PPO-based Agent Training for Adaptive Decoy Deployment in Cyber Defense

Zhenhong Zhong*, Jie Gao*, and Thomas Kunz†
*School of Information Technology, Carleton University, Ottawa, ON, Canada, K1S 5B6
†Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, K1S 5B6
Emails: {kevinzhong@cmail.carleton.ca, jie.gao6@carleton.ca, tkunz@sce.carleton.ca}

***Abstract*—In this paper, we present integrated PPO-based agent training for autonomous cyber defense, where enhanced state tracking and adversary behavior recognition are designed to provide PPO with the contextual information required to learn adaptive defense policies. Central to these policies is strategic decoy deployment, while the agent also learns to coordinate deception with analysis, remediation, and system restoration actions. The agent is trained to mitigate threats through adaptive deception strategies, situational awareness, and dynamic policy learning. Central to our approach is adaptive decoy deployment that reallocates deception resources based on temporal scan state tracking and adversary behavior recognition. The agent identifies the attacker type, i.e., targeted versus exploratory, based on their behavioral patterns and selects specialized defense policies accordingly. A context-aware reward shaping scheme accelerates learning by incorporating domain-specific signals to incentivize early intervention and efficient remediation. We evaluate the proposed approach, in comparison with state-of-the-art benchmarks, including Hierarchical PPO (HPPO) and Ensembled Dueling Double Deep Q Networks (DDDQN), across both attacker types. Experiment results demonstrate that our agent outperforms these baselines in many scenarios, particularly against an exploratory attacker and highlight the potential of PPO-based agent training augmented with domain-specific enhancements in cyber defense.**

***Index Terms*—Autonomous cyber defense, proximal policy optimization, blue agent training, decoy deployment.**

I. Introduction

As cyber threats grow increasingly sophisticated, autonomous cyber defense has become critical for protecting modern digital infrastructure. Traditional cyber defense methods, which rely on predefined rules or static signatures, struggle when confronting attackers with varying behaviors. This limitation has spurred interest in approaches based on machine learning, in particular Reinforcement Learning (RL) and Deep RL (DRL), which enable agents to learn effective defense through interactions with their environment [1], [2]. For example, an anomaly detection framework based on DRL was proposed in [3], with a focus on transferability across various datasets. In [4], the authors introduced an entity-based RL framework for autonomous cyber defense, using a transformer-based policy to enable generalization across different network topologies.

Among various DRL methods, Proximal Policy Optimization (PPO) has emerged as a promising choice for training agents in the case of high-dimensional decision spaces. The robust performance of PPO in sequential decision-making with continuous state spaces makes it particularly suitable for learning defense strategies [5]. For example, the authors of [6] incorporated domain-specific features, such as normalized rewards that reflect the severity of the attack and the cost of recovery, into PPO-based agents to improve the quality of the decision. The approach, however, does not incorporate decoys, which are a critical component of modern cyber defense.

Decoys serve three primary defensive purposes, i.e., diverting attackers from real assets, triggering early warning signals upon interaction, and exposing attacker behavior patterns without risking production systems [7]. In [8], cyber deception based on low- and high-interaction honeypots with different complexities was investigated, and a game-theoretic model was developed to deploy honeypots against malicious reconnaissance activities. The authors of [9] proposed a honeypot framework for software-defined networking (SDN), using honeypots to attract attackers, exploiting machine learning algorithms such as random forest and BayesNet to classify attacks, and dynamically configuring the SDN rules. However, a DRL approach that automatically adapts the decoy deployment strategy based on the attacker type for effective defense requires further investigation.

In this paper, we present a PPO-based approach for training an autonomous cyber defense agent, with a strong focus on adaptive decoy deployment. The trained agent is evaluated within the CybORG simulation framework [10], which mimics realistic enterprise networks and attacker behaviors. Our agent is designed to defend against network intrusions through timely analysis, remediation, and deception strategies. A temporal scan state tracking mechanism enables the agent to infer attacker movement and behavioral patterns, allowing it to distinguish between targeted and exploratory attacks and respond accordingly. For example, it focuses on chokepoints against targeted attackers while spreading defenses across subnets for exploratory threats. Additionally, our customized reward shaping function injects domain knowledge into the learning process, accelerating policy convergence by reinforcing desirable defensive behaviors such as early intervention and efficient resource usage. These core design components are seamlessly integrated into the PPO learning loop to improve training efficiency and runtime adaptability.

