Learning Security Strategies through Game Play and Optimal Stopping

Kim Hammar†‡ and Rolf Stadler†‡
† Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden
‡ KTH Center for Cyber Defense and Information Security, Sweden
Email: \{kimham, stadler\}@kth.se
May 31, 2022

Abstract—We study automated intrusion prevention using reinforcement learning. Following a novel approach, we formulate the interaction between an attacker and a defender as an optimal stopping game and let attack and defense strategies evolve through reinforcement learning and self-play. The game-theoretic perspective allows us to find defender strategies that are effective against dynamic attackers. The optimal stopping formulation gives us insight into the structure of optimal strategies, which we show to have threshold properties. To obtain the optimal defender strategies, we introduce T-FP, a fictitious self-play algorithm that learns Nash equilibria through stochastic approximation. We show that T-FP outperforms a state-of-the-art algorithm for our use case. Our overall method for learning and evaluating strategies includes two systems: a simulation system where defender strategies are incrementally learned and an emulation system where statistics are produced that drive simulation runs and where learned strategies are evaluated. We conclude that this approach can produce effective defender strategies for a practical IT infrastructure.

Index Terms—Network security, automation, optimal stopping, reinforcement learning, game theory, Markov decision process, Dynkin games, 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 [2]. 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 security strategies. Examples of this research include: automated creation of threat models [3]; computation of defender strategies using dynamic programming and control theory [4, 5]; computation of exploits and corresponding defenses through evolutionary methods [6, 7]; identification of infrastructure vulnerabilities through attack simulations and threat intelligence [8, 9]; computation of defender strategies through game-theoretic methods [10, 11]; and use of machine learning techniques to estimate model parameters and strategies [12, 13, 14].

In this paper, we present a novel approach to automatically learn security strategies for an attacker and a defender. We apply this approach to an intrusion prevention use case, which 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 providing services to a client population. (We use the term “intrusion prevention” as suggested in the literature, e.g. in [1], it means that the attacker is prevented from reaching its goal, rather than prevented from accessing any part of the infrastructure.)

We formulate the use case as an optimal stopping game, i.e. a stochastic game where each player faces an optimal stopping problem [15, 16, 17]. The stopping game formulation enables us to gain insight into the structure of optimal strategies, which we show to have threshold properties. To obtain effective defender strategies, we use reinforcement learning and self-play. Based on the threshold properties of optimal strategies, we design an efficient self-play algorithm that iteratively computes optimal defender strategies against a dynamic attacker.

Our method for learning and evaluating strategies includes building two systems (see Fig. 2). First, we develop 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
The attacker's goal is to intrude on the infrastructure and compromise its servers. To achieve this, the attacker explores the infrastructure through reconnaissance and exploits vulnerabilities while avoiding detection by the defender. The attacker decides when to start an intrusion and may stop the intrusion at any moment. During the intrusion, the attacker follows a pre-defined strategy. When deciding the time to start or stop an intrusion, the attacker considers both the gain of compromising additional servers and the risk of getting detected. The optimal strategy for the attacker is to compromise as many servers as possible without being detected.

The defender continuously monitors the infrastructure through accessing and analyzing IPS alerts and other statistics. It can take a fixed number of defensive actions, each of which has a cost and a chance of preventing an ongoing attack. An example of a defensive action is to drop network traffic that triggers IPS alerts of a certain priority. The defender takes defensive actions in a pre-determined order, starting with the action that has the lowest cost. The final action blocks all external access to the gateway, which disrupts any ongoing intrusion as well as the services to the clients.

When deciding the time for taking a defensive action, the defender balances two objectives: (i) maintain services to its clients; and (ii), prevent a possible intrusion at lowest cost. The optimal strategy for the defender is to monitor the infrastructure and maintain services until the moment when the attacker enters through the gateway, at which time the attack must be prevented at minimal cost through defensive actions. The challenge for the defender is to identify the precise time for this moment.

### III. FORMALIZING 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 providing services to a client population (Fig. 1). The infrastructure includes a set of servers that run the services and an Intrusion Prevention System (IPS) that logs events in real-time. Clients access the services through a public gateway, which also is open to the attacker.

The attacker’s goal is to intrude on the infrastructure and compromise its servers. To achieve this, the attacker explores the infrastructure through reconnaissance and exploits vulnerabilities while avoiding detection by the defender. The attacker

![Simulation System](image)

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

runs produce system measurements and logs from which we estimate distributions of infrastructure metrics. We then use the estimated distributions to instantiate the simulation model. Second, we build a *simulation system* where game episodes are simulated and strategies are incrementally learned. Learned strategies are then extracted from the simulation system and evaluated in the emulation system. (A video demonstration of our software framework that implements the emulation and simulation systems is available at [18].)

We make three contributions with this paper. First, we formulate intrusion prevention as an optimal stopping game. This novel formulation allows us a) to derive structural properties of optimal strategies using results from optimal stopping theory; and b) to find defender strategies that are effective against attackers with dynamic strategies. We thus address a limitation of many related works that consider static attackers only [19], [20], [21], [22], [23], [14], [24], [25]. Second, we propose T-FP, an efficient reinforcement learning algorithm that exploits structural properties of optimal stopping strategies and outperforms a state-of-the-art algorithm for our use case. Third, we provide evaluation results from an emulated infrastructure. This addresses a common drawback in related research, which we provide evaluation results from an emulated infrastructure.

We model the use case as a partially observed stochastic game. The attacker wins the game when it can intrude on infrastructure and hide its actions from the defender. In contrast, the defender wins the game when it manages to prevent an intrusion. We model this as a zero-sum game, which means that the gain of one player equals the loss of the other player.

The attacker and the defender have different observability in the game. The defender observes alerts from an Intrusion Prevention System (IPS) but has no certainty about the presence of an attacker or the state of a possible intrusion. The attacker, on the other hand, is assumed to have complete observability. It has access to all the information that the defender has access to, as well as the defender’s past actions. The asymmetric observability requires the defender to find strategies that are effective against any attacker, including attackers with inside information about its monitoring capabilities.

The reward function of the game encodes the defender’s objective. An optimal defender strategy maximizes reward when facing a worst-case attacker. Similarly, an optimal attacker strategy minimizes reward when facing a worst-case defender. In game-theoretical terms, this means that a pair of optimal strategies is a Nash equilibrium [38]. Formally, we model the game as a finite and zero-sum Partially Observed Stochastic Game (POSG) with one-sided partial observability: $\Gamma = (\mathcal{N}, \mathcal{S}, (A_i)_{i \in \mathcal{N}}, \mathcal{T}, (R_i)_{i \in \mathcal{N}}, \gamma, \rho_1, T, (O_i)_{i \in \mathcal{N}}, \mathcal{Z})$. It is a discrete-time game that starts at time $t = 1$. In the following, we describe the components of the game, its evolution, and the objectives of the players.
Players $\mathcal{N}$. The game has two players: player 1 is the defender and player 2 is the attacker. Hence, $\mathcal{N} = \{1, 2\}$.

State space $\mathcal{S}$. The game has three states: $s_t = 0$ if no intrusion is occurring, $s_t = 1$ if an intrusion is ongoing, and $s_t = 0$ if the game has ended. Hence, $\mathcal{S} = \{0, 1, \emptyset\}$. The initial state is $s_1 = 0$ and the initial state distribution is the degenerate distribution $\rho_1(0) = 1$.

Action spaces $\mathcal{A}_i$. Each player $i \in \mathcal{N}$ can invoke two actions: “stop” ($S$) and “continue” ($C$). The action spaces are thus $\mathcal{A}_1 = \mathcal{A}_2 = \{S, C\}$. $S$ results in a change of state and $C$ means that the game remains in the same state. We encode $S$ with 1 and $C$ with 0.

The attacker can invoke the stop action two times: the first time to start the intrusion and the second time to stop it. The defender can invoke the stop action $L \geq 1$ times. Each stop of the defender can be interpreted as a defensive action against a possible intrusion. The number of stops remaining of the defender at time-step $t$ is known to both the attacker and the defender and is denoted by $l_t \in \{1, \ldots, L\}$.

