

Learning Intrusion Response Strategies for OT Systems

Duc Huy Le, Rolf Stadler
 Dept. of Network and Systems Engineering
 KTH Royal Institute of Technology, Stockholm, Sweden
 Email: {dhle, stadler}@kth.se

Abstract—Cyberattacks against Operational Technology (OT) systems, which monitor and control industrial processes, pose an increasing threat to essential societal services. For this reason, developing automated intrusion response strategies is highly important. In this paper, we present a formal model of an OT intrusion response use case using the POMDP framework. It includes a realistic model of partial observability that is based on traffic measurements. This approach allows us to develop tractable, learning-based solution methods for automated intrusion response, which are based on PPO. We evaluate the obtained response strategies on an emulated OT system and find that they are effective against several types of MITRE attacks for the studied use case.

Index Terms—Operational Technology (OT), security management, automated security, defender strategy, reinforcement learning, Partially Observable Markov Decision Process (POMDP)

I. INTRODUCTION

Operational Technology (OT) systems, which monitor and control industrial processes, are increasingly exposed to cyber threats that target critical infrastructure and industrial production. The number of OT cyberattacks grew by more than 90% annually between 2019 and 2023 [1]. Ransomware attacks against industrial organizations increased by 87% in 2024 [2]. Although the number of OT attacks is currently lower than that of Information Technology (IT) attacks, their consequences can be more severe, by disrupting essential services and affecting communities beyond the targeted organizations [1]. Notable examples of such attacks include TRITON/TRISIS [3], Industroyer2 [4], and FrostyGoop [2].

A factor contributing to the rise of OT attacks is the IT/OT convergence, where IT systems are integrated with physical OT networks to facilitate monitoring, data collection, and efficient operation. This integration expands the attack surface and creates paths from enterprise networks into OT environments, which adversaries can exploit [5], [6]. Traditionally, OT infrastructures have a lower level of cybersecurity than IT infrastructures [7].

OT cybersecurity guidelines, e.g. [8], [9], [10], emphasize proactive measures, such as network segmentation and vulnerability analysis, as well as monitoring-based control, including resetting hosts and isolating network segments. They leave intrusion response actions largely to human operators and do not stress automated responses.

Early research on automated intrusion response for IT and OT systems has focused on rule-based policies, which configure firewall rules [11] or SDN flows [12], [13], for instance. These rules are defined and maintained by human experts.

Recently, methods for OT intrusion response have been developed that do not rely on predefined rules, but are based on learning from system measurements. These methods use Reinforcement Learning (RL) as the dominant concept. Some RL approaches learn heuristic policies without a formal system model [14], [15], [16], which precludes the understanding of an achievable optimal strategy. Other works apply formal models, such as Markov Decision Process (MDP) [17], [18], [19], Partially Observable Markov Decision Process (POMDP) [20], [21], and stochastic game [22], [23]. Most of these lines of research assume full observability of the system state or attacker actions, which is unrealistic and limits the practical relevance of the results. Alternatively, some studies assume specific observation models, without detailing how such observations can be obtained in real systems [20], [21].

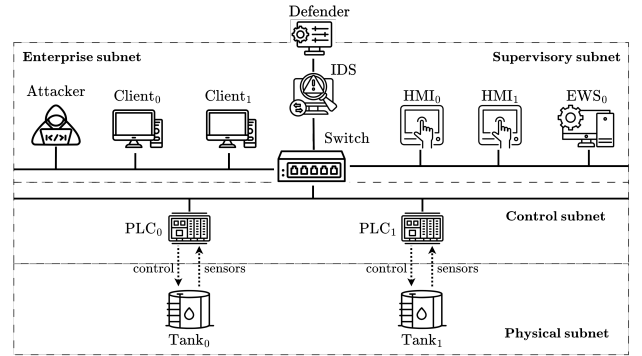


Fig. 1: The OT infrastructure of the intrusion response use case

In this paper, we investigate an intrusion response use case for an OT infrastructure (see Fig. 1). The infrastructure follows the Purdue model architecture with network segmentation [24]. The objective of the attacker is to disrupt the industrial processes that control the tanks in the OT infrastructure. The attacker creates a path through the infrastructure by executing a sequence of actions, including reconnaissance,

vulnerability scanning, exploitation, and host inspection. The defender's objective is to maintain normal process operation and mitigate the attacker actions. The defender monitors the infrastructure and executes a sequence of defensive actions based on continuous observations of the network traffic.

We model this scenario as a discrete-time dynamical system and formalize the defender's problem as a Partially Observable Markov Decision Process (POMDP) [25]. The model allows us to capture attacker and defender actions and define an optimal defender strategy for the use case. To learn effective defender strategies, we propose two learning methods, which are based on Proximal Policy Optimization (PPO), a state-of-the-art RL algorithm. Applying PPO to our problem formulation is not computationally feasible, since the state space (and thus the belief space) is too large. The first method, k -Obs-PPO, simplifies the structure of the input for the defender strategy, while the second method, BF-PPO, uses particle filter [26] to estimate the system state.

We have developed an emulation system, which serves as a replica of the target infrastructure shown in Fig. 1. On this system, we perform the attacks for this scenario. While the attacks are occurring, we collect traffic measurements. These measurements are used to estimate the observation function of the POMDP model, which allows us to simulate this model and to compute effective defender strategies. The obtained defender strategies are evaluated on the emulation system.

The key results of this investigation are as follows. We can compute effective defender strategies for the selected scenario using our methods. They converge in reasonable time on a simulator (about 1 hour on an Apple M3 Pro processor). BF-PPO produces a defender strategy that performs on the emulation system close to a strategy that assumes full observability, for three different types of attacks.

We make four contributions with this paper:

- We present a formal model of the OT intrusion response use case using the POMDP framework and formally define an optimal defender strategy.
- We present two solution methods, k -Obs-PPO and BF-PPO, for automated response. The methods are learning based, not rule-based, and not set up by experts. They efficiently learn effective defender strategies.
- Our POMDP model includes a realistic model of partial observability, which is based on online traffic measurements.
- We evaluate the solution methods in a realistic emulation environment of the OT infrastructure, demonstrating that the learned defender strategies are effective. In particular, BF-PPO achieves performance close to a defender strategy that has been computed under the assumption of full observability.

We see the main limitations of this work as follows. First, like all formal models, our model relies on simplifying assumptions. For instance, we assume that attacker and defender actions are taken simultaneously at discrete times and have immediate effects. In reality, such actions occur asynchronously in continuous time, have different durations, and affect the

system with delays. Capturing such dynamics would make the model more realistic, at the expense of increasing the computational complexity of the learning defender strategies. Also, we could have modeled the state space with higher granularity, with the same drawback. Second, our POMDP model is valid for the specific use case, system configuration and attack types. In this work, we did not study the generalization property of our model with respect to those aspects. Third, this work currently does not consider operational safety, which constrains defender strategy. We will do so in the future.