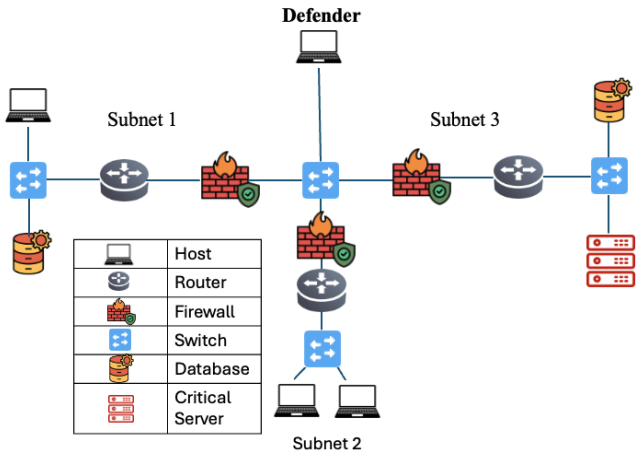

Fig. 1: An Example Network Scenario.

II. Problem Description

A. Network Scenario

An enterprise network comprising N_s segmented subnets with different roles and N_h hosts is considered. An example with 3 subnets, 6 hosts, and 1 defender is illustrated in Fig. 1, while the proposed approach applies to larger networks with more subnets and hosts. In Fig. 1, Subnet 1 contains a user workstation and a database server, forming part of the user zone. Subnet 2 includes two user workstations, representing a secondary user or administrative zone. Subnet 3 houses a database server and a critical server, representing the operational zone. The critical server in Subnet 3 is the ultimate target of the red agent.

A centralized defense agent, labeled as "defender" in Fig. 1 aims to protect the network, continuously monitoring it for signs of intrusion, performing system restoration when compromise is detected, and deploying decoys. Decoys are deployable on selected hosts that support compatible network services, allowing deception efforts to align with each host's role and exposure within the environment. The segmented network architecture introduces natural control points, and the network defender should allocate resources strategically across the subnets.

An attacker aims to compromise the critical server. It can be either of two types, both reflecting common attack patterns: 1) a *B_Line Agent (Targeted Attacker)*, which follows a direct path from an entry point to the critical server, progressing quickly through chokepoints with minimal scan. This type of attacker performs focused reconnaissance and aims to reach critical infrastructure with high efficiency; 2) a *Meander Agent (Exploratory Attacker)*, which gradually compromises an entire subnet, by scanning broadly and establishing persistence across multiple hosts before progressing to another subnet. Such an attacker prioritizes complete control over speed. Both types of attackers can perform attack actions such as reconnaissance, exploitation, privilege escalation, and lateral movement.

B. Decoy Deployment

Decoys are deployed to mislead attackers and protect real infrastructure [11]. They can mimic a variety of services typically targeted by attackers, such as web servers, application platforms (for example, Tomcat), authentication endpoints (e.g., SSH), and mail servers. In the scenario considered, a decoy should match the role of the corresponding host to increase its believability and maximize the value of the deception.

The strategic placement of the decoys plays a central role in the proposed approach. For example, gateway servers and other chokepoints, through which attackers must pass to reach operational zone, should be prioritized for deploying decoys of diverse types. Similarly, critical server of Subnet 3 in Fig. 1 should receive enhanced protection to increase the likelihood of engaging the attacker with decoys.

Our agent implements a dynamic decoy management strategy that prevents redundant deployments and ensures broad network coverage. A deployment tracker maintains a record of decoy types assigned to each host, enabling the agent to place decoys intelligently and avoid over-saturation. This can extend the time window for the defense agent in detection and remediation to improve the overall security posture.

III. The Proposed PPO-Based Defense Agent Training

This section introduces the core components of the proposed approach and provides a brief overview of how PPO supports defense agent training.

A. Core Components

Fig. 2 presents the workflow of our PPO-based approach, which involves four core components, detailed as follows.

1) Enhanced State Tracking: Our defense agent maintains an N_h-element scan vector s that tracks attacker scan activities across the network. Each entry s_i equals 2 if host i was scanned in the current step (only one host can be scanned per step, at most one entry will be 2), 1 if it was scanned in any earlier step of the episode, and 0 if it has never been scanned. At each time step, we first decay the vector by setting any entry currently at 2 down to 1 (while letting entries at 1 stay unchanged), and then mark the newly scanned host with a value of 2. By this two-phase update, the vector both highlights the single most recent scan and retains a record of all prior scans in the same episode, giving the agent robust situational awareness across the network.

