t + 14$	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}	login(N_{25}), backdoor(N_{25}), RECON_3, CVE-2017-7494 exploit on N_{27}
$I_t + 15$-$I_t + 16$		login(N_{18}), backdoor(N_{18})	login(N_{27}), backdoor(N_{27})
$I_t + 17$-$I_t + 19$		RECON_2, CVE-2015-1427 on N_{25}, login(N_{25})	

TABLE IV
NUMBER OF STOPS REQUIRED TO PREVENT THE ATTACKER L AND RECONNAISSANCE COMMANDS OF THE ATTACKER PROFILES

Attacker	L	Reconnaissance
NOVICEATTACKER	1	TCP/UDP scan
EXPERIENCEDATTACKER	2	ICMP ping scan
EXPERTATTACKER	3	ICMP ping scan

is completed before the defender performs the final stop, the sequence is restarted.

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 II and Table III). Further, the defender observations in the emulation system are not sampled but are obtained through reading log files and metrics of the emulated infrastructure. 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 B. 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 [66].

A. Evaluation Process

We train three defender policies against the NOVICE, EXPERIENCED and EXPERT attacker until convergence. For each attacker we run 10,000 training episodes to estimate an optimal defender policy using the method described in Section IV-C. After each episode we evaluate the current defender policy.

To evaluate a defender 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 10,000 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 4 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 policies learned through T-SPSA with three baseline policies. The first baseline prescribes the stop action whenever an IDS alert occurs, i.e., whenever $(\Delta x + \Delta y) \geq 1$. The second baseline is obtained by configuring the Snort IDS as an Intrusion Prevention System (IPS) which drops network traffic following its internal recommendation system (see Appendix C for the Snort configuration). To calculate the reward, we define 100 dropped IP packets of the Snort IPS to be a stop action of the defender. Lastly, the third baseline is an ideal policy which presumes knowledge of the exact intrusion time and performs all stop actions at exactly that time.

We evaluate our algorithm, T-SPSA, by comparing it with three baseline algorithms: Proximal Policy Optimization (PPO) [51], Heuristic Search Value Iteration (HSVI) [69], and Shiryaev's algorithm [70]. PPO is a state-of-the-art reinforcement learning algorithm, HSVI is a state-of-the-art dynamic programming algorithm for POMDPs, and Shiryaev's algorithm is an optimal algorithm for change detection. The main difference between T-SPSA and the first two baselines (PPO and HSVI) is that T-SPSA exploits the threshold structure expressed in Theorem 1 and the main difference in comparison with Shiryaev's algorithm is that T-SPSA learns L thresholds whereas Shiryaev's algorithm uses a single

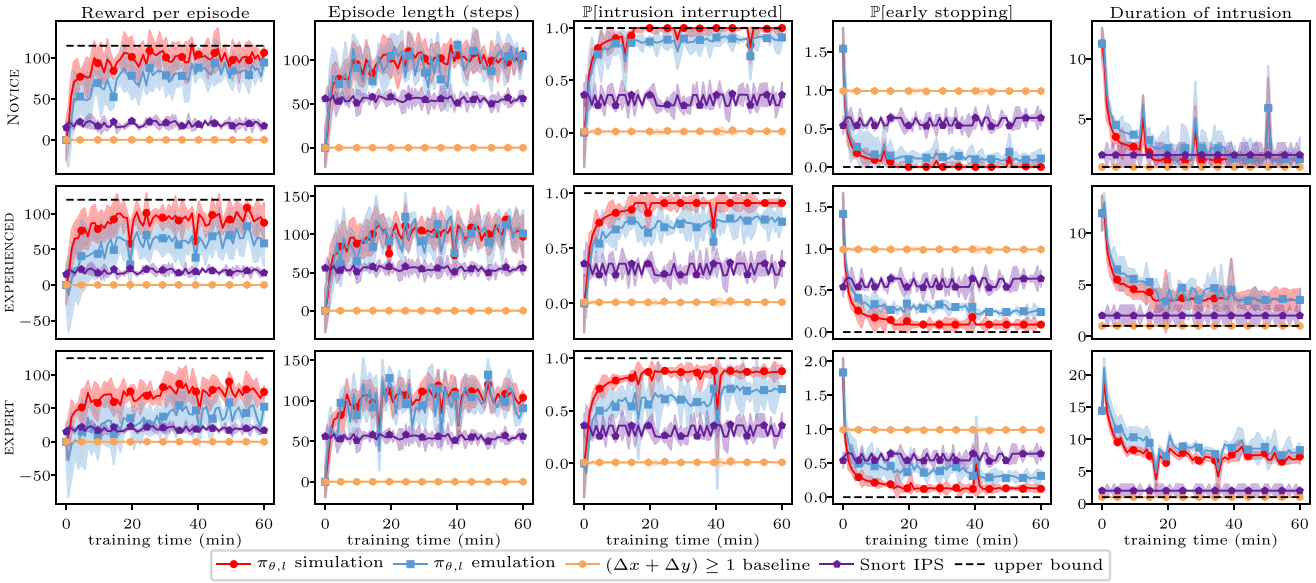

Fig. 8. Learning curves obtained during training of T-SPSA; red curves show simulation results and blue curves show emulation results; the purple, orange, and black curves relate to baseline policies; the rows from top to bottom relate to: NOVICEATTACKER, EXPERIENCEDATTACKER, and EXPERTATTACKER; the 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 Lth stop action; the curves show the mean and 95% confidence interval for five training runs with different random seeds.

predefined threshold. We set this threshold to 0.75 based on a hyperparameter search (see Appendix B).

B. Learning Intrusion Prevention Policies

Fig. 8 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 purple and orange curves give the performance of the Snort IPS baseline and the baseline policy that mandates a stop action whenever an IDS alert occurs, respectively. 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. 8 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. 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. 8) and that the empirical probability of stopping before the intrusion starts is close to 0 (second rightmost column of Fig. 8). This suggests that the learned policies are close to optimal. We also observe that all learned policies do significantly better than the Snort IPS baseline and the baseline that stops whenever an IDS alert occurs (leftmost column in Fig. 8).