II. RELATED WORK

Many studies have investigated the intrusion response problem in IT systems. They use a variety of modeling frameworks, including POMDP [27], [28], game theory [29], [30], or causal models [31]. They apply numerous solution techniques, such as RL [27], [28], attack-graph-based planning [32], and large language models [33].

Automated intrusion response for OT systems has been studied using rule-based and learning-based approaches. Rule-based methods define and enforce predefined response actions, for example through automated firewall configuration [11] or software-defined networking [12], [13]. These methods automatically detect possible incidents and their response strategies are defined by human experts.

Recent advances in RL have initiated research on learning-based intrusion response solutions for OT systems, which allows defender strategies to be learned from measurements. Some studies propose heuristic solution methods that do not require a formal system model of the use case [14], [15], [16], [34]. However, without a formal model, it cannot be assessed to which extent the learned defender strategies are optimal.

Other works formulate intrusion response problems using decision-theoretic frameworks, including MDP [17], [18], [19], POMDP [20], [21], and stochastic games [22], [23]. Most of them investigate scenarios in which the defender can observe the attacker's actions and progress. This assumption of full observability is unrealistic. Some works consider partial observability using abstract observation models, but do not discuss how such models are obtained for real systems [16], [20], [21]. A methodology for developing a defender strategy, which first computes the strategy based on a formal model and subsequently evaluates it on an emulation platform, is presented in [35].

The work in this paper is unique in that it considers the following aspects in combination. It formulates a formal model of an OT intrusion response use case, presents learning-based solution methods for automated intrusion response, includes a realistic model of partial observability and evaluation results from an emulated OT system.

III. THE OT INTRUSION RESPONSE USE CASE

We study an intrusion response use case for the OT infrastructure shown in Fig. 1. The infrastructure follows the Purdue model [24] and is partitioned into enterprise, supervisory, control, and physical subnets. The *enterprise* subnet

contains client systems that provide access to the industrial environment. The *supervisory* subnet contains servers with Human–Machine Interfaces (HMIs) and Engineering Workstations (EWSs) for process monitoring, control, and engineering operations. The *control* subnet contains Programmable Logic Controllers (PLCs), which manage two water tanks in the *physical* subnet.

We assume the attacker has a foothold in the enterprise subnet, either through a compromised enterprise host or through insider access to the IT network. The attacker’s objective is to disrupt the industrial processes by tampering with the tanks. We consider attacks from the MITRE ATT&CK framework [36], where an attacker performs reconnaissance, exploitation, and inspection on supervisory hosts, and then uses a compromised host to interact with the PLCs controlling the tanks. We consider three attacker strategies for this use case, which differ in the way they perform reconnaissance, exploitation, etc.

The defender cannot directly observe the attacker’s progression through the infrastructure. Instead, it receives network measurements from the switch through the IDS in Fig. 1. The defender’s objective is to maintain normal operation of the industrial processes and mitigate the attacker actions. The defender can perform three types of defensive actions: reset a supervisory host, reset an industrial process, and reset all hosts in the supervisory subnet and the control subnet. A reset action on a component reboots the machine, renews its credentials, changes its IP address, etc. Such an action can cause a temporary disruption of industrial operations.

Note that the traffic measurements not only relate to the attacker actions but also to normal process operations, which makes their interpretation for the defender difficult.

IV. FORMALIZING THE USE CASE USING A POMDP MODEL

We study the intrusion response use case introduced in §III from the defender’s perspective. We assume that the system evolves in discrete time steps with a finite horizon. At each time, both the defender and the attacker take an action. The defender cannot observe the attacker’s progression and obtains knowledge about the attack through network measurements. We formalize the problem of intrusion response as a sequential decision-making problem under partial observability. We choose POMDP as the modeling framework.

A. Partially Observable Markov Decision Process

A Partially Observable Markov Decision Process (POMDP) models the decision process of a discrete-time Markovian system with partial observability [25]. It is defined by a 10-tuple $\mathcal{P} = (\mathbf{S}, \mathbf{D}, \mathcal{T}, \mathbf{O}, \mathcal{Z}, \mathcal{C}, \gamma, \rho_1, T, \mathbf{B})$. $\mathbf{S}$ denotes the state space and $\mathbf{D}$ denotes the action space. The probability measure $\mathcal{T} : \mathbf{S} \times \mathbf{S} \times \mathbf{D} \rightarrow [0, 1]$ denotes the system dynamics, where $\mathcal{T}(S_{t+1}|S_t, D_t)$ is the transition probability from state S_t to S_{t+1} by taking action D_t . The state transition is partially observable through variable $O_t \in \mathbf{O}$, where $\mathbf{O}$ is the observation space. The conditional observation distribution

is denoted as $\mathcal{Z}$. Taking an action D_t in state S_t induces a cost $C_t = \mathcal{C}(S_t, D_t) \in \mathbb{R}$. The objective in a POMDP is to find the sequence of T actions, $D_1, \dots, D_T$, to minimize the expected cumulative cost $\mathbb{E}[J]$, with discount factor $\gamma \in (0, 1]$:

$$J = \sum_{t=1}^T \gamma^{t-1} C_t \quad (1)$$

A belief state $b_t = \langle b_t(s_t) \rangle_{s_t \in \mathbf{S}}$ is associated with time t , where $b_t(s) = \mathbb{P}[S_t = s | h_t]$ with $h_t = (\rho_1, o_1, d_1, o_2, \dots, d_{t-1}, o_t) \in \mathcal{H}$. The belief state is a distribution over the state space $\mathbf{S}$. At every time t , the belief is recursively computed:

$$b_{t+1}(s) = \eta^{-1} \mathcal{Z}(o_{t+1}|s, d_t) \sum_{s_t \in \mathbf{S}} \mathcal{T}(s|s_t, d_t) b_t(s_t) \quad (2)$$

where $\eta = \sum_{s \in \mathbf{S}} \mathcal{Z}(o_{t+1}|s, d_t) \sum_{s_t \in \mathbf{S}} \mathcal{T}(s|s_t, d_t) b_t(s_t)$ is the normalization factor. At the beginning of an episode, the initial state S_1 is sampled from the initial state distribution $\rho_1 : \mathbf{S} \rightarrow [0, 1]$, which also defines the initial belief state.

B. A POMDP Model of the Intrusion Response Use Case

This subsection defines the POMDP model of the defender’s problem. The system state captures attacker progression and process integrity, while the observation represents IDS-derived network measurements. The transition function models the interaction between attacker and defender actions, and the cost function encodes the defender’s objective. The resulting POMDP defines the optimization problem for computing an optimal defender strategy.