2) Adversary Behavior Recognition: The sum of the elements in the state vector reflects the number of hosts recently scanned. Since a B_Line agent follows a direct path to the final target, which results in fewer scan activities, a lower sum suggests a higher chance of a targeted attack by a B_Line agent. By contrast, a higher sum suggests a higher chance of a Meander agent as it explores and scans widely.

3) Strategic Decoy Deployment: Our defense agent is trained to assign decoys based on a prioritized host list, focusing on key assets, and the attacker type. For example,

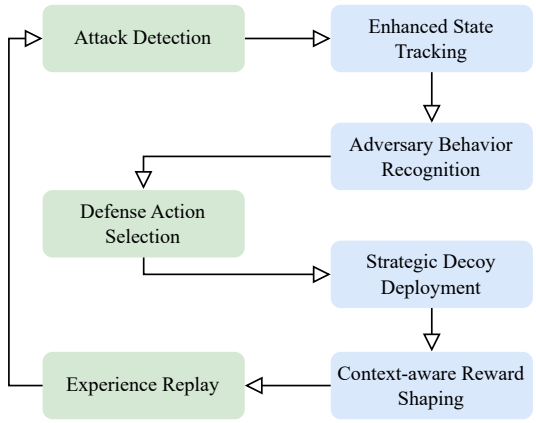

Fig. 2: Workflow of the PPO-based Agent Training. The 3 blocks on the left represent standard training steps, while the 4 blocks on the right highlight our design.

the agent may deploy multiple distinct decoys on a high-value server against a B_Line agent. In contrast, it may distribute different decoys across hosts, creating a wide deception net when defending against a Meander agent. For realism, the agent matches decoys to the port and service configurations of each host. By only deploying decoys that mimic services actually supported by a host, the agent avoids inconsistencies that may be recognized and increases the likelihood of engaging the attacker.

4) Context-Aware Reward Shaping: To accelerate learning, we implement reward shaping, during the training phase only, to provide immediate feedback based on the timing and strategic value of defense actions. This design encourages the agent to adopt proactive rather than reactive defense strategies. Specifically, we define the following reward shaping variables.

- R_1^s: the reward for deploying decoys within early steps of an episode, which encourages early deception.
- R_2^s: the reward for correctly analyzing compromised hosts, which reinforces effective detection.
- $-R_3^s$: the negative reward (penalty) for restoring clean systems or deploying redundant decoys, which discourages wasteful actions.
- R_1^e: the bonus for completing an episode with no infrastructure compromised.
- R_2^e: the final reward for ending an episode with no host under attacker control.

The above rewards guide the agent toward learning both efficient low-level actions and high-level strategic goals.

The reward shaping complements the baseline reward, which decreases when the attacker successfully compromises a host or progresses in the attack. It incentivizes the defense agent to minimize the attacker persistence, avoid costly remediation, and protect high-value assets. Together, the reward components provide informative learning signals that promote faster convergence and more robust defense behavior [12]. With the reward shaping, the reward received by the defense agent at each time step is given by[1]:

$$R_{\text{Blue}} = -0.1(h_{\text{red}} - 1) - s_{\text{red}} - r_{\text{blue}} - 10(i_{\text{red}}),$$

where h_{red} and s_{red} are the number of hosts and servers that the attack (red) agent has root access to at the current step, respectively, r_{blue} is 1 if the defense (blue) agent has taken an action to recover a compromised system at that step and 0 otherwise, and i_{red} is 1 if the attacker has successfully impacted the critical server and 0 otherwise. This reward formulation is adapted to our environment and training goals, building on insight drawn from prior work such as [13]. It incentivizes the agent to minimize compromise and prevent escalation while efficiently allocating its defense actions.

B. PPO-based Training

We use PPO, an on policy method known for training stability and sample efficiency [14], to train the defense agent. Between the two main PPO variants, PPO-Clip is adopted, which uses a clipped surrogate objective function given below to constrain the policy change into a specified range [15]:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right],$$

where $r_t(\theta)$ is the ratio of the probability of taking action a_t at state s_t in the current policy to that in the previous policy, ϵ specifies how far away the new policy can deviate from the previous policy, the clip function clips $r_t(\theta)$ into the range $(1-\epsilon, 1+\epsilon)$ for training stability, and $\hat{A}_t$ is the estimated advantage.

