

Learning Optimal Defender Strategies for CAGE-2 using a POMDP Model

Duc Huy Le, Rolf Stadler

Dept. of Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

Email: {dhle, stadler}@kth.se

Abstract—CAGE-2 is an accepted benchmark for learning and evaluating defender strategies against cyberattacks. It reflects a scenario where a defender agent protects an IT infrastructure against various attacks. Many defender methods for CAGE-2 have been proposed in the literature. In this paper, we construct a formal model for CAGE-2 using the framework of Partially Observable Markov Decision Process (POMDP). Based on this model, we define an optimal defender strategy for CAGE-2 and introduce a method to efficiently learn this strategy. Our method, called BF-PPO, is based on PPO, and it uses particle filter to mitigate the computational complexity due to the large state space of the CAGE-2 model. We evaluate our method in the CAGE-2 CybORG environment and compare its performance with that of CARDIFF, the highest ranked method on the CAGE-2 leaderboard. We find that our method outperforms CARDIFF regarding the learned defender strategy and the required training time.

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

I. INTRODUCTION

Traditionally, systems for intrusion detection and response have relied on rule sets that trigger alarms (e.g., [1], [2]). The rule sets have been defined and maintained by human experts. The increasing complexity and rapid changes of digital services and infrastructures have made the maintenance of these rule sets a challenging and a time-consuming task. As a response, research efforts into *automated cyberdefence* have started based on the idea that defender strategies can be dynamically learned and then executed by defender agents with minimal human intervention. One can say that the rules are no longer defined by humans, but automatically constructed from observing systems under attack.

Over the last decade, various learning approaches have been proposed for automated cyberdefence, including those based on reinforcement learning [3]–[5], stochastic modelling [6]–[8], game theory [9]–[11], and most recently, causal inference [12]–[14].

DISTRIBUTION STATEMENT A: Approved for public release; dissemination unlimited. This research is supported by the Defense Advanced Research Project Agency (DARPA) through the CASTLE program under Contract No.W912CG23C0029. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Currently, the most popular benchmark environment for learning and evaluating defender strategies is the Cyber Autonomy Gym for Experimentation Challenge 2 (CAGE-2) [15]. It is based on a scenario where a defender agent protects an IT infrastructure against different types of attacks (Fig. 1). CAGE-2 includes a simulation environment, named CybORG [16], in which defender agents can be trained and evaluated.

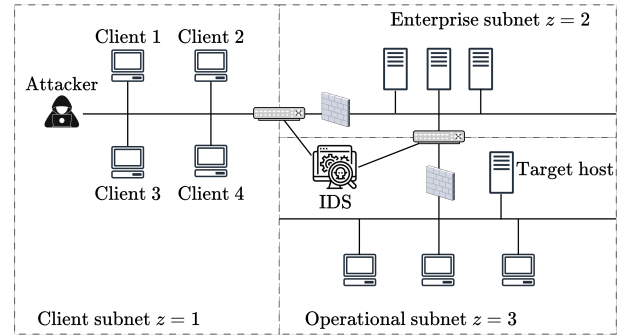


Fig. 1: The network topology of CAGE-2 scenario

A large number of defender methods for CAGE-2 have been proposed and published. A CAGE-2 leaderboard ranks the solutions according to a score that measures the effectiveness of the defenders against the attacks. The solutions on the leaderboard are based on heuristic approaches, and none of them is built on a formal model. As a consequence, it is not clear to which extent these solutions are optimal or close to optimal. (There is a recent work that uses a formal model, which we discuss in §II).

In this paper, we formalise the CAGE-2 scenario and develop a formal model using the framework of Partially Observable Markov Decision Process (POMDP) [17]. The model is obtained from studying the documentation and the source code of CAGE-2, as well as from conducting experiments in CybORG. Note that we do not assess the extent to which CAGE-2 captures a realistic scenario, or how CAGE-2 can be modified to better reflect reality. Our goal is to accurately model the CAGE-2 benchmark and its implementation.

The model allows us to define an optimal defender strategy for CAGE-2. To learn this strategy, we propose a learning method, which we call *Belief Filter Policy Proximal Optimisation* (BF-PPO). It is based on the state-of-the-art reinforcement

learning algorithm Policy Proximal Optimisation (PPO) [18], and it uses the concept of particle filter [19] to address the computational complexity arising from the large state space of the CAGE-2 model. We evaluate our method and compare its performance with CARDIFF [20], the method with the highest rank on the CAGE-2 leaderboard. We find that our method outperforms CARDIFF in terms of the learned defender strategy. Also, our method converges faster, requiring fewer training episodes.

The contributions of this paper are:

- We present a formal model of the CAGE-2 scenario using the POMDP framework (presented in §IV-B) and formally define an optimal defender strategy for CAGE-2 (presented in §IV-C).
- We present BF-PPO, a solution method that iteratively approximates the optimal strategy based on PPO and particle filter (presented in §IV-D)
- We evaluate BF-PPO in the CybORG environment and show that it outperforms the state-of-the-art method CARDIFF, the top performer on the CAGE-2 leaderboard (presented in §V).

II. RELATED WORK

The CAGE-2 challenge has attracted much attention in the cybersecurity community. More than 35 solution methods have been developed to learn defender strategies for the CAGE-2 scenario, for example [3], [12], [15], [21]–[25]. All of these methods, except for [12], rely on heuristic techniques and are not based on a formal model. As a consequence, it cannot be shown that any of these approaches lead to an optimal defender strategy. In contrast, we present in this paper a formal model of the CAGE-2 scenario, based on which we define an optimal strategy for the defender, and develop a solution to approximate this strategy.

The only published method that is based on a formal model of the CAGE-2 scenario is included in [12]. The study presents a structural causal model and proposes a defender strategy using Monte Carlo Planning. The method, however, is an online solution and is therefore of a different type than the above cited works, which all use offline training. This paper also uses offline training, which allows us to directly compare its performance with those works.

The POMDP framework has been recently used by many researchers to study intrusion detection and response [6]–[8], [26]–[28]. These works consider scenarios that are much simpler than CAGE-2 and the corresponding state spaces are much smaller. In order to address the computational complexity that results from larger state spaces, this paper uses the concept of particle filter for state estimation.

III. INTRUSION RESPONSE USE CASE: CAGE-2

The CAGE-2 scenario presents a cybersecurity use case in which a defender defends an IT infrastructure against an attacker with a fixed strategy. The network contains 11 hosts, divided into 3 subnets (see Fig. 1).