1) *Scenario configuration*: To simplify notation, we denote the supervisory hosts and the processes in Fig. 1 by $h_0 = \text{HMI}_0$, $h_1 = \text{HMI}_1$, and $h_2 = \text{EWS}_0$, $p_0 = \text{Tank}_0$ and $p_1 = \text{Tank}_1$. Let $\mathbf{H} = \{h_0, h_1, h_2\}$ be the set of hosts and $\mathbf{P} = \{p_0, p_1\}$ be the set of tanks. In this topology, each supervisory host $h \in \mathbf{H}$ can monitor and control both p_0 and p_1 through their corresponding control units, PLC_0 and PLC_1 . During an episode, the attacker follows a fixed strategy $\pi_A \in \Pi_A$, where Π_A is the attacker strategy space.

2) *System state space $\mathbf{S}$ and initial state distribution ρ_1* : The system state S_t at time t represents the states of the supervisory hosts, the states of the industrial processes, and the attacker action state from the previous time step. Formally, S_t is defined as the tuple $S_t = (S_t^{h_0}, S_t^{h_1}, S_t^{h_2}, S_t^{p_0}, S_t^{p_1}, A_{t-1})$.

For each host $h \in \mathbf{H}$, its state $S_t^h \in \mathbf{S}^h = \{\text{U}, \text{D}, \text{S}, \text{E}, \text{I}\}$ represents the attacker’s progression on host h at time t . The values denote whether h is undiscovered by the attacker (U), discovered by the attacker (D), scanned by the attacker (S), exploited and accessed by the attacker (E), or inspected for PLC control by the attacker (I). These states are motivated by MITRE ATT&CK framework [36] and reflect the main attack stages of an OT cyberattack, where an attacker first discovers assets, then gathers information, gains access, and learns control patterns.

For each process $p \in \mathbf{P}$, its state $S_t^p \in \mathbf{S}^p = \{\text{W}, \text{C}\}$ denotes the operation condition of the water tanks. The values denote

whether p is in its normal operation state ($\mathbb{W}$) or has been corrupted by the attacker ($\mathbb{C}$).

The component A_{t-1} denotes the attacker action at time $t-1$ and is defined as $A_{t-1} = (A_{t-1}^h, A_{t-1}^p)$. A_{t-1}^h represents the targeted host at time $t-1$ and takes three types of values: $\emptyset$, if no host is targeted; $h \in \mathbf{H}$, if host h is targeted; and $\mathbf{H}$, if the attacker scans the supervisory subnet. A_{t-1}^p represents the targeted process at time $t-1$ and can be either $\emptyset$ or a process $p \in \mathbf{P}$. We require $A_{t-1}^p \neq \emptyset$ only when $A_{t-1}^h \in \mathbf{H}$, meaning that process tampering is performed through a targeted host. The attacker action space is $\mathbf{A} = \{\mathbf{H} \cup \{\emptyset, \mathbf{H}\}\} \times \{\mathbf{P} \cup \{\emptyset\}\}$.

The state space is defined as $\mathbf{S} = \mathbf{S}^h \times \mathbf{S}^p \times \mathbf{A}$. At time $t = 1$, no intrusion has occurred, hence, the initial state distribution ρ_1 is a degenerate distribution with $\rho_1(s) = 1$, where $s = (\mathbb{U}, \mathbb{U}, \mathbb{U}, \mathbb{W}, (\emptyset, \emptyset))$.

3) *Defender action space $\mathbf{D}$* : At time t , the defender takes an action $D_t = (D_t^h, D_t^p)$. D_t^h is the target host(s) and D_t^p is the target process(es) at time t . D_t can take one of the following values: $(D_t^h = \emptyset, D_t^p = \emptyset)$ if the defender passively monitors the system; $(D_t^h = h \in \mathbf{H}, D_t^p = \emptyset)$ if the defender resets host h ; $(D_t^h = \emptyset, D_t^p = p \in \mathbf{P})$ if the defender resets process p ; and $(D_t^h = \mathbf{H}, D_t^p = \mathbf{P})$ if the defender resets the whole system. The defender action space is thereby defined as $\mathbf{D} = \{(\emptyset, \emptyset)\} \cup \{(h, \emptyset) : h \in \mathbf{H}\} \cup \{(\emptyset, p) : p \in \mathbf{P}\} \cup \{(\mathbf{H}, \mathbf{P})\}$.

4) *Transition function $\mathcal{T}$* : The state transition $\mathcal{T}(S_{t+1}|S_t, D_t)$ from time t to $t+1$ can be described through the transitions of each component of the system state: (a) $A_{t-1} \rightarrow A_t$; (b) $S_t^h \rightarrow S_{t+1}^h$; and (c) $S_t^p \rightarrow S_{t+1}^p$. The transitions are presented as follows.

a) *Attacker action $A_{t-1} \rightarrow A_t$* : The attacker action A_t at time t is defined by the system state S_t and its strategy π_A , i.e., $A_t = \pi_A(S_t)$.

b) *Host state $S_t^h \rightarrow S_{t+1}^h$* : The state S_{t+1}^h of a host $h \in \mathbf{H}$ is dependent on its previous state S_t^h , the attacker action A_t^h and the defender action D_t^h .

$$S_t^h = \mathbb{U} \rightarrow S_{t+1}^h = \mathbb{D} \text{ if } A_t^h = \mathbf{H}, h \notin D_t^h \quad (3a)$$

$$S_t^h = \mathbb{D} \rightarrow S_{t+1}^h = \mathbb{S} \text{ if } A_t^h = \mathbf{h}, h \notin D_t^h \quad (3b)$$

$$S_t^h = \mathbb{S} \rightarrow S_{t+1}^h = \mathbb{E} \text{ if } A_t^h = \mathbf{h}, N_e = 1, h \notin D_t^h \quad (3c)$$

$$S_t^h = \mathbb{S} \rightarrow S_{t+1}^h = \mathbb{S} \text{ if } A_t^h = \mathbf{h}, N_e = 0, h \notin D_t^h \quad (3d)$$

$$S_t^h = \mathbb{E} \rightarrow S_{t+1}^h = \mathbb{I} \text{ if } A_t^h = \mathbf{h}, h \notin D_t^h \quad (3e)$$

$$S_t^h \rightarrow S_{t+1}^h = \mathbb{U} \text{ if } h \in D_t^h \quad (3f)$$

$$S_t^h \rightarrow S_{t+1}^h = S_t^h \text{ otherwise} \quad (3g)$$

where N_e is a binary random variable that determines the success of the attacker's *Exploit* action on host h .

