

# ACDZero: MCTS Agent for Mastering Automated Cyber Defense

Yu Li<sup>†,1</sup>, Sizhe Tang<sup>†,1</sup>, Rongqian Chen<sup>1</sup>, Fei Xu Yu<sup>1</sup>, Guangyu Jiang<sup>1</sup>, Mahdi Imani<sup>2</sup>, Nathaniel D. Bastian<sup>3</sup>, Tian Lan<sup>1</sup>

<sup>1</sup>Dept of ECE, George Washington University, Washington, D.C., USA

<sup>2</sup>Dept of ECE, Northeastern University, Boston, MA, USA

<sup>3</sup>Dept of EECS, United States Military Academy, West Point, NY, USA

**Abstract**—Automated cyber defense (ACD) seeks to protect computer networks with minimal or no human intervention, reacting to intrusions by taking corrective actions such as isolating hosts, resetting services, deploying decoys, or updating access controls. However, existing approaches for ACD, such as deep reinforcement learning (RL), often face difficult exploration in complex networks with large decision/state spaces and thus require an expensive amount of samples. Inspired by the need to learn sample-efficient defense policies, we frame ACD in CAGE Challenge 4 (CAGE-4 / CC4) as a context-based partially observable Markov decision problem and propose a planning-centric defense policy based on Monte Carlo Tree Search (MCTS). It explicitly models the exploration-exploitation tradeoff in ACD and uses statistical sampling to guide exploration and decision making. We make novel use of graph neural networks (GNNs) to embed observations from the network as attributed graphs, to enable permutation-invariant reasoning over hosts and their relationships. To make our solution practical in complex search spaces, we guide MCTS with learned graph embeddings and priors over graph-edit actions, combining model-free generalization and policy distillation with look-ahead planning. We evaluate the resulting agent on CC4 scenarios involving diverse network structures and adversary behaviors, and show that our search-guided, graph-embedding-based planning improves defense reward and robustness relative to state-of-the-art RL baselines.

**Index Terms**—Automated cyber defense, Monte Carlo Tree Search, Reinforcement learning.

## I. INTRODUCTION

Automated cyber defense (ACD) systems are designed to monitor network environments and execute corrective actions—such as host isolation, service restoration, decoy deployment, and credential rotation—with minimal human input [1], [2]. While reinforcement learning (RL) and deep reinforcement learning (DRL) have demonstrated potential in training defense policies within simulated environments [3], [4], the approaches often face the challenge of balancing exploration and exploitation [5]–[7] in complex networks with large decision/state spaces, and thus require an expensive amount of samples to learn a reasonable defense policy. As modern cyber-adversaries become more sophisticated [8]–[10] and leverage multi-step strategies. Existing solutions relying

on RL and DRL struggle to capture multi-step look-ahead defense planning while also meeting the required sample efficiency for agile cyber defense.

To bridge this gap, we propose **ACDZERO**, a framework for learning automated cyber defense policies through Monte Carlo Tree Search (MCTS) and graph-based latent-space planning. More precisely, we leverage MCTS, which has demonstrated strong performance in multi-step reasoning and complex planning problems [11] such as mastering chess and board games [12]–[14]—to explicitly model the exploration-exploitation tradeoff in ACD and use statistical sampling to guide decision making. By building a dynamic model of the ACD problem (in a latent graph-embedding space as introduced later), each search consists of a series of simulated ACD games of defense action self-play to traverse a tree from root state to leaf state of the ACD game. Each simulation proceeds by selecting in each node/state a defense action with respect to a dynamically-updated upper confidence tree (UCT) bound to balance exploration-exploitation. The final game result of each rollout is then used to weight the nodes in the ACD game tree and to update the UCT bounds for learning a search policy with optimal rewards.

Unique challenges arise from applying MCTS to a complex ACD environment like the CAGE Challenge 4 (CC4) [15], involving multiple subnets, dynamic communications, random initialization, and a large set of servers/hosts and defense actions. We note that structural heterogeneity and unknown environment dynamics due to partial observability lead to significant challenges in representing the dynamic observations and network states to learn a dynamic model in MCTS. This dynamism and uncertainty render standard fixed-length vector representations brittle and prone to failure when deployed across varying network topologies [16]–[21]. To this end, ACDZERO makes novel use of a Graph Neural Network (GNN) as an invariant structural engine to encode network entities as typed nodes and edges [22], [23]. This representation serves as the foundation for a learned latent dynamics model, which enables MCTS to perform “virtual” look-ahead simulations without access to the ground-truth simulator [12].

Crucially, our architecture integrates this high-fidelity search into a decentralized Actor-Critic framework. During training,

<sup>†</sup>Equal contribution.