The attacker aims to access the target host located in the *operational subnet* and to disrupt the services it provides. After compromising one of the clients in the *client subnet*, the attacker tries to break into one of the hosts in the *enterprise subnet*, before finally attacking the target host. To achieve this goal, it performs a sequence of attack actions of the following types: (1) Discover the hosts on a subnet; (2) Scan for vulnerabilities in a host; (3) Exploit a discovered vulnerability to get access to a host; (4) Escalate the access to root access; (5) Interrupt the services on the host.

The defender cannot directly observe the progress of the attacker. Instead, it relies on events produced by an Intrusion Detection System (IDS), and responds by taking actions of the following types: (A) Analyse a host to check whether the attacker has performed an action (3), (4) or (5) on it; (B) Deploy a decoy service to deceive the attacker; (C) Neutralise and remove malware installed by the attacker; and (D) Restore the host into a safe state and restart the services. The goal of the defender is balancing the two objectives: maximising the service availability while minimising the access of the attacker.

An attack evolves over a finite number of time steps. During each time, both the attacker and the defender take an action.

IV. FORMALISING CAGE-2 WITH A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS MODEL

We formalise the CAGE-2 scenario introduced in Section III using the framework of Partially Observable Markov Decision Process (POMDP) and describe a learning solution method, BF-PPO, which is based on the state-of-the-art reinforcement learning algorithm PPO and the concept of particle filter.

In the following, we use the term CAGE-2 to refer to the CAGE-2 scenario as well as to the implementation of CAGE-2 in the CybORG platform.

A. Partially Observable Markov Decision Process

A Partially Observable Markov Decision Process (POMDP) models the progression of a discrete-time stochastic system with partial observability. It is defined by a 9-tuple $\mathcal{P} = (\mathbf{S}, \mathbf{D}, \mathcal{P}_{S_t, S_{t+1}}^{D_t}, \mathbf{O}, \mathcal{Z}_{O_{t+1}}^{D_t, S_{t+1}}, \mathcal{R}_{S_t}^{D_t}, \gamma, T, \mathbf{B})$. $\mathbf{S}$ denotes the state space and $\mathbf{D}$ denotes the action space. An episode in a POMDP begins in an initial state S_t . At every time $t = 1, \dots, T$ the system performs an action D_t , which moves the system state S_t to a new state S_{t+1} with transition probability $\mathcal{P}_{S_t, S_{t+1}}^{D_t} = \mathbb{P}[S_{t+1}|S_t, D_t]$. The state transition is partially observable through variable $O_t \in \mathbf{O}$, where $\mathbf{O}$ is the observation space. The observation function is defined as $\mathcal{Z}_{O_{t+1}}^{D_t, S_{t+1}} = \mathbb{P}[O_{t+1}|S_{t+1}, D_t]$. Associating with a state transition is a reward $R_t = \mathcal{R}_{S_t}^{D_t} \in \mathbb{R}$. The objective with a POMDP is to identify a sequence of T actions that maximises the expected cumulative reward $\mathbb{E}[J]$, with discount factor $\gamma \in (0, 1]$:

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

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 $h_t =$

$(S_1, D_1, O_2, \dots, D_{t-1}, O_t)$. The belief state is a distribution over the state space $\mathbf{S}$. At every time t , the belief is recursively computed:

$$b_t(S_t) = \eta Z_{O_t}^{D_{t-1}, S_t} \sum_{S_{t-1} \in \mathbf{S}} \mathcal{P}_{S_{t-1}, S_t}^{D_{t-1}} b_{t-1}(S_{t-1}) \quad (2)$$

where $\eta = \sum_{S_t \in \mathbf{S}} Z_{O_t}^{D_{t-1}, S_t} \sum_{S_{t-1} \in \mathbf{S}} \mathcal{P}_{S_{t-1}, S_t}^{D_{t-1}} b_{t-1}(S_{t-1})$ is the normalisation factor. The initial belief state is $b_1(S_1) = 1$ (the initial state is observable).

B. Formalising CAGE-2 using POMDP

We formulate an attack in the CAGE-2 scenario as a POMDP episode. The POMDP explicitly models the state of the infrastructure, the defender action and the observation produced by the IDS. At time t , both the defender and the attacker each perform an action.

1) *Infrastructure model*: Let $\mathbf{H}$ be the set of n hosts, $\mathbf{Z}$ be the set of subnets, and $\mathbf{E}$ be the set of m services. Each host $h \in \mathbf{H}$ belongs to a subnet defined by $z(h)$. $\mathbf{E}_h \subset \mathbf{E}$ denotes the set of services provided by host h . Exploiting a service $e \in \mathbf{E}_h$ grants the attacker one of the following accesses to the host: N (No access), U (User access), and S (Superuser access). The access is determined by function $t(h, e)$. In the CAGE-2 scenario, a host h_1 where the attacker has root access provides the attacker with the knowledge of a different host h_2 . We model this fact with a function g_M that maps h_1 to h_2 . For details of the infrastructure model, see Appendix.

2) *State space $\mathbf{S}$* : The system state S_t represents the collective states of all hosts at time t . Formally, $S_t = (S_{1,t}, \dots, S_{n,t})$, where $S_{h,t}$ is the local state of host h . $S_{h,t}$ has three components: the attacker access state $I_{h,t}$, the running services $E_{h,t}$, and the scanned services $F_{h,t}$.

$I_{h,t}$ represents the attacker access to host h and takes one of the following vales: H if the host is unknown to the attacker; K if the host is known to the attacker; S if the host has been scanned by the attacker; U or R if the attacker has performed a successful exploit on the host; P if the attacker has root access on the host; and I if the services on the host are interrupted by the attacker.

$E_{h,t}$ represents the set of services deployed on host h and $F_{h,t}$ represents the knowledge of the attacker about the services running on host h . $E_{h,t}$ and $F_{h,t}$ are subsets of $\mathbf{E}$. The state space $\mathbf{S}$ can be written as $\{\{H, K, S, U, R, P, I\} \times 2^{\mathbf{E}} \times 2^{\mathbf{E}}\}^n$.

An episode starts with $S_{h,1} = (I_{h,1} = H, E_{h,1} = \mathbf{E}_h, F_{h,1} = \emptyset)$ for all hosts $h \in \mathbf{H}$.