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. 8 that the learned policies do better against NOVICEATTACKER than 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. 8). 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. 9 shows a comparison between our reinforcement learning algorithm (T-SPSA) and the three baseline algorithms in the simulation system. We observe in Fig. 9 that both T-SPSA and PPO converge to close approximations of an optimal policy within an hour of training whereas HSVI does not converge within the measured time. The slow convergence of HSVI manifests the intractability of using dynamic programming to compute policies in large POMDPs [39]. We also see in Fig. 9 that T-SPSA converges significantly faster than PPO. This is expected since T-SPSA considers a smaller space of policies than PPO. Finally, we also note in Fig. 9 that T-SPSA outperforms Shiryaev's algorithm, which demonstrates the benefit of using L thresholds instead of a single threshold.

VII. RELATED WORK

Traditional approaches to intrusion prevention use packet inspection and static rules for detection of intrusions and selection of response actions [1], [71], [72]. Their main drawback lies in the need for domain experts to configure the rule sets. As a consequence, much effort has been devoted to developing methods for finding security policies in an automatic way. This

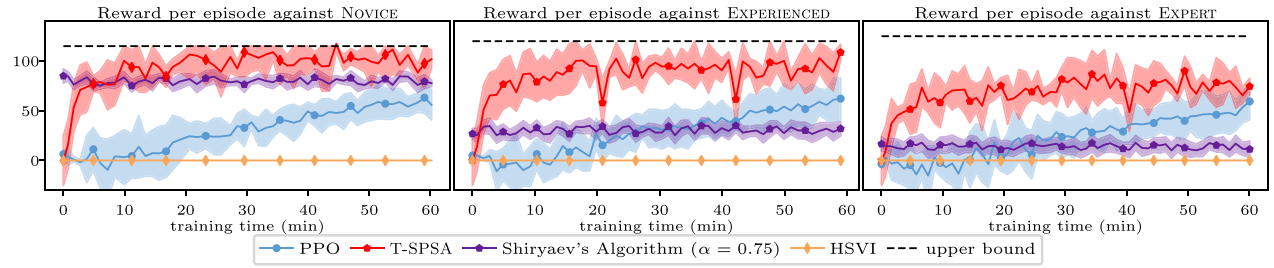

Fig. 9. Comparison between T-SPSA and three baseline algorithms; all curves show simulation results; red curves relate to T-SPSA; blue curves relate to PPO; orange curves relate to HSVI; purple curves relate to Shiryaev's algorithm with threshold $\alpha = 0.75$; the columns from left to right relate to: NOVICEATTACKER, EXPERIENCEDATTACKER, and EXPERTATTACKER; all curves show the mean and 95% confidence interval for five training runs with different random seeds.

research uses concepts and methods from various areas, most notably from anomaly detection (see example [73]), change-point detection (see example [37]), statistical learning (see examples [74], [75], [76]), control theory (see survey [6]), game theory (see textbooks [10], [77], [78], [79]), artificial intelligence (see survey [80]), dynamic programming (see example [5]), reinforcement learning (see surveys [81], [82]), evolutionary methods (see example [7]), and attack graphs (see example [83]).

While the research reported in this paper is informed by all the above works, we limit the following discussion to prior work that centers around finding security policies through reinforcement learning, a topic area that has grown considerably in recent years. Three seminal papers: [84], [85], and [86], published in 2000, 2005, and 2008, respectively, analyze intrusion prevention use cases and evaluate traditional reinforcement learning algorithms for this task. These papers have inspired much follow-up research, e.g., on studying *deep* reinforcement learning algorithms for intrusion prevention [12], [13], [25] and studying new use cases, such as defense against jamming attacks [87], mitigation of denial of service attacks [88], [89], defense against advanced persistent threats [90], placement of honeypots [91], botnet detection [92], [93], detection of flip attacks [94], detection of network traffic anomalies [95], greybox fuzzing [96], and defense against topology attacks [97].

Among the recent works that use reinforcement learning to find security policies, many focus on intrusion prevention use cases similar to the one we discuss in this paper [12], [13], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [38], [41]. These works use a variety of models, including MDPs [20], [22], [24], [25], [29], Markov games [12], [18], [23], and POMDPs [13], [26], [38], as well as various reinforcement learning algorithms, including Q-learning [18], [20], [22], SARSA [38], PPO [12], [13], hierarchical reinforcement learning [24], DQN [25], Thompson sampling [26], MuZero [23], NFQ [27], DDQN [29], and DDPG [28].

This paper differs from the works referenced above in two main ways. First, we formulate the intrusion prevention problem as a multiple stopping problem. The other works formulate the problem as solving 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-B).

Problem formulations based on optimal stopping theory can be found in prior research on change detection [13], [37], [38], [70], [94], [98]. Compared to these papers, our approach is more general by allowing multiple stop actions within an episode. Another difference is that we model intrusion *prevention* rather than intrusion *detection*. Further, compared with traditional change detection algorithms, e.g., CUSUM [98] and Shiryaev's algorithm [70], our algorithm *learns* thresholds and does not assume them to be preconfigured.

Second, our solution method to find effective policies for intrusion prevention includes using an emulation system in addition to a simulation system. The advantage of our method compared to the simulation-only approaches [12], [13], [18], [19], [20], [21], [22], [24], [25], [26], [38], [41] is that the parameters of our simulation system are determined by measurements from an emulation system instead of being chosen by a human expert. 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 would provide.

Some prior works on reinforcement learning for intrusion prevention that make use of emulation are: [23], [27], [28], and [29]. They emulate software-defined networks based on Mininet [99]. The main differences between these efforts and the work described in this paper are: (1) we develop our own emulation system which allows for experiments with a large variety of exploits; (2) we focus on a different intrusion prevention use case; (3) we do not assume that the defender has perfect observability; and (4), we use an underlying theoretical framework to formalize the use case, derive structural properties of optimal policies, and test these properties in an emulation system.