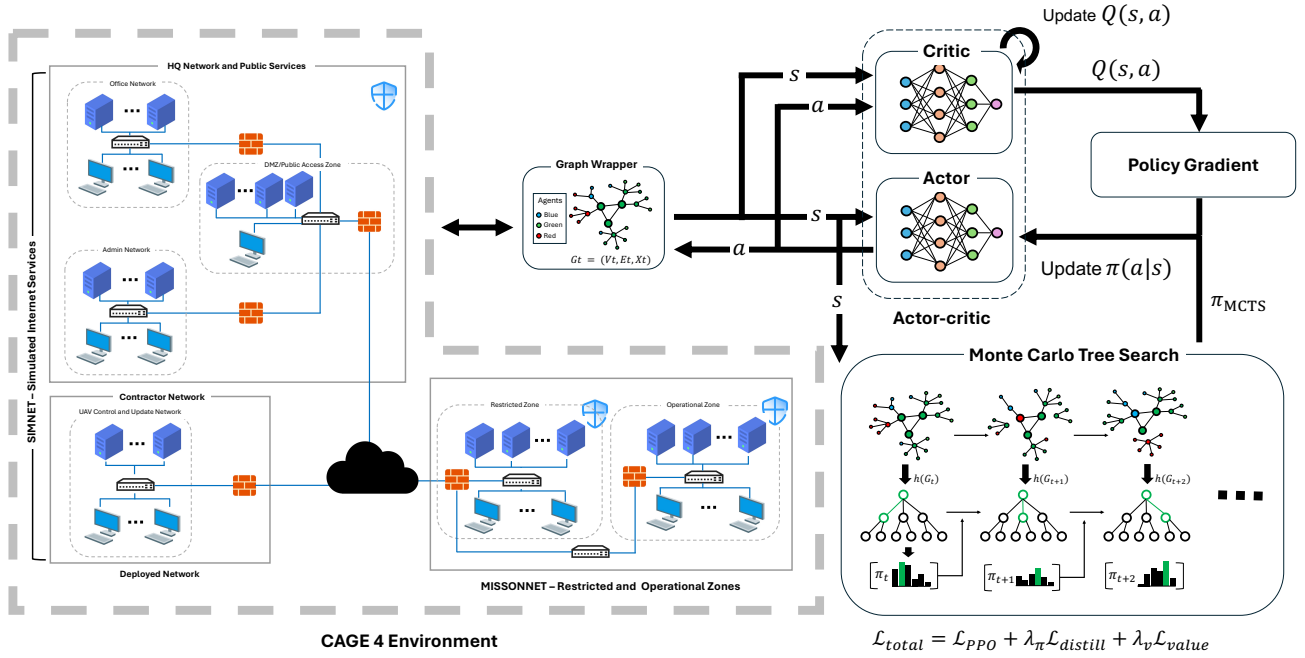


Fig. 1. Overview of the ACDZero framework applied to the CAGE Challenge 4 environment. CAGE-4 simulates a high-fidelity enterprise network where autonomous defenders must protect critical assets against adaptive adversaries in a partially observable, multi-agent setting. To address the brittleness of standard reactive policies that degrade under topological changes, our approach treats defense not only as policy learning but as online decision-time planning. The system first transforms local observations into attributed graphs ( $G_t$ ) to enable permutation-invariant reasoning over hosts and their relationships. Uniquely, ACDZero utilizes a Monte Carlo Tree Search (MCTS) module (bottom right) to perform look-ahead planning within a learned latent space, simulating alternative futures to discover optimal strategies. These high-quality search policies serve as distillation targets for the GNN-based Actor (top right), allowing the agent to internalize strategic foresight while maintaining the inference speed required for real-time autonomous cyber defense.

the MCTS serves as a powerful policy improvement operator, generating strategic targets that are distilled into a GNN-based actor via policy gradient updates. This distillation process allows the agent to internalize complex multi-step reasoning within its neural weights. At deployment, the distilled policy can be executed directly as a fast, reactive actor, retaining the strategic foresight of search-guided training without the computational overhead of real-time planning. ACDZERO is implemented in the CC4 environment and evaluated against state-of-the-art RL baselines.

Our contributions are summarized as follows:

- **ACDZero Framework:** We introduce a graph-embedding-based MCTS framework that unifies GNN-based state representation with a learned latent dynamics model and an Actor-Critic distillation pipeline for topology-robust planning.
- **Formalization of Graph-Based POMDP:** We formalize ACD in CC4 as a partially observable Markov decision process over dynamic graphs, necessitating representations that are strictly invariant to node permutation and network scale.
- **State-of-the-Art Performance:** We demonstrate that ACDZERO achieves a 29.2% improvement in defense success over the current state-of-the-art graph-based baselines, while exhibiting superior convergence stability across diverse and unseen network configurations.
- **Empirical Analysis of Search-Guided Training:**

Through extensive ablation studies, we quantify the contributions of latent-space planning and policy distillation, confirming that MCTS-guided supervision is the primary driver for achieving strategic robustness in complex cyber environments.