(3a-3g) describe the transition of the host state $S_t^h \rightarrow S_{t+1}^h$, $\forall h \in \mathbf{H}$. (3a) describes the attacker's network reconnaissance action on the supervisory subnet, which exposes the hosts $\mathbf{H}$ to the attacker and causes the transition $\mathbb{U} \rightarrow \mathbb{D}$. (3b) describes the transition $\mathbb{D} \rightarrow \mathbb{S}$, where the attacker performs a host-level discovery on host h . (3c) and (3d) describe the exploitation attempt of the attacker on host h , which has a success probability defined by a binary random variable N_e . After gaining access to host h , the attacker inspects it to learn

control patterns to the underlying physical processes, which is presented by the transition $\mathbb{E} \rightarrow \mathbb{I}$. All transitions in (3a-3e) occur under the condition that the defender does not reset host h , which brings S_{t+1}^h to the secure state $\mathbb{U}$ (3f).

The transitions are illustrated in Fig. 2, which describes the attacker actions on a host toward state $\mathbb{I}$. The defender can reset the host and bring it to the secure state $\mathbb{U}$.

c) *Process state $S_t^p \rightarrow S_{t+1}^p$* : The state S_{t+1}^p of a process $p \in \mathbf{P}$ is dependent on its state S_t , the attacker action A_t^p and the defender action D_t^p . The transition is illustrated in Fig. 2.

$$S_t^p \rightarrow S_{t+1}^p = \mathbb{C} \text{ if } A_t^p = p, A_t^h = h, S_t^h = \mathbb{I}, p \notin D_t^p \quad (4a)$$

$$S_t^p \rightarrow S_{t+1}^p = \mathbb{W} \text{ if } p \in D_t^p \quad (4b)$$

$$S_t^p \rightarrow S_{t+1}^p = S_t^p \text{ otherwise} \quad (4c)$$

(4a-4c) describe the transition of the process state $S_t^p \rightarrow S_{t+1}^p$, $\forall p \in \mathbf{P}$. (4a) describes the process tampering action, which requires an inspected host ($\exists h \in \mathbf{H} : S_t^h = \mathbb{I}$), and the process is not reset by the defender at the same time ($p \notin D_t^p$). (4b) describes the defender resetting process p to its normal working state.

5) *Defender observation space $\mathbf{O}$* : The defender cannot observe the system state S_t . Instead, the defender observes

$$O_t = (G_t, O_t^{h_0}, O_t^{h_1}, O_t^{h_2}, O_t^{p_0}, O_t^{p_1}) \quad (5)$$

where G_t represents the number of IP packets between the subnets, $O_t^{h_i}$ represents the number of IP packets between supervisory host i and the enterprise subnet, and $O_t^{p_j}$ represents the number of IP packets between PLC $_j$ and the supervisory subnet, during time step t .

6) *Defender observation function $\mathcal{Z}$* : The observation function is the conditional probability mass function of the discrete observation vector O_{t+1} :

$$\mathcal{Z}(o, s, d) = \mathbb{P}[O_{t+1} = o \mid S_{t+1} = s, D_{t+1} = d] \quad (6)$$

where $s \in \mathbf{S}$, $d \in \mathbf{D}$, and $o \in \mathbf{O}$.

7) *Cost function $\mathcal{C}$ and defender objective J* : At each time t , the defender pays a cost C_t :

$$C_t = \sum_{h \in \mathbf{H}} \sigma_h(S_t^h) + \sum_{p \in \mathbf{P}} \sigma_p(S_t^p) + \sigma_d(D_t) \quad (7)$$

where σ_h , σ_p , and σ_d are parameter functions, which reflect the defender's objective by denoting costs for the host states, the process states and the defensive actions. The functions are specified in Tab. I. The high cost for process corruption, $\sigma_p(\mathbb{C}) = 10$, prioritizes the safety of the physical processes. The increasing host state costs σ_h penalize the defender for attacker progression and encourage preventive responses. The action cost σ_d penalizes resets, preventing unnecessarily disruptive responses.

Hence, the objective of the defender is formulated as minimizing the expected cumulative cost C_t over the time horizon T with discount factor $\gamma = 1$:

$$J = \sum_{t=1}^T \mathbb{E}[C_t] \quad (8)$$

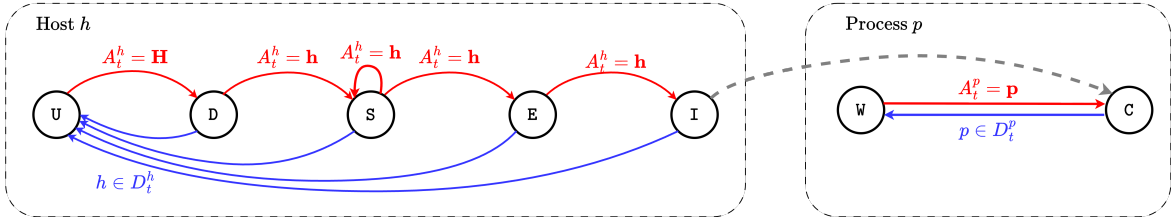


Fig. 2: The transitions of host states S_t^h and the process states S_t^p caused by attacker action A_t (red arrows) and defender action D_t (blue arrows). The dashed arrow illustrates that host h must be in state I for the attacker to tamper with process p using host h ($A_t^h = h, A_t^p = p$).

TABLE I: Parameters of the cost function $\mathcal{C}$

Cost function	Value
σ_h	$\sigma_h(U) = \sigma_h(D) = 0, \sigma_h(S) = 0.2, \sigma_h(E) = 1.5, \sigma_h(I) = 3.5$
σ_p	$\sigma_p(W) = 0, \sigma_p(C) = 10$
σ_d	$\sigma_d(D_t^h, D_t^p) = 2 \times D_t^h + 3 \times D_t^p $

8) *The defender's optimization problem:* We define $\pi_D : \mathcal{H} \rightarrow \mathbf{D}$ or $\pi_D : \mathbf{B} \rightarrow \mathbf{D}$ as the defender strategy, where $\mathcal{H}$ is the history space and $\mathbf{B}$ is the belief space. The defender problem is defined as finding the optimal strategy π_D^* that minimizes the expected cumulative cost J .

Problem 1. Find the optimal defender strategy :

$$\pi_D^* = \arg \min_{\pi_D} \mathbb{E}_{\pi_D}[J] \quad (9a)$$

$$\text{subject to } D_t = \pi_D(b_t) \forall t \quad (9b)$$

$$\pi_A \sim P(\Pi_A) \quad (9c)$$

Because the POMDP has finite state, action, and observation spaces over a finite horizon, an optimal strategy π_D^* exists [25, Thm. 7.4.1].

V. LEARNING DEFENDER STRATEGIES WITH REINFORCEMENT LEARNING

Problem 1 can be solved with Dynamic Programming methods [37]. Due to the large state space, such methods are computationally intractable for our use case [38]. We therefore parameterize the defender strategy π_D with parameter θ . We apply PPO [39], a state-of-the-art RL algorithm, to learn the (almost) optimal defender strategy. PPO is an actor-critic policy-gradient method that uses a clipped surrogate objective to improve training stability by limiting policy changes between updates.