Finally, [100] and [101] describe ongoing efforts in building emulation platforms for reinforcement learning, which resemble our emulation system. In contrast to these papers, our emulation system has been built to investigate the specific use case of intrusion prevention and forms an integral part of our general solution method (see Fig. 2).

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 (Figs. 8-9). We also demonstrated that our reinforcement learning algorithm (T-SPSA), which takes advantage of the threshold structure (Theorem 1), outperforms state-of-the-art algorithms on our use case.

We make assumptions in this paper that limit the practical applicability of the results: the attacker follows a static policy, and the defender learns only the times of taking defensive actions but not the types of actions. Therefore, the question arises whether our approach can be extended so that (1) the attacker can pursue a wide range of realistic policies and (2) the defender learns optimal policies that express not only when defensive actions needs to be taken but also the specific measure to be executed.

Addressing these points is part of our research agenda. The dynamic attacker can be studied using a game-theoretic extension of the introduced framework. The theory tells us that an optimal solution can be found through self-play in a similar manner as described in this paper, but further work is needed to show that such a solution is feasible in practice. Scenarios involving several attackers can also be studied in this context.

We also plan to extend the defender model to include the selection of defensive actions. One possible approach is to learn two orthogonal policies: a policy that decides when to take a defensive action and another policy that decides which action to take.

APPENDIX A
PROOF OF THEOREM 1

Given the POMDP introduced in Section IV-A, let L denote the number of stop actions, f_{XYZ} the observation distribution, $\mathcal{B} = [0, 1]$ the belief space (see Section IV-B), $b(1)$ the belief state, $\mathscr{S}_l$ the stopping set, and $\mathscr{C}_l$ the continuation set.

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

Lemma 1: During a POMDP episode, an optimal policy π_L^* prescribes L stop actions.

Proof: The proof follows directly from the definition of the transition probabilities (see Eqs. (11)–(14)) and the reward function (see Eqs. (17)–(19)). ■

Lemma 2: $\mathscr{S}_1$ is a convex subset of $\mathcal{B}$.

Proof: The proof can be found in [13, p. 10, Lemma 3] and in [17, p. 258, Th. 12.2.1]. ■

Lemma 3: $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ is TP2 and $\mathcal{R}^S_{b(1),l_t} - \mathcal{R}^C_{b(1),l_t}$ is increasing in $b(1)$ for $l_t \in \{1, \ldots, L\}$.

Proof: The transition probabilities (see Section IV-A) are given by the following two row-stochastic matrices:

$$\begin{array}{c} \\ 0 \\ 1 \\ \emptyset \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 0 & 1 & \emptyset \end{array} \\ \begin{bmatrix} 0.99 & 0.01 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}, \qquad \begin{array}{c} \\ 0 \\ 1 \\ \emptyset \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 0 & 1 & \emptyset \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}, \tag{30}$$

The left matrix corresponds to the transition probabilities when $a_t = C$, or, when $a_t = S$ and $l_t > 1$. The right matrix represents the transition probabilities when $a_t = S$ and $l_t = 1$. To show that $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ is TP2, it is sufficient to show that all $\binom{3}{2}^2$ second order minors of both matrices are non-negative. The second-order minors of the first matrix are $M_{1,2} = M_{1,3} = M_{2,3} = M_{3,1} = M_{3,2} = 0$, $M_{1,1} = 1$, $M_{2,1} = 0.01$, $M_{2,2} = M_{3,3} = 0.99$, where $M_{i,j}$ denotes the determinant of the submatrix formed by deleting the ith row and jth column. For the second matrix all second order minors are zero. Hence, $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ is TP2.

$\mathcal{R}^S_{b(1),l_t} - \mathcal{R}^C_{b(1),l_t}$ is expanded to:

$$\mathcal{R}^S_{b(1),l_t} - \mathcal{R}^C_{b(1),l_t} = b(1)\left(\frac{50}{4l_t} + 10/L\right) - 1 \tag{31}$$

which is increasing in $b(1)$ for all $l_t \in \{1, \ldots, L\}$. ■

Lemma 4: Given two beliefs $b'(1) \geq b(1)$ and two observations $o \geq \bar{o}$, if $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ and $f_{XYZ|s}$ are TP2, then the following holds for any $a \in \mathcal{A}$, $k \in \mathcal{O}$, and $l_t \in \{1, \ldots, L\}$:

1) $b'^{,o}_a(1) \geq b^o_a(1)$
2) $\mathbb{P}[o \geq k|b', a] \geq \mathbb{P}[o \geq k|b, a]$
3) $b^o_a(1) \geq b^{\bar{o}}_a(1)$

where $b'^{,o}_a(1)$ and $b^o_a(1)$ denote the beliefs updated with Eq. (3) after taking action $a \in \mathcal{A}$ and observing $o \in \mathcal{O}$.

Proof: The proof is published in [17, Th. 10.3.1, pp. 225, 238]. (Remark: in the referenced proof, the monotone likelihood ratio (MLR) order is considered; in our case $|\mathcal{S} \setminus \emptyset| = 2$, hence the MLR order reduces to the natural order $b'(1) \geq b(1)$.) ■

We now use Lemmas 1-4 to prove Theorem 1. The proof uses the value iteration algorithm to establish structural properties of V_l^* and π_l^* [16], [17].

Let V_l^k, $\mathscr{S}_l^k$, and $\mathscr{C}_l^k$, denote the value function, the stopping set, and the continuation set at iteration k of the value iteration algorithm, respectively. Then, $\lim_{k\to\infty} V_l^k = V_l^*, \lim_{k\to\infty} \mathscr{S}_l^k = \mathscr{S}_l$, and $\lim_{k\to\infty} \mathscr{C}_l^k = \mathscr{C}_l$ [16], [17]. We define $V_l^0(b(1)) = 0$ for all $b(1) \in [0, 1]$ and $l \in \{1, \ldots, L\}$.