## II. BACKGROUNDS

**Automated Cyber Defense.** ACD has transitioned from traditional rule-based heuristics [1] toward RL frameworks capable of discovering optimal defensive policies through autonomous environment interaction [3], [24]–[26]. However, the efficacy of RL in network security is often constrained by state representation. Early approaches predominantly utilized fixed-vector encodings, where network features are mapped to static indices [27]–[30]. Such representations impose an implicit dependency on node ordering; because they lack permutation invariance, even a minor re-indexing of hosts results in disparate feature vectors for functionally identical network states, severely inhibiting generalization.

These architectural limitations are particularly pronounced in high-fidelity benchmarks such as the CC4 [15]. Unlike static environments, CC4 features a stochastically initialized topology where the number of hosts (5–15 per subnet) and active services (1–5 per host) vary across episodes. This structural fluidity prevents agents from memorizing specific configurations and demands strategies that are robust to topological shifts. To mitigate this, recent research has pivoted

toward graph-based representations [31]–[33]. By encoding the network as an attributed graph—where entities are nodes and relations are edges—defenders can leverage GNNs to achieve the permutation invariance necessary for reasoning about structural patterns across heterogeneous network configurations [22].

**Monte Carlo Tree Search.** MCTS is applied widely to solve planning problems through sequential decision-making [34], [35]. A typical MCTS involves four phases, i.e., *Selection* to choose actions from candidates via UCB-style strategies [36], *Expansion* to sample new candidate actions for existing nodes, *Simulation* to obtain the corresponding payoffs, and *Backup* to update the cumulated returns along the search path. We denote the state by  $s$  and action by  $a$ . For each node  $(s, a)$  in the tree, there are statistics including the estimated value  $\Phi(s, a)$ , visiting count  $N(s, a)$ .

MuZero [12] is a classic MCTS framework learning an internal dynamics model, allowing it to perform tree-based planning without access to the environment’s ground-truth rules or a simulator. This framework is composed of three learnable components parameterized by  $\theta$ : (i) a representation function  $s_0 = h_\theta(o_1, \dots, o_t)$  that transforms observations into a latent state, (ii) a dynamics function  $(s_k, r_k) = g_\theta(s_{k-1}, a_k)$  that predicts the next latent state and immediate reward, and (iii) a prediction function  $(\mathbf{p}_k, v_k) = f_\theta(s_k)$  which outputs the policy prior and state value.

During the MCTS process, MuZero navigates the search tree entirely within this learned latent space. Starting from the root node, it applies a variant of the predictor Upper Confidence Bound (pUCT) [12] to select actions:  $a = \arg \max_{a \in \mathcal{A}} Q(s, a) + P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{N(s, a) + 1} c_1$  where  $P(s, a)$  is the prior probability from the policy head and  $c_1$  is a constant. Statistics along the search path are updated during the *Backup* phase using a cumulative discounted payoff  $G_{t,k} = \sum_{\tau=0}^{l-1-k} \gamma^t \hat{r}_{k+1+\tau} + \gamma^{l-k} v^l$ . The mean action value  $Q(s, a)$  is then updated as the average of these bootstrapped returns:  $Q(s, a) = \frac{N(s, a)Q(s, a) + G_{t,k}}{N(s, a) + 1}$ .

### III. METHOD

### A. Problem Formulation

We formalize the automated cyber defense task within the CC4 networks environment as a Decentralized Partially Observable Markov Decision Process. Therefore the problem can be defined by the tuple  $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ . Here,  $\mathcal{N}$  denotes the set of defender agents,  $\mathcal{S}$  the global state space encompassing network topology and host compromise status,  $\mathcal{A}$  the joint action space, and  $\mathcal{O}$  the observation space. The transition function  $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$  and reward function  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  capture the dynamics arising from interactions among red (attacker), green (user), and blue (defender) agents.

Due to partial observability, each agent  $i \in \mathcal{N}$  receives only a local observation  $o_t^{(i)} \in \mathcal{O}$  corresponding to its assigned subnet. Therefore, a key challenge is that the network topology is stochastically initialized at each episode, causing the dimensionality of  $\mathcal{S}$  and  $\mathcal{O}$  to vary. To address this, we frame policy

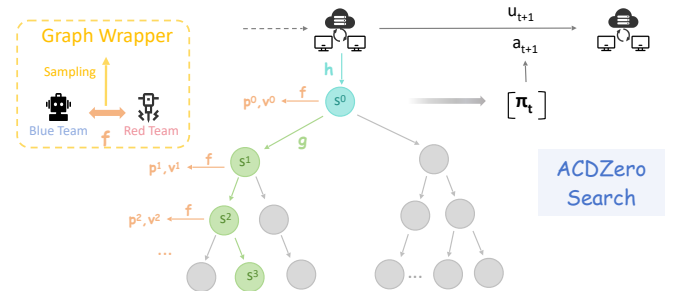


Fig. 2. ACDZERO framework. Graph observations are encoded into latent states. MCTS performs tree search with learned dynamics  $g$  to generate improved policy and value estimates  $(p, v)$ , which are distilled via prediction head  $f$  into the actor  $\pi_\theta$  for action selection.

learning over dynamic graphs rather than fixed-dimensional vectors, requiring representations invariant to input size and node permutation.