A direct application of PPO to Problem 1 is not feasible due to the large state space and belief space. We therefore propose two PPO-based solution methods that use different techniques to address the issue.

A. PPO with k latest observations: k -Obs-PPO

The first method, k -Obs-PPO, uses the k most recent IDS observations, $(O_t, \dots, O_{t-k+1})$ as input for π_D^θ . It allows training the defender strategy directly from observation histories, without computing the belief state. The parameter k controls the trade-off between the amount of information used to learn the strategy on the one hand and the computational

complexity and sampling efficiency on the other hand. A larger k provides more temporal information to the defender strategy but increases the input dimension and the learning complexity. A smaller k keeps the input small and simplifies defender strategy learning.

B. Belief Filter Proximal Policy Optimization: BF-PPO

The second method, BF-PPO, uses an approximate and simplified belief state to learn the defender strategy. In a POMDP, belief updates are computed using Bayes filter (Eq. (2)). Such an update has a quadratic time complexity with respect to the size of the state space. Therefore, the Bayes Filter is computationally infeasible in our case. We address the issue by approximating the belief using particle filter [26]. At time t , the particle filter represents the belief by M sampled states, $\mathcal{P}_t = \{s_t^{(1)}, \dots, s_t^{(M)}\}$. The belief state is then approximated by the relative frequency of each system state in $\mathcal{P}_t$, i.e., $\hat{b}_t(s) = \frac{1}{M} \sum_{i=1}^M \mathbb{1}\{s_t^{(i)} = s\}$.

Due to the large state space, the belief vector $\langle \hat{b}_t(s) \rangle$ is high dimensional, which increases the learning complexity and makes it inefficient as input to the defender strategy. We therefore exploit the factorized structure of the system state with components $S_t^{h_0}, S_t^{h_1}, S_t^{h_2}, S_t^{p_0}, S_t^{p_1}, A_{t-1}$. For each such component X_t of the state and component state x , we compute the marginal belief

$$\hat{b}_t^X(x) = \frac{1}{M} \sum_{i=1}^M \mathbb{1}\{X_t^{(i)} = x\}. \quad (10)$$

The concatenation of these marginal beliefs is used as input to the defender strategy π_D^θ , providing a compact input while preserving the information about each component of the system state.

VI. CONSTRUCTING AN EMULATION ENVIRONMENT AND A SIMULATION ENVIRONMENT TO TRAIN AND EVALUATE DEFENDER STRATEGIES

Learning defender strategies using our solution methods and evaluating them requires repeated POMDP episodes. Such episodes cannot be performed directly in an operating OT infrastructure, which is both inefficient due to the large number of episodes required for training and impractical because it requires performing attacker actions in the system. For these reasons, we develop an emulation environment and a simulation environment. The emulation environment is designed to closely match the operational conditions of

the target infrastructure. It serves two purposes: collecting IDS measurements to identify the observation model of the POMDP, and evaluating defender strategies under realistic conditions. The simulation environment generates POMDP episodes to efficiently train the defender strategies.

A. Constructing the Emulation Environment

The infrastructure in Fig. 1 is emulated with Docker [40] containers and orchestrated by ContainerLab [41]. Each functional component is instantiated as a separate container, including the enterprise clients, supervisory hosts, PLCs, water tanks, router, attacker, defender, and IDS. The PLCs are implemented using OpenPLCv3 [42]. The main communication protocol between OT components (HMI, PLC, EWS, and water tanks) is implemented using ModbusTCP [43]. The configuration of the components is listed in Appendix A.

The network is implemented using Linux virtual Ethernet interfaces. Logical network segmentation is enforced through VLAN configurations using Open vSwitch [44]. Inter-subnet routing and firewall rules are implemented by the router container. The IDS is connected to a mirrored Open vSwitch port, which allows it to observe and collect network measurements.

The defender's reset action on a host or a process changes the IP address and login credentials of the target component, thereby removing potential attacker access. The updated configuration is then propagated to the components that require it. For example, when a PLC is reset, its new IP address is distributed to the HMIs and the EWS so that supervisory communication can be re-established.

The attacker actions, including reconnaissance and exploitation, are implemented by a sequence of commands executed from the attacker container. To capture a broad range of traffic patterns for the same action, the implementation varies command parameters, including target networks and ports, probe types, command arguments, and payload sizes. This allows us to observe traffic under a wide range of attacker behaviors. More details of the implementation are presented in Appendix A.

Background traffic is generated by three types of activities: by enterprise clients using services on the HMI and EWS containers; by HMI-PLC communication, where HMIs periodically read process measurements from the PLCs and issue routine control commands; and by EWS-PLC communication, representing engineering operations performed by an operator. All non-periodic background activities are generated by independent Poisson processes.

B. Estimating the observation function in the emulation environment

In the emulation environment, we perform operation in periods of 30 seconds wall clock time. During such a period, the attacker performs an attack action according to its strategy. Also, the defender performs a defensive action based on the observation data collected during the period. This data consists of network statistics obtained from the IDS, namely, $o = (g, o^{h_0}, o^{h_1}, o^{h_2}, o^{p_0}, o^{p_1})$ (Eq. (5)).

To estimate the observation function $\mathcal{Z}$ (Eq. (6)) of the POMDP, we collect monitoring observation data from 40000 periods, which amounts to 14 days. Effectively estimating $\mathcal{Z}(o, s, d) = \Pr(O_{t+1} = o \mid S_{t+1} = s, D_t = d)$ is not feasible since it would require approximately 10^8 observations ($|S||D| = 52500$ possible conditioning combinations of (s, d) and assuming about 2000 observations needed for each combination). To work with 40000 observation measurements, we simplify the observation function $\mathcal{Z}$ to $\hat{\mathcal{Z}}$ by setting $\hat{\mathcal{Z}}(o, a) = \Pr(O_{t+1} = o \mid A_t = a)$, where a denotes the attacker action. Fig. 3 shows empirical distributions for selected attacker actions.

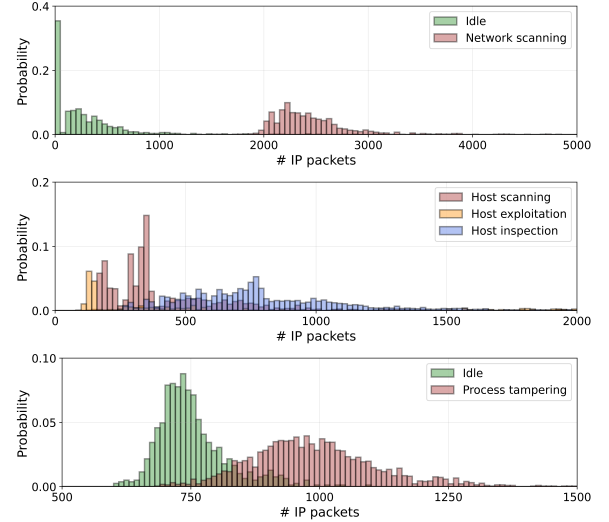


Fig. 3: Empirical distribution of IP packet counts measured on the switch in Fig. 1. The top figure relates to the total traffic passed through the switch; the middle figure relates to the traffic between EWS-0 and entities in the enterprise subnet; the lower figure relates to the traffic between PLC-0 and entities in the supervisory subnet. Different colors refer to measurements from different attacker actions.

