Collaborative Reinforcement Learning for Cyber Defense: Analysis of Strategies, and Policies

Davide Rigoni[1], Rafael F. Cunha[2], Frank Fransen[2], Puck de Haan[3], Amir Javadpour[4] and Fatih Turkmen[1,*]

[1]*Independent*
[2]*University of Groningen, The Netherlands*
[3]*TNO, The Netherlands*
[4]*MOSA!C Lab / ICTFICIAL Oy, Finland*

Abstract

As cybersecurity threats grow in scale and sophistication, traditional defenses increasingly struggle to detect and counter them. Recent work applies reinforcement learning (RL) to develop adaptive defensive agents, but challenges remain, particularly in how agents learn, the environments used, and the strategies they adopt. These issues are amplified in multi-agent settings, where coordination becomes especially complex. This paper presents an empirical analysis of collaborative RL for cybersecurity defense, focusing on environment models, RL methods, and agent policies. Specifically, it compares several multi-agent RL algorithms in the context of CAGE Challenge 4 to identify effective defense configurations. The study also evaluates the learned policies to assess their real-world applicability and highlight gaps between agent behavior and practical defense strategies.

1. Introduction

As cybersecurity threats grow in volume and complexity, traditional detection and defense methods face increasing challenges. Advances in machine learning, especially deep learning and reinforcement learning, have improved cybersecurity practices [1]. RL, in particular, has gained attention for automating defense mechanisms and testing strategies [2].

Given this progress, it is crucial to explore how reinforcement learning can be effectively applied to cybersecurity, focusing on methods with real-world applicability. This is especially important in complex environments with multiple agents collaborating to defend a system. This study addresses two key research questions in the context of CAGE Challenge[1]:

- **RQ.1:** Which reinforcement learning methods are the most effective for intrusion prevention in the chosen baseline environment?
- **RQ.2:** How can collaborative reinforcement learning policies be extracted and applied to defend against intrusion in a selected environment?

2. Background

2.1. Multi-Agent Reinforcement Learning

This section discusses Multi-Agent Reinforcement Learning (MARL) as per Albrecht et al. [3]. MARL offers robustness and scalability since the system can handle agent failures and new agents can be easily added [4]. In intrusion prevention, a multi-agent setup naturally accommodates multiple attackers and defenders, thereby enhancing simulation realism.

MARL configurations include fully **cooperative** agents working toward a shared reward, **competitive** agents opposing each other, or **mixed** settings where groups cooperate internally but compete

SPAIML'25: International Workshop on Security and Privacy-Preserving AI/ML, October 26, 2025, Bologna, Italy
*Corresponding author.
f.turkmen@rug.nl (F. Turkmen)

[1]https://github.com/cage-challenge

Table 1
Overview of the multi-agent reinforcement learning methods used in this study, highlighting key differences in training paradigms, coordination mechanisms, and architecture.

	IPPO	MAPPO	QMIX	MADDPG
Method type	On-policy, Policy-based	On-policy, Policy-based	Off-policy, Value-based	Off-policy, Actor-Critic
Framework	DTDE	CTDE	CTDE	CTDE
Collaboration	Low	High	High	High
Scalability	High	Low	Low	Low

externally. Training frameworks in MARL range from Decentralized Training and Decentralized Execution (DTDE), where each agent learns independently, to centralized approaches that use global information. These can be a fully Centralized Training and Centralized Execution (CTCE) model with a single policy controlling all agents, or a hybrid Centralized Training and Decentralized Execution (CTDE) framework, which trains individual policies together for coordinated, decentralized action. Key challenges in MARL stem from these designs, including the credit assignment problem, non-stationarity in DTDE, scalability issues in CTCE, and achieving robust coordination in CTDE.

