A System for Interactive Examination
of Learned Security Policies

Kim Hammar [†‡] and Rolf Stadler[†‡]

[†] Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden
[‡] KTH Center for Cyber Defense and Information Security, Sweden
Email: {kimham, stadler}@kth.se
January 28, 2022

Abstract—We present a system for interactive examination of learned security policies. It allows a user to traverse episodes of Markov decision processes in a controlled manner and to track the actions triggered by security policies. Similar to a software debugger, a user can continue or or halt an episode at any time step and inspect parameters and probability distributions of interest. The system enables insight into the structure of a given policy and in the behavior of a policy in edge cases. We demonstrate the system with a network intrusion use case. We examine the evolution of an IT infrastructure's state and the actions prescribed by security policies while an attack occurs. The policies for the demonstration system have been obtained through a reinforcement learning approach that includes a simulation system where policies are incrementally learned and an emulation system that produces statistics that drive the simulation runs.

Index Terms—Network security, automation, reinforcement learning, Markov decision processes, MDP, POMDP.

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. 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: computation of security policies using dynamic programming and control theory; computation of exploits and corresponding defenses through evolutionary methods; computation of security policies through game-theoretic methods; and use of machine learning techniques to estimate model parameters and policies [2], [3], [4], [5].

A promising direction of recent research is automatically learning security policies through reinforcement learning methods (see survey [2]). This paper builds on our earlier work in this area [3], [4], [5]. In this line of research, the problem of finding a security policy is generally modeled as a Markov decision problem and policies are learned through simulation. Following this approach, the learned policies are evaluated by metrics that are aggregated over a large number of simulation runs. This kind of of evaluation provides a useful, overall, assessment of the policies' performance, but it is incomplete.
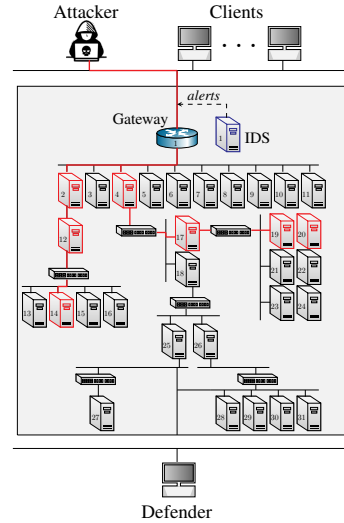

Fig. 1: The IT infrastructure and the actors in the intrusion prevention use case: an attacker, a client population, and a defender.

For instance, it does not explain the structural differences between effective and non-effective policies. Also, it does not give insight into the behavior of policies in edge-cases, which is vital to understand before deploying security policies in practice.

In this demo paper, we describe a system that complements quantitative evaluations of learned security policies by allowing a user to interactively inspect the behavior of policies in different states and against different attackers. We demonstrate that the system enables insight into the structure of a given policy and in the behavior of a policy in edge cases. We also open-source the code [6], explain the implementation, and describe our reinforcement learning approach for finding security policies.

II. THE INTRUSION PREVENTION USE CASE

We demonstrate the system with an *intrusion prevention* use case (see Fig. 1). 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.
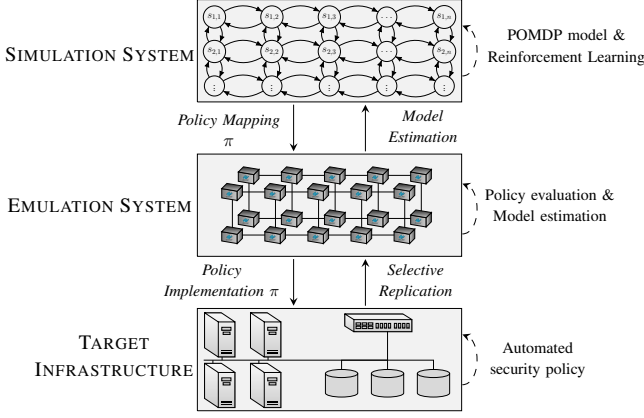
Fig. 2: Our framework for learning and evaluating security policies.

Fig. 3: Architecture of the system for inspecting and examining learned security policies.

The 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 an attacker while, at the same time, providing a service to a client population. 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.

In our demonstration, we assume that the defender can take a fixed number of actions, each of which has a cost. An example of such an action is to update the firewall of the gateway. When deciding on an action, the defender balances two objectives: (*i*) maintaining service to its clients; and (*ii*), keeping a possible attacker out of the infrastructure.