3) *Action space $\mathbf{D}$* : At time t , the defender takes the action $D_t = (\Delta_t, T_t)$, whereby Δ_t is the action type, namely, *Analyse* a host (A); *Deploy* a decoy service $e \in \mathbf{E}$ (D_e); *Neutralise* and remove malware on a host (N); and *Restore* the host to a secure state and restart the services (R). T_t is the target host. In addition, the defender has the option to perform no action at time t , in which case we write $D_t = I$. Thus, the action space is $\mathbf{D} = I \cup \{A, D_1, \dots, D_m, N, R\} \times \mathbf{H}$, where $D_1, \dots, D_m$ are the decoy actions for each of the m services in $\mathbf{E}$.

4) *State transition*: The attacker in CAGE-2 follows a fixed strategy π_A that maps the state S_t to an action A_t at time t . We model $A_t = (\Lambda_t, T_t)$, where Λ_t is an action type, namely, *Discover* (D); *Scan* (S); *Exploit* service $e \in \mathbf{E}$ (E_e); *Privileged escalate* (P); and *Interrupt* (I) (see §III). T_t is the target of Λ_t , which is either a subnet $z \in \mathbf{Z}$ (for action D) or a host $h \in \mathbf{H}$ (for other actions).

In CAGE-2, we can describe the state transition $S_t \rightarrow S_{t+1}$ through the transition of the host states $S_{h,t} \rightarrow S_{h,t+1}$, $h \in \mathbf{H}$. Fig. 2 shows the state transition diagram of a host.

At time $t = 1, \dots, T$, the defender performs an action D_t , followed by the attacker performing an action A_t . The sequencing may look strange, but this is the way CAGE-2 is designed. We can therefore decompose the transition $S_{h,t} \rightarrow S_{h,t+1}$ into two consecutive steps.

a) $S_{h,t} \rightarrow S'_{h,t}$ (*defender action*):

$$I_{h,t} = U \rightarrow I'_{h,t} = S \text{ if } D_t = (N, h) \quad (3a)$$

$$I_{h,t} \in \{U, R, P, I\} \rightarrow I'_{h,t} = S \text{ if } D_t = (R, h) \quad (3b)$$

$$I_{h,t} \rightarrow I'_{h,t} = I_{h,t} \text{ otherwise} \quad (3c)$$

(3a - 3c) describe the transition of the attacker access state $I_{h,t} \rightarrow I'_{h,t}$. (3a) captures the effect of the *Neutralise* action, which removes the attacker from the host if $I_{h,t} = U$. (3b) states that the *Restore* action sets the host in the secure state S (see Fig. 2).

$$E_{h,t} \rightarrow E'_{h,t} = E_{h,t} \cup \{e\} \text{ if } D_t = (E_e, h), e \notin E_{h,t} \quad (4a)$$

$$E_{h,t} \rightarrow E'_{h,t} = \mathbf{E}_h \text{ if } D_t = (R, h) \quad (4b)$$

$$E_{h,t} \rightarrow E'_{h,t} = E_{h,t} \text{ otherwise} \quad (4c)$$

(4a-4c) describe the transition of the running services $E_{h,t} \rightarrow E'_{h,t}$. (4a) presents the installation of a new decoy service on host h . (4b) presents the removal of all decoy services as an effect of the *Restore* action.

Lastly, the scanned services component is not affected by the defender action, i.e., $F'_{h,t} = F_{h,t} \forall D_t \in \mathbf{D}$.

b) $S'_{h,t} \rightarrow S_{h,t+1}$ (*attacker action*):

$$I'_{h,t} = H \rightarrow I_{h,t+1} = K \text{ if}$$

$$\begin{cases} A_t = (D, z(h) = 1) \\ A_t = (D, z(h)) \exists h' : z(h') = z(h), I'_{h',t} \in \{P, I\} \\ \exists h' : I'_{h',t} = P, g_M(h') = h \end{cases} \quad (5a)$$

$$I'_{h,t} = K \rightarrow I_{h,t+1} = S \text{ if } A_t = (S, h) \quad (5b)$$

$$I'_{h,t} = S \rightarrow I_{h,t+1} =$$

$$\begin{cases} S \text{ if } A_t = (E_e, h), e \notin \mathbf{E}_h \text{ or } t(h, e) = N \\ U \text{ if } A_t = (E_e, h), e \in \mathbf{E}_h, t(h, e) = U \\ R \text{ if } A_t = (E_e, h), e \in \mathbf{E}_h, t(h, e) = S \end{cases} \quad (5c)$$

$$I'_{h,t} = U \rightarrow I_{h,t+1} = P \text{ if } A_t = (P, h) \quad (5d)$$

$$I'_{h,t} = R \rightarrow I_{h,t+1} = P \text{ if } A_t = (P, h) \quad (5e)$$

$$I'_{h,t} = P \rightarrow I_{h,t+1} = I \text{ if } A_t = (I, h) \quad (5f)$$

$$I'_{h,t} \rightarrow I_{h,t+1} = I'_{h,t} \text{ otherwise} \quad (5g)$$

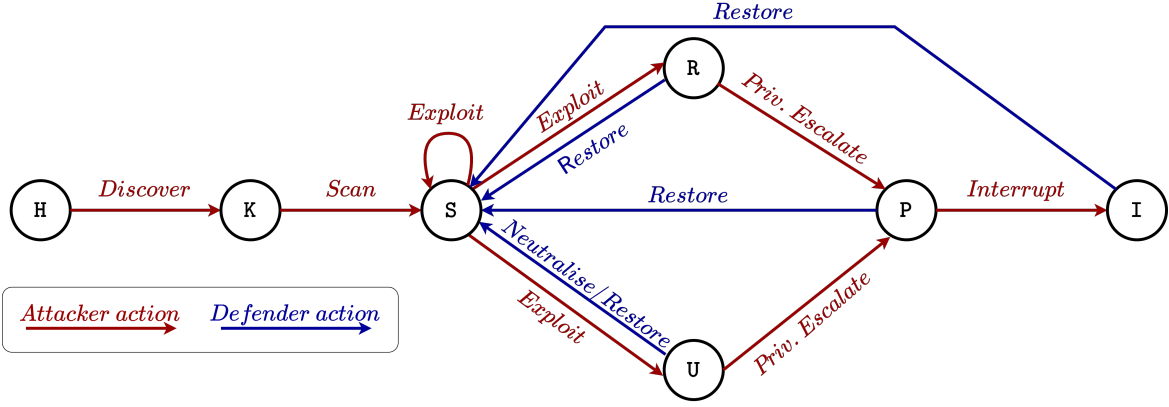


Fig. 2: The transition of the attacker access state $I_{h,t}$ caused by an action from the attacker A_t or the defender D_t ; nodes present the access states; arrows present the actions that cause state transitions. The defender actions *Analyse* and *Decoy* do not change the state $I_{h,t}$ and they are therefore not included.