2.1.1. Dec-POMDP

The **Markov Decision Process (MDP)** provides a formal framework for single-agent decision-making in fully observable environments. This is extended to the Partially Observable Markov Decision Process (POMDP) when the agent's view is incomplete. For multi-agent systems, particularly in the fully cooperative setting where agents work towards a common goal, the Decentralized POMDP (Dec-POMDP) offers a suitable formalization. A Dec-POMDP models a team of agents, each receiving only a local observation of the environment and acting decentrally. This inherent challenge of coordinating with limited information is central to our intrusion prevention scenario. More details on our implementation are provided in Section 2.3.

2.2. Reinforcement Learning Methods

Our study evaluates a selection of representative MARL algorithms. As a DTDE baseline, we employ Independent PPO (IPPO) [5], where each agent learns its policy entirely separately. The majority of our analysis focuses on the CTDE framework, for which we examine both policy-based and value-based methods. For policy-based CTDE, we include: MAPPO [6], which uses a single centralized critic to train decentralized policies, and MADDPG [7], a deterministic actor-critic method where each agent's policy is trained using a dedicated critic that has access to global information. For value-based CTDE, we use QMIX [8], which learns a monotonic joint action-value function through a central mixing network. Table 1 highlights the key differences among these methods.

2.3. CAGE Challenge 4 and CybORG

CAGE Challenge 4 features a network divided into four subnetworks connected via the internet, as shown in Figure 1. These include two deployed networks (each with restricted and operational zones), a Headquarters network with three security zones, and a contractor network used by red team attackers. Server, host, and service distributions across zones are randomized. Hosts contain information ranging from basic system details to session credentials, which red agents can exploit to gain user or root access. Each of the four blue agents defends one subnetwork, and the fifth agent (Agent 5) defends three subnetworks.

The challenge proceeds in episodes, and each episode is non-uniform. Episodes progress through phases that prioritize missions in specific deployed networks, guiding agent actions accordingly. There are three different phases with varying levels of priorities over the missions the agents can carry on:

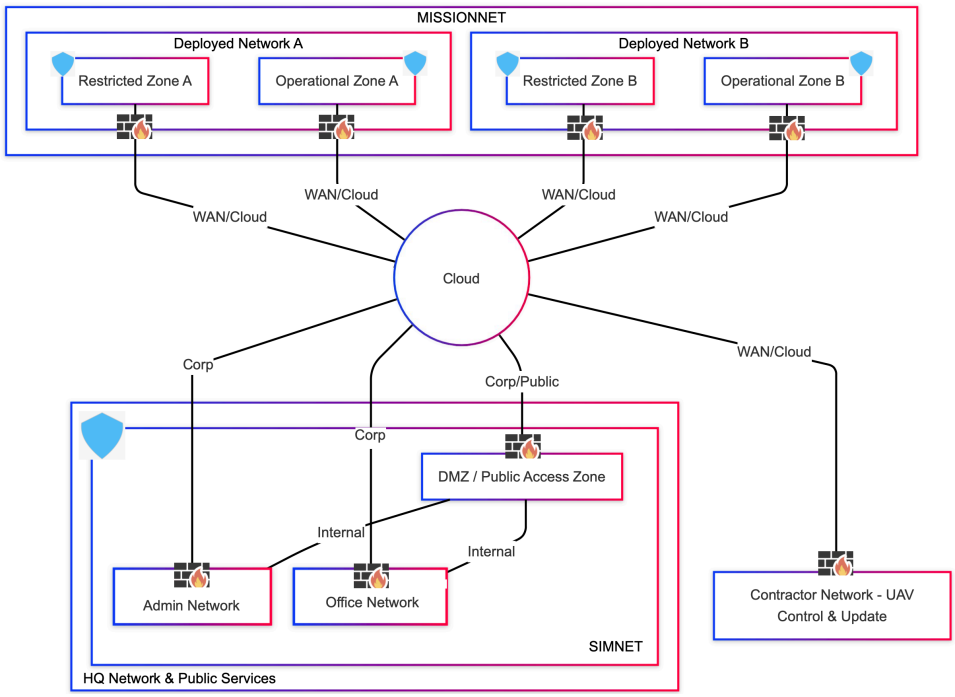


Figure 1: Implemented Network layout based on CybORG CAGE Challenge 4.