The defense agent is trained with PPO following an actor-critic model: the actor selects defensive actions based on current observations, while the critic estimates the values of those states. The core components of our proposed approach mentioned in Section III-A, including state tracking and reward shaping, are integrated into the PPO training loop, allowing the agent to learn defensive strategies that adapt to changing attack patterns and network states as the attacker progresses through its attack sequence.

IV. Experiments and Results

We conducted our experiments in the CybORG environment, which was adopted in prior research to support training pipelines for autonomous defense (blue) agents against varying attack (red) agent behaviors [16]. The environment simulates interactions between the red and the blue agents within a segmented network comprising 3 subnets with 10 hosts (i.e., $N_s = 3$ and $N_h = 10$), enterprise gateways, and an operational zone [13], [17].

[1]The subtraction of 1 from h_{red} accounts for the attacker's initial foothold on Host 0 at the start of each episode.

A. Agent and Environment Setup

1) PPO Agent Architecture: Our PPO-based agent utilizes a three-layer actor-critic neural network architecture with a 64-64 hidden node structure and ReLU activation functions. The agent observes a combined input of 62 features, including 52 environmental features and a 10-element scan vector described in Section III.A. The 52 environmental features are derived from host-level observations such as operating system type, process and service states, privilege levels, and connectivity information across the network. This combined observation allows the agent to reason both spatially and temporally about the current threat landscape. The actor outputs a probability distribution through a softmax activation. The critic network produces scalar value estimates to support advantage ($\hat{A}_t$) calculation, guiding PPO training toward long-term, efficient defense behavior.

2) Action Space: The agent's action space consists of both standard defense actions and decoy-related actions. The policy outputs a probability distribution over 36 actions. The actions include 27 standard options such as host analysis, malware removal, and system restoration, and 9 decoy deployment actions. Each decoy action represents deploying a specific decoy type such as Apache, Tomcat, SSH, or mail servers, on a selected host. These actions are chosen based on the agent's current state space, balancing between direct remediation and deception. The policy network learns not only when to deploy a decoy, but also to match the type of decoy to host characteristics, enhancing the effectiveness of decoys. This integrated action space allows the agent to coordinate strategic deception alongside traditional defense operations. To cope with more diverse or dynamic threat environment, however, further studies are required to extend beyond the limited discrete action set and handcrafted input features.

3) Parameters: Table I summarizes the key hyperparameters used in our PPO implementation. Basically, the training was conducted over 100,000 episodes, each with a maximum length of 100 steps. PPO updates were performed every 512 steps with each update involving 10 training epochs, using a discount factor ($\gamma = 0.99$) and Generalized Advantage Estimation (GAE, $\lambda = 0.95$). Table II summarizes the specific numerical values assigned to each component of the reward shaping scheme in the experiments, chosen to balance early intervention and detection accuracy.

4) Attacker Types: We evaluated our defense agent against two types of attackers, i.e., the B_line agent and the Meander agent, as mentioned in Section II-A. Experiments were conducted separately and independently for defense agent training against the two types of attackers. For each attacker type, 20+ experiment runs were conducted. By evaluating against both types of attackers, we aim to provide insights into the agent's ability to defend against different attack behaviors.

B. Evaluation Metric

The defense agent was evaluated using the metric of the cumulative reward. The per-step reward is non-positive, which represents the impact of the attacker on the considered network. Larger penalties are applied for more severe compromise events, depending on the number of hosts and servers compromised, the stage of the attack (e.g., root access or impact on operational zone), and ineffective remediation (e.g., failed restore attempts). The cumulative reward is negative, and a higher value represents better defense performance. The experiments are conducted over three time durations: 30, 50, and 100 steps for evaluating defense strategies across tactical (short-term), operational (medium-term), and strategic (long-term) timeframes.

- *30-step evaluation:* assesses the agent's ability to rapidly identify attacks and deploy initial countermeasures during the reconnaissance and early exploitation phases.
- *50-step evaluation:* assesses the agent's sustained defensive performance as the attacker progresses through lateral movement phases.
- *100-step evaluation:* assesses comprehensive defensive resilience throughout complete attack sequences, including defense of high-value targets.