Proof of Theorem 1.A: The proof has originally been published in [34, Propositions 4.5-4.8, pp. 437–441]. It is also available in a more accessible form in [33, Th. 1.C, Th. 8, pp. 389–397]. We give our own version of the proof since the referenced proofs assume zero reward for the continue action and assume that rewards are independent of l.

If $b(1) \in \mathscr{S}_{l-1}$, the Bellman equation and the fact that $\mathbb{P}[o|a, b] = \mathbb{P}[o|b] = \mathbb{P}^o_{b(1)}$ for all $a \in \mathcal{A}$ and $o \neq \emptyset$ (see Eq. (15)) implies that:

$$\mathcal{R}^S_{b(1),l-1} - \mathcal{R}^C_{b(1),l-1} \tag{32}$$

$$+ \sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \big(V^*_{l-2}(b^o(1)) - V^*_{l-1}(b^o(1)) \big) \geq 0$$

We show that $b(1) \in \mathscr{S}_l$ follows from the above inequality.

Let $W^k_l(b(1)) = \mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^k_{l-1}(b(1)) - V^k_l(b(1))$. To show that $b(1) \in \mathscr{S}_{l-1} \implies b(1) \in \mathscr{S}_l$, it is sufficient to show that $W^k_l(b(1))$ is non-decreasing in l for all $k \geq 0$. We proceed to show this statement by mathematical induction.

For iteration $k = 0$ of value iteration, $W^0_l(b(1)) = V^0_l(b(1)) - V^0_{l-1}(b(1)) = 0$, which is trivially non-decreasing in l. Assume by induction that $W^{k-1}_l(b(1))$ is non-decreasing in l for iterations $k-1, k-2, \ldots, 1$. To show that $W^k_l(b(1))$ is non-decreasing in l also for iteration k, we show that $W^k_l(b(1)) - W^k_{l-1}(b(1)) \geq 0$.

There are four cases to consider:

1) If $b(1) \in \mathscr{S}^k_l \cap \mathscr{S}^k_{l-1} \cap \mathscr{S}^k_{l-2}$, then:

$$W^k_l(b(1)) - W^k_{l-1}(b(1)) = \sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \Big(W^{k-1}_{l-1}(b^o(1)) - W^{k-1}_{l-2}(b^o(1)) \Big) \quad (33)$$

which is non-negative by the induction assumption.

2) If $b(1) \in \mathscr{S}^k_l \cap \mathscr{S}^k_{l-1} \cap \mathscr{C}^k_{l-2}$, then:

$$W^k_l(b(1)) - W^k_{l-1}(b(1)) = \mathcal{R}^S_{b(1),l-1} - \mathcal{R}^C_{b(1),l-1} + \sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \Big(V^{k-1}_{l-2}(b^o(1)) - V^{k-1}_{l-1}(b^o(1)) \Big) \quad (34)$$

which is non-negative because $b(1) \in \mathscr{S}^k_{l-1}$ (it is implied by Eq. (10)).

3) If $b(1) \in \mathscr{S}^k_l \cap \mathscr{C}^k_{l-1} \cap \mathscr{C}^k_{l-2}$, then:

$$W^k_l(b(1)) - W^k_{l-1}(b(1)) = \mathcal{R}^C_{b(1),l-1} - \mathcal{R}^S_{b(1),l-1} + \sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \Big(V^{k-1}_{l-1}(b^o(1)) - V^{k-1}_{l-2}(b^o(1)) \Big) \quad (35)$$

which is non-negative because $b(1) \in \mathscr{C}^k_{l-1}$ (it is implied by Eq. (10)).

4) If $b(1) \in \mathscr{C}^k_l \cap \mathscr{C}^k_{l-1} \cap \mathscr{C}^k_{l-2}$, then:

$$W^k_l(b(1)) - W^k_{l-1}(b(1)) = \quad (36)$$

$$\sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \Big(W^{k-1}_l(b^o(1)) - W^{k-1}_{l-1}(b^o(1)) \Big) \quad (37)$$

which is non-negative by the induction assumption.

The other cases, e.g., $b(1) \in \mathscr{C}^k_l \cap \mathscr{C}^k_{l-1} \cap \mathscr{S}^k_{l-2}$, can be discarded due to the induction assumption. Hence, $W^k_l(b(1))$ is non-decreasing in l for all $k \geq 0$.

Since the left-hand side of Eq. (32) is non-decreasing in l it follows that if Eq. (32) holds, i.e., if $b(1) \in \mathscr{S}_{l-1}$, then $b(1) \in \mathscr{S}_l$. ∎

Proof of Theorem 1.B: The proof follows the chain of reasoning in [17, Corollary 12.2.2, p. 258].

Using Lemma 2, we know that the stopping set $\mathscr{S}_1$ 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 $\beta^* = 1$.

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

$$\pi^*_1(1) \in \arg\max_{\{S,C\}} \Bigg[\underbrace{50 + V^*_0(\emptyset)}_{a=S}, \quad \underbrace{-9 + \sum_{o \in \mathcal{O}} \mathcal{Z}(o, 1, C)\, V^*_1(b^o_C(1))}_{a=C} \Bigg] \quad (38)$$

As $L = 1$, it follows from Lemma 1 that an optimal policy prescribes one stop action during a POMDP episode and that the intrusion is prevented after the first stop. Hence, $V^*_0(\emptyset) = \mathcal{R}_{\emptyset,0} = 0$. Moreover, since $s = 1$ is an absorbing state until the stop, it follows from the definition of b^o_C (Eq. (3)) that $b^o_C(1) = 1$ for all $o \in \mathcal{O} \setminus \emptyset$. Thus, since $V^*_1(1) \leq 50$ (see Eqs. (17)–(19)), we get:

$$\pi^*_1(1) \in \arg\max_{\{S,C\}} \Bigg[\underbrace{50}_{a=S}, \underbrace{-9 + V^*_1(1)}_{a=C} \Bigg] = S \quad (39)$$

This means that $\beta^* = 1$ and therefore $\mathscr{S}_1 = [\alpha^*, 1]$. ∎

Corollary 1: If $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ and $f_{XYZ|s}$ are TP2, the stopping set $\mathscr{S}_l$ is connected for all $l \in \{1, \ldots, L\}$.