C. Running the POMDP simulation

Simulating a POMDP episode starts the system in state $s = (U, U, U, W, W, (\emptyset, \emptyset))$. At each time step, the attacker and the defender perform an action following their respective strategies. The system state is then updated following the transition function $\mathcal{T}$ specified in §IV. The observation that the defender makes during the time step is sampled from $\hat{\mathcal{Z}}$. Further, the cost incurred for this time step is computed using the cost function $\mathcal{C}$. The episode ends when the T time steps have been executed. The trajectories of observations, actions, and costs per period are used to update the defender strategies during training.

VII. COMPUTING THE DEFENDER STRATEGIES IN THE SIMULATION ENVIRONMENT AND EVALUATING THEM IN THE EMULATION ENVIRONMENT

We train BF-PPO and two k -Obs-PPO variants, with $k = 1$ and $k = 4$ in the simulation environment. We then evaluate the obtained defender strategies in the emulation environment.

We compare the proposed methods with two baselines, a threshold-based defender strategy and an idealized PPO baseline, where we assume the defender has full observation of the system states and the attacker actions. The training and evaluation are conducted on an Apple M3 Pro processor. The hyperparameters for the learning methods are listed in Appendix B. The source code is available at [45].

A. Evaluation setup

1) *Attacker strategies*: We train and evaluate the methods against three attacker strategies:

a) *Opportunistic*: The attacker prioritizes reaching process tampering as quickly as possible. It selects actions that advance the attack along the shortest available path toward tampering with a process.

b) *Explorative*: The attacker prioritizes gathering knowledge about the infrastructure before tampering with a process. It scans supervisory hosts, exploits available entry points, and inspects host-process control relationships before tampering with the processes.

c) *Adaptive*: This attacker adapts to defender actions by avoiding recently reset components and switching to alternative attack paths when possible.

2) Baselines:

a) *MDP-PPO*: The method applies PPO whereby the system state can be observed by the defender. This baseline method represents an idealized setting and a cost lower bound for the PPO-based methods proposed in this paper.

b) *Threshold strategy*: At each time step t , the defender computes a cost score R_h^t for each host and a cost score R_p^t for each process. If the score of any process exceeds its respective threshold, the defender following this strategy resets the process with the highest score. Otherwise, if the score of any host exceeds its respective threshold, the defender resets the host with the highest score. The thresholds are determined by the reset costs σ_d (see Tab. I). The cost scores are given by the formulas in Appendix C.

3) *Evaluation process*: We train the defender strategies using BF-PPO, 1-Obs-PPO, and 4-Obs-PPO, as well as the baseline method MDP-PPO in the simulation environment. For each method, we perform four training runs with different seeds. A training run consists of 500 iterations, where each iteration contains 100 episodes with time horizon 100. After each iteration, we evaluate the current policy to obtain a point in the learning curve.

We finally evaluate the learned defender strategies in the emulation environment. The strategy obtained by each method is evaluated against the three attacker strategies for 20 episodes with time horizon 100. Evaluating a defender strategy against an attacker strategy takes about 17 hours on an Apple M3 Pro processor.

B. Evaluating the methods in the simulation environment

Fig. 4 shows the learning curves of the proposed methods and the baselines against three attacker strategies. BF-PPO, 1-Obs-PPO, and 4-Obs-PPO, are represented by the green, blue,

and orange curves, respectively. MDP-PPO and Threshold strategy are shown by red curves and dashed lines, respectively. Each plot corresponds to an attacker strategy and shows the natural logarithm of the cumulative cost during training.

The learning curves show that 1-Obs-PPO and 4-Obs-PPO converge to higher cumulative costs than BF-PPO for all attacker strategies. They also exhibit larger variation. 1-Obs-PPO tends to perform better than 4-Obs-PPO, which suggests that, in this setting, increasing the observation history does not provide sufficient additional information to improve performance.

BF-PPO converges faster and to lower cumulative costs than the k -Obs-PPO variants for all attacker strategies. Its learning curves are also close to those of MDP-PPO, the full-observability baseline. This indicates that using particle filter and marginal beliefs provides a solution for learning effective defender strategies. Further, Fig. 4 also shows that BF-PPO consistently outperforms the Threshold strategy.

C. Evaluating the defender strategies in the emulation environment

Fig. 5 shows the evaluation results of the defender strategies in the emulation environment (green color) and compares them with the results from the simulation environment (red color). The box plots show the distributions of cumulative costs over 20 evaluation episodes for BF-PPO, 1-Obs-PPO and 4-Obs-PPO. The dashed line shows the average cumulative cost of MDP-PPO, the full-observability baseline method.

The plots show that BF-PPO outperforms the k -Obs-PPO variants for all attacker strategies. Its cumulative costs are also close to those of MDP-PPO, which is consistent with the simulation results in Fig. 4.

BF-PPO shows a gap between the performance in the emulation environment and the simulation environment. We expect that the defender strategy performs less well in the emulation environment than in the simulation environment. This is due to the fact that the formal model we have presented is an approximation of the more realistic emulated system, and the defender strategy has been optimized for the simulation environment.

Overall, we find that BF-PPO is the strongest proposed method for our use case and the considered attacker strategies.

VIII. CONCLUSION AND FUTURE WORK

Based on the POMDP model of the use case, we have defined an optimal defender strategy and proposed two solution methods for learning the approximation of this strategy. The solution methods allow us to learn the defender strategies at a reasonable cost and with low sample complexity. The evaluation in the emulation environment shows that effective defender strategies can be learned with these methods. In particular, BF-PPO approaches the performance of a baseline with full observability.