### B. State Representation and Environment Interface

To bridge CC4 simulation data and the graph-based policy architecture, we implement a specialized *Environment Interface* that (i) constructs semantically rich attributed graphs from local observations, and (ii) maps policy decisions to executable simulation commands.

At each timestep  $t$ , the interface transforms the agent’s local observation into an attributed graph  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t)$ , where  $\mathcal{V}_t$  denotes the node set,  $\mathcal{E}_t$  the edge set, and  $\mathbf{X}_t \in \mathbb{R}^{|\mathcal{V}_t| \times d}$  the  $d$ -dimensional node feature matrix. We represent environment entities as typed nodes: *Hosts* (servers and workstations), *Subnets* (network segments), *Ports* (services and connections), and *Files* (analyzed assets). Each node  $v \in \mathcal{V}_t$  encodes type-specific attributes—hosts include OS metadata (version, distribution, architecture) and role indicators; ports encode process information (port number, service type) and status flags (ephemeral, default, decoy); files capture analysis meta-data (density, signature status). Categorical attributes are one-hot encoded.

A critical component is inter-agent communication integration. CC4 restricts communication to 8-bit messages, which the interface parses and encodes as features of corresponding *Subnet* nodes, enabling implicit coordination without centralized training. Environment-wide variables are encoded into a global context vector  $\mathbf{g}_t \in \mathbb{R}^{d_g}$  for temporal and phase awareness. For action mapping, the interface abstracts simulator actions into graph operations: defensive operations (*Analyze*, *Restore*, *DeployDecoy*) target *Host* nodes, while network operations (*AllowTraffic*, *BlockTraffic*) modify edges between subnet nodes. This decoupling enables seamless adaptation to variable network topologies.

### C. ACDZERO

Our proposed algorithm, ACDZERO, adopts a MuZero-like framework [12] for decentralized cyber defense. The core objective is to employ MCTS as a policy improvement operator that enhances a GNN-based PPO [37] agent. MCTS

performs lookahead search in learned latent space, generating an improved policy  $\pi_{\text{mcts}}$  and value estimates that are distilled into decentralized actor-critic networks.

In our ACDZERO, network state evolution is modeled as transitions in a latent search tree. For each agent  $i$ , search initializes at a root node representing the latent belief  $s_{t,0} = h_\theta(o_{\leq t})$ , where  $h_\theta$  is a GNN-based representation network processing graph-structured observation history  $o_{\leq t}$ . To capture dependencies in stochastically initialized CC4 topologies,  $h_\theta$  employs two-stage hierarchical aggregation: *Intra-Entity Aggregation* pools port and file attributes into host embeddings  $\mathbf{h}_{\text{host}}$ , then *Inter-Subnet Aggregation* propagates these to subnet nodes, creating a representation invariant to node permutations and network size. Tree edges represent defensive actions  $a \in \mathcal{A}^{(i)}$ , and child nodes are latent states predicted by dynamics function  $s_{t,k+1} = g_\theta(s_{t,k}, a_{t+k})$ . To handle variable action dimensionality,  $g_\theta$  projects each action into fixed-dimensional embeddings processed by a GRU, capturing temporal dependencies across heterogeneous topologies. The algorithm integrates a reward function  $r_{t,k} = R_\theta(s_{t,k}, a_{t+k})$  and prediction head  $(\mathbf{p}_{t,k}, v_{t,k}) = f_\theta(s_{t,k})$  to forecast rewards, policy priors, and state values.

a) *MCTS Procedure*: To perform multi-step reasoning in CC4's latent space, each agent executes a fixed number of look-ahead simulations before taking real actions, mimicking rehearsal of attack-defense trajectories. The procedure follows three iterative phases:

- **Selection**: Starting from root node  $s_{t,0}$ , the agent traverses the tree by selecting actions balancing exploitation and exploration. We employ the pUCT rule [12] with Dirichlet noise. At each node  $s$ , the agent selects action  $a^*$  according to:

$$a^* = \arg \max_{a \in \mathcal{A}^{(i)}} \left[ Q(s, a) + P(s, a) \cdot \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)} c_1 \right] \quad (1)$$

where  $Q(s, a)$  tracks the action's historical performance, and  $P(s, a)$  from prediction head  $f_\theta$  represents the agent's prior intuition. Selection continues until reaching a leaf node.