At each time-step, the attacker and the defender simultaneously choose an action each: $a_t = (a_t^{(1)}, a_t^{(2)})$, where $a_t^{(i)} \in \mathcal{A}_i$.

Observation space $\mathcal{O}$. The attacker has complete information and knows the game state, the defender’s actions, and the defender’s observations. The defender, however, only sees the observations $o_t \in \mathcal{O}$, where $\mathcal{O}$ is a discrete set. (In our use case, $o_t$ relates to the number of IPS alerts during time-step $t$.)

Both players have perfect recall, meaning that they remember their respective play history. The history of the defender at time-step $t$ is $h_t^{(1)} = (\rho_1, a_1^{(1)}, a_1, \ldots, a_t^{(1)}, a_t)$ and the history of the attacker is $h_t^{(2)} = (\rho_1, a_1^{(1)}, a_2^{(1)}, a_1, s_1, \ldots, a_{t-1}^{(1)}, a_t^{(2)}, o_t, s_t)$.

Belief space $\mathcal{B}$. Based on its history $h_t^{(1)}$, the defender forms a belief about $s_t$, which is expressed in the belief state $b_t(s_t) = \mathbb{P}[s_t|h_t^{(1)}] \in \mathcal{B}$. Since $s_t \in \{0, 1\}$ and $b_t(0) = 1 - b_t(1)$, $b_t$ is determined by $b_t(1)$. Hence, we can model $\mathcal{B}$ as $[0, 1]$. 

Transition probabilities $\mathcal{T}$. At each time-step $t$, a state transition occurs. The probabilities of the state transitions are defined by $\mathcal{T}_t(s_{t+1}|s_t, (a_t^{(1)}, a_t^{(2)})) = \mathbb{P}[s_{t+1}|s_t, (a_t^{(1)}, a_t^{(2)})]$: $\mathcal{T}_{t}^{>0}(0, 0, (S, C)) = \mathcal{T}_{t}^{>0}(0, 0, (C, C)) = 1$ (1) $\mathcal{T}_{t}^{>0}(1, 1, (\cdot, C)) = \mathcal{T}_{t}^{>0}(1, 1, (C, C)) = 1 - \phi_t$ (2) $\mathcal{T}_{t}^{>0}(1, 0, (\cdot, S)) = \mathcal{T}_{t}^{>0}(1, 0, (C, S)) = 1$ (3) $\mathcal{T}_{t}^{>0}(0, 1, (\cdot, C)) = \mathcal{T}_{t}^{>0}(0, 1, (C, C)) = \phi_t$ (4) $\mathcal{T}_{t}(0, 0, (\cdot, S)) = \mathcal{T}_{t}(0, 0, (\cdot, C)) = \mathcal{T}_{t}(0, 1, (\cdot, S)) = 1$ (5)

All other state transitions have probability 0.

Eqs. (1)-(5) define the probabilities of the recurrent state transitions $0 \rightarrow 0$ and $1 \rightarrow 1$. The game stays in state 0 with probability 1 if the attacker selects the action $C$ and $l_t - a_t^{(1)} > 0$. Similarly, the game stays in state 1 with probability $1 - \phi_t$ if the attacker chooses action $C$ and $l_t - a_t^{(1)} > 0$. $\phi_t$ is a parameter of the use case that defines the probability that the intrusion is prevented, which increases with each stop action that the defender takes.

Fig. 3: State transition diagram of a game episode: each disk represents a state; an arrow represents a state transition; a label indicates the conditions for the state transition; a game episode starts in state $s_1 = 0$ with $l_1 = L$ and ends in state $s_T = \emptyset$.

The evolution of the game can be described with the state transition diagram in Fig. 3. The figure describes a game episode, which starts at $t = 1$ and ends at $t = T$. The time horizon $T$ is a random variable that depends on both players’ strategies and takes values in the set $\{t = 2, 3, \ldots, \infty\}$.

Reward function $\mathcal{R}_t$. At time-step $t$, the defender receives the reward $r_t = \mathcal{R}_t(s_t, (a_t^{(1)}, a_t^{(2)}))$ and the attacker receives the reward $-r_t$. The reward function is parameterized by the reward that the defender receives for stopping an intrusion ($R_{int}$), the defender’s cost of taking a defensive action ($R_{\text{cost}} < 0$), and its loss when being intruded ($R_{\text{cost}} < 0$):

$$\mathcal{R}_t(0, \cdot, \cdot) = 0, \quad \mathcal{R}_t(1, \cdot, S) = 0$$ (6) $$\mathcal{R}_t(0, C, \cdot) = 0$$ (7) $$\mathcal{R}_t(0, (S, \cdot)) = R_{\text{cost}}/l_t$$ (8) $$\mathcal{R}_t(1, C, \cdot) = R_{\text{cost}}/l_t$$ (9) $$\mathcal{R}_t(1, (C, \cdot)) = R_{\text{int}}$$ (10)

Eq. 6 states that the reward is zero in the terminal state and when the attacker ends an intrusion. Eq. 7 states that the defender incurs no cost when it is not under attack and not taking defensive actions. Eq. 8 indicates that the defender incurs a cost when stopping if no intrusion is ongoing, which is decreasing with the number of stops remaining $l_t$. Eq. 9 states that the defender receives a reward that is decreasing in $l_t$ when taking a stop action that affects an ongoing intrusion. Lastly, Eq. 10 indicates that the defender receives a loss for each time-step when under intrusion.

Observation function $\mathcal{Z}$. At time-step $t$, $o_t \in \mathcal{O}$ is drawn from a random variable $O$ whose distribution $f_O$ depends on the current state $s_t$. We define $\mathcal{Z}(o_t, s_t, (a_t^{(1)}, (a_t^{(2)})) = $
\[ P[a_t | s_t, (a_{t-1}^{(1)}, a_{t-1}^{(2)})] \] as follows:
\[
\begin{align*}
Z(a_t, 0, \cdot) &= f_0(a_t | 0) \quad (11) \\
Z(a_t, 1, \cdot) &= f_0(a_t | 1) \quad (12) \\
Z(0, 0, \cdot) &= 1 \quad (13)
\end{align*}
\]

**Belief update.** At time-step \( t \), the belief state \( b_t \) is updated as follows:
\[
\begin{align*}
b_{t+1}(s_{t+1}) &= C \sum_{s_t \in S} \sum_{a_t^{(1)} \in A_t} \sum_{a_t^{(2)} \in A_t} b_t(s_t) \pi_{2,t}(a_t^{(1)} | s_t, b_t). \\
Z(a_{t+1}, s_{t+1}, (a_{t}^{(1)}, a_{t}^{(2)})) &= \mathcal{T}(s_{t+1}, s_t, (a_{t}^{(1)}, a_{t}^{(2)})) \quad (14)
\end{align*}
\]

where \( C = 1/P[a_t | s_t, (a_{t-1}^{(1)}, a_{t-1}^{(2)})] \) is a normalizing factor that makes \( b_{t+1} \) sum to 1. The initial belief is \( b_t(0) = 1 \).

**Player strategies** \( \pi_{i,t} \). A strategy of the defender is a function \( \pi_{1,t} : \Pi_1 : B \rightarrow \Delta(A_t) \). Analogously, a strategy of the attacker is a function \( \pi_{2,t} : \Pi_2 : S \times B \rightarrow \Delta(A_t) \). \( \Delta(A_t) \) denotes the set of probability distributions over \( A_t \). \( \Pi_1 \) denotes the strategy space of player \( i \), and \( \pi_{i,-t} \) denotes the strategy of player \( j \in N \setminus \{i\} \). For both players, a strategy is dependent on \( l \) but independent of \( t \), i.e., strategies are stationary. If \( \pi_{i,t} \) always maps on to an action with probability 1, it is called pure, otherwise it is called mixed.

**Objective functions** \( J_i \). The goal of the defender is to maximize the expected discounted cumulative reward over the time horizon \( T \). Similarly, the goal of the attacker is to minimize the same quantity. Therefore, the objective functions \( J_1 \) and \( J_2 \) are:
\[
\begin{align*}
J_1(\pi_{1,t}, \pi_{2,t}) &= \mathbb{E}_{(\pi_{1,t}, \pi_{2,t})} \left[ \sum_{t=1}^{T} \gamma^{t-1} R_{i,t}(s_t, a_t) \right] \quad (15) \\
J_2(\pi_{1,t}, \pi_{2,t}) &= -J_1(\pi_{1,t}, \pi_{2,t}) \quad (16)
\end{align*}
\]

where \( \gamma \in [0, 1) \) is the discount factor.