For simulation of the use case and formal analysis, we model it with a Partially Observed Markov Decision Process (POMDP) [5], [4]. The POMDP evolves in discrete time-steps: $t = 1, \ldots, T$. The states $s_t \in \mathcal{S}$ and observations $o_t \in \mathcal{O}$ of the POMDP represent the infrastructure's states and the defender's observations, respectively. The defender policy $\pi_\theta(a_t | h_t)$ determines the actions $a_t \in \mathcal{A}$ that the defender takes after observing the history $h_t = (\rho_1, a_1, o_1, \ldots, a_{t-1}, o_t)$, where ρ_1 is the initial state distribution.

III. Automated Learning of Security Policies

We integrate the system for interactive examination of learned security policies with a reinforcement learning framework for learning security policies that we developed in our previous work [3], [5], [4]. Our framework includes two systems (see Fig. 2): first, we use an *emulation system* where key functional components of the target infrastructure are replicated. This system closely approximates the functionality of the target infrastructure and is used to run attack scenarios and defender responses. These runs produce system metrics and logs that we use to estimate distributions of infrastructure
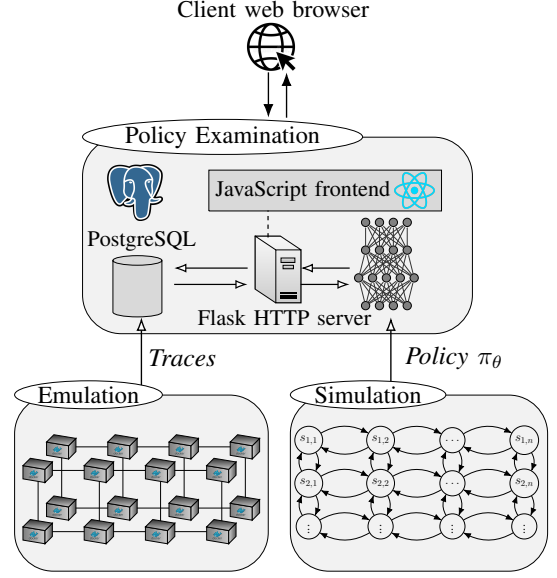
metrics, which we use to instantiate a POMDP. Second, we use a *simulation system* where POMDP episodes are simulated and policies are incrementally learned. After learning policies, they are extracted from the simulation system and evaluated in the emulation system. Using this approach, the emulation system provides the statistics needed to simulate the use case and to evaluate the learned policies, whereas the simulation system automatically learns policies through reinforcement learning in an efficient manner.

IV. The System for Interactive Policy Examination

The architecture of the system for policy examination is shown in Fig. 3. The system provides an interface to inspect learned policies either from emulation traces or from simulating a POMDP. Users access the system through a web browser where they can interactively step through a POMDP episode. At each step, the users can inspect relevant parameters and probability distributions.

V. Implementation

The web interface is implemented in JavaScript using the React framework (see Fig. 3). It makes requests to an HTTP server written in Python based on the Flask framework, which provides a REST API that can be queried to fetch data and to make policy predictions. The data on the server is collected from the emulation system and is stored in a PostgreSQL database. The policy is extracted from the simulation system and is stored in the main memory of the server.