- **Expansion and Evaluation**: At leaf node  $s_{t,l}$ , ACDZERO performs virtual expansion using learned dynamics. Unlike traditional MCTS requiring a simulator, ACDZERO uses dynamics function  $g_\theta$  to generate the next latent state  $s_{t,l+1} = g_\theta(s_{t,l}, a_{t+l})$  and predicts reward  $\hat{r}_{t,l}$ . Simultaneously, prediction head  $f_\theta$  evaluates the node to obtain value  $v_{t,l+1}$  and policy prior  $\mathbf{p}_{t,l+1}$ , enabling anticipation of defensive operations without environment interaction latency.
- **Backup**: Evaluation results propagate backwards to update ancestor node statistics. For each state-action pair  $(s, a)$ , we increment visit count  $N(s, a)$  and update mean value  $Q(s, a)$  using  $n$ -step bootstrapped return  $G_{t,k}$ :

$$G_{t,k} = \sum_{j=0}^{l-k-1} \gamma^j \hat{r}_{t,k+j} + \gamma^{l-k} v_{t,l}, \quad (2)$$

$$Q(s, a) \leftarrow \frac{N(s, a)Q(s, a) + G_{t,k}}{N(s, a) + 1} \quad (3)$$

where  $v_{t,l}$  is the terminal value and  $\hat{r}$  are rewards predicted by  $R_\theta$ . This recursive update ensures root statistics converge toward an optimal defensive strategy, providing robust targets for policy distillation.

b) *Optimization Objectives*: The training ACDZERO is formulated as multi-task learning, integrating MCTS's deliberate reasoning with PPO's reactive efficiency. Each agent minimizes a joint loss, ensuring stable updates, effective distillation, and accurate latent dynamics:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PPO}} + \lambda_\pi \mathcal{L}_{\text{distill}} + \lambda_v \mathcal{L}_{\text{value}} \quad (4)$$

where  $\lambda_\pi$  and  $\lambda_v$  are scaling coefficients that balance the contribution of each objective.

- **Decentralized Policy Optimization ( $\mathcal{L}_{\text{PPO}}$ )**: To maintain baseline stability in the non-stationary multi-agent environment, we employ the standard clipped surrogate objective:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}_t \left[ \min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right] \quad (5)$$

where  $r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$  is the probability ratio and  $\hat{A}_t$  is the advantage estimated by the GNN-critic. This loss ensures that the agent's policy does not deviate excessively from its previous iterations, facilitating safe exploration.

- **MCTS-Guided Policy Distillation ( $\mathcal{L}_{\text{distill}}$ )**: A pivotal feature of ACDZERO is the use of MCTS as a "policy improvement" operator. The search process yields a visit count distribution at the root node, which constitutes an improved search policy  $\pi_{\text{mcts}}(a|s) \propto N(s, a)^{1/\tau}$ . We use the Kullback-Leibler (KL) divergence to force the GNN actor  $\pi_\theta$  to internalize the multi-step look-ahead logic:

$$\mathcal{L}_{\text{distill}} = \mathbb{E}_t [D_{KL}(\pi_{\text{mcts}}(\cdot|s_t) || \pi_\theta(\cdot|o_t))] \quad (6)$$

By minimizing this loss, the lightweight actor learns to approximate the high-fidelity search policy, allowing it to exhibit "strategic foresight" even during real-time inference when the search tree is omitted.

- **Latent Dynamics and Value Prediction ( $\mathcal{L}_{\text{value}}$ )**: To ensure the latent space provides a reliable foundation for planning, the dynamics function  $g_\theta$ , the reward head  $\hat{r}$ , and the value head  $v$  are optimized to minimize the prediction error over an unrolled trajectory of length  $K$ . This process facilitates *indirect optimization*: the dynamics model is not supervised by raw state observations but is instead shaped by its utility in predicting rewards and long-term values. This ensures that the latent transitions capture the most semantically relevant features for cyber defense, such as host compromise status and subnet connectivity.

Through joint optimization, ACDZERO bridges slow deliberation and fast execution. During training, MCTS supervises

the GNN to learn complex defensive patterns; during deployment, the agent maintains neural network inference speed, achieving robustness against high-velocity cyber-attacks.

#### IV. RESULTS

We evaluate ACDZero on the CAGE Challenge 4 environment, comparing it against tabular RL baselines (DQN, PPO) and the graph-based GCN method. Our experiments demonstrate that combining MCTS-guided planning with graph neural networks yields substantial improvements in both final performance and sample efficiency.

##### A. Experimental Setting

All methods are evaluated on CAGE Challenge 4, a multi-agent cyber defense scenario where five blue agents defend against adaptive red adversaries across four network zones. The network topology is stochastically initialized at each episode, with 5-15 hosts per subnet and 1-5 services per host. Following the official protocol, we evaluate against FiniteStateRedAgent over 100 episodes of 500 timesteps each, reporting mean reward and standard deviation.