(5a-5g) describe the transition of the attacker access state $I'_{h,t} \rightarrow I_{h,t+1}$. (5a) describes the three cases causing the transition $H \rightarrow K$, where host h becomes exposed to the attacker. The first case presents the initial phase of an attack with the *Discover* action on the clients on subnet $z = 1$. The second case represents the *Discover* action other subnets $z = 2, 3$, the attacker must have root access to another host on that subnet. The attacker can also discover host h by having root access to host h' that is connected to h , i.e., $g_M(h') = h$ (the last case). (5b) describes the transition $K \rightarrow S$, where the attacker scans a known host h for vulnerabilities.

After scanning host h ($I'_{h,t} = S$), the attacker attempts to gain access by exploiting one of the scanned services $e \in F'_{h,t}$. (5c) describes the three possible outcomes of this action, which are illustrated in Fig. 2. The exploit fails if the target service is a decoy ($e \notin \mathbf{E}_h$) or not exploitable ($t(h, e) = N$) (first case of (5c)). Otherwise, a successful exploit, presented by the last two cases of (5c), gives the attacker access to host h depending on the access right of the target service, either a regular user ($t(h, e) = U$) or a SuperUser ($t(h, e) = S$). Finally, (5d) and (5e) describe the attacker gaining root access, and (5f) describes the attacker interrupting the services running on host h .

$$F'_{h,t} \rightarrow F_{h,t+1} = E'_{h,t} \text{ if } A_t = (S, h) \quad (6a)$$

$$F'_{h,t} \rightarrow F_{h,t+1} = F'_{h,t} \text{ otherwise} \quad (6b)$$

(6a) and (6b) describe the transition of the scanned services $F'_{h,t} \rightarrow F_{h,t+1}$, which is only triggered when the attacker perform the *Scan* action to explore the services running on host h (6a).

Lastly, the running services component is not affected by the attacker action, i.e., $E_{h,t+1} = E'_{h,t} \forall A_t$.

5) *Time horizon T*: An episode in CAGE-2 has a finite time horizon T .

6) *Observation space O*: At time t , the defender observes $O_t = (O_{1,t}, \dots, O_{n,t})$. The host observation $O_{h,t}$ takes one

of the following values: H if the host has not been scanned by the attacker; S if the host has been scanned at time $t - 1$ by the attacker; C if the IDS has issued an alarm for the host; P if the attacker has root access to the host; U if the *Neutralise* action has been performed by the defender; and N if there is no detected attacker activity on a scanned host. Thus, the observation space is $\mathbf{O} = \{H, S, C, P, U, N\}^n$. The initial observation $O_{h,1} = H, \forall h \in \mathbf{H}$.

7) *Observation function*: We describe the observation function for O_t at time $t = 2, \dots, T$ using the host observations $O_{h,t}$, which depend on the host state $S_{h,t}$, the attacker action A_{t-1} and the defender action D_{t-1} :

$$O_{h,t} = \begin{cases} H & \text{if } I_{h,t} \in \{H, K\} & (7a) \\ S & \text{if } A_{t-1} = (S, h) & (7b) \\ S & \text{if } A_{t-1} = (E_e, h), N_d = 0 \forall e \in \mathbf{E} & (7c) \\ C & \text{if } A_{t-1} = (E_e, h), N_d = 1 \forall e \in \mathbf{E} & (7d) \\ U & \text{if } D_{t-1} = (N, h) & (7e) \\ N & \text{if } D_{t-1} = (R, h) & (7f) \\ C & \text{if } I_{h,t} \in \{U, R\}, D_{t-1} = (A, h) & (7g) \\ P & \text{if } I_{h,t} \in \{P, I\}, D_{t-1} = (A, h) & (7h) \\ N & \text{otherwise} & (7i) \end{cases}$$

where N_d is a binary random variable that determines the observation $O_{h,t}$ when the attacker performs the *Exploit* action on host h .

In (7a) and (7b), $O_{h,t}$ refers to the observations before and after the attacker scans host h , respectively. After the attacker performs the *Exploit* action at time $t-1$, $O_{h,t}$ has value S if the exploitation is not detected by the IDS (7c), otherwise, it has value C (7d). (7e) and (7f) describe the observations produced by the defender actions *Neutralise* and *Restore*, respectively. The defender can perform the *Analyse* action to learn about the compromise state of host h (7g-7h). Otherwise, the observation $O_{h,t} = N$ (7i).

8) *Reward function* $\mathcal{R}_{S_t}^{A_t}$: In CAGE-2, upon performing an action at time t , the defender receives a reward R_t :

$$R_t = \sigma_{D_t} + \sum_{h \in \mathbf{H}} (\sigma_{z(h)} \mathbb{1}_{I_{h,t} \in \{\mathbf{U}, \mathbf{R}, \mathbf{P}, \mathbf{I}\}} + \sigma_h \mathbb{1}_{I_{h,t} = \mathbf{I}}) \quad (8)$$

where $\sigma_{D_t} < 0$ defines the reward (actually, the cost) for performing the action D_t ; $\sigma_{z(h)} < 0$ is the reward for each compromised host in subnet $z(h)$; and $\sigma_h < 0$ is the reward for service interruption on host h . The values of these parameters in CAGE-2 are presented in the Appendix.

C. Defender Problem

The objective of the defender in CAGE-2 is to maximise the expected cumulative reward J with the discount factor $\gamma = 1$:

$$J(\pi_D) = \sum_{t=1}^T \mathbb{E}_{\pi_D} [R_t] \quad (9)$$

whereby π_D is the defender strategy, which defines a mapping from the belief space to the action space.

Therefore, the defender problem is to find the optimal strategy π_D^* that maximises the expected cumulative reward over the time horizon T .