**Best response strategies** \( \tilde{\pi}_{1,t} \). A defender strategy \( \tilde{\pi}_{1,t} \in B_1(\pi_{1,t}) \) is called a best response against \( \pi_{2,t} \in \Pi_2 \) if it maximizes \( J_1 \) (Eq. 17). Similarly, an attacker strategy \( \tilde{\pi}_{2,t} \in B_2(\pi_{1,t}) \) is called a best response against \( \pi_{1,t} \in \Pi_1 \) if it minimizes \( J_1 \) (Eq. 18).
\[
\begin{align*}
B_1(\pi_{2,t}) &= \arg \max_{\pi_{1,t} \in \Pi_1} J_1(\pi_{1,t}, \pi_{2,t}) \quad (17) \\
B_2(\pi_{1,t}) &= \arg \min_{\pi_{2,t} \in \Pi_2} J_1(\pi_{1,t}, \pi_{2,t}) \quad (18)
\end{align*}
\]

**Optimal strategies** \( \pi_{1,t}^* \). An optimal defender strategy \( \pi_{1,t}^* \) is a best response strategy against any attacker strategy that minimizes \( J_1 \). Similarly, an optimal attacker strategy \( \pi_{2,t}^* \) is a best response against any defender strategy that maximizes \( J_1 \). Hence, when both players follow optimal strategies, they play best response strategies against each other:
\[
(\pi_{1,t}^*, \pi_{2,t}^*) \in B_1(\pi_{2,t}^*) \times B_2(\pi_{1,t}^*) \quad (19)
\]

This means that no player has an incentive to change its strategy and that \( (\pi_{1,t}^*, \pi_{2,t}^*) \) is a Nash equilibrium \[38\].
Based on [47, 48, 49, 12, 43], we formulate Theorem 1 which contains an existence result for equilibria and a structural result for best response strategies in the game.

**Theorem 1.** Given the one-sided POSG $\Gamma$ in Section III with $L \geq 1$, the following holds.

(A) $\Gamma$ has a mixed Nash equilibrium. Further, $\Gamma$ has a pure Nash equilibrium when $s = 0 \iff b(1) = 0$.

(B) Given any attacker strategy $\pi_2, l \in \Pi_2$, if the probability mass function $f_{O|b}$ is totally positive of order 2 (i.e., TP2 [27 Definition 10.2.1, pp. 223]), there exist values $\tilde{\alpha}_1 \geq \tilde{\alpha}_2 \geq \ldots \geq \tilde{\alpha}_L \in [0, 1]$ and a best response strategy $\hat{\pi}_1, l \in B_1(\pi_2, l)$ of the defender that satisfies:

$$\hat{\pi}_1, l(b(1)) = S \iff b(1) \geq \tilde{\alpha}_l, \quad l \in 1, \ldots, L \quad (26)$$

(C) Given a defender strategy $\pi_1, l \in \Pi_1$, where $\pi_1, l(S|b(1))$ is non-decreasing in $b(1)$ and $\pi_1, l(S|1) = 1$, there exist values $\tilde{\beta}_0, 1, \tilde{\beta}_1, \ldots, \tilde{\beta}_0, L, \tilde{\beta}_1, L \in [0, 1]$ and a best response strategy $\hat{\pi}_2, l \in B_2(\pi_1, l)$ of the attacker that satisfies:

$$\hat{\pi}_2, l(0, b(1)) = C \iff \pi_1, l(S|b(1)) \geq \tilde{\beta}_0, l \quad (27)$$

$$\hat{\pi}_2, l(1, b(1)) = S \iff \pi_1, l(S|b(1)) \geq \tilde{\beta}_1, l \quad (28)$$

for $l \in 1, \ldots, L$.

**Proof.** See Appendices A-C.

Theorem 1 tells us that $\Gamma$ has a mixed Nash equilibrium. It also tells us that, under certain assumptions, the best response strategies have threshold properties (see Fig. 5). In the following, we describe an efficient algorithm that takes advantage of these properties to approximate Nash equilibria of $\Gamma$.

**B. Finding Nash Equilibria through Fictitious Self-Play**

Computing Nash equilibria for a POSG is generally intractable [43]. However, approximate solutions can be obtained through iterative approximation methods. One such method is fictitious self-play, where both players start from random strategies and continuously update their strategies based on the outcomes of played game episodes [50].

Fictitious self-play evolves through a sequence of iteration steps, which is illustrated in Fig. 6. An iteration step includes three procedures. First, player 1 learns a best response strategy against player 2’s current strategy. The roles are then reversed and player 2 learns a best response strategy against player 1’s current strategy. Lastly, the iteration step is completed by having each player adopt a new strategy, which is determined by the empirical distribution over its past best response strategies. The sequence of iteration steps continues until the strategies of both players have sufficiently converged to a Nash equilibrium [50, 51].

**C. Our Self-Play Algorithm: T-FP**

We present a fictitious self-play algorithm, which we call T-FP, that exploits the statements in Theorem 1 to efficiently approximate Nash equilibria of $\Gamma$.

T-FP implements the fictitious self-play process described in Section IV-B and generates a sequence of strategy profiles $(\pi_1, l, \pi_2, l), (\pi_1, l, \pi_2, l), \ldots, (\pi_1, L, \pi_2, L)$ that converges to a Nash equilibrium. During each step of this process, T-FP learns best responses against the players’ current strategies and then updates the strategies of both players to be the empirical distribution over the past strategies (see Fig. 6).

T-FP parameterizes the best response strategies $\hat{\pi}_1, l, \hat{\theta}(1) \in B_1(\pi_2, l)$ and $\hat{\pi}_2, l, \hat{\theta}(2) \in B_2(\pi_1, l)$ by threshold vectors. The defender’s best response strategy is parameterized with the vector $\hat{\theta}(1) \in \mathbb{R}^L$ (Eq. 26). Similarly, the attacker’s best response strategy is parameterized with the vector $\hat{\theta}(2) \in \mathbb{R}^{2L}$ (Eq. 31).

$$\varphi(a, b) = \left(1 + \frac{b(1 - \sigma(a))}{\sigma(a)(1 - b)}\right)^{-1} \quad (29)$$
Fig. 6: The fictitious self-play process; in every iteration each player learns a best response strategy \( \tilde{\pi}_{1,t} \in B_1(\pi_{-1,t}) \) and updates its strategy based on the empirical distribution of its past best responses; the horizontal arrows indicate the iterations of self-play and the vertical arrows indicate the learning of best responses; if the process is convergent, it reaches a Nash equilibrium \( (\pi_{1,t}^*, \pi_{2,t}^*) \).

Algorithm 1 T-FP

**Input**  
\( \Gamma, N \): the POSG and \# best response iterations  
\( a, c, \lambda, A, \epsilon, \delta \): scalar coefficients