TABLE I: PPO Hyperparameters

Hyperparameter	**Value**
Training Episodes	100,000
Steps per Update	512
Epochs per Update	10
Batch Size	64
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Clip Range (ϵ)	0.2
Learning Rate	3×10^{-4}
Entropy Coefficient	0.01
Activation Function	ReLU
Network Architecture	[64, 64] (Actor-Critic)

TABLE II: Reward Shaping Parameters for Training

Reward Component	**Value**
R_1^s (early decoy deployment)	+5
R_2^s (correct host analysis)	+3
$-R_3^s$ (negative reward for wasteful actions)	-2
R_1^e (episode without infrastructure compromise)	+5
R_2^e (episode with no host compromised)	+10

TABLE III: Performance Comparison of Defense Agents

Steps	Against B_line Agent			Against Meander Agent		
	Proposed	HPPO	DDDQN	Proposed	HPPO	DDDQN
30	-5.35	-4.44	-5.87	-5.54	-6.57	-10.93
50	-12.96	-7.70	-10.82	-8.82	-11.78	-26.29
100	-36.89	-15.68	-23.96	-16.47	-29.52	-67.89

C. Results and Discussion

Table III summarizes the cumulative reward of our PPO-trained agent, in comparison with two benchmarks, an HPPO

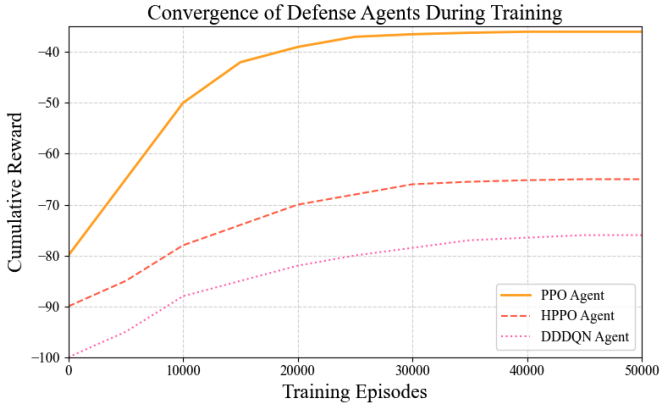

Fig. 3: Convergence of the Cumulative Reward in Training.

agent (ranked 11 in the global leader board of TTCP Cage 2 submissions [10]) and a DDDQN agent (ranked 17) over 30, 50, and 100 steps.

Against the exploratory Meander Agent, our PPO-based defense consistently outperformed both HPPO and DDDQN over all timeframes. This strong result highlights our agent's ability to distribute decoys in response to widespread scanning by an exploratory attacker. The results are achieved due to the adaptability gained through scan state tracking and context-aware reward shaping, which are core components in our proposed approach. Against the B_line Agent, our agent ranked between the two benchmarks in the 30-step evaluation, due to rapid situational awareness and effective early stage decoy placements. In the 50-step or 100-step evaluation, our agent does not outperform the benchmarks. This indicates room for improvement in sustaining long-term defense effectiveness against targeted attackers, potentially by refining the temporal decay mechanism of our scan state tracking.

These results highlight a trade-off. Broad decoy distribution across subnets is effective against the Meander agent but dilutes coverage at critical chokepoints, reducing effectiveness against the B_line agent. This explains why the proposed agent consistently outperforms baselines against Meander but not the HPPO baseline against B_line in Table III.

Note that the enhanced performance of our PPO agent against the Meander Agent was not achieved at the cost of extensive training episodes. As shown in Fig. 3, the performance of the PPO agent stabilized after approximately 30,000 episodes, comparable to that of the benchmarks. The figure is based on the average of 14 independent training runs, all of which exhibit similar convergence behavior. The agent's ability to learn a well-performing defense strategy quickly suggests that the reward shaping design is effective in accelerating the learning process by providing timely and context-aware feedback.

Table IV shows that our agent developed a highly adaptive decoy deployment strategy. Against targeted attacks by the B_line agent, it consistently placed decoys near critical chokepoints such as the firewall/router at subnet boundaries to mislead the attacker, which attempts direct traversal toward

TABLE IV: Decoy placement distribution by host category

Host category	**B_line agent (%)**	**Meander agent (%)**
critical server	42	18
database	17	23
host (workstation)	10	37
router/switch (chokepoint)	31	22

TABLE V: Performance with and without Decoy Deployment

Steps	Against B_line Agent			Against Meander Agent		
	Proposed	w/o Decoy	Heuristic	Proposed	w/o Decoy	Heuristic
30	-5.35	-68.07	-58.83	-5.54	-33.07	-44.29
50	-12.96	-110.61	-93.05	-8.82	-80.34	-85.75
100	-36.89	-186.99	-184.34	-16.47	-194.89	-192.63

the critical server. In contrast, against exploratory attacks by a Meander agent, our PPO agent distributed decoys more broadly across the subnets to cover various potential access points. The distinct deployment strategies enabled effective deception according to the attacker type.