The emulation system executes on a cluster of machines that runs a virtualization layer provided by Docker [7] containers and virtual connections. It implements network isolation and traffic shaping on the containers using network namespaces and the NetEm module in the Linux kernel. Resource constraints of the containers, e.g. CPU and memory constraints,
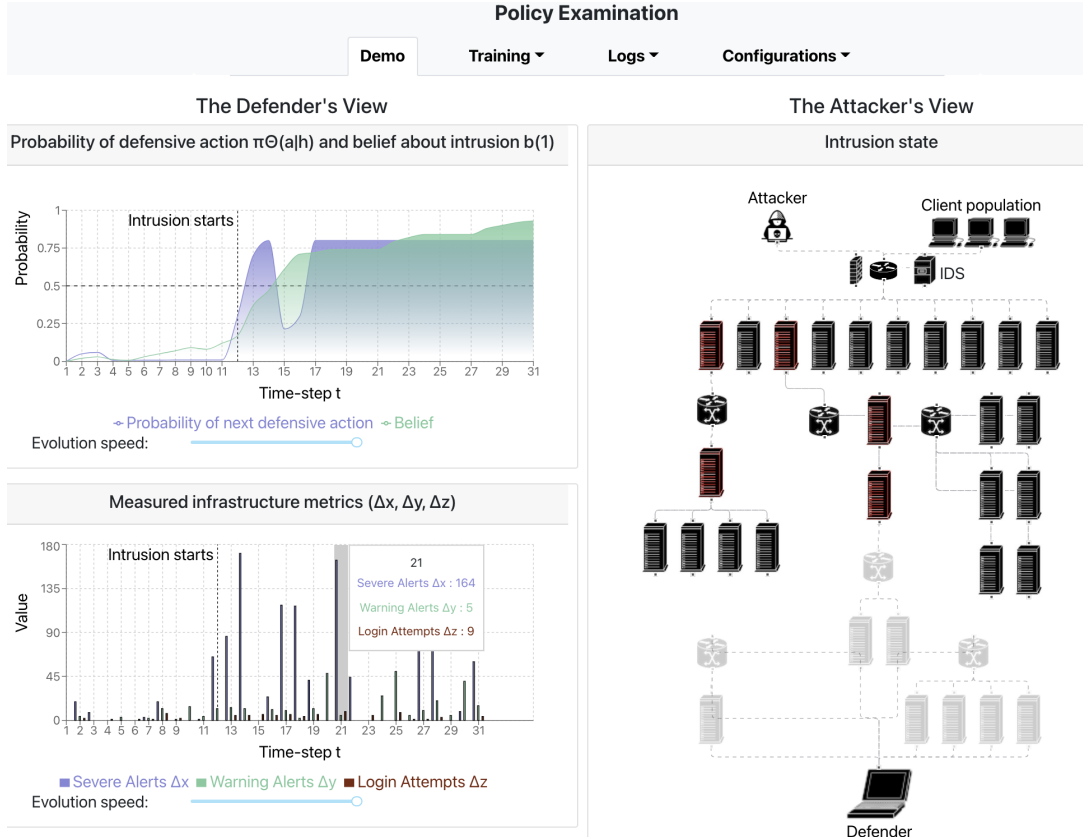
Fig. 4: The demonstration view of the system which allows the user to interactively step through a POMDP episode.

are enforced using cgroups. The software running in the containers replicates important components of the target infrastructure, such as, web servers, databases, and an IDS.

The defender policy $\pi_\theta(a|h)$ is represented by a neural network that takes infrastructure metrics h as input and outputs a probability distribution over actions $a \in \mathcal{A}$.

The simulation system is written in Python, and the policy training is performed using the Proximal Policy Optimization (PPO) algorithm, which we have implemented using the PyTorch auto-differentiation library.

VI. DEMONSTRATION

We demonstrate the system by interactively stepping through a POMDP episode of the intrusion prevention use case. The POMDP is traversed based on traces and a defender policy, which were obtained through the reinforcement learning method described in Section III.

Figure 4 pictures the main user interface of the system. The left part of the interface shows the defender's view and the right part shows the attacker's view. The plot on the upper left shows the defender's belief about the infrastructure's state and the probability that the defender takes an action at each time-step. The plot on the lower left shows the distribution of infrastructure metrics that the defender observes. The graphic on the right shows the attacker's view, depicted as an overlay on the IT-infrastructure's topology.

The demonstration provides insight into the structure of the defender policy and its behavior in edge-cases. We observe correlations among the attacker's actions, the infrastructure metrics, and the actions that the defender policy prescribes. We also examine which actions of the attacker are difficult for the defender to detect or trigger actions by the defender.

For example, we can observe that if the attacker performs reconnaissance by executing a port scan, there is a spike in infrastructure metrics, which causes a defensive action with a high probability. If the attacker uses other means of reconnaissance, e.g. a ping scan, we can observe that the defender will not detect it since no change in infrastructure metrics is generated. Finally, we can discover an edge-case where the defender policy mistakes regular client traffic for an attacker and wrongly takes defensive actions. This happens, for instance, if the attacker is passive for a long time.

REFERENCES

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

[2] N.Dhir *et al.*, "Prospective artificial intelligence approaches for active cyber defence," 2021.

[3] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and Self-Play," in *International Conference on Network and Service Management (CNSM 2020)*, Izmir, Turkey, 2020.

[4] ——, "Learning intrusion prevention policies through optimal stopping," in *International Conference on Network and Service Management (CNSM 2021)*, Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.07160.pdf.

[5] ——, "Intrusion prevention through optimal stopping," 2021, https://arxiv.org/abs/2111.00289.

[6] ——, "automated-intrusion-prevention-dashboard," 2022, https://github.com/Limmen/automated-intrusion-prevention-dashboard.

[7] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.