**Output**  
\( (\pi_{1,t}^*, \pi_{2,t}^*) \): an approximate Nash equilibrium

1: **procedure** T-FP
2: \( \tilde{\theta}^{(1)} \sim U_L(\{-1, 1\}), \tilde{\theta}^{(2)} \sim U_L(\{-1, 1\}) \)
3: \( \Theta^{(1)} \leftarrow \{\tilde{\theta}^{(1)}\}, \Theta^{(2)} \leftarrow \{\tilde{\theta}^{(2)}\}, \delta \leftarrow \infty \)
4: \( \pi_{1,t} \leftarrow \text{EMP(DISTRIBUTION}(\Theta^{(1)}) \)
5: \( \pi_{2,t} \leftarrow \text{EMP(DISTRIBUTION}(\Theta^{(2)}) \)
6: **while** \( \delta \geq \delta \) **do**
7: \( \text{for } i \in \{1, 2\} \) **do**
8: \( \tilde{\theta}^{(i)} \sim U_L(\{-1, 1\}) \)
9: \( \text{for } n \in \{1, \ldots, N\} \) **do**
10: \( a_n \leftarrow \frac{a}{n+A}, c_n \leftarrow \frac{c}{n} \)
11: \( \text{for } k \in \{1, \ldots, iL\} \) **do**
12: \( (\Delta a_n) \sim U_L(\{-1, 1\}) \)
13: **end for**
14: \( R_{\text{high}} \sim J_i(\pi_i, \tilde{\theta}^{(i)}) + c_n \Delta a_n, \pi_{-i} \)
15: \( R_{\text{low}} \sim J_i(\pi_i, \tilde{\theta}^{(i)}) - c_n \Delta a_n, \pi_{-i} \)
16: **for** \( k \in \{1, \ldots, iL\} \) **do**
17: \( \Delta \pi_i \leftarrow \frac{(\Delta a_n)}{a_n} \)
18: \( \tilde{\theta}^{(i)} \leftarrow \tilde{\theta}^{(i)} + a_n \nabla \tilde{\theta}^{(i)} J_i(\pi_i, \tilde{\theta}^{(i)}, \pi_{-i}) \)
19: **end for**
20: \( \Theta^{(i)} \leftarrow \Theta^{(i)} \cup \tilde{\theta}^{(N+1)} \)
21: **end for**
22: **end while**
23: **end procedure**

V. EMULATING THE TARGET INFRASTRUCTURE TO INSTANTIATE THE SIMULATION

To simulate a game episode we must know the observation distribution conditioned on the system state (see Eqs. [11][13]). We estimate this distribution using measurements from the emulation system shown in Fig. [2]. Moreover, to evaluate the performance of strategies learned in the simulation system, we run game episodes in the emulation system by having the attacker and the defender take actions at the times prescribed by the learned stopping strategies.

A. EMULATING THE TARGET INFRASTRUCTURE

The emulation system executes on a cluster of machines that runs a virtualization layer provided by Docker [54] containers and virtual links. The system implements network isolation and traffic shaping on the containers using network namespaces and the NetEm module in the Linux kernel [55]. Resource constraints on the containers, e.g., CPU and memory constraints, are enforced using cgroups.

The network topology of the emulated infrastructure is given in Fig. [1] and the configuration is given in Appendix [4]. The system emulates the clients, the attacker, the defender, network connectivity, and 31 physical components of the system.
The software functions replicate important components of the target infrastructure, such as, web servers, databases, and the Snort IPS, which is deployed using Snort’s community ruleset v2.9.17.1.

We emulate connections between servers as full-duplex loss less connections with capacity 1 Gbit/s in both directions. We emulate external connections between the gateway and the client population as full-duplex connections of 100 Mbit/s capacity and 0.1% packet loss with random bursts of 1% packet loss. (These numbers are drawn from empirical studies on enterprise and wide area networks [56, 57, 58].)

### B. Emulating the Client Population

The client population is emulated by processes that run inside Docker containers and interact with the application servers through the gateway. The clients select functions uniformly at random from the list given in Table 1. We emulate client arrivals using a stationary Poisson process with parameter \( \lambda = 20 \) and exponentially distributed service times with parameter \( \mu = \frac{1}{4} \). The duration of a time-step in the emulation is 30 seconds.

### C. Emulating Defender and Attacker Actions

The attacker and the defender observe the infrastructure continuously and take actions at discrete time-steps \( t = 1, 2, \ldots, T \). During each step, the defender and the attacker can perform one action each.

The attacker executes either a continue action or a stop action. Only the stop action affects the progression of the emulation. We have implemented \( L = 7 \) stop actions, which are listed in Table 2. The first stop action revokes user certificates and recovers user accounts thought to be compromised by the attacker. The second stop action updates the firewall configuration of the gateway to drop traffic from IP addresses that have been flagged by the IPS. Stop actions \( 3 - 6 \) update the configuration of the IPS to drop traffic that generates alerts of priorities \( 1 - 4 \). The final stop action blocks all incoming traffic. (Contrary to Snort’s terminology, we define 4 to be the highest priority.)

Like the defender, the attacker executes either a stop action or a continue action during each time-step. The attacker can take two stop actions. The first determines when the intrusion starts and the second determines when it ends (see Section III). A continue action in state \( s = 0 \) has no affect on the emulation, but a continue action in state \( s = 1 \) has. When the attacker takes a stop action in state \( s = 0 \) or a continue action in state \( s = 1 \), an intrusion command is executed. We have implemented 25 such commands, which are listed in Table 3.

### D. Estimating the IPS Alert Distribution

At the end of every time-step, the emulation system collects the metric \( o_t \), which contains the number of IPS alerts that occurred during the time-step, weighted by priority. For the evaluation reported in this paper we collect measurements from 23000 time-steps of 30 seconds each.

Using these measurements, we fit a Gaussian mixture distribution \( \hat{f}_O(\cdot|s_t) \) as an estimate of \( f_O(\cdot) \) in the target infrastructure (Eqs. 11-12). For each state \( s_t \), we obtain the conditional distribution \( f_O(s_t|\cdot) \) through expectation-maximization [59].

![Fig. 7: Empirical distributions of \( o_t \) when no intrusion occurs (\( s_t = 0 \)) and during intrusion (\( s_t = 1 \)); the black lines show the fitted Gaussian mixture models.](image-url)
mass within the range components, respectively. Both mixtures have most probability
8). Log updates are read periodically by a program that
implemented with Kafka [60] (see Fig.
and computing functions (see Table 2 and Table 3).
actions in the emulation system include executing networking
generated by the client population, the attacker, as well as
emulated infrastructure, which depend on the network traffic
issues requests to the emulated application servers (see Section
programming algorithm for one-sided POSGs.
Fig. 2). This section describes our evaluation results for
the intrusion prevention use case.

A. Learning Equilibrium Strategies through Self-Play

We run T-FP for 500 iterations to estimate a Nash equilib-
using the iterative method described in Section [V-B] At
the end of each iteration step, we evaluate the current strategy
pair (π1,l,π2,l) by running 500 evaluation episodes in the
simulation system and 5 evaluation episodes in the emulation
This allows us to produce learning curves for different
performance metrics (see Fig. 9).

To estimate the convergence of the sequence of strategy
pairs to a Nash equilibrium, we use the approximate ex-
ploitability metric \( v^{exp} \) [63]:

\[
v^{exp} = J_1(\hat{π}_1,l,\hat{π}_2,l) + J_2(π_1,l,\hat{π}_2,l)
\]

where \( \hat{π}_1,l \) denotes an approximate best response strategy
for player \( i \) obtained through dynamic programming. The
closer \( v^{exp} \) becomes to 0, the closer \( (π_1,l,π_2,l) \) is to a Nash
equilibrium.

The 500 training iterations constitute one training run. We
run four training runs with different random seeds. A single
training run takes about 5 hours of processing time on a P100
GPU. In addition, it takes around 12 hours to evaluate the
strategies on the emulation system.

Defender baseline strategies. We compare the learned
defender strategies with three baselines. The first baseline
prescribes the stop action whenever an IPS alert occurs, i.e.,
whenever \( o_t \geq 1 \). The second baseline follows the Snort
IPS’s internal recommendation system and takes a stop action
whenever 100 IP packets have been dropped by the Snort
IPS (see Appendix E for the Snort configuration). The third
baseline assumes knowledge of the exact intrusion time and
performs all stop actions at subsequent time-steps.

Baseline algorithms. We compare the performance of T-
FP with two baseline algorithms: Neural Fictitious Self-Play
(NFSP) [64] and Heuristic Search Value Iteration (HSV)
for one-sided POSGs [65]. NFSP is a state-of-the-art deep
reinforcement learning algorithm for imperfect-information
games. Similar to T-FP, NFSP is a fictitious self-play al-
gorithm. However, contrary to T-FP, NFSP does not exploit
the threshold structures expressed in Theorem [1] and as a
result is more complex. HSVI is a state-of-the-art dynamic
programming algorithm for one-sided POSGs.
Defender reward per episode

Fig. 9: Learning curves from the self-play process with T-FP; the red curve show simulation results and the blue curves show emulation results; the purple, orange, and black curves relate to baseline strategies; the figures show different performance metrics: exploitability, episodic reward, and the length of intrusion; the curves indicate the mean and the 95% confidence interval over four training runs with different random seeds.