Problem 1. Find the optimal defender strategy under the POMDP model of CAGE-2:

$$\pi_D^* = \arg \max_{\pi_D} \mathbb{E}_{\pi_D} [J] \quad (10a)$$

$$\text{subject to } D_t = \pi_D(b_t) \forall t \quad (10b)$$

$$\pi_A \sim P(\Pi_A) \quad (10c)$$

As the POMDP is stationary with a finite time horizon, we know that an optimal strategy π_D^* exists [17, Thm. 7.4.1].

D. Computing the optimal defender strategy: BF-PPO

The optimal strategy π_D^* can be computed by *dynamic programming* methods such as value iteration [29]. However, the large size of the state space (in the order of 10^{39}) leads to a high-dimensional belief. As a result, exact computation or estimation of π_D^* with the mentioned methods are computationally intractable [30], [31].

We therefore apply an iterative approximation strategy using Reinforcement Learning in form of Proximal Policy Optimization (PPO) [18]. It uses a neural network to represent the policy and performs policy optimisation using gradient ascent with a clipped objective function to prevent large policy updates.

In this work, we make two additions to the traditional algorithm regarding the estimation of the belief state and its representation. First, the evolution of a POMDP requires a belief update every time t , which is calculated with the Bayes Filter (2). The update has a quadratic time complexity with respect to the size of the state space $\mathbf{S}$. For that reason, the Bayes Filter is computationally impractical in our case. We address the issue by updating the belief state using particle filter [19], which is an approximate, non-parametric implementation of the Bayes Filter. The method approximates

b_t with a set of M random state samples (or *particles*), denoted as $\mathcal{P}_t = \{s_t^{(1)}, \dots, s_t^{(M)}\}$ with $s_t^{(i)} \in \mathbf{S}$. The belief state (2) is estimated using the frequency of the particle states in $\mathcal{P}_t$, i.e., $\hat{b}_t(s_t) = \frac{1}{M} \sum_i \mathbb{1}_{s_t^{(i)} = s_t}$. The particles at time t are sampled using rejection sampling, as presented in Alg. 1.

Algorithm 1 Particle filter in CAGE-2

```

1: Input: Input particle set  $\mathcal{P}_{t-1} = \{s_{t-1}^{(1)}, \dots, s_{t-1}^{(M)}\}$ ; Defender
   action  $D_{t-1}$ ; Observation  $O_t$ ; Simulator  $\mathcal{S}$ ;  $\mathcal{P}_t = \emptyset$ 
2: if  $t = 1$  then
3:    $\mathcal{P}_t \leftarrow \{S_1\}$ 
4: else
5:   while  $|\mathcal{P}_t| < M$  do
6:      $\bar{s} \sim \text{Uniform}(\mathcal{P}_{t-1})$ 
7:     Set  $\mathcal{S}$  to state  $\bar{s}$ 
8:     State  $\bar{S}_t$ , Observation  $\bar{O}_t \leftarrow \text{execute } D_{t-1} \text{ on } \mathcal{S}$ 
9:     if  $\bar{O}_t = O_t$  then
10:       $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \{\bar{S}_t\}$ 
11:   end if
12: end while
13: end if
14: Output: Set of particles  $\mathcal{P}_t$ 

```

Line 3 of Alg. 1 presents the initial particle set at the beginning of an episode, where the defender has full knowledge of the system state S_1 . Otherwise, the sampling routine consists of two parts. The first part (lines 6-8) samples the candidate particle states of $\mathcal{P}_t$ by executing action D_{t-1} on each particle state $s_{t-1}^{(i)} \in \mathcal{P}_{t-1}$. The second part (lines 9-11) concentrates the particles in states that are most likely to generate the observation O_t .

Second, based on the output of particle filter, we find a representation of the belief state to be used as input to the neural network encoded strategy. We cannot use $\{b_t(S_t)\}_{S_t \in \mathbf{S}}$ as the representation since the input layer of the neural network would be very large ($\sim 10^{39}$ in our case). Instead, we decide to represent the belief state through a representative sample state, which is sampled from $\mathcal{P}_t$ (An alternative would have been taking the most likely particle state from $\mathcal{P}_t$):

$$\hat{S}_t \sim \text{Uniform}(\mathcal{P}_t) \quad (11)$$

PPO, particle filter and particle sampling are the key elements of our solution method, which we call Belief Filter Policy Proximal Optimisation (BF-PPO). At time t , the method selects action D_t in three steps, which is illustrated in Fig. 3 and detailed in Alg. 2. First, it approximates the belief state b_t with particle filter (Alg. 2, Line 2). From the output of the particle filter $\mathcal{P}_t$, it samples a representative particle state $\hat{S}_t$ (Alg. 2, Line 3). Lastly, $\hat{S}_t$ is input to a neural network that is trained with PPO to generate action D_t (Alg. 2, Line 4). The policy π_D^θ used in the action selection step is trained through PPO, presented in Alg. 3.

V. EVALUATING BF-PPO IN CAGE-2

We evaluate our solution method BF-PPO for the CAGE-2 scenario, and compare its performance with that of a state-of-the-art solution. We implement BF-PPO in Python. The hyperparameters are listed in Appendix.

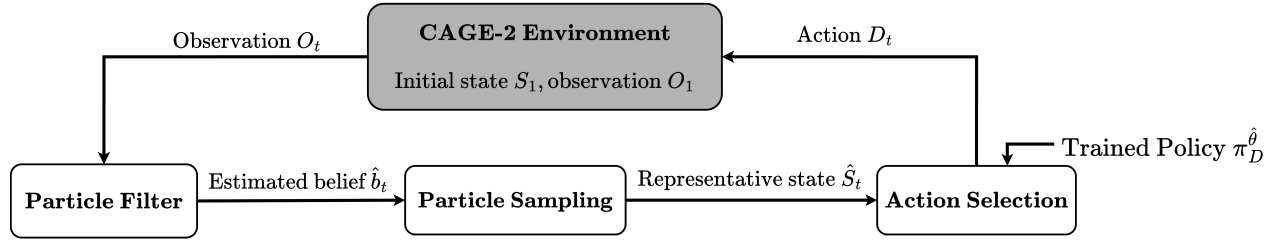


Fig. 3: Belief Filter Proximal Policy Optimisation (BF-PPO)