We compare against: (1) DQN and PPO using fixed-vector representations via the EnterpriseMAE wrapper, and (2) GCN, a graph-based method using graph convolutional networks, as they have announced, ranked 5th on the official CAGE-4 leaderboard.

ACDZero uses a GNN backbone with 256-dimensional hidden layers and 128-dimensional embeddings. MCTS performs 16 simulations per action with dynamic  $c_1$  scheduling ( $c_{\text{base}} = 19652$ ,  $c_{\text{init}} = 1.25$ ). During training, we apply Dirichlet noise ( $\alpha = 0.3$ ,  $\epsilon = 0.25$ ) and use temperature  $\tau = 1.0$ ; during evaluation,  $\tau = 0.1$ . The joint loss uses  $\lambda_\pi = 0.5$  and  $\lambda_v = 0.5$ . We train with 5 parallel workers using PPO clipping  $\epsilon = 0.2$  and discount factor  $\gamma = 0.99$ .

##### B. Main Result

Table I presents the final performance of all methods. ACDZero achieves a mean reward of  $-150.03 \pm 19.85$ , representing a 29.2% improvement over the GCN baseline ( $-193.68 \pm 21.07$ ). The improvement demonstrates the effectiveness of integrating MCTS-guided planning with graph-based policy learning, which stems from MCTS systematically exploring multi-step defensive strategies, policy distillation providing high-quality training targets, and learned dynamics enabling anticipation of attacker behavior. DQN and PPO obtain mean rewards of  $-606.20$  and  $-597.28$ , respectively, highlighting the fundamental limitation of fixed-vector representations: they cannot generalize across varying network topologies because they implicitly memorize specific node orderings.

Beyond final performance, ACDZero exhibits 5.8% lower variance ( $\pm 19.85$  vs  $\pm 21.07$ ) than GCN, indicating more consistent defense across diverse network configurations. The MCTS stable planning framework is adaptable to different topologies, and the learned strategies capture generalizable defense principles rather than topology-specific heuristics.

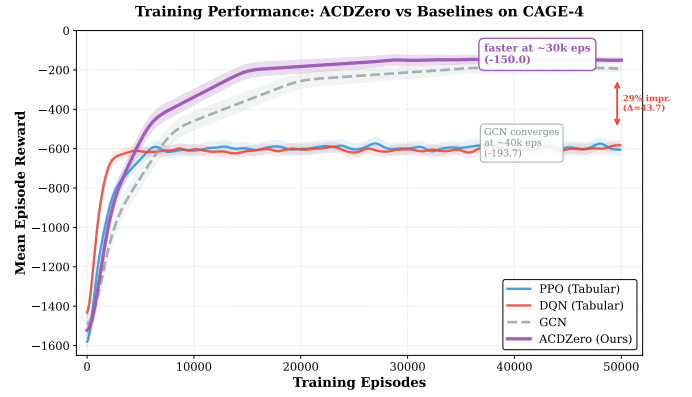


Fig. 3. Training performance on CAGE Challenge 4. ACDZero converges faster ( $\sim 30k$  episodes) and to better performance ( $-150$ ) than the GCN baseline ( $-193.68$  at  $\sim 40k$  episodes). Tabular methods plateau early at suboptimal performance. Shaded regions indicate standard deviation.

Figure 3 shows learning curves over 50,000 episodes. Tabular methods plateau within 10,000 episodes at  $-600$ , while GCN continues improving until 40,000 episodes ( $-193.68$ ), demonstrating that permutation-invariant processing extracts topology-independent strategies. ACDZero converges at 30,000 episodes (25% faster than GCN) with superior performance. This acceleration stems from MCTS providing multi-step search supervision rather than noisy single-step rewards. Training requires  $2.5\times$  more computation per step than GCN, but fewer episodes yield comparable total time. At inference, ACDZero uses only  $\pi_\theta$  without MCTS, achieving reactive-speed decision-making.

##### C. Ablation Study

To isolate the contributions of individual components, we systematically remove key elements of ACDZero. Table II summarizes the results.

Removing MCTS (using only the GNN-based PPO agent) reduces performance to  $-193.68$ , equivalent to the GCN baseline. This 29.2% performance gap directly quantifies the benefit of search-guided planning. Disabling policy distillation while retaining MCTS yields  $-175.23 \pm 23.41$ , demonstrating that knowledge transfer from search to policy network provides substantial gains beyond using MCTS for action selection alone. Removing Dirichlet noise results in  $-162.45 \pm 20.72$ , showing that stochastic root exploration is important for discovering diverse defensive strategies. Using fixed  $c_1 = 1.25$  instead of dynamic scheduling yields  $-158.91 \pm 21.33$ , indicating that adaptive exploration control provides modest but consistent improvements.