Exploitability

Fig. 10: Probability of the stop action $S$ by the learned equilibrium strategies in function of $b(1)$ and $l$; the left and middle plots show the attacker’s stopping probability when $s = 0$ and $s = 1$, respectively; the right plot shows the defender’s stopping probability.

B. Discussion of the Evaluation Results

Fig. 9 shows the learning curves of the strategies obtained during the T-FP self-play process. The red curve represents 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 strategy that mandates a stop action whenever an IPS alert occurs, respectively. The dashed black curve gives the performance of the baseline strategy that assumes knowledge of the exact intrusion time.

The results in Fig. 9 lead us to the following conclusions. First, the fact that all learning curves seem to converge suggests to us that the learned strategies have converged as well. Second, we observe that the exploitability of the learned strategies converges to small values (left plot of Fig. 9). This indicates that the learned strategies approximate a Nash equilibrium both in the simulation system and in the emulation system. Third, we see from the middle plot in Fig. 9 that both baseline strategies show decreasing performance as the attacker updates its strategy. In contrast, the learned defender strategy improves its performance over time. This shows the benefit of using a game-theoretic approach, whereby the defender’s strategy is optimized against a dynamic attacker.

Fig. 10 illustrates some of the structural properties of the learned strategies. The y-axis shows the probability of the stop action $S$ and the x-axis shows the defender’s belief $b(1) \in B$. We observe that the strategies are stochastic. Hence, since

$$
\pi_1(S|b(1)), \pi_2(S|b(1))
$$

Fig. 11 suggests that the learned strategies converge to a Nash equilibrium, Fig. 10 suggests that this equilibrium is mixed, which we expect based on Theorem 1A. As we further expect from Theorem 1B-C, we see that that the defender’s stopping probability is increasing with $b(1)$ and decreasing with $l$ (right plot of Fig. 10). Similarly, we observe that the attacker’s stopping probability is decreasing with the defender’s stopping probability when $s = 0$ and is increasing when $s = 1$ (left and middle plot of Fig. 10).

Fig. 11 allows a comparison between T-FP and the two baseline algorithms (NFSP and HSVI) as they execute in the simulation system. Since T-FP and NFSP both implement fictitious self-play, they allow for a direct comparison. We observe that T-FP converges much faster to a Nash equilibrium than NFSP. We expect the fast convergence of T-FP due to its design to exploit structural properties of the stopping game.

The right plot of Fig. 11 shows that HSVI reaches an HSVI approximation error $< 5$ within an hour. We expected slower convergence due to findings in [66, 43]. A direct comparison between T-FP and HSVI is not possible due to the different nature of the two algorithms.

Lastly, Fig. 12 shows the computed value function of the game $V^*$ (Eq. 25). We see that $V^*$ is piece-wise linear and convex, as expected by the theory of one-sided POSGs [66]. We also observe that the value of $V^*$ is minimal when $b(1) \approx
Fig. 12: The value function $V^*(b(1))$ computed through the HSVI algorithm with approximation error $4$; the blue and red curves relate to $l = 7$ and $l = 1$, respectively.

0.25 and is 0 when $b(1) = 1$. Moreover, we note that this value is slightly lower for $l = 1$ compared to $l = 7$.

VII. RELATED WORK

Traditional approaches to intrusion prevention use packet inspection and static rules for detection of intrusions and selection of response actions [67], [68], [11]. 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 strategies in an automatic way. This research uses concepts and methods from various areas, most notably from anomaly detection (see example [69]), change-point detection (see example [70]), statistical learning (see examples [71], [72], [73]), control theory (see survey [5]), game theory (see textbooks [10], [24], [75], [76]), artificial intelligence (see survey [77] and textbook [78]), dynamic programming (see example [4]), reinforcement learning (see surveys [79], [80] and textbook [81]), evolutionary methods (see examples [6], [7]), and attack graphs (see example [82]).

While the research reported in this paper is informed by all the above works, we limit the following discussion to prior work that uses game-theoretic models and centers around finding strategies through reinforcement learning.

A. Reinforcement Learning in Network Security

Many recent results of automating security strategies have been obtained using reinforcement learning methods. In particular, a large number of studies have focused on intrusion prevention use cases similar to the one we discuss in this paper [13], [14], [26], [27], [19], [25], [23], [20], [29], [21], [22], [24], [83], [84], [85], [23], [86], [83], [34], [34].

These works use a variety of models, including MDPs [19], [20], [21], [22], [23], [35], Markov games [26], [13], [83], [85], attack graphs [34], and POMDPs [14], [24], [25], as well as various reinforcement learning algorithms, including Q-learning [26], [19], [20], [36], SARSA [25], PPO [13], [14], [34], [35], hierarchical reinforcement learning [21], DQN [22], Thompson sampling [24], MuZero [83], NFQ [84], DDQN [23], NFSP [37], and DDPG [85], [33].

This paper differs from the works referenced above in three main ways. First, we model the intrusion prevention use case as a partially observed stochastic game. Most of the other works model the use case as a single-agent MDP or POMDP. The advantage of using a game-theoretic model is that it allows finding defender strategies that are effective against dynamic attackers.

Second, in a novel approach, we derive structural properties of strategies in the game using optimal stopping theory.

Third, our method to find effective defender strategies includes using an emulation system in addition to a simulation system. The advantage of our method compared to the simulation-only approaches [13], [14], [26], [27], [19], [25], [28], [20], [29], [21], [22], [24], [83], [84], [34], [35], [36], [37] 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 strategies are evaluated in the emulation system, not in the simulation system. As a consequence, the evaluation results give higher confidence of the obtained strategies’ performance in the target infrastructure than what simulation results would provide.

Some prior work on automated intrusion prevention that make use of emulation are: [83], [84], [85], [87], and [23]. They emulate software-defined networks based on Mininet [88]. 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; (4) we do not assume a static attacker; and (5), we use an underlying theoretical framework to formalize the use case, derive structural properties of optimal strategies, and test these properties in an emulation system.

Finally, [89], [90], and [91] describe ongoing efforts in building emulation platforms for reinforcement learning and cyber defense, 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).

B. Game Theoretic Modeling in Network Security

Several examples of game theoretic security models can be found in the literature, e.g. advanced persistent threat games [30], [31], [32], honeypot placement games [92], [93], [94], resource allocation games [95], authentication games [11], distributed denial-of-service games [87], [96], and intrusion prevention games [97], [13], [83], [98], [99], [100].

This paper differs from the works referenced above in two main ways. First, we model the intrusion prevention use case as an optimal stopping game. The benefit of our model is that it provides insight into the structure of best response strategies through the theory of optimal stopping.

Game-theoretic formulations based on optimal stopping theory can be found in prior research on Dynkin games [16], [101], [102], [103], [104]. Compared to these papers, our approach is more general by (1) allowing each player to take multiple stop actions within an episode; and (2), by not assuming a game of perfect information. Another difference is that the referenced papers either study purely mathematical problems or problems in mathematical finance. To the best of our knowledge, we are the first to apply the stopping game formulation to the use case of intrusion prevention.
Our stopping game has similarities with the FlipIt game [30] and signaling games [105], both of which are commonplace in the security literature (see survey [106] and textbooks [10, 24, 25, 75, 76]). Signaling games have the same information asymmetry as our game and FlipIt uses the same binary state space to model the state of an intrusion. The main differences are as follows. FlipIt models the use case of advanced persistent threats and is a symmetric non-zero-sum game. In contrast, our game models an intrusion prevention use case and is an asymmetric zero-sum game. Compared to signaling games, the main differences are that our game is a sequential and simultaneous-move game. Signaling games are typically two-stage games where one player moves in each stage.

Second, as we noted above, we evaluate obtained strategies on an emulated IT infrastructure. This contrasts with most of the prior works that use game-theoretic approaches, which only evaluate strategies analytically or in simulation [30, 31, 32, 92, 93, 94, 11, 96, 97, 13, 98, 99, 100].