Algorithm 2 Action Selection with BF-PPO

```

1: Input: CAGE-2 simulator  $\mathcal{S}$ ; Particle set  $\mathcal{P}_{t-1} = \{s_{t-1}^{(1)}, \dots, s_{t-1}^{(M)}\}$ ; Observation  $O_t$ ; Action  $D_{t-1}$ ; defender strategy  $\pi_D^\theta$  (trained by Alg. 3)
2:  $\mathcal{P}_t \leftarrow \text{ParticleFilter}(\mathcal{P}_{t-1}, O_t, D_{t-1}, \mathcal{S})$   $\triangleright$  Alg. 1
3:  $\hat{S}_t \sim \text{Uniform}(\mathcal{P}_t)$   $\triangleright$  Eq. (11)
4:  $D_t \leftarrow \pi_D^\theta$ 
5: Output: Action  $D_t$ 

```

Algorithm 3 Defender strategy training with BF-PPO

```

1: Input: CAGE-2 simulator  $\mathcal{S}$ , Time horizon  $T$ , # iterations  $n_I$ , # training episodes  $n_E$ , Initial strategy  $\pi_D^\theta$ 
2: for iteration  $i \leftarrow 1, \dots, n_I$  do
3:    $\triangleright$  Collect trajectories
4:   Initialise trace buffer  $\mathcal{B} \leftarrow \emptyset$ 
5:   for episode  $e \leftarrow 1, \dots, n_E$  do
6:     for  $t \leftarrow 1, \dots, T$  do
7:        $\mathcal{P}_t \leftarrow \text{ParticleFilter}(\mathcal{P}_{t-1}, O_t, \mathcal{S})$   $\triangleright$  Alg. 1
8:        $\hat{S}_t \sim \text{Uniform}(\mathcal{P}_t)$ 
9:        $p_{D_t}, D_t \sim \pi_\theta(\cdot | \hat{S}_t)$ 
10:       $O_{t+1}, R_t \leftarrow \text{Execute } D_t \text{ in } \mathcal{S}$ 
11:      Store  $(\hat{S}_t, D_t, p_{D_t}, R_t, \hat{S}_{t+1})$  in  $\mathcal{B}$ 
12:    end for
13:  end for
14:   $\triangleright$  Update  $\theta$  with PPO [18, Alg. 1]
15:   $\hat{A} \leftarrow \text{Monte Carlo advantage}$ 
16:  Update  $\theta$  using clipped surrogate objective:

```

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_t \left[\min \left(\rho_t \hat{A}, \text{clip}(\rho_t, 1 - \epsilon, 1 + \epsilon) \hat{A} \right) \right]$$

```

17: end for
18: Output: Learned defender strategy  $\pi_D^{\hat{\theta}} = \pi_D^\theta$ 

```

A. Evaluation setup

1) *Baseline:* We compare the performance of BF-PPO with CARDIFF [20], a state-of-the-art method for CAGE-2, which achieves the highest performance among the methods on the CAGE-2 leaderboard [32]. CARDIFF is not based on a formal model. It combines PPO with hierarchical reinforcement learning. CARDIFF is open source at [20].

2) *Evaluation metrics:* We use the average cumulative reward $\hat{J}$, which is the average of cumulative rewards across multiple episodes, as the main evaluation metric.

3) *Attacker scenarios:* The CAGE-2 challenge includes two main attacker scenarios.

B-LINE. The defender defends the system against the B-LINE attacker [15]. The attacker has prior knowledge of

the network topology and the intrusion proceeds directly to the target server (see Fig. 1).

MEANDER. The defender defends the system against the Meander attacker [15]. The attacker explores the network topology and attempts to gain privilege access to each host it encounters, until it reaches the target server (see Fig. 1).

4) *Evaluation Process:* We train BF-PPO and CARDIFF to estimate an optimal defender strategy against each attacker scenario. A training run consists of 400 iterations. Each iteration corresponds to an update to the strategy parameters and consists of 100 episodes with time horizon $T = 100$. The defender strategies are evaluated at every iteration. We perform four training runs with different random seeds (listed in Appendix).

Second, we evaluate the defender strategy learned through BF-PPO against each attacker scenario with different time horizons, $T = 30, 50, 100$. Each combination of attacker and time horizon is run for 100 episodes. We compare the performance of BF-PPO with the published performance of CARDIFF on the CAGE-2 challenge leaderboard [32].

B. Evaluation results

Fig. 4 shows the training performance of the study methods against the two attacker scenarios. The blue curves represent the performance of our solution method BF-PPO. The green curves represent the performance of the baseline CARDIFF. Each row corresponds to the training results against an attacker scenario. The left column shows the cumulative reward of the defender strategies during training run with 400 iterations. The right column enlarges the second half of the training.

Fig. 4 shows that the learning curves of BF-PPO quickly converge to a stable mean value for both attackers, indicating its learned strategies have converged. On the other hand, we observe an increasing trend for the learning curves of CARDIFF throughout the training period, indicating its strategies have not yet converged. We conclude that our method BF-PPO exhibits significantly faster convergence than CARDIFF.

We also observe that the learning curves of BF-PPO remain above those of CARDIFF for the entire training period. This shows that our method produces more effective defender strategies at every point in the training period, especially for the MEANDER attacker scenario.

We compare the converged strategies of BF-PPO with the published score of CARDIFF. The result is shown in Tab. I. BF-PPO achieves higher cumulative rewards for both attacker