Phase 1, Phase 2A, and Phase 2B. In line with the CAGE Challenge 4 [9], we employ the CybORG environment in our study. As with most multi-agent environments, CybORG models the challenge scenario as a Dec-POMDP. In what follows, we will detail its components as outlined in Section 2.

Agents: In a multi-agent environment, *Red* and *Blue* team agents operate within the network summarized previously. The Red Agent aims to disrupt availability by infecting hosts, while the Blue Agent works to prevent this. Each agent can perform a specific set of actions (> 80), such as *Sleep* or *Restore* each of which take a certain number of (time) 'ticks' to execute—simple actions occur immediately, while more complex ones take longer. One of the blue agents (agent 4) is special since it has an action set of more than 200 actions.

One of the key features of CAGE Challenge 4 is the inclusion of *green agents*, representing standard network users vulnerable to possible attacks, such as phishing, by red agents. Maintaining network availability for these green agents is crucial, with penalties applied to blue agents if access to the network by green agents is disrupted.

Lastly, the red agent is modeled as a Finite State Machine (FSM), with states representing host interactions based on discovery and other offensive actions.

Rewards: As mentioned earlier, the rewards for the blue agent are based upon the network's availability and the quality of work performed by the green agents at any given time. To minimize these disruptions, the blue agent begins with a reward of 0, which gradually decreases each time the green agents are unable to perform their tasks or if there are any disruptions caused by the red agents. Furthermore, Phase 2A focuses on maintaining operational activities for Zone A, while Phase 2B pertains to maintaining operational activities for Zone B. Consequently, failure to ensure service availability for these operational zones during their respective phases will result in significant penalties for the blue agents. In summary, the blue agents can achieve a **maximum reward** of 0 if they successfully prevent all red agent attacks proactively.

Observations: The observation space for agents is a key consideration in CAGE 4. At each time step, agents receive observations that include the current *mission phase*, details about each *observable*

subnet (such as its state, communication policy, and any detected malicious activity), and 8-bit long *messages* from other agents to support coordination. The observation space size varies by agent, as those defending multiple subnets receive more information. Most agents have a flat observation space of size 92, while agent 4, which oversees multiple subnets, has a space of size 210.

3. Methodology

3.1. Reinforcement Learning Methods

In our analysis, we employed various variants of RL algorithms that enhance the methods presented in Section 2. In this section, we provide an overview of these methods in relation to the original RL methods and the CybORG environment, as they are relevant to the experiments.

Global State Representation: Yu et al. [6] highlight inefficiencies in MAPPO's standard global state representation, which concatenates all agent observations. To reduce redundancy, they propose a more compact method to combine message states and initial mission values. We adopt this approach in our study, referring to it as C_MAPPO, which results in a global state that is approximately 20% smaller.

Exploration Strategy: For an alternative approach to the ϵ-greedy exploration strategy commonly used in QMIX, we considered the Boltzmann softmax action selection operator [10]. Unlike ϵ-greedy, whose exploration consists of selecting an action uniformly at random and can thus be suboptimal, Boltzmann softmax assigns a weighted probability distribution to actions based on their Q-values (expected cumulative reward of taking a joint action in the multi-agent setting), with the balance between exploration and exploitation controlled by a temperature parameter τ. Higher values of τ increase exploration by making action probabilities more similar, while lower values emphasize exploitation by favoring actions with higher Q-values ($Q_t(a)$) as can be seen from Equation 1. As a result of the discussion of this study, we will refer to the QMIX implementation using the Boltzmann softmax as B_QMIX.

$$\Pr(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{a' \in A} e^{Q_t(a')/\tau}} \tag{1}$$

Prioritized Experience Replay: Previously, we introduced QMIX and MADDPG as off-policy methods for the CybORG use case, both of which rely on replay buffers. Instead of using standard uniform sampling, we adopt Prioritized Experience Replay (PeR) as proposed by Schaul et al. [11], allowing more informative transitions, based on Temporal-Difference (TD) error, to be sampled more often. To prevent overfitting to high-TD transitions, prioritization (p_i) is balanced using a formula (Equation 2) with a tunable α parameter. New transitions are initially assigned the highest priority to ensure they are sampled at least once. We refer to these modified implementations as PeR-QMIX and PeR-MADDPG in the remainder of the paper.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{2}$$