#### V. CONCLUSION

We presented ACDZero, a graph-guided planning framework combining graph neural networks with Monte Carlo Tree Search for automated cyber defense. ACDZero addresses topology generalization and multi-step reasoning challenges in dynamic network environments. By combining graph neural



TABLE I  
PERFORMANCE AND INTERPRETABLE CYBERSECURITY METRICS ON CAGE CHALLENGE 4.

Method	Reward (mean $\pm$ std)	Clean Hosts (ratio)	Non-Escalated (ratio)	Recovery Prec. (TP/TP+FP)	Mean TTR (timesteps)	Impact Count (per episode)	Recovery Error (%)
DQN (Tabular)	-606.20 $\pm$ 43.22	0.19	0.82	0.12	142.3	9.84	88
PPO (Tabular)	-597.28 $\pm$ 41.98	0.21	0.84	0.14	138.6	9.51	86
GCN	-193.68 $\pm$ 21.07	0.74	0.96	0.61	58.7	2.45	39
<b>ACDZero</b>	<b>-150.03 <math>\pm</math> 19.85</b>	<b>0.82</b>	<b>0.98</b>	<b>0.71</b>	<b>46.2</b>	<b>1.28</b>	<b>32</b>

TABLE II  
ABLATION STUDY ON ACDZERO COMPONENTS.

Configuration	Mean Reward	Std.
ACDZero (Full)	-150.03	$\pm$ 19.85
w/o MCTS	-193.68	$\pm$ 21.07
w/o Policy Distill	-175.23	$\pm$ 23.41
w/o Dirichlet Noise	-162.45	$\pm$ 20.72
w/o Dynamic $c_1$	-158.91	$\pm$ 21.33

networks' representational flexibility with tree search's deliberative reasoning, ACDZero achieves robust performance across diverse configurations while maintaining computational efficiency for real-time deployment. Evaluation on CAGE Challenge 4 demonstrates 29.2% performance improvement over the state-of-the-art GCN baseline, with 25% faster convergence and 5.8% lower variance. Looking ahead, ACDZERO method enables two promising extensions: (1) integrating with pre-trained policies to leverage domain knowledge as MCTS priors, accelerating exploration and convergence, and (2) learning the graph-based dynamics model from offline trajectories collected by existing systems, such as rule-based defenders or learned baselines, thereby reducing online interaction costs while preserving adaptive planning capabilities.

## REFERENCES

- [1] S. Vyas, J. Hannay, A. Bolton, and P. P. Burnap, "Automated cyber defence: A review," *arXiv preprint arXiv:2303.04926*, 2023.
- [2] S. Tang, X. Xia, M. Bilal, W. Dou, and X. Xu, "Human-centric service offloading with cnn partitioning in cloud-edge computing-empowered metaverse networks," *IEEE Transactions on Consumer Electronics*, 2025.
- [3] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 3779–3795, 2021.
- [4] S. Tang, M. Cui, L. Qi, and X. Xu, "Edge intelligence with distributed processing of dnns: A survey," *CMES-Computer Modeling in Engineering & Sciences*, vol. 136, no. 1, 2023.
- [5] T. Dam, K. Panaganti, B. Driss, and A. Wierman, "Online robust reinforcement learning through monte-carlo planning," in *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. [Online]. Available: <https://openreview.net/forum?id=m25ma707Ec>
- [6] D. Chang and Y. Li, "Mixed text recognition with efficient parameter fine-tuning and transformer," in *Neural Information Processing*. Singapore: Springer Nature Singapore, 2025, pp. 17–31.
- [7] Y. Li, T. Lan, and Z. Qi, "Inspo: Unlocking intrinsic self-reflection for llm preference optimization," *arXiv preprint arXiv:2512.23126*, 2025.
- [8] R. Chen, A. Andreyev, Y. Xiu, J. Chilukuri, S. Sen, M. Imani, B. Li, M. Gorlatova, G. Tan, and T. Lan, "A neurosymbolic framework for interpretable cognitive attack detection in augmented reality," *arXiv preprint arXiv:2508.09185*, 2025.
- [9] R. Chen, S. Hong, R. Islam, M. Imani, G. Tan, and T. Lan, "Perception graph for cognitive attack reasoning in augmented reality," in *Proceedings of the Twenty-sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2025, pp. 505–506.
- [10] S. Hong, R. Chen, R. Islam, M. Imani, G. Tan, and T. Lan, "Poster: Time-aware lstm for gaze prediction in mixed reality under latency perturbations," in *Proceedings of the Twenty-sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2025, pp. 514–515.
- [11] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.
- [12] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.
- [13] J. Czech, J. Blüml, K. Kersting, and H. Steingrimsson, "Representation matters for mastering chess: Improved feature representation in alphazero outperforms switching to transformers," in *27th European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2024, pp. 2378–2385.
- [14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017. [Online]. Available: <https://arxiv.org/abs/1712.01815>
- [15] M. Kiely, M. Ahiskali, E. Borde, B. Bowman, D. Bowman, D. Van Bruggen, K. Cowan, P. Dasgupta, E. Devendorf, B. Edwards *et al.*, "Cage challenge 4: A scalable multi-agent reinforcement learning gym for autonomous cyber defence," *AI Magazine*, vol. 46, no. 3, p. e70021, 2025.
- [16] G.-A. Vargas-Hakim, E. Mezura-Montes, and H.-G. Acosta-Mesa, "A review on convolutional neural network encodings for neuroevolution," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 1, pp. 12–27, 2021.
- [17] Ü. Çavuşoğlu, D. Akgun, and S. Hizal, "A novel cyber security model using deep transfer learning," *Arabian Journal for Science and Engineering*, vol. 49, no. 3, pp. 3623–3632, 2024.
- [18] Y. Li, J. Huang, Y. Zhang, J. Deng, J. Zhang, L. Dong, D. Wang, L. Mei, and C. Lei, "Dual branch segment anything model-transformer fusion network for accurate breast ultrasound image segmentation," *Medical Physics*, vol. 52, no. 6, pp. 4108–4119, 2025.
- [19] Y. Li, D. Chang, D. Luo, J. Huang, L. Dong, D. Wang, L. Mei, and C. Lei, "Sfmdiffusion: self-supervised monocular depth estimation in endoscopy based on diffusion models," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–9, 2025.
- [20] S. Zeng, Y. Li, K. Yang, and Y. Chen, "Local optimum time-reassigned synchrosqueezing transform for bearing fault diagnosis of rotating equipment," *IEEE Sensors Journal*, vol. 24, no. 7, pp. 10528–10539, 2024.
- [21] Z. Li, S. Tang, H. Tian, H. Xiang, X. Xu, and W. Dou, "A crowdsensing service pricing method in vehicular edge computing," in *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*. IEEE, 2024, pp. 82–89.
- [22] F. Guan, T. Zhu, W. Zhou, and K.-K. R. Choo, "Graph neural networks: a survey on the links between privacy and security," *Artificial Intelligence Review*, vol. 57, no. 2, p. 40, 2024.
- [23] Q. Yan, Z. Liang, Y. Song, R. Liao, and L. Wang, "Swingnn: Rethinking