VIII. CONCLUSION AND FUTURE WORK

We formulate the interaction between an attacker and a defender in an intrusion prevention use case as an optimal stopping game. The theory of optimal stopping provides us with insight about optimal strategies for attackers and defenders. Based on this knowledge, we develop a fictitious self-play algorithm, T-FP, which allows us to compute near optimal strategies in an efficient way. This approach provides us with a complete formal framework for analyzing and solving the intrusion prevention use case. The simulation results from executions of T-FP show that the exploitability of the computed strategies converges, which suggests that the strategies converge to a Nash equilibrium and thus to an optimum in the game-theoretic sense. The results also demonstrate that T-FP converges faster than a state-of-the-art fictitious self-play algorithm by taking advantage of structural properties of optimal stopping strategies.

To assess the computed strategies in a real environment, we evaluate them in a system that emulates our target infrastructure. The results show that the strategies achieve almost the same performance in the emulated infrastructure as in the simulation. This gives us a high confidence of the obtained strategies’ performance in the target infrastructure.

We plan to extend this work in several directions. First of all, the model of the attacker and the defender in this paper is simplistic as it only models the timing of actions and not their selection. We plan to combine our current model for deciding when to take defensive actions with a model for the selection of which action to take.

IX. ACKNOWLEDGMENTS

This research has been supported in part by the Swedish armed forces and was conducted at KTH Center for Cyber Defense and Information Security (CDIS). The authors would like to thank Pontus Johnson for his useful input to this research, and Forough Shahab Samani and Xiaoxuan Wang for their constructive comments on a draft of this paper. The authors are also grateful to Branislav Bosanský for sharing the code of the HSVI algorithm for one-sided POSGs and to Jakob Styrmne for contributing to our implementation of NFSP.

APPENDIX A
PROOF OF THEOREM I.A

Proof of Theorem I.A. Since the POSG $\Gamma$ introduced in Section III is finite and $\gamma \in (0, 1)$, the existence proofs in [41] and [66] applies, which state that a mixed Nash equilibrium exists.

We prove that a pure Nash equilibrium exists when $s = 0 \iff b(1) = 0$ using a proof by construction. It follows from Eqs. 6, 9, and Eq. 17 that the pure strategy defined by $\pi_{1,i}(0) = C$ and $\pi_{2,i}(1) = S \iff b(1) > 0$ is a best response for the defender against any attacker strategy when $s = 0 \iff b(1) = 0$. Similarly, given $\pi_{1,i}$, we get from Eqs. 6 and Eq. 17 that the pure strategy defined by $\pi_{2,i}(0, b(1)) = C$ and $\pi_{2,i}(1, b(1)) = S$ for all $b(1) \in \{0, 1\}$ is a best response for the attacker. Hence, $(\pi_{1,i}, \pi_{2,i})$ is a pure Nash equilibrium (see Eq. 19).

APPENDIX B
PROOF OF THEOREM I.B.

Given the POSG $\Gamma$ introduced in Section III and a fixed attacker strategy $\pi_{2,i}$, the best response strategy of the defender $\pi_{1,i} \in B_1(\pi_{2,i})$ is an optimal strategy in a POMDP $\mathcal{M}^P$ (see Section IV). Hence, it is sufficient to show that there exists an optimal strategy $\pi_{1,i}^*$ in $\mathcal{M}^P$ that satisfies Eq. 26. The conditions for Eq. 26 to hold and the proof are given in our previous work [13] (Theorem 1.C). Since $f_{|0|s}$ is TP2 by assumption and all of the remaining conditions hold by definition of $\Gamma$, the result follows.

APPENDIX C
PROOF OF THEOREM I.C.

Given the POSG $\Gamma$ introduced in Section III and a fixed defender strategy $\pi_{1,i}$, the best response strategy of the attacker $\pi_{2,i} \in B_2(\pi_{1,i})$ is an optimal strategy in an MDP $\mathcal{M}$ (see Section IV). Hence, it is sufficient to show that there exists an optimal strategy $\pi_{2,i}^*$ in $\mathcal{M}$ that satisfies Eqs. 27, 27. To prove this, we use properties of $\mathcal{M}$’s value function $V_{\pi_{1,i}, \cdot}$.

We use the value iteration algorithm to establish properties of $V_{\pi_{2,i}, \cdot}$ [44, 47]. Let $V_{\pi_{1,i}, \cdot}$, $\gamma_{s, l, \pi_{1,i}, \cdot}$, and $\gamma_{s, l, \pi_{1,i}, \cdot}$ denote the value function, the stopping set, and the continuation set at iteration $k$ of the value iteration algorithm, respectively. Then, $\lim_{k \to \infty} V_{\pi_{1,i}, \cdot}^{k} = V_{\pi_{1,i}, \cdot}^{*}$, $\lim_{k \to \infty} \gamma_{s, l, \pi_{1,i}, \cdot}^{k} = \gamma_{s, l, \pi_{1,i}, \cdot}^{*}$, and $\lim_{k \to \infty} \gamma_{s, l, \pi_{1,i}, \cdot}^{k} = \gamma_{s, l, \pi_{1,i}, \cdot}$ [44, 47]. We define $V_{\pi_{2,i}, \cdot}^{0}(s, b(1)) = 0$ for all $b(1) \in \{0, 1\}$, $s \in S$ and $l \in \{1, \ldots, L\}$.

Towards the proof of Theorem I.C, we state the following six lemmas.

Lemma 1. Given any defender strategy $\pi_{1,i}$, $V_{\pi_{1,i}, \cdot}^{\infty}(s, b(1)) \geq 0$ for all $s \in S$ and $b(1) \in \{0, 1\}$.

Proof. Consider $\pi_{2,i}$ defined by $\pi_{2,i}(0, \cdot) = C$ and $\pi_{2,i}(1, \cdot) = S$. Then it follows from Eqs. 6 that for any $\pi_{1,i} \in \Pi_1$,
We know that the second term in the left-hand side of Eq. 34 is thus zero. Hence, Lemma 5.

**Lemma 2.** \( V^*_{\pi_{1,i},l}(1, b(1)) \) is non-increasing with \( \pi_{1,i}(S|b(1)) \) and non-decreasing with \( l \in \{1, \ldots, L\} \).

**Proof.** We prove this statement by mathematical induction. Let

\[
V^k_{\pi_{1,i},l}(s, b(1)) = \max \left[ 0, -R(1, (C, a(1))) \right] + (1 - \phi) \sum_{o} f_o(o|1) V^{k-1}_{\pi_{1,i},l}(1, b(1))
\]

The first term in the maximization in Eq. 33 is trivially non-increasing with \( \pi_{1,i}(S|b(1)) \) and non-decreasing with \( l \). Assume by induction that the conditions hold for \( V^{k-1}_{\pi_{1,i},l}(s, b(1)) \). Then the second term in Eq. 33 is non-increasing with \( \pi_{1,i}(S|b(1)) \) and non-decreasing with \( l \) by Eqs. 10 and the induction hypothesis. Hence, \( V^k_{\pi_{1,i},l}(s, b(1)) \) is non-increasing with \( \pi_{1,i}(S|b(1)) \) and non-decreasing with \( l \) for all \( k \geq 0 \).

**Lemma 3.** If \( f_o \) is TP2 and \( \pi_{1,i}(S|b(1)) \) is increasing with \( b(1) \), then \( V_{\pi_{1,i},l}(b(1), 1) \geq \sum_{o} f_o(o|1) V^{k-1}_{\pi_{1,i},l}(1, b(1)) \), where \( b^*(1) \) denotes \( b(1) \) updated with Eq. 14 after observing \( o \in \mathcal{O} \).

**Proof.** Since \( f_o \) is TP2, it follows from [47] Theorem 10.3.1, pp. 225,238 and [12] Lemma 4, pp. 12] that given two beliefs \( b'(1) \geq b(1) \) and two observations \( o \geq \bar{a} \), the following holds for any \( k \in \mathcal{O} \) and \( l_i \in \{1, \ldots, L\} : b'(1) \geq b(1), \mathbb{P}[o \geq k|b'(1)] \geq \mathbb{P}[o \geq a|b(1)], \) and \( \phi_o(1) \geq \phi_{a}(1) \).