Hyperparameter search: Hyperparameter selection is critical for training RL agents, especially in complex environments like CybORG. Full tuning is computationally costly—each training run takes approximately 30 hours on an Intel Core i7-1065G7. However, the main bottleneck lies in the environment steps rather than agent training itself. This can be mitigated using GPUs and parallelized methods such as PPO. We focused on two key hyperparameters: the discount factor γ and learning rate lr. Based on tuning with IPPO, the best values were $\gamma = 0.99$ and $lr = 0.00025$ (Table 2).

Table 2
Hyperparameter search performed on the CybORG environment with IPPO

Parameter	Values
Discount Factor (γ)	[0.6, 0.9, 0.99]
Learning Rate (lr)	[0.00025, 0.005, 0.01]

3.2. Recurrency in CybORG

A key enhancement to the methods discussed is adding recurrence to the network architecture. This is especially useful in partially observable environments, such as CybORG, where actions have delayed effects and temporal dependencies are crucial. We train on full episodes without truncation, so there is no need to store hidden states mid-sequence. Hidden states are reset at the start of each episode, keeping the buffer and algorithm logic unchanged.

We implemented recurrent versions of PPO variants and QMIX, using the R prefix (e.g., R_QMIX). For the recurrent layers, we use LSTM for PPO-based methods and GRU for QMIX to compare both architectures in the CybORG setting.

3.3. Messages

As detailed earlier, CybORG allows blue agents to send 8-bit messages to support coordination. We designed three message types based on coordination theory, falling into two categories: action messages and state messages. **Action messages** inform others of intended actions, helping reduce non-stationarity due to delayed action effects. Each action is encoded in 8 bits. **State messages** share subnetwork status. A 2-bit version signals the presence of malicious activity, while the 8-bit version provides counts of infected processes and events (4 bits each). These strategies improve coordination without expanding the observation space or sharing sensitive data, maintaining decentralized training and secure network separation.

3.4. Rewards

A key challenge in MARL is attributing rewards when all agents share the same global reward. This can make it difficult for agents to identify whether their actions directly influenced the outcome or if it was due to others. Atrazhev et al. [12] emphasize that different reward structures can significantly impact agent performance, especially in cooperative settings.

Individual Rewards: CybORG uses a shared global reward based on network-wide outcomes. We propose an alternative reward system where each agent receives a reward based on its own actions and the state of its assigned subnet, thereby keeping the total reward unchanged but distributing it differently. Additionally, we introduce a hybrid reward combining both individual and scaled global rewards. The scaling is proportional to the number of subnetworks controlled by each agent, balancing local feedback with the overall network performance. This approach helps agents better understand their contributions while maintaining coordination.

Intrinsic Rewards: Besides external rewards from the environment, intrinsic rewards encourage exploration, which is especially useful in environments with sparse feedback. We incorporate the Never Give Up (NGU) module [13], which combines two components:

- **Episodic novelty**: rewards agents for encountering states that differ from those seen earlier in the current episode.
- **Life-long novelty**: uses Random Network Distillation (RND) to reward states that remain novel over the agent's lifetime by measuring prediction errors against a fixed random network.

The combined intrinsic reward is added during training to promote exploration and improve learning. Algorithms using NGU include the suffix in their names (e.g., IPPO_NGU).

4. Results

4.1. Results for RQ.1

With RQ.1, we aim to evaluate the performance of various RL methods in the baseline CybORG environment. Agents were trained using different seeds, each over 500-step episodes, and results were averaged across five runs. Figure 2 shows that PPO-based methods consistently outperform others, with centralized versions starting stronger but converging similarly to decentralized ones. Recurrent versions (Figure 3) perform slightly worse than non-recurrent ones.

