Comparative Analysis of Reinforcement Learning Algorithms for Autonomous Blue Team Defense in CybORG

Research Study on CybORG Blue Agent Training
Vasanth Iyer
Based on the CybORG Framework
Standen et al., IJCAI-21 1st International Workshop on Adaptive Cyber Defense

February 2026

Abstract

This study presents a comprehensive comparison of multiple reinforcement learning algorithms for training autonomous Blue team (defender) agents in the CybORG (Cyber Operations Research Gym) environment. We evaluate Proximal Policy Optimization (PPO), Long Short-Term Memory PPO (LSTM-PPO), and Deep Q-Network (DQN) approaches on Scenario 1b with a hierarchical action space. Our experiments reveal that DQN significantly outperforms policy gradient methods in learning state-dependent defensive strategies, achieving 39 action changes per episode compared to 0 for PPO-based methods. The DQN agent learned a cyclic defense pattern utilizing Monitor, Analyse, Remove, and Restore actions across multiple hosts, demonstrating genuine reactive behavior to the Red agent's attack sequence.

1 Introduction

Autonomous Cyber Operations (