Since \( \pi_{1,i} \) is increasing with \( b(1) \) and \( V_{\pi_{1,i},l}(b(1), 1) \) is decreasing with \( b(1) \) (Lemma 2), it follows that \( E_{\pi_{1,i}}[b^*(1)] \geq b(1) \) and thus \( V_{\pi_{1,i},l}(b(1), 1) \geq \sum_{o} f_o(o|1) V^{k-1}_{\pi_{1,i},l}(1, b(1)) \).

**Lemma 4.** If \( f_o \) is TP2, \( \pi_{1,i}(S|b(1)) = 1 \), and \( \pi_{1,i}(S|b(1)) \) is increasing with \( b(1) \), then \( V^*_{\pi_{1,i},l}(s, b(1)) = 0 \) and for any \( \pi_{2,i} \in \mathcal{B}(\pi_{1,i}), \pi_{2,i}(b(1)) = 1 \).

**Proof.** From Eqs. 22 and 25, we know that \( \pi_{2,i}(1, b(1)) = 1 \) iff:

\[
R_{st}/l + (\phi_l - 1) \sum_{o} f_o(o|1) V^*_{\pi_{2,i},l-a(1)}(1, b^*(1)) \geq 0 \tag{34}
\]

We know that \( R_{st} \geq 0 \) (see Section III). Further, since \( f_o \) is TP2, \( \pi_{1,i}(S|b(1)) = 1 \), and \( \pi_{1,i}(S|b(1)) \) is increasing with \( b(1) \), we have by Lemma 3 that \( E_{\pi_{1,i}}[\pi_{1,i}(S|b^*(1))] = 1 \). The second term in the left-hand side of Eq. 34 is thus zero. Hence, the inequality holds and \( \pi_{2,i}(1, b(1)) = 1 \), which implies that \( V^*_{\pi_{1,i},l}(s, b(1)) = 0 \).

**Lemma 5.** Given any defender strategy \( \pi_{1,i} \in \Pi_1 \), if \( \pi_{2,i}(1, b(1)) = S \), then \( \pi_{2,i}(0, b(1)) = C \).

**Proof.** \( \pi^2_{2,i}(1, b(1)) = S \) implies that \( V^*_{\pi_{2,i},l}(1, b(1)) = 0 \). Hence, by Lemma 5 we get that:

\[
(1 - \phi_l) \sum_{o \in \mathcal{O}} f_o(o|1) V^*_{\pi_{2,i},l}(1, b^o) \leq 0
\]

\[
\sum_{o \in \mathcal{O}} f_o(o|1) V^*_{\pi_{2,i},l}(1, b^o) \leq \sum_{o \in \mathcal{O}} f_o(o|0) V^*_{\pi_{2,i},l}(0, b^o)
\]

\[
\pi^2_{2,i}(0, b(1)) = C
\]

**Lemma 6.** If \( \pi_{1,i}(S|b(1)) \) is non-decreasing with \( b(1) \) and \( f_o \) is TP2, then \( V^*_{\pi_{1,i},l}(0, b(1)) = V^*_{\pi_{1,i},l}(1, b(1)) \) is non-decreasing with \( \pi_{1,i}(S|b(1)) \).

**Proof.** We prove this statement by mathematical induction. Let \( W^k_{\pi_{1,i},l}(b(1)) = V^k_{\pi_{1,i},l}(0, b(1)) - V^k_{\pi_{1,i},l}(1, b(1)). \) For \( k = 0 \), it follows from Eqs. 11 that \( W^0_{\pi_{1,i},l}(b(1)) \) is non-decreasing with \( \pi_{1,i}(S|b(1)) \). Assume by induction that the conditions hold for \( W^{k-1}_{\pi_{1,i},l}(b(1)) \). We show that then the conditions hold also for \( W^k_{\pi_{1,i},l}(b(1)) \).

There are three cases to consider:

- If \( b(1) \in \mathcal{I}(2)_{0,\pi_{1,i},l} \cap \mathcal{I}(2)_{1,\pi_{1,i},l} \), then:

\[
W^k_{\pi_{1,i},l}(b(1)) = R_{st}/(l - R_{cost}/l - R_{int})
\]

which is non-decreasing with \( \pi_{1,i}(S|b(1)) \) since \( R_{st}/l - R_{cost}/l - R_{int} \geq 0 \) (see Section III).

- If \( b(1) \in \mathcal{I}(2)_{0,\pi_{1,i},l} \cap \mathcal{I}(2)_{2,\pi_{1,i},l} \) then:

\[
W^k_{\pi_{1,i},l}(b(1)) = \pi_{1,i}(S|b(1)) \left( R_{st}/l - R_{cost}/l \right)
\]

\[
+ R_{int} \left( V^{k-1}_{\pi_{1,i},l}(1, b(1)) - R_{int} \right) + R_{int} \sum_{o \in \mathcal{O}} f_o(o|0)
\]

\[
V^k_{\pi_{1,i},l}(0, b(1)) - (1 - \phi_l) f_o(o|1) V^k_{\pi_{1,i},l}(1, b(1))
\]

The first term is non-decreasing with \( \pi_{1,i}(S|b(1)) \) since \( R_{st}/l - R_{cost}/l - R_{int} \geq 0 \) (see Section III) and \( V^{k-1}_{\pi_{1,i},l}(1, b(1)) \geq 0 \) (it is a consequence of Lemma 3 and Eqs. 22 and 25). The second term is non-decreasing with \( \pi_{1,i}(S|b(1)) \) by the induction hypothesis and the assumption that \( f_o \) is TP2.

- If \( b(1) \in \mathcal{I}(2)_{0,\pi_{1,i},l} \cap \mathcal{I}(2)_{1,\pi_{1,i},l} \), then:

\[
W^k_{\pi_{1,i},l}(b(1)) = \pi_{1,i}(S|b(1)) (-R_{cost}/l)
\]

\[
+ \sum_{o} f_o(o|0) V^k_{\pi_{1,i},l}(0, b^o(1))
\]

\[
= \pi_{1,i}(S|b(1)) (-R_{cost}/l) + \sum_{o} f_o(o|0)
\]

\[
V^k_{\pi_{1,i},l}(0, b(1)) - (1 - \phi_l) f_o(o|1) V^k_{\pi_{1,i},l}(1, b(1))
\]

The first term is non-decreasing with \( \pi_{1,i}(S|b(1)) \) since \( -R_{cost}/l \geq 0 \). The second term is non-decreasing with \( \pi_{1,i}(S|b(1)) \) by the induction hypothesis and the assumption that \( f_o \) is TP2. The second equality in Eq. 38 follows from Lemma 3 and because \( b(1) \in \mathcal{I}(2)_{0,\pi_{1,i},l} \).

The other cases, e.g. \( b(1) \in \mathcal{I}(2)_{0,\pi_{1,i},l} \cap \mathcal{I}(2)_{2,\pi_{1,i},l} \), can be discarded due to Lemma 5. Hence, \( W^k_{\pi_{1,i},l}(b(1)) \) is non-decreasing with \( \pi_{1,i}(S|b(1)) \) for all \( k \geq 0 \).
We now use Lemmas [1][6] to prove Theorem [1][C]. The main idea behind the proof is to show that the stopping sets in state \( s = 1 \) have the form: \( \mathcal{A}_{1,t,\pi_{1,t}}^{2} = \{ \tilde{\beta}_{1,t,1} \} \), and that the continuation sets in state \( s = 0 \) have the form: \( \mathcal{C}_{0,t,\pi_{0,t}}^{2} = \{ \tilde{\beta}_{0,1,1} \} \), for some values \( \tilde{\beta}_{0,1,1}, \tilde{\beta}_{1,1,1}, \ldots, \tilde{\beta}_{0,L,1}, \tilde{\beta}_{1,L,1} \in [0,1] \).