Figure 4 compares different message types, revealing no significant benefit from adding messages to the observation space; in fact, the 2 Bits message type performed worse. Figure 5 shows that QMIX struggles to learn effectively, though improved exploration boosts the method slightly. Figure 6 presents MAPPO with two global state representations, and Figure 7 shows that altering the reward structure or adding the NGU module negatively impacts IPPO's training, confirming the baseline reward setup yields the best performance.

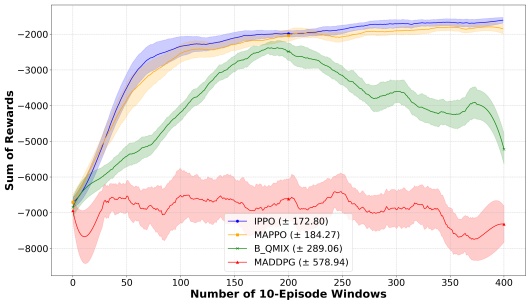
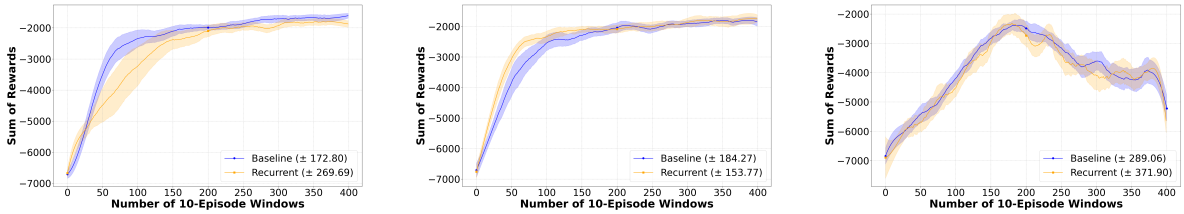
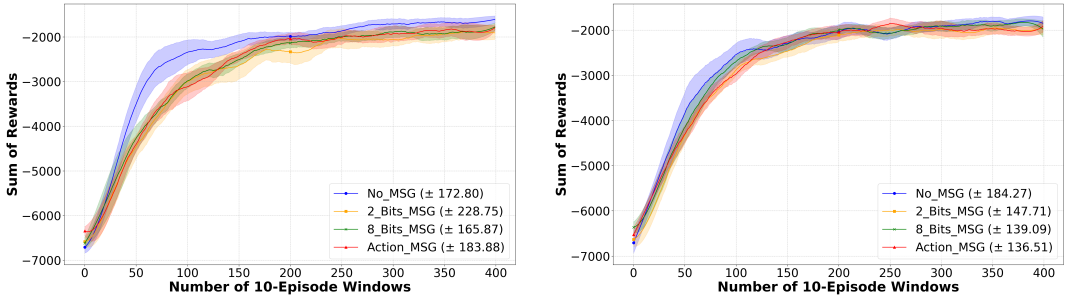


Figure 2: Results obtained from using different reinforcement learning methods in the baseline CybORG environment.



(a) IPPO **(b)** MAPPO **(c)** QMIX

Figure 3: Results obtained from using different reinforcement learning methods and their recurrent versions in the baseline CybORG environment.



(a) IPPO **(b)** MAPPO

Figure 4: Results obtained from using different reinforcement learning methods with different message versions in the baseline CybORG environment.