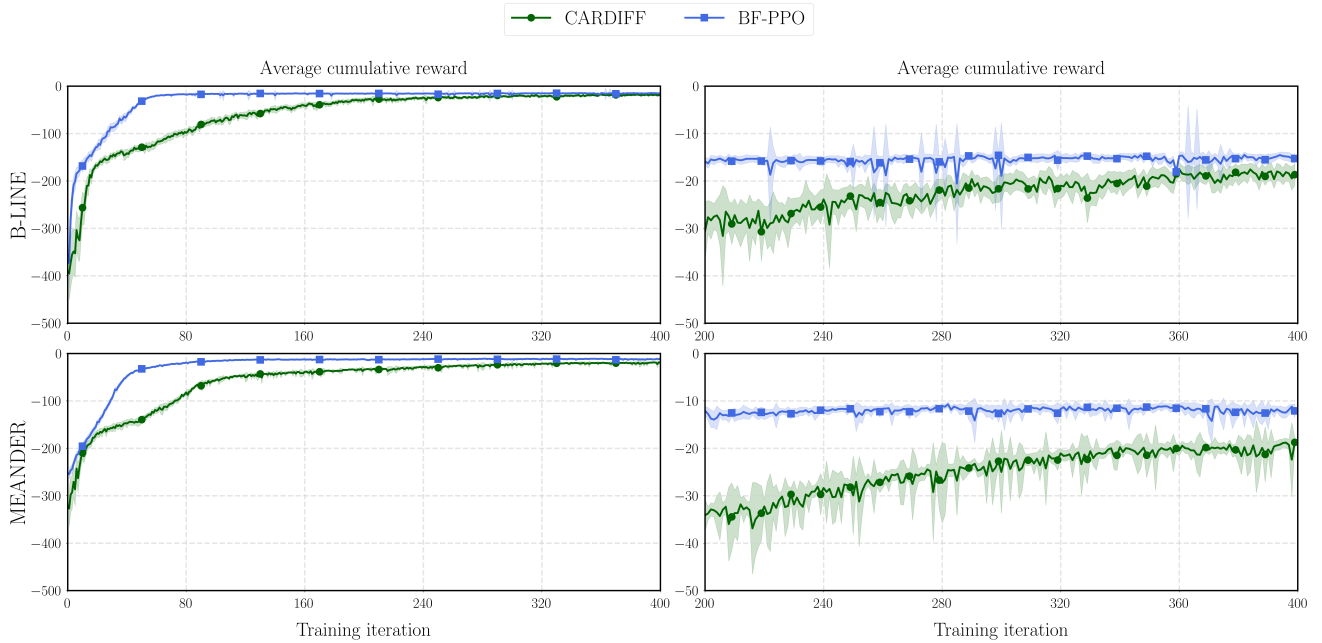


Fig. 4: The learning curves for our solution method, BF-PPO (blue curves) and the baseline, CARDIFF (green curves). Each row indicates a CAGE-2 attacker scenario, B-LINE and MEANDER. The left column shows the average cumulative rewards over the training period. The right column enlarges the right half of the graph on the left column. The curves show the mean and the 95% confidence interval for four training runs with different random seeds.

TABLE I: The evaluation results of our solution method BF-PPO and the baseline CARDIFF for the CAGE-2 scenario. Each subcolumn corresponds to a different time horizon T for an episode. The table cells show the mean and standard deviation of the cumulative rewards, recorded for 100 episodes.

Model	BLINE			MEANDER		
	$T = 30$	$T = 50$	$T = 100$	$T = 30$	$T = 50$	$T = 100$
CARDIFF	-3.41 ± 1.77	-6.41 ± 2.41	-13.76 ± 4.25	-5.64 ± 1.29	-8.69 ± 2.20	-16.6 ± 3.83
BF-PPO	-3.42 ± 1.35	-6.24 ± 2.09	-12.82 ± 3.07	-4.22 ± 1.96	-6.92 ± 2.84	-11.56 ± 4.22

scenarios and for most time horizon, with significantly better performance against MEANDER. This observation is consistent with the performance gaps between the methods in Fig. 4.

The analysis of Fig. 4 and Tab. I leads us to the conclusion that BF-PPO provides an offline strategy learning method that performs better than CARDIFF, in both terms of convergence rate and convergence value.

VI. CONCLUSION AND FUTURE WORK

This paper formalises the CAGE-2 scenario using the POMDP framework. For this formal model, we define an optimal defender strategy and propose an iterative method to learn it. We call this method BF-PPO (Alg. 2 and Alg. 3). It is based on the state-of-the-art reinforcement learning algorithm PPO. We use particle filter (Alg. 1) to address the challenges of computational complexity due to the large state space of the CAGE-2 model. We evaluate BF-PPO for the CAGE-2 scenario and we find that our method outperforms CARDIFF, the highest ranked method on the CAGE-2 leaderboard. Our method produces a higher reward (Tab. I) and requires fewer training episodes than CARDIFF (Fig. 4).

The formal model developed in this paper allows for further investigation beyond computing approximate optimal strategies. For instance, an analysis of the state transitions of the model may allow us to find system configurations where an attacker cannot reach a certain target if the defender performs the correct actions. Second, given a fixed defender strategy, we can formulate an optimal attacker strategy and use our method BF-PPO to compute it. Also, we can formulate the interaction between attacker and defender as a game where both players follow dynamic strategies, and we can analyse and hopefully solve the game.

We are currently developing a different formalisation of CAGE-2, which uses causal modelling. The model captures the causal relations between key variables describing the network infrastructure as well as the attacker and the defender. It allows us to significantly reduce the policy search space of the corresponding solution method.

VII. ACKNOWLEDGEMENT

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

TABLE II: Configuration of the network infrastructure in CAGE-2.

Host h	Subnet $z(h)$	Services E_h	Access $t(h, e)$	Connectivity $g_M(h)$
CLIENT-1	1	SSH FTP	S U	ENT-1
CLIENT-2	1	SMB RDS	N S	ENT-1
CLIENT-3	1	MYSQL APACHE2 SMTP	S U S	ENT-0
CLIENT-4	1	SSHD MYSQL APACHE2 SMTP	S S U S	ENT-0
ENT-0	2	SSHD	S	-
ENT-1	2	SSHD RDS SMB TOMCAT8	S N S U	-
ENT-2	2	SSHD RDS SMB TOMCAT8	S N S U	OP-SERVER
OP-SERVER	3	SSHD	S	-
OP-HOST-0	3	SSHD	S	-
OP-HOST-1	3	SSHD	S	-
OP-HOST-2	3	SSHD	S	-

TABLE III: Reward parameters for the defender in CAGE-2 in equation (8)

Parameter	Value	Description
$\sigma_{D_t=R}$	-1	cost for performing action $D_t = R$
$\sigma_{D_t \neq R}$	0	cost for performing other actions
$\sigma_{z=1}$	-0.1	cost for each host in one of the states $\{U, R, P, I\}$ in the client subnet $z = 1$
$\sigma_{z=2}$	-1	cost for each host in one of the states $\{U, R, P, I\}$ in the ENT subnet $z = 2$
$\sigma_{z=3}$	-1	cost for each host in one of the states $\{U, R, P, I\}$ in operational subnet $z = 3$
$\sigma_{h=OP-SERVER}$	-10	cost for interrupted services on OP-SERVER
$\sigma_{h \neq OP-SERVER}$	0	cost for interrupted services on other host