D. Impact of Decoy Deployment

In training, the defense agent learns to take actions beyond deploying decoys, including host analysis, malware removal, and system restoration. To understand the contribution of decoy deployment to the overall effectiveness of the approach, we conducted an ablation study by disabling decoy deployment while keeping all other components unchanged. The performance of the modified agent was compared to the full PPO agent and a heuristic agent provided by the CybORG environment. The heuristic agent prioritizes restoring compromised systems and analyzing suspicious hosts following predefined rules but lacks support for decoy deployment or adaptive decision-making.

As shown by the performance results in Table V, strategic decoy deployment is critical to the effectiveness of the PPO agent. Compared to the PPO agent without decoy deployment and the heuristic agent, the proposed agent consistently achieves significantly better performance against both B_line and Meander attackers across all timeframes.

Moreover, as the timeframe increases from 30 steps to 100 steps, the gap between the full PPO agent and the PPO agent without decoy deployment becomes more evident. This underscores the growing importance of strategic decoy deployment in prolonged engagements, particularly against a Meander agent, where broader network coverage is essential. As shown in Table IV, the decoy placement pattern adapts to the attacker type, which contributes to the overall performance.

E. Impact of Reward Shaping Components

Next, we tested the performance of the PPO agent with the contrast parameters in Table VI, which represents a non-optimal baseline as compared to the tuned and empirically optimized parameters in Table II. Fig. 4 shows the performance of the PPO agent under both parameter sets, demonstrating a significant performance gap. This indicates the importance of proper reward shaping settings in the training stage.

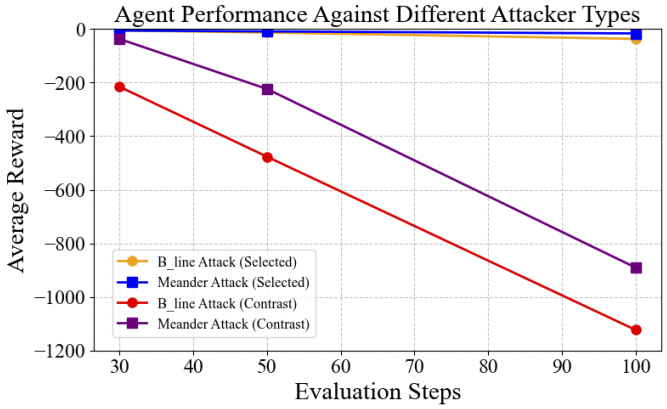

Fig. 4: Comparison of agent performance under two reward-shaping schemes against B_line and Meander attackers.

TABLE VI: Contrast Reward Shaping Parameters

Reward Component	**Value**
R_1^s (early decoy deployment)	+3
R_2^s (correct host analysis)	+2
$-R_3^s$ (penalty for wasteful actions)	0
R_1^e (no infrastructure compromised)	+3
R_2^e (no host compromised)	+5

In Table VI, $R_2^e = +5$ is the highest, causing the agent to primarily focus on host protection; the values $R_1^s = +3$ and $R_1^e = +3$ render early decoy deployment and infrastructure protection as secondary priorities; the choice of $R_2^s = +2$ encourages host analysis for anomaly detection; and setting $-R_3^s$ to 0 avoids penalizing wasteful actions. The absence of such a penalty allows redundant defensive actions and can potentially distract the agent from decoy deployment, contributing to the low reward in Fig. 4.

In transitioning from Table. VI to Table. II, each reward component was re-weighted to better align with operational priorities. The early-decoy deployment reward R_1^s was raised from +3 to +5, and the host-analysis reward R_2^s increased from +2 to +3, thereby sharpening the focus on decoy deployment while motivating host analysis. The penalty for wasteful actions $-R_3^s$ shifted from 0 to –2 to discourage redundant restore and removal actions. Finally, the episode-level integrity bonuses R_1^e and R_2^e were increased from +3 to +5 and from +5 to +10, respectively, to underscore the critical importance of preserving both infrastructure and host integrity. These coordinated adjustments serve to enforce efficient decoy use and strengthen the agent's strategic defense capability. While these results highlight the effectiveness of carefully tuned parameters, the current reward shaping design is manually configured and scenario-specific. Future work could explore automating parameter selection, for example through meta-learning or Bayesian optimization, to improve generalization across diverse environments.