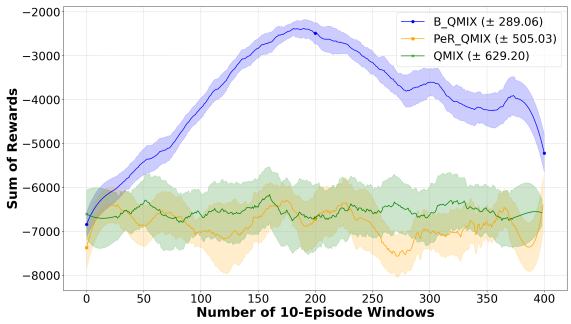
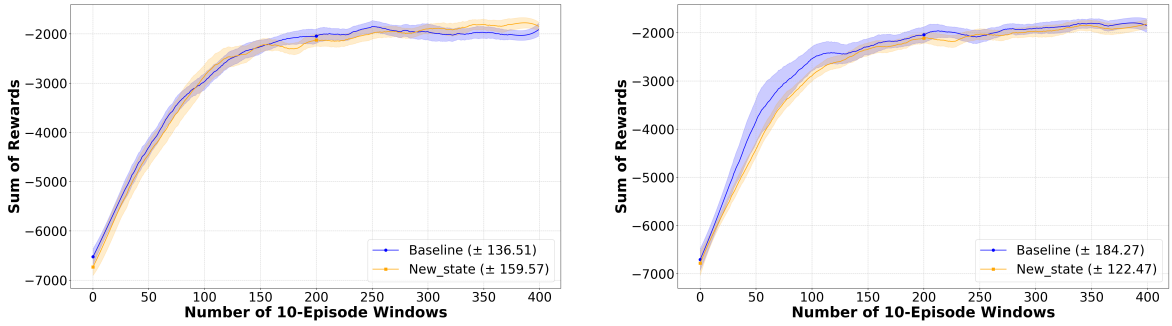


Figure 5: Results obtained from using QMIX with different buffer and exploration configurations in the baseline CybORG environment.



(a) Action Messages **(b)** No Messages

Figure 6: Results obtained from using MAPPO with a different global state representation in the baseline CybORG environment with the inclusion of Action Messages.

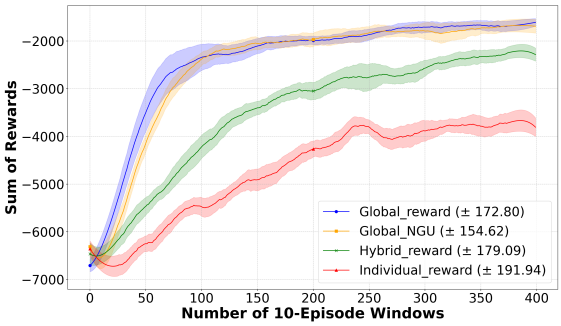


Figure 7: Impact of different reward structures on the performance of the `IPPO` algorithm. The y-axis shows the mean episodic return, averaged over 5 independent runs. Shaded regions denote the standard deviation. The results demonstrate that global reward signals (`Global_reward` and `Global_NGU`) lead to substantially higher performance and faster convergence compared to the `Hybrid` and purely `Individual` reward schemes, highlighting the critical role of a shared team objective in solving the credit assignment problem for this task.

4.2. Results for RQ.2

RQ.2 aims to analyze the policies learned by agents in the baseline CybORG environment and its variations (Section 3). We focus on the most frequently selected actions by trained blue agents across different methods and message implementations. The *Sleep* action was masked in order to allow for clearer interpretation of agent behavior. We also evaluated blue agent reactions to red agent activity but found no meaningful or consistent correlations. Similarly, policy patterns remained largely unchanged across different environment modifications and reward schemes, indicating that agent behavior was generally robust to these variations.

Figure 8 presents a simplified Finite State Machine (FSM) of the red agent's behavior. It captures key states in the red agent's attack process, from discovering host IPs (states K/KD), identifying services (S/SD), gaining user access (U/UD), to achieving root access (R/RD). Transitions between these states reflect the actions of the red agent. While the figure provides a structural view of red agent behavior, it primarily serves as context for interpreting the blue agent's defensive responses across environments.

4.3. Discussion

4.3.1. Discussion for RQ.1

Our results show that **PPO-based methods** consistently outperform off-policy approaches like MADDPG and QMIX in the CybORG environment. This is likely due to PPO's on-policy learning and its inherently stochastic policy, which promotes more effective exploration in partially observable and dynamic environments. In contrast, the deterministic policies used by off-policy methods, even when enhanced with strategies like epsilon-greedy or Gumbel-Softmax, often struggle, especially when combined with large replay buffers that can introduce outdated data. While Boltzmann exploration improved QMIX's performance, it still performs worse than PPO.