TABLE IV: Hyperparameters for the training of defender strategies

Parameter	Value
Training configuration	
Random seeds	[0, 108, 153, 701]
Time horizon T	100
# iterations, # episodes/iteration	400, 100
Neural network policy and PPO parameters	
# hidden layer, # neurons/layer	2, 64
Learning rate	$5 \cdot 10^{-4}$
# epochs, discount factor	10, 0.99
Optimiser (parameters)	Adam ($\beta_1 = 0.9, \beta_2 = 0.99$)
Clipping parameter ϵ	0.2

REFERENCES

- [1] M. Roesch, “Snort - lightweight intrusion detection for networks,” in *Proceedings of the 13th USENIX Conference on System Administration (LISA)*, Seattle, WA, USA: USENIX Association, 1999, pp. 229–238.
- [2] V. Paxson, “Bro: A system for detecting network intruders in real-time,” *Computer Networks*, vol. 31, no. 23, pp. 2435–2463, 1999, ISSN: 1389-1286. DOI: [https://doi.org/10.1016/S1389-1286\(99\)00112-7](https://doi.org/10.1016/S1389-1286(99)00112-7).
- [3] M. Foley, M. Wang, Z. M. C. Hicks, and V. Mavroudis, “Inroads into autonomous network defence using explained reinforcement learning,” in *CAMLIS*, E. Raff, S. Samtani, and L. Deason, Eds., ser. CEUR Workshop Proceedings, vol. 3391, CEUR-WS.org, 2022, pp. 1–19.
- [4] K. Hammar and R. Stadler, “Learning intrusion prevention policies through optimal stopping,” in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 509–517. DOI: 10.23919/CNSM52442.2021.9615542.
- [5] Y. Tang, J. Sun, H. Wang, J. Deng, L. Tong, and W. Xu, “A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning,” *Comput. Secur.*, vol. 142, no. C, Jul. 2024, ISSN: 0167-4048. DOI: 10.1016/j.cose.2024.103871.
- [6] K. Hammar, T. Li, R. Stadler, and Q. Zhu, “Adaptive security response strategies through conjectural online learning,” *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4055–4070, 2025, ISSN: 1556-6021. DOI: 10.1109/tifs.2025.3558600.
- [7] 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. DOI: 10.1109/TIFS.2018.2819967.
- [8] A. Mcabee, M. Tummala, and J. Mceachen, “The use of partially observable markov decision processes to

- optimally implement moving target defense,” Jan. 2021. DOI: 10.24251/HICSS.2021.840.
- [9] P. Lau, W. Wei, L. Wang, Z. Liu, and C.-W. Ten, “A cybersecurity insurance model for power system reliability considering optimal defense resource allocation,” *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4403–4414, 2020. DOI: 10.1109/TSG.2020.2992782.
 - [10] K. Hammar and R. Stadler, “Learning near-optimal intrusion responses against dynamic attackers,” *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 1158–1177, 2024. DOI: 10.1109/TNSM.2023.3293413.
 - [11] 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. DOI: 10.1109/TIFS.2022.3229595.
 - [12] K. Hammar, N. Dhir, and R. Stadler, *Optimal defender strategies for cage-2 using causal modeling and tree search*, 2024. arXiv: 2407.11070 [cs.LG].
 - [13] A. Andrew, S. Spillard, J. Collyer, and N. Dhir, *Developing optimal causal cyber-defence agents via cyber security simulation*, 2022. arXiv: 2207.12355 [cs.CR].
 - [14] D. Shi, Z. Guo, K. H. Johansson, and L. Shi, “Causality countermeasures for anomaly detection in cyber-physical systems,” *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 386–401, 2018. DOI: 10.1109/TAC.2017.2714646.
 - [15] M. Kiely, D. Bowman, M. Standen, and C. Moir, “On autonomous agents in a cyber defence environment,” *arXiv preprint arXiv:2309.07388*, 2023.
 - [16] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, *Cyborg: A gym for the development of autonomous cyber agents*, 2021. arXiv: 2108.09118 [cs.CR].
 - [17] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.
 - [18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].
 - [19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005, ISBN: 9780262201629.
 - [20] S. Vyas, J. Hannay, A. Bolton, and P. P. Burnap, *Automated cyber defence: A review*, code <https://github.com/john-cardiff/cyborg-cage-2>, 2023.
 - [21] M. Wolk, A. Applebaum, C. Dennler, et al., *Beyond cage: Investigating generalization of learned autonomous network defense policies*, 2022. arXiv: 2211.15557 [cs.LG].
 - [22] M. Foley, C. Hicks, K. Highnam, and V. Mavroudis, “Autonomous network defence using reinforcement learning,” in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ACM, May 2022. DOI: 10.1145/3488932.3527286.
 - [23] K. Heckel, “Neuroevolution for autonomous cyber defense,” in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO ’23 Companion, Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 651–654, ISBN: 9798400701207. DOI: 10.1145/3583133.3590596.
 - [24] B. Prebot, Y. Du, X. Xi, and C. Gonzalez, “Cognitive models of dynamic decision in autonomous intelligent cyber defense,” in *International Conference on Autonomous Intelligent Cyber-defense Agents*, 2022.
 - [25] 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. DOI: 10.1109/NOMS54207.2022.9789835.
 - [26] 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. DOI: 10.1109/TSG.2018.2878570.
 - [27] Z. Hu, M. Zhu, and P. Liu, “Adaptive cyber defense against multi-stage attacks using learning-based pomdp,” vol. 24, no. 1, Nov. 2020, ISSN: 2471-2566. DOI: 10.1145/3418897.
 - [28] K. Pisal and S. Roychowdhury, “Cyber-defense mechanism considering incomplete information using pomdp,” in *Proceedings of International Conference on Network Security and Blockchain Technology*, D. Giri, J. K. Mandal, K. Sakurai, and D. De, Eds., Singapore: Springer Nature Singapore, 2022, pp. 3–17.
 - [29] 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, ISSN: 0030364X, 15265463.
 - [30] C. H. Papadimitriou and J. N. Tsitsiklis, “The complexity of markov decision processes,” *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987, ISSN: 0364765X, 15265471. (visited on 03/10/2025).
 - [31] 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, ISSN: 0304-3975. DOI: [https://doi.org/10.1016/0304-3975\(95\)00158-1](https://doi.org/10.1016/0304-3975(95)00158-1).
 - [32] CAGE, “Tcp cage challenge 2,” in *AAAI-22 Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2022. [Online]. Available: <https://github.com/cage-challenge/cage-challenge-2>.