As for future work, we plan to evaluate the proposed model and solution methods on an industrial testbed in collaboration with an industrial partner, for an intrusion response use case

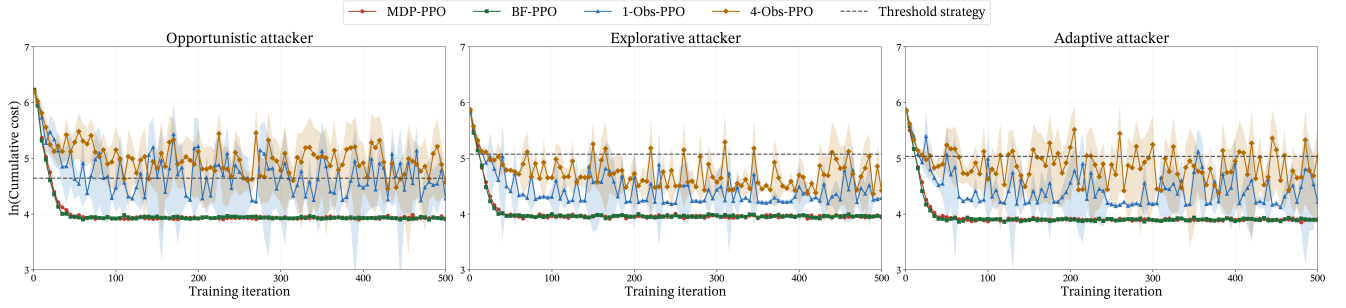


Fig. 4: The learning curves for the solution methods proposed in §V (BF-PPO – green curves, 1-Obs-PPO – blue curves, and 4-Obs-PPO – orange curves) and the baselines (MDP-PPO – red curves and Threshold strategy – dashed lines). The solution methods computed effective defender strategies for three different attacker strategies (Opportunistic, Explorative and Adaptive). The learning curves show the average and the confidence intervals of 120 episodes. The best performing solution method with respect to convergence speed and cost is BF-PPO.

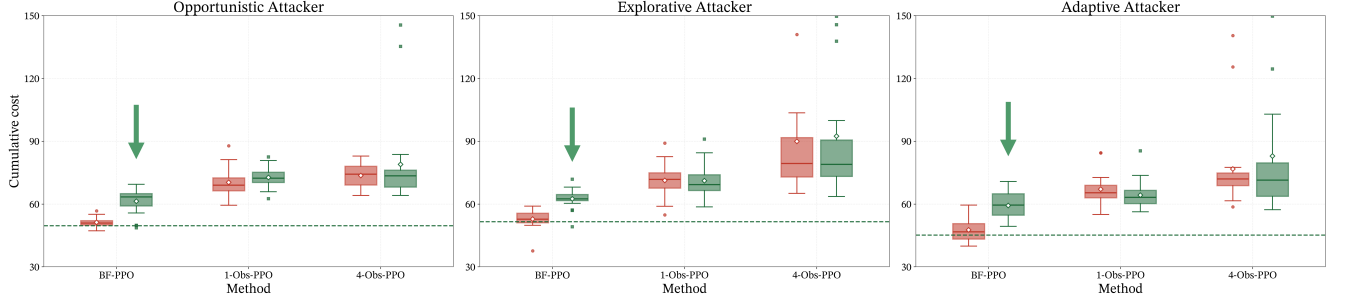


Fig. 5: Evaluation of the learned defender strategies for the proposed solution methods, BF-PPO, 1-Obs-PPO and 4-Obs-PPO. The green box plots relate to evaluation results in the emulation environment. For comparison, we add the results from the simulation environment, presented by red box plots. The baseline refers to the learned defender strategies under the assumption of full observability of the system states.

similar to the one studied in this paper. This evaluation will allow us to study our approach to modeling and computing defender strategies in a different and realistic environment and further to include safety constraints.

IX. ACKNOWLEDGMENT

This work has been supported by the DARPA CASTLE program through project ORLANDO and by the WASP NEST program through project AIRR. The authors thank KTH researchers Kim Hammar and Xiaoxuan Wang for their constructive comments.

APPENDIX

A. Emulation configurations: Tab. II and Tab. III

B. Training hyperparameters: Tab. IV

TABLE II: Configuration of the emulated physical components

Component	Services	Vulnerabilities
HMI ₀ , HMI ₁	HTTP	Weak credentials
EWS ₀	SSH, Telnet, SMB	SSH/Telnet weak credentials, CVE-2017-7494
PLC ₀ , PLC ₁	ModbusTCP, HTTP	–
Tank ₀ , Tank ₁	ModbusTCP	–

TABLE III: Implementation of attacker actions in the emulation

Attacker action	Implementation
Scan network	ICMP scan
Scan host	TCP/UDP/OS scan, lightweight probes
Exploit host	HMI: HTTP brute-force, SQL injection, EWS: SSH/Telnet brute-force, SambaCry
Inspect host	HMI: HTTP probes, EWS: ModbusTCP control
Tamper process	HMI: PLC parameter modifications via web interface, EWS: Craft & Upload PLC programs, PLC parameter control via ModbusTCP

TABLE IV: Hyperparameters for the training of defender strategies

Parameter	Value
# particles (BF-PPO) M	500
# hidden layer, # neurons, random seeds	2, 128, [77, 108, 433, 841]
l.r. α , γ , GAE λ , ENT-COEF, clip ϵ	0.0001, 0.99, 0.95, 0.001, 0.2
Optimiser (parameters)	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.99$)

C. Threshold strategy's cost score formulas

Eq. (11) defines the formulation of the cost score for each host and process at time t , where b_t denotes the belief state.

$$R_h^t = \sum_{s \in S^h} b_t^h(s) \sigma_h(s); \quad R_p^t = \sum_{s \in S^p} b_t^p(s) \sigma_p(s) \quad (11)$$

REFERENCES

- [1] Waterfall Security Solutions and ICS Strive, “2024 threat report: Ot cyberattacks with physical consequences,” Waterfall Security Solutions,