Between PPO variants, *IPPO slightly outperformed MAPPO*, though both *converged similarly*. This echoes findings from prior work [14][6], where a centralized critic, like in MAPPO, can introduce unnecessary complexity without clear benefits. Our use of a new global state representation did not improve MAPPO's performance, nor did adding messages to the observation space; this suggests that agents already learn effectively without this added complexity.

The use of recurrent layers, designed for partially observable environments, did not improve performance in our setup, possibly due to implementation details or a lack of architectural enhancements. Message passing also showed limited benefits; while richer message types, such as 8-bit and action-based messages, helped early in training, they did not lead to long-term gains. This suggests that the messages were either insufficiently useful to change learning trajectories, or that PPO agents quickly converged to local optima, where added information was ineffective.

Finally, restructuring the reward scheme to use individual or hybrid rewards also failed to outperform

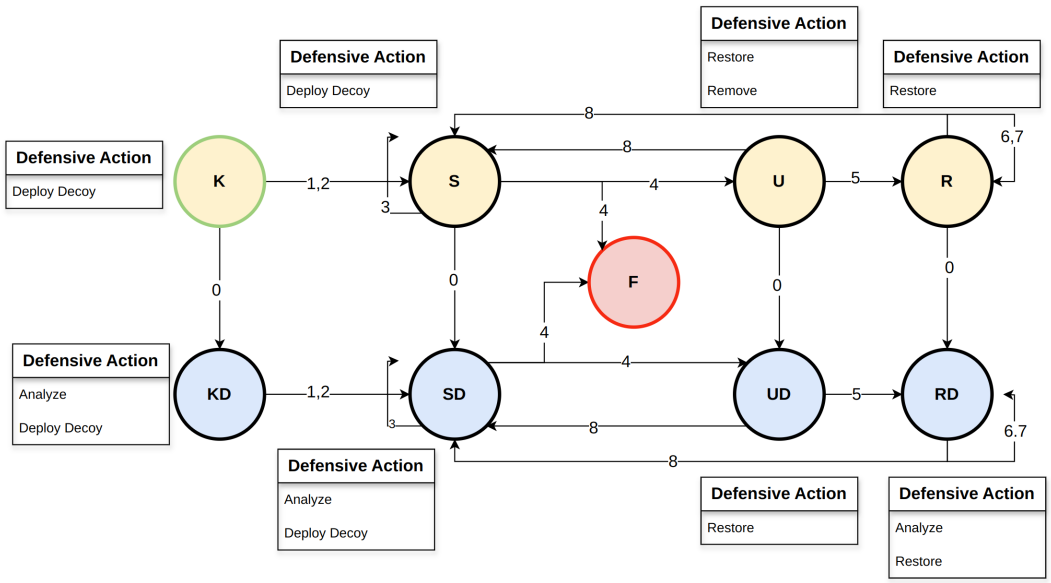


Figure 8: Most chosen action of Blue Agent per each state of the network host as presented in the red agent FSM where transitions between these states are labeled with the action numbers of the agents as described in CybORG CAGE Challenge 4 [9]

the global reward baseline. Sparse feedback in individual rewards, especially for agents with fewer responsibilities, hinders learning. Even normalization or hybrid schemes could not consistently improve results, likely due to local maxima or a limited additional learning signal.

In summary, PPO's simplicity, built-in exploration, and compatibility with the global reward setup explain its strong performance. Attempts to improve learning through message passing, recurrent models, or modified rewards did not yield clear benefits in this environment.

4.3.2. Discussion for RQ.2

We analyzed the learned defensive policies by examining the red agent's finite state machine and the response of blue agents in each state. Figure 8 reveals that blue agents tend to take more decisive actions when the red agent has escalated access, such as deploying stronger defenses in user or root shell states (U, R, RD), while preferring milder actions like Analyze or Deploy Decoy in earlier states (K, S). However, suboptimal patterns persist. For instance, analysis actions are frequently used even in critical RD states, suggesting a lack of appropriate threat response. Agents trained independently showed more consistent per-state actions but still failed to develop truly effective policies. Overall, while the FSM-based analysis reveals some alignment between threat level and response, the learned policies still fall short of cybersecurity best practices.