**Proof of Theorem 7C.** We first show that \( \mathcal{A}_{1,t,\pi_{1,t}}^{2} \) and that \( \mathcal{C}_{0,t,\pi_{0,t}}^{2} \). Since \( \pi_{1,t}(S|1) = 1 \), by Lemma 4 we have that \( \mathcal{A}_{1,t,\pi_{1,t}}^{2} \) and it follows from Eqs. [22][25] that \( \pi_{2,t}(0,b(1)) = C \) i.f.f:

\[
\sum_{o} f_{0}(0)V_{\pi_{1,t},t-1}(0,b(o(1))) - f_{0}(1)V_{\pi_{1,t},t-1}(1,b(o(1))) \geq 0
\]

(39)

The left-hand side of the above equation is positive due to the assumption that \( f_{0} \) is TP2 and since

\[
\sum_{o} f_{0}(0)V_{\pi_{1,t},t-1}(0,b(o(1))) \geq 0 \text{ by Lemma 7}
\]

and

\[
f_{0}(1)V_{\pi_{1,t},t-1}(1,b(o(1))) = 0 \text{ by Lemma 8}
\]

Hence, \( \mathcal{A}_{1,t,\pi_{1,t}}^{2} \).

Now we show that \( \mathcal{C}_{0,t,\pi_{0,t}}^{2} = \{ \tilde{\beta}_{1,t,1} \} \) and that \( \mathcal{C}_{0,t,\pi_{0,t}}^{2} = \{ \tilde{\beta}_{0,1,1} \} \) for some values \( \tilde{\beta}_{0,1,1}, \tilde{\beta}_{1,1,1}, \ldots, \tilde{\beta}_{0,L,1}, \tilde{\beta}_{1,L,1} \in [0,1] \).

From Eqs. [22][25] we know that \( \pi_{2,t}(1,b(1)) = S \) i.f.f:

\[
\mathbb{E}_{\pi_{t-1}}[\mathcal{R}_{t}|(1,a(1),C)] \geq 0
\]

(40)

The first term in the above expectation is increasing with \( b(1) \) (Eqs. [6][10]). The second term is decreasing with \( b(1) \) (Lemma 3). Hence, we conclude that if \( \pi_{2,t}(1,b(1)) = S \), then for any \( b'(1) \geq b(1) \), \( \pi_{2,t}(1,b'(1)) = S \). As a consequence, there exists \( \tilde{\beta}_{1,1,1}, \ldots, \tilde{\beta}_{1,L,1} \) such that \( \mathcal{A}_{1,t,\pi_{1,t}}^{2} = \{ \tilde{\beta}_{1,t,1} \} \).

Similarly, from Eqs. [22][25] we know that \( \pi_{2,t}(0,b(1)) = C \) i.f.f:

\[
\mathbb{E}_{\pi_{t-1}}[\sum_{o} f_{0}(0)V_{\pi_{1,t},t-1-o}(1,b(o(1))) - f_{0}(1)V_{\pi_{1,t},t-1-o}(1,b(o(1))) \geq 0
\]

(41)

Since \( f_{0} \) is TP2 and \( \pi_{1,t}(S|b(1)) \) is increasing with \( b(1) \), the left-hand side in the above inequality is decreasing (Lemma 2 and Lemma 6). Hence, we conclude that if \( \pi_{2,t}(0,b(1)) = C \), then for any \( b'(1) \geq b(1) \), \( \pi_{2,t}(0,b'(1)) = C \). As a result, there exists \( \tilde{\beta}_{0,1,1}, \ldots, \tilde{\beta}_{0,L,1} \) such that \( \mathcal{C}_{0,t,\pi_{0,t}}^{2} = \{ \tilde{\beta}_{0,t,1} \} \).

**APPENDIX D**

**HYPERPARAMETERS**

The hyperparameters used for the evaluation are listed in Table 4 and were obtained through grid search.

**APPENDIX E**

**CONFIGURATION OF THE INFRASTRUCTURE IN FIG. 1**

The configuration of the target infrastructure (Fig. 1) is available in Table 5.
[13] ——. “Finding effective security strategies through reinforcement learning and Self-Play,” in International Conference on Network and Service Management (CNSM 2020), Izmir, Turkey, 2020.

[14] ——. “Learning intrusion prevention policies through optimal stopping,” in International Conference on Network and Service Management (CNSM 2021), Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.1294.pdf.

[15] A. Wald, Sequential Analysis. Wiley and Sons, New York, 1947.

[16] E. Dynkin, “A game-theoretic version of an optimal stopping problem,” Dokl. Akad. Nauk SSSR, vol. 385, pp. 16–19, 1969.

[17] A. N. Shiryaev, “On optimum methods in quickest detection problems,” Theory Probab. Appl., vol. 8, no. 1, pp. 22–46, 1963.

[18] K. Hammar and R. Stadler, “A software framework for building self-learning security systems,” 2022, https://www.youtube.com/watch?v=18P7MPKNDNg.

[19] F. M. Zemaro and L. Erdodi, “Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular q-learning,” CoRR, 2020, https://arxiv.org/abs/2005.12652.

[20] A. Ridley, “Machine learning for autonomous cyber defense,” 2018, the Next Wave, Vol 22, No 1 2018.

[21] K. Tran, A. Akella, M. Standen, J. Kim, D. Bowman, T. Richer, and C.-T. Lin, “Deep hierarchical reinforcement agents for automated penetration testing,” 2021, https://arxiv.org/abs/2109.06449.

[22] R. Gangapantulatu, T. Cody, P. Park, A. Rahman, L. Eisenbeiser, D. Radke, and R. Clark, “Using cyber terrain in reinforcement learning for penetration testing,” 2021, https://arxiv.org/abs/2108.07129.

[23] T. V. Phan and T. Bauschert, “Deepair: Deep reinforcement learning for adaptive intrusion response in software-defined networks,” IEEE Transactions on Network and Service Management, pp. 1–1, 2022.

[24] Z. Hu, M. Zhu, and P. Liu, “Adaptive cyber defense against multi-stage attacks using learning-based pomdp,” ACM Trans. Priv. Secur., vol. 24, no. 1, Nov. 2020.

[25] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, “Online cyber-attack detection in smart grid: A reinforcement learning approach,” IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5174–5185, 2019.

[26] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Dragan, and M. Wiering, “Adversarial reinforcement learning in a cyber security simulation,” in ICAART, 2017.

[27] J. Schwartz, H. Kurniawati, and E. El-Mahassni, “Pomdp + information-decay: Incorporating defender’s behaviour in autonomous penetration testing,” Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30, no. 1, pp. 235–243, Jun. 2020.

[28] W. Blum, “Gamifying machine learning for stronger security and ai models,” 2021, https://www.microsoft.com/security/blog/2021/04/20/gamifying-machine-learning-for-stronger-security-and-ai-models/.

[29] M. Zhu, Z. Hu, and P. Liu, “Reinforcement learning algorithms for adaptive cyber defense against heartbleed,” in Proceedings of the First ACM Workshop on Moving Target Defense, ser. MTD ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 51–58.

[30] van Dijk, A. Juels, A. Oprea, and R. L. Rivest, “Flipping: The game of “stealthy takeover”,” Journal of Cryptology, no. 4, Oct 2013.

[31] L. Huang and Q. Zhu, “A dynamic games approach to proactive defense strategies against advanced persistent threats in cyber-physical systems,” Computers & Security, vol. 89, p. 101660, 11 2019.

[32] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, General Sum Markov Games for Strategic Detection of Advanced Persistent Threats Using Moving Target Defense in Cloud Networks, 10 2019.

[33] L. Zhang, Y. Pan, Y. Liu, Q. Zheng, and Z. Pan, “Multiple domain cyberspace attack and defense game based on reward maximization reinforcement learning,” 2022.

[34] Y. Du, Z. Song, S. Milani, C. Gonzales, and F. Fang, “Learning to play an adaptive cyber defense game,” The 13th Workshop on Optimization and Learning in Multiagent Systems, AMAS 2022, 2022.

[35] A. Dutta, E. Al-Shaer, and S. Chatterjee, “Constraints satisfiability driven reinforcement learning for autonomous cyber defense,” CoRR, vol. abs/2104.08994, 2021.

[36] Q. Xu, Z. Su, and R. Lu, “Game theory and reinforcement learning based secure edge caching in mobile social networks,” IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3415–3429, 2020.

[37] K. Li, B. Jiu, W. Pu, H. Liu, and X. Peng, “Neural fictitious self-play for radar-inter-jamming dynamic game with imperfect information,” IEEE Transactions on Aerospace and Electronic Systems, pp. 1–1, 2022.

[38] J. F. Nash, “Non-cooperative games,” Annals of Mathematics, vol. 54, pp. 286–295, 1951.