Proof: We adapt the proof from [33, Th. 1.B, pp. 389–397] to our model. In contrast to the referenced proof, our model includes non-zero rewards for the continue action and $|\mathcal{S} \setminus \emptyset| = 2$.

If $b(1) \in \mathscr{S}_l$, the Bellman equation and the fact that $\mathbb{P}[o|a, b] = \mathbb{P}[o|b] = \mathbb{P}^o_{b(1)}$ for all $a \in \mathcal{A}$ and $o \neq \emptyset$ (see Eq. (15)) implies that:

$$\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + \sum_{o \in \mathcal{O}} \mathbb{P}^o_{b(1)} \big(V^*_{l-1}(b^o(1)) - V^*_l(b^o(1)) \big) \geq 0 \quad (40)$$

We show that the above inequality implies that $b'(1) \in \mathscr{S}_l$ for any $b'(1) \geq b(1)$, which means that $\mathscr{S}_l$ is connected.

Since $\mathcal{B} = [0, 1]$, the beliefs are totally ordered according to the standard ordering. Further, since $f_{XYZ|s}$ is TP2 by assumption and $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ is TP2 by Lemma 3, $b^o(1)$ is weakly increasing in both $b(1)$ and $o \in \mathcal{O}$. Further, $\mathbb{P}[o \geq k|b', a] \geq \mathbb{P}[o \geq k|b, a]$ for any $k \in \mathcal{O}$ (Lemma 4). Thus, since $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ (Theorem 1.A) and $\mathscr{S}_1 = [\alpha^*_1, 1]$ (Theorem 1.B), it is sufficient to show that $\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^*_{l-1}(b(1)) - V^*_l(b(1))$ is weakly increasing in $b(1)$. We proceed to show this by mathematical induction.

For iteration $k = 0$ of value iteration, $\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^0_{l-1}(b^o(1)) - V^0_l(b^o(1)) = \mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l}$ which is weakly increasing in $b(1)$ by Lemma 3. Assume by induction that the expression is weakly increasing in $b(1)$ for iterations $k-1, k-2, \ldots, 1$. We show that this implies that the induction assumption holds also for iteration k.

Since $\mathscr{S}_{l-1} \subseteq \mathscr{S}_l$ (Theorem 1.A) and $\mathscr{S}_1 = [\alpha^*_1, 1]$ (Theorem 1.B), there are three cases to consider

TABLE V
HYPERPARAMETERS OF THE POMDP AND THE ALGORITHMS USED FOR EVALUATION

Hyperparameters for the POMDP	*Values*
γ, $\Delta x_{max}, \Delta y_{max}, \Delta z_{max}$	1, $6 \cdot 10^2$, $3 \cdot 10^2$, 10^2
Hyperparameters for T-SPSA	*Values*
$c, \epsilon, \lambda, A, a$	1, 0.101, 0.602, 100, 1
Hyperparameters for PPO	*Values*
lr α, batch, # layers, # neurons, clip ϵ	10^{-4}, $4 \cdot 10^3 t$, 2, 32, 0.2
GAE λ, ent-coef, activation	0.95, 10^{-4}, ReLU
Hyperparameters for HSVI	*Values*
ϵ	0.01
Hyperparameter for Shiryaev's algorithm	*Value*
α	0.75

1) If $b(1) \in \mathscr{S}_l \cap \mathscr{S}_{l-1}$, then:

$$\begin{aligned}&\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^k_{l-1}(b(1)) - V^k_l(b(1)) \\ &= \mathcal{R}^S_{b(1),l-1} - \mathcal{R}^C_{b(1),l-1} \\ &\quad + \sum_{o\in\mathcal{O}} \mathbb{P}^o_{b(1)} \Big(V^{k-1}_{l-2}(b^o(1)) - V^{k-1}_{l-1}(b^o(1)) \Big) \end{aligned} \tag{41}$$

which is weakly increasing in $b(1)$ by the induction assumption.

2) If $b(1) \in \mathscr{S}_l \cap \mathscr{C}_{l-1}$, then:

$$\begin{aligned}&\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^k_{l-1}(b(1)) - V^k_l(b(1)) \\ &= \sum_{o\in\mathcal{O}} \mathbb{P}^o_{b(1)} \Big(V^{k-1}_{l-1}(b^o(1)) - V^{k-1}_{l-1}(b^o(1)) \Big) = 0 \end{aligned} \tag{42}$$

which is trivially weakly increasing in $b(1)$.

3) If $b(1) \in \mathscr{C}_l \cap \mathscr{C}_{l-1}$, then:

$$\begin{aligned}&\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} + V^k_{l-1}(b(1)) - V^k_l(b(1)) \\ &= \mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l} \\ &\quad + \sum_{o\in\mathcal{O}} \mathbb{P}^o_{b(1)} \Big(V^{k-1}_{l-1}(b^o(1)) - V^{k-1}_{l}(b^o(1)) \Big) \end{aligned} \tag{43}$$

which is weakly increasing in $b(1)$ by the induction assumption.