5. Conclusions

In this paper, we evaluated different reinforcement learning methods for intrusion prevention in the context of CAGE Challenge 4 and analyzed the resulting policies to assess the viability of collaborative RL agents for real-world cybersecurity applications.

A key finding is the lack of a standardized multi-agent RL library with diverse implementations, reflecting the relative immaturity of the field. Among the tested methods, those using stochastic policies (PPO-based methods) for balancing exploration and exploitation performed best. However, they still fell short of top-performing CAGE 4 solutions—highlighting the need for further tuning and configuration. Policy analysis showed that current defensive strategies are not yet suitable for real-world deployment. Extracting and evaluating learned policies remains challenging, though training agents directly in CAGE 4 environment may improve results. The interpretation and validation of these policies against operational standards is also challenging.

In conclusion, while collaborative reinforcement learning holds promise, it is still in its early stages of development. Significant refinement is needed before such systems can reliably support or automate network defense. Future research can focus on creating standard libraries for MARL, which would facilitate easier comparison of methods and more reproducible results. Additionally, exploring combined simulation and emulation-based environments could make experiments more realistic and help apply learned policies more effectively in real-world cybersecurity.

Declaration on Generative AI

A generative AI language model (ChatGPT) was used to improve grammar, clarity, and readability of this manuscript. The tool has not been used to generate original content, analysis, or interpretation. The authors take full responsibility for all claims and conclusions presented.

References

[1] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, J. F. Connolly, Cybersecurity threats and their mitigation approaches using machine learning—a review, Journal of Cybersecurity and Privacy 2 (2022) 527–555. URL: https://doi.org/10.3390/jcp2030027. doi:10.3390/jcp2030027.

[2] A. M. K. Adawadkar, N. Kulkarni, Cyber-security and reinforcement learning — A brief survey, Engineering Applications of Artificial Intelligence 114 (2022) 105116. URL: https://linkinghub.elsevier.com/retrieve/pii/S0952197622002512. doi:10.1016/j.engappai.2022.105116.

[3] S. V. Albrecht, F. Christianos, L. Schäfer, Multi-agent reinforcement learning: Foundations and modern approaches, MIT Press, 2024.

[4] L. Buşoniu, R. Babuška, B. De Schutter, Multi-agent reinforcement learning: An overview 310 (2010) 183–221. doi:10.1007/978-3-642-14435-6_7.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, CoRR abs/1707.06347 (2017). URL: http://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[6] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. WU, The surprising effectiveness of ppo in cooperative multi-agent games, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 24611–24624. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

[7] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

[8] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, S. Whiteson, Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018. URL: https://arxiv.org/abs/1803.11485. arXiv:1803.11485.

[9] T. C. W. Group, Ttcp cage challenge 4, https://github.com/cage-challenge/cage-challenge-4, 2023.

[10] K. Asadi, M. L. Littman, An alternative softmax operator for reinforcement learning, 2017. URL: https://arxiv.org/abs/1612.05628. arXiv:1612.05628.

[11] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, 2016. URL: https://arxiv.org/abs/1511.05952. arXiv:1511.05952.

[12] P. Atrazhev, P. Musilek, It's all about reward: Contrasting joint rewards and individual reward in centralized learning decentralized execution algorithms, Systems 11 (2023). URL: https://www.mdpi.com/2079-8954/11/4/180. doi:10.3390/systems11040180.

[13] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, C. Blundell, Never give up: Learning directed exploration strategies, 2020. URL: https://arxiv.org/abs/2002.06038. arXiv:2002.06038.

[14] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, S. Whiteson, Is independent learning all you need in the starcraft multi-agent challenge?, CoRR abs/2011.09533 (2020). URL: https://arxiv.org/abs/2011.09533. arXiv:2011.09533.