V. Conclusion

In this work, we presented a PPO-based autonomous cyber defense agent that leverages four core components to improve adaptability and performance in adversarial network environments. By integrating enhanced state tracking, adversary behavior recognition, strategic decoy deployment, and context-aware reward shaping, our agent is able to learn effective defense strategies against both targeted and exploratory attackers and achieve competitive cumulative rewards in all timeframes. The results demonstrate the potential of PPO-based defense agent training for adaptive decoy deployment. Future research may incorporate adversary modeling for decoy deployment based on predicted attack paths, address scalability challenges in larger networks with expanded state and action spaces, and extend the approach beyond CybORG to real-world enterprise environments.

References

[1] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 4793–4813, Nov. 2021.

[2] J. F. Loevenich et al., "Training Autonomous Cyber Defense Agents: Challenges & Opportunities in Military Networks," *IEEE MILCOM'24*, Washington, DC, USA, 2024, pp. 158-163.

[3] M. He, X. Wang, P. Wei, L. Yang, Y. Teng and R. Lyu, "Reinforcement Learning Meets Network Intrusion Detection: A Transferable and Adaptable Framework for Anomaly Behavior Identification," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 2, pp. 2477-2492, Apr. 2024.

[4] I. S. Thompson, A. Caron, C. Hicks, and V. Mavroudis, "Entity-based Reinforcement Learning for Autonomous Cyber Defence," *ACM AutonomousCyber'24*, Salt Lake City, USA, 2024, pp 56–67.

[5] M. Wolk, et. al., "Beyond CAGE: Investigating Generalization of Learned Autonomous Network Defense Policies," *NeurIPS'22 Workshop*, New Orleans, U.S., 2022.

[6] 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," *IEEE ICPADS'23*, Ocean Flower Island, China, 2023, pp. 928–935.

[7] P. Radoglou-Grammatikis et al., "Strategic Honeypot Deployment in Ultra-Dense Beyond 5G Networks: A Reinforcement Learning Approach," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 2, pp. 643-655, Apr.-June 2024.

[8] A. H. Anwar, M. Zhu, Z. Wan, J. -H. Cho, C. A. Kamhoua and M. P. Singh, "Honeypot-Based Cyber Deception Against Malicious Reconnaissance via Hypergame Theory," *IEEE GLOBECOM'22*, Rio de Janeiro, Brazil, 2022, pp. 3393-3398

[9] J. Franco, A. Aris, L. Babun and A. S. Uluagac, "S-Pot: A Smart Honeypot Framework with Dynamic Rule Configuration for SDN," *IEEE GLOBECOM'22*, Rio de Janeiro, Brazil, 2022, pp. 2818-2824.

[10] "TTCP CAGE Challenge 2" [Online]. Available at: https://github.com/cage-challenge/cage-challenge-2

[11] A. H. Anwar, C. A. Kamhoua, N. O. Leslie and C. Kiekintveld, "Honeypot Allocation for Cyber Deception Under Uncertainty," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 3, pp. 3438-3452, Sept. 2022.

[12] E. Bates, V. Mavroudis, and C. Hicks, "Reward Shaping for Happier Autonomous Cyber Security Agents," *ACM AISec'23*, Copenhagen, Denmark, 2023, pp. 221-232.

[13] M. Kiely, D. Bowman, M. Standen, and C. Moir, "On Autonomous Agents in a Cyber Defence Environment," in *ACD Workshop'23*, Melbourne, USA, 2023, pp. 1-8.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, arXiv:1707.06347.

[15] N.-C. Huang, P.-C. Hsieh, K.-H. Ho, and I.-C. Wu, "PPO-Clip Attains Global Optimality: Towards Deeper Understandings of Clipping", *AAAI*, vol. 38, no. 11, pp. 12600-12607, Mar. 2024.

[16] M. O. Farooq and T. Kunz, "A Generic Blue Agent Training Framework for Autonomous Cyber Operations," in *Proc. IFIP Networking*, Thessaloniki, Greece, 2024, pp. 515-521.

[17] 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.