■

Proof of Theorem 1.C: Since $\mathcal{B} = [0, 1]$ (see Section IV-B), $f_{XYZ|s}$ is TP2 by assumption, $\mathcal{P}^{a_t}_{s_t,s_{t+1},l_t}$ is TP2 by Lemma 3, and $\mathcal{R}^S_{b(1),l} - \mathcal{R}^C_{b(1),l}$ is increasing in $b(1)$ (Lemma 3), it follows from Corollary 1 that $\mathscr{S}_l$ is a connected subset of [0, 1] for $l \in \{1, \dots, L\}$. Further, from Theorem 1.B we know that $\mathscr{S}_1 = [\alpha^*_1, 1]$. Then, because $\mathscr{S}_l \subseteq \mathscr{S}_{l+1}$ for $l \in \{1, \dots, L-1\}$ (Theorem 1.A), we conclude that $\mathscr{S}_l = [\alpha^*_l, 1]$ for $l \in \{1, \dots, L\}$ and that $\alpha^*_l \geq \alpha^*_{l+1}$ for $l \in \{1, \dots, L-1\}$. ■

APPENDIX B
HYPERPARAMETERS

The hyperparameters used for the evaluation are listed in Table V and were obtained through grid search.

TABLE VI
CONFIGURATION OF THE TARGET INFRASTRUCTURE (FIG. 1)

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:-

Algorithm 1 T-SPSA

Input
$\mathcal{M}_\mathcal{P}, \theta_{(1)} \in \mathbb{R}^L$: the POMDP, initial L thresholds
N: number of iterations
$a, c, \lambda, A, \epsilon$: scalar coefficients
Output
$\theta_{(N+1)}$: learned threshold vector

procedure T-SPSA($\mathcal{M}_\mathcal{P}$, $\theta_{(1)}$, N, a, c, λ, A, ϵ)
  for $n \in \{1, \dots, N\}$ **do**
    $a_n \leftarrow \frac{a}{(n+A)^\epsilon}$, $c_n \leftarrow \frac{c}{n^\lambda}$
    for $i \in \{1, \dots, L\}$ **do**
      $(\Delta_n)_i \sim \mathcal{U}(\{-1, 1\})$
    end for
    $R_{high} \sim \hat{J}(\theta_{(n)} + c_n \Delta_n)$
    $R_{low} \sim \hat{J}(\theta_{(n)} - c_n \Delta_n)$
    for $i \in \{1, \dots, L\}$ **do**
      $(\hat{\nabla}_{\theta_{(n)}} J(\theta_{(n)}))_i \leftarrow \frac{R_{high} - R_{low}}{2 c_n (\Delta_n)_i}$
    end for
    $\theta_{(n+1)} \leftarrow \theta_{(n)} + a_n \hat{\nabla}_{\theta_{(n)}} J(\theta_{(n)})$
  end for
  return $\theta_{(N+1)}$
end procedure

APPENDIX C
CONFIGURATION OF THE INFRASTRUCTURE IN FIG. 1

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

APPENDIX D
THE T-SPSA ALGORITHM

Algorithm 1 contains the pseudocode of T-SPSA.

ACKNOWLEDGMENT

The authors would like to thank Pontus Johnson for his useful input to this research and Vikram Krishnamurthy for helpful discussions. The authors are also grateful to Forough Shahab Samani and Xiaoxuan Wang for their constructive comments on a draft of this paper.

REFERENCES

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

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

[3] E. Zouave, M. Bruce, K. Colde, M. Jaitner, I. Rodhe, and T. Gustafsson, "Artificially intelligent cyberattacks," Swedish Defence Res. Agency, Stockholm, Sweden, Rep. FOI-R–4947–SE, Mar. 2020.

[4] P. Johnson, R. Lagerström, and M. Ekstedt, "A meta language for threat modeling and attack simulations," in *Proc. 13th Int. Conf. Avail. Rel. Security*, New York, NY, USA, 2018, pp. 1–8.

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

[6] E. Miehling, M. Rasouli, and D. Teneketzis, *Control-Theoretic Approaches to Cyber-Security*, Cham, Switzerland: Springer, 2019.

[7] R. Bronfman-Nadas, N. Zincir-Heywood, and J. T. Jacobs, "An artificial arms race: Could it improve mobile malware detectors?" in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, 2018, pp. 1–8.

[8] N. Wagner *et al.*, "Towards automated cyber decision support: A case study on network segmentation for security," in *Proc. IEEE Symp. Series Comput. Intell. (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 *Proc. ACM Workshop Inf. Sharing Collaborative Security*, 2016, pp. 49–56.

[10] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed. New York, NY, USA: Cambridge Univ. 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, Switzerland: 2019, pp. 439–458.

[12] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and self-play," in *Proc. Int. Conf. Netw. Service Manage. (CNSM)*, Izmir, Turkey, 2020, pp. 1–9.

[13] K. Hammar and R. Stadler, "Learning intrusion prevention policies through optimal stopping," in *Proc. Int. Conf. Netw. Service Manage. (CNSM)*, Izmir, Turkey, 2021, pp. 509–517.

[14] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, 1957.

[15] A. Wald, *Sequential Analysis*. New York, NY, USA: Wiley, 1947.

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: Wiley, 1994.

[17] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[18] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Drugan, and M. Wiering, "Adversarial reinforcement learning in a cyber security simulation," in *Proc. ICAART*, 2017, pp. 1–8.

[19] J. Schwartz, H. Kurniawati, and E. El-Mahassni, "POMDP + information-decay: Incorporating defender's behaviour in autonomous penetration testing," in *Proc. Int. Conf. Autom. Plan. Schedul.*, vol. 30, Jun. 2020, pp. 235–243.

[20] F. M. Zennaro and L. Erdodi, "Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular Q-learning," 2020, *arxiv:2005.12632*.

[21] W. Blum. "Gamifying Machine Learning for Stronger Security and AI Models." 2021. [Online]. Available: https://www.microsoft.com/security/blog/2021/04/08/gamifying-machine-learning-for-stronger-security-and-ai-models/

[22] A. Ridley, "Machine learning for autonomous cyber defense," *Next Wave*, vol. 22, no. 1, pp. 1–43, 2018.