- Technical Report, 2024.
- [2] Dragos, Inc., “2025 ot/ics cybersecurity report: A year in review,” Dragos Inc., Technical Report, 2025.
 - [3] A. Di Pinto, Y. Dragoni, and A. Carcano, “Triton: The first ics cyber attack on safety instrument systems,” *Proc. Black Hat USA*, vol. 2018, pp. 1–26, 2018.
 - [4] D. D. McCarthy, “Precursor analysis report: Industroyer2 and wiper malware targeting ukrainian energy provider 2022,” Idaho National Laboratory (INL), Idaho Falls, ID (United States), Tech. Rep., 2025.
 - [5] M. Dobler, M. Hellwig, N. Lopes, K. Oakley, and M. Winterburn, “Systematic review and characterisation of malicious industrial network traffic datasets: M. dobler et al.” *International Journal of Information Security*, vol. 24, no. 5, p. 208, 2025.
 - [6] G. M. Makrakis, C. Kolias, G. Kambourakis, C. Rieger, and J. Benjamin, “Industrial and critical infrastructure security: Technical analysis of real-life security incidents,” *IEEE Access*, vol. 9, p. 165295–165325, 2021.
 - [7] M. Rodda and V. Mavroudis, “Analysis of publicly accessible operational technology and associated risks,” *arXiv preprint arXiv:2508.02375*, 2025.
 - [8] P. Ackerman, *Industrial Cybersecurity: Efficiently monitor the cybersecurity posture of your ICS environment*. Packt Publishing Ltd, 2021.
 - [9] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule, and M. Thompson, “Guide to operational technology (ot) security,” Gaithersburg, MD, 9 2023.
 - [10] International Society of Automation, “IEC 62443 Series of Standards,” <https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards>, 2026.
 - [11] J. Vaudey, S. Mocanu, G. Delaval, and E. Rutten, “Reconfiguration of firewall filter rules as a response to industrial control system intrusion,” in *2025 IEEE Conference on Communications and Network Security (CNS)*, 2025, pp. 1–6.
 - [12] A. F. Murillo Piedrahita, V. Gaur, J. Giraldo, Á. A. Cárdenas, and S. J. Rueda, “Leveraging software-defined networking for incident response in industrial control systems,” *IEEE Software*, vol. 35, no. 1, pp. 44–50, 2018.
 - [13] A. F. M. Piedrahita, V. Gaur, J. Giraldo, A. A. Cardenas, and S. J. Rueda, “Virtual incident response functions in control systems,” *Computer Networks*, vol. 135, pp. 147–159, 2018.
 - [14] A. Babar, T. Halabi, and M. Zulkernine, “Autonomous and adaptive cyber incident detection and response in industrial cyber-physical systems using hierarchical reinforcement learning,” *ACM Trans. Cyber-Phys. Syst.*, vol. 10, no. 1, Jan. 2026.
 - [15] Y. Yue, D. Zhao, Y. Zhou, L. Xu, Y. Tang, and H. Peng, “An intrusion response approach based on multi-objective optimization and deep q network for industrial control systems,” *Expert Systems with Applications*, vol. 272, p. 126664, 2025.
 - [16] X. Li, C. Zhou, Y.-C. Tian, and Y. Qin, “A dynamic decision-making approach for intrusion response in industrial control systems,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2544–2554, 2019.
 - [17] H. Chen, Y. Lai, J. Liu, and H. Wanyan, “Interpretable cross-layer intrusion response system based on deep reinforcement learning for industrial control systems,” *IEEE Transactions on Industrial Informatics*, vol. 20, no. 7, pp. 9771–9781, 2024.
 - [18] S. Xu, Z. Xie, C. Zhu, X. Wang, and L. Shi, “Enhancing cybersecurity in industrial control system with autonomous defense using normalized proximal policy optimization model,” in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 928–935.
 - [19] L. Chen, Y. Lai, P. Zhao, B. Xie, and Y. Zhang, “Safety-aware intrusion response system based on safe reinforcement learning for cyber-physical systems,” *IEEE Transactions on Consumer Electronics*, vol. 72, no. 2, pp. 2711–2723, 2026.
 - [20] J. Mern, K. B. Hatch, R. Silva, J. S. Brush, and M. J. Kochenderfer, “Reinforcement learning for industrial control network cyber security orchestration,” *ArXiv*, vol. abs/2106.05332, 2021.
 - [21] Y. Lai, P. Zhao, and Z. Wang, “Mfir: Model-free intrusion response for partially observable industrial control systems,” *IEEE Internet of Things Journal*, pp. 1–1, 2026.
 - [22] P. Yao, Z. Jiang, B. Yan, Q. Yang, and W. Wang, “Bayesian and stochastic game joint approach for cross-layer optimal defensive decision-making in industrial cyber-physical systems,” *Information Sciences*, vol. 662, p. 120216, 2024.
 - [23] K. Zhong, Z. Yang, G. Xiao, X. Li, W. Yang, and K. Li, “An efficient parallel reinforcement learning approach to cross-layer defense mechanism in industrial control systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2979–2990, 2022.
 - [24] T. J. Williams, “The purdue enterprise reference architecture,” *Computers in Industry*, vol. 24, no. 2, pp. 141–158, 1994.
 - [25] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.
 - [26] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
 - [27] E. Miehling, M. Rasouli, and D. Teneketzis, “A pomdp approach to the dynamic defense of large-scale cyber networks,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2490–2505, 2018.
 - [28] D. H. Le and R. Stadler, “Learning optimal defender strategies for cage-2 using a pomdp model,” in *2025 21st International Conference on Network and Service Management (CNSM)*, 2025, pp. 1–9.
 - [29] L. Zhang, T. Zhu, F. K. Hussain, D. Ye, and W. Zhou, “A game-theoretic method for defending against advanced persistent threats in cyber systems,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1349–1364, 2023.
 - [30] K. Hammar and R. Stadler, “Finding effective security strategies through reinforcement learning and self-play,” in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.
 - [31] K. Hammar, N. Dhir, and R. Stadler, “Optimal defender strategies for cage-2 using causal modeling and tree search,” *arXiv preprint arXiv:2407.11070*, 2024.
 - [32] J. Nyberg, P. Johnson, and A. Méhes, “Cyber threat response using reinforcement learning in graph-based attack simulations,” in *2022 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2022, pp. 1–4.
 - [33] K. Hammar, T. Alpcan, and E. C. Lupu, “Incident response planning using a lightweight large language model with reduced hallucination,” 2025.
 - [34] A. Wilson, R. Menzies, N. Morarji, D. Foster, M. C. Mont, E. Turkbeyler, and L. Gralowski, “Multi-agent reinforcement learning for maritime operational technology cyber security,” 2024.
 - [35] K. Hammar, “Optimal security response to network intrusions in it systems,” PhD dissertation, Kungliga Tekniska högskolan, Stockholm, 2024.
 - [36] B. Al-Sada, A. Sadighian, and G. Oliveri, “Mitre att&ck: State of the art and way forward,” *ACM Comput. Surv.*, vol. 57, no. 1, Oct. 2024.
 - [37] E. J. Sondik, “The optimal control of partially observable markov processes over the infinite horizon: Discounted costs,” *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.
 - [38] D. Burago, M. de Rougemont, and A. Slissenko, “On the complexity of partially observed markov decision processes,” *Theoretical Computer Science*, vol. 157, no. 2, pp. 161–183, 1996.
 - [39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.
 - [40] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” Houston, TX, Mar. 2014.
 - [41] Nokia SR Linux Labs, “Containerlab,” 2026, container-based networking lab orchestrator. [Online]. Available: <https://containerlab.dev>
 - [42] T. Alves and T. H. Morris, “OpenPLC: An IEC 61131–3 compliant open source industrial controller for cyber security research,” *Computers & Security*, vol. 78, pp. 364–379, 2018.
 - [43] *MODBUS Messaging on TCP/IP Implementation Guide*, Modbus Organization, 2006, accessed: 2026-06-02. [Online]. Available: https://assets.noviams.com/novi-file-uploads/modbus/pdfs-and-documents/Modbus_Messaging_Implementation_Guide_V1_0b.pdf
 - [44] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, “The design and implementation of open vswitch,” in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’15. USA: USENIX Association, 2015, p. 117–130.
 - [45] D. H. Le, “Intrusion Response in OT systems (implementation),” <https://github.com/duchuy108/viper>, May 2026.