permutation invariance in diffusion models for graph generation,” *arXiv preprint arXiv:2307.01646*, 2023.

- [24] 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)*. IEEE, 2020, pp. 1–9.
- [25] T. Kunz, C. Fisher, J. La Novara-Gsell, C. Nguyen, and L. Li, “A multi-agent cyberbattlesim for rl cyber operation agents,” in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2022, pp. 897–903.
- [26] Q. Jiang, X. Zhou, R. Wang, W. Ding, Y. Chu, S. Tang, X. Jia, and X. Xu, “Intelligent monitoring for infectious diseases with fuzzy systems and edge computing: A survey,” *Applied Soft Computing*, vol. 123, p. 108835, 2022.
- [27] A. Ridley, “Machine learning for autonomous cyber defense,” *The Next Wave*, vol. 22, no. 1, pp. 7–14, 2018.
- [28] M. Kiely, D. Bowman, M. Standen, and C. Moir, “On autonomous agents in a cyber defence environment, 2023,” URL <https://arxiv.org/abs/2309.07388>, vol. 1, no. 2, p. 4.
- [29] E. Miehling, M. Rasouli, and D. Teneketzis, “Optimal defense policies for partially observable spreading processes on bayesian attack graphs,” in *Proceedings of the second ACM workshop on moving target defense*, 2015, pp. 67–76.
- [30] X. Xu, S. Tang, L. Qi, X. Zhou, F. Dai, and W. Dou, “Cnn partitioning and offloading for vehicular edge networks in web3,” *IEEE Communications Magazine*, vol. 61, no. 8, pp. 36–42, 2023.
- [31] I. J. King, B. Bowman, and H. H. Huang, “Automated cyber defense with generalizable graph-based reinforcement learning agents,” *arXiv preprint arXiv:2509.16151*, 2025.
- [32] J. Collyer, A. Andrew, and D. Hodges, “Acd-g: Enhancing autonomous cyber defense agent generalization through graph embedded network representation.” International Conference on Machine Learning, 2022.
- [33] D. Chang, P. Xue, Y. Li, Y. Liu, P. Xu, and S. Zhang, “Calibrating and rotating: A unified framework for weight conditioning in peft,” *arXiv preprint arXiv:2511.00051*, 2025.
- [34] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of monte carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.
- [35] S. Tang, J. Chen, and T. Lan, “Malinzero: Efficient low-dimensional search for mastering complex multi-agent planning,” in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- [36] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.
- [37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.