[23] J. Gabirondo-López, J. Egaña, J. Miguel-Alonso, and R. O. Urrutia, "Towards autonomous defense of SDN networks using MuZero based intelligent agents," *IEEE Access*, vol. 9, pp. 107184–107199, 2021.

[24] K. Tran *et al.*, "Deep hierarchical reinforcement agents for automated penetration testing," 2021, *arXiv:2109.06449*.

[25] R. Gangupantulu *et al.*, "Using cyber terrain in reinforcement learning for penetration testing," 2021, *arxiv:2108.07124*.

[26] Z. Hu, M. Zhu, and P. Liu, "Adaptive cyber defense against multi-stage attacks using learning-based pomdp," *ACM Trans. Privacy Security*, vol. 24, no. 1, pp. 1–26, Feb. 2021.

[27] I. Akbari, E. Tahoun, M. A. Salahuddin, N. Limam, and R. Boutaba, "ATMoS: Autonomous threat mitigation in SDN using reinforcement learning," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2020, pp. 1–9.

[28] Y. Liu, M. Dong, K. Ota, J. Li, and J. Wu, "Deep reinforcement learning based smart mitigation of ddos flooding in software-defined networks," in *Proc. IEEE 23rd Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, 2018, pp. 1–6.

[29] T. V. Phan and T. Bauschert, "DeepAir: Deep reinforcement learning for adaptive intrusion response in software-defined networks," *IEEE Trans. Netw. Service Manag.*, early access, Mar. 2022, doi: 10.1109/TNSM.2022.3158468.

[30] G. Sofronov, J. Keith, and D. Kroese, "An optimal sequential procedure for a buying-selling problem with independent observations," *J. Appl. Probab.*, vol. 43, no. 2, pp. 454–462, 2006.

[31] R. Carmona and N. Touzi, "Optimal multiple stopping and valuation of swing options," *Math. Finan.*, vol. 18, pp. 239–268, Apr. 2008.

[32] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions," in *Proc. 16th Annu. ACM-SIAM Symp. Discr. Algorithms*, 2005, pp. 630–631.

[33] 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, Sep. 2018.

[34] T. Nakai, "The problem of optimal stopping in a partially observable Markov chain," *J. Optim. Theory Appl.*, vol. 45, no. 3, pp. 425–442, Mar. 1985.

[35] J. du Toit and G. Peskir, "Selling a stock at the ultimate maximum," *Ann. Appl. Probab.*, vol. 19, no. 3, pp. 983–1014, Jun. 2009.

[36] A. Roy, V. S. Borkar, A. Karandikar, and P. Chaporkar, "Online reinforcement learning of optimal threshold policies for Markov decision processes," 2019, *arxiv:1912.10325*.

[37] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blaďek, and H. Kim, "Detection of intrusions in information systems by sequential change-point methods," *Stat. Methodol.*, vol. 3, no. 3, pp. 252–293, 2006.

[38] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5174–5185, Sep. 2019.

[39] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Math. Oper. Res.*, vol. 12, pp. 441–450, Aug. 1987.

[40] K. Hammar and R. Stadler. *A Software Framework for Building Self-Learning Security Systems*. (Apr. 14, 2022). [Online Video]. Available: https://www.youtube.com/watch?v=18P7MjPKNDg

[41] M. Zhu, Z. Hu, and P. Liu, "Reinforcement learning algorithms for adaptive cyber defense against heartbleed," in *Proc. 1st ACM Workshop Moving Target Defense*, 2014, pp. 51–58.

[42] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.

[43] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, 1998.

[44] K. J. Åström, "Optimal control of Markov processes with incomplete state information," *J. Math. Anal. Appl.*, vol. 10, no. 1, pp. 174–205, 1965.

[45] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Sci., 1996.

[46] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[47] R. Bellman, *Dynamic Programming*. Mineola, NY, USA: Dover Publ., 1957.

[48] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Oper. Res.*, vol. 26, no. 2, pp. 282–304, 1978.

[49] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, 3rd ed. Belmont, MA, USA: Athena Sci., 2005.

[50] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, London, U.K., 1989.

[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[52] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, vol. 6. Red Hook, NY, USA: Curran Assoc., 1994.

[53] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, 1951.

[54] A. N. Shirayev, *Optimal Stopping Rules*. Berlin, Germany: Springer-Verlag, 2007.

[55] G. Peskir and A. Shiryaev, *Optimal Stopping and Free-Boundary Problems* (Lectures in Mathematics. ETH Zürich). Basel, Switzerland: Springer, 2006.

[56] Y. Chow, H. Robbins, and D. Siegmund, *Great Expectations: The Theory of Optimal Stopping*. Boston, MA, USA: Houghton Mifflin, 1971.

[57] S. M. Ross, *Introduction to Stochastic Dynamic Programming: Probability and Mathematical*. San Diego, CA, USA: Academic, 1983.

[58] J. Bather, *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. Sussex, NJ, USA: Wiley, 2000.

[59] H. V. Poor and O. Hadjiliadis, *Quickest Detection*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[60] M. Rabi and K. H. Johansson, "Event-triggered strategies for industrial control over wireless networks," in *Proc. 4th Annu. Int. Conf. Wireless Internet*, 2008, pp. 1–7.
[61] J. L. Snell, "Applications of martingale system theorems," *Trans. Amer. Math. Soc.*, vol. 73, no. 2, pp. 293–312, 1952.
[62] S. Karlin, "Total positivity, absorption probabilities and applications," *Trans. Amer. Math. Soc.*, vol. 111, no. 1, pp. 105–108, 1964.
[63] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 1999.
[64] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992.
[65] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 3, pp. 817–823, Jul. 1998.
[66] K. Hammar and R. Stadler. "Gym-Optimal-Intrusion-Response." 2021. [Online]. Available: https://github.com/Limmen/gym-optimal-intrusion-response
[67] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, 2014.
[68] S. Hemminger, "Network emulation with NetEm," in *Proc. Linux Conf.*, 2005, pp. 1–9.
[69] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, Arlington, TX, USA, 2004, pp. 520–527.
[70] A. N. Shiryaev, "On optimum methods in quickest detection problems," *Theory Probab. Appl.*, vol. 8, no. 1, pp. 22–46, 1963.
[71] M. Roesch, "Snort—Lightweight intrusion detection for networks," in *Proc. 13th USENIX Conf. Syst. Admin.*, 1999, pp. 229–238.
[72] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.
[73] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 34–47, Mar. 2017.
[74] C. J. Fung, J. Zhang, and R. Boutaba, "Effective acquaintance management based on bayesian learning for distributed intrusion detection networks," *IEEE Trans. Netw. Service Manag.*, vol. 9, no. 3, pp. 320–332, Sep. 2012.
[75] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Dirichlet-based trust management for effective collaborative intrusion detection networks," *IEEE Trans. Netw. Service Manag.*, vol. 8, no. 2, pp. 79–91, Jun. 2011.
[76] S. Huang *et al.*, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2064–2076, Dec. 2020.
[77] M. Tambe, *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*, 1st ed. New York, NY, USA: Cambridge Univ. Press, 2011.
[78] C. J. Fung and R. Boutaba, *Intrusion Detection Networks—A Key to Collaborative Security*. Boca Raton, FL, USA: CRC Press, 2013.
[79] L. Buttyan and J.-P. Hubaux, *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. New York, NY, USA: Cambridge Univ. Press, 2007.
[80] N. Dhir, H. Hoeltgebaum, N. Adams, M. Briers, A. Burke, and P. Jones, "Prospective artificial intelligence approaches for active cyber defence," 2021, *arxiv:2104.09981*.
[81] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," 2019, *arxiv:1906.05799*.
[82] ****** Y. Huang, L. Huang, and Q. Zhu, "Reinforcement learning for feedback-enabled cyber resilience," *Annu. Rev. Control*, to be published.
[83] E. Miehling, M. Rasouli, and D. Teneketzis, "A POMDP approach to the dynamic defense of large-scale cyber networks," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2490–2505, 2018.
[84] J. C. Georgia, "Next generation intrusion detection: Autonomous reinforcement learning of network attacks," in *Proc. 23rd Nat. Inf. Syst. Secuity Conf.*, 2000, pp. 1–12.
[85] X. Xu and T. Xie, "A reinforcement learning approach for host-based intrusion detection using sequences of system calls," in *Advances in Intelligent Computing*. Heidelberg, Germany: Springer, 2005, pp. 995–1003.
[86] A. Servin and D. Kudenko, "Multi-agent reinforcement learning for intrusion detection," in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*. Heidelberg, Germany: Springer, 2008.
[87] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "AFRL: Adaptive federated reinforcement learning for intelligent jamming defense in FANET," *J. Commun. Netw.*, vol. 22, no. 3, pp. 244–258, Jun. 2020.
[88] K. Malialis and D. Kudenko, "Multiagent router throttling: Decentralized coordinated response against DDoS attacks," in *Proc. IAAI*, 2013, pp. 1551–1556.
[89] K. A. Simpson, S. Rogers, and D. P. Pezaros, "Per-host DDoS mitigation by direct-control reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 103–117, Mar. 2020.
[90] B. Ning and L. Xiao, "Defense against advanced persistent threats in smart grids: A reinforcement learning approach," in *Proc. 40th Chin. Control Conf. (CCC)*, 2021, pp. 8598–8603.
[91] L. Huang and Q. Zhu, "Adaptive honeypot engagement through reinforcement learning of semi-Markov decision processes," in *Decision and Game Theory for Security*. Cham, Switzerland: Springer, 2019, pp. 196–216.
[92] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, and K. K. R. Choo, "An efficient reinforcement learning-based Botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102479.
[93] G. Apruzzese *et al.*, "Deep reinforcement adversarial learning against botnet evasion attacks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 1975–1987, Dec. 2020.
[94] H. Liu, Y. Li, J. Mårtensson, L. Xie, and K. H. Johansson, "Reinforcement learning based approach for flip attack detection," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, 2020, pp. 3212–3217.
[95] S. Dong, Y. Xia, and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4197–4212, Dec. 2021.
[96] J. Wang, C. Song, and H. Yin, "Reinforcement learning-based hierarchical seed scheduling for greybox fuzzing," in *Proc. NDSS*, 2021, pp. 1–17.
[97] J. Yan, H. He, X. Zhong, and Y. Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 200–210, 2017.
[98] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, 1954.
[99] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, pp. 1–6.
[100] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "CybORG: A gym for the development of autonomous cyber agents," 2021, *arXiv:2108.09118*.
[101] A. Molina-Markham, C. Miniter, B. Powell, and A. Ridley, "Network environment design for autonomous cyberdefense," 2021, *arxiv:2103.07583*.

Kim Hammar (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering in distributed systems from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Division of Network and Systems Engineering. He has also held engineering positions with Ericsson, Allstate, Logical Clocks AB, MIC Nordic, and Tele 2. His research interests are in the intersection between decision theory, machine learning, and large-scale systems, focusing on networking and security applications.

Rolf Stadler (Senior Member, IEEE) received the M.Sc. degree in mathematics and the Ph.D. degree in computer science from the University of Zurich. He is a Professor with the KTH Royal Institute of Technology, Stockholm, Sweden, and the Head of the Division of Network and Systems Engineering. Before joining KTH in 2001, he held positions with the IBM Zurich Research Laboratory, Columbia University, and ETH Zürich. His group made contributions to real-time monitoring, resource management, and automation for large-scale networked systems. His current interests include data-driven methods for network engineering and management, as well as AI techniques for cybersecurity. He was the Editor-in-Chief of IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT from 2014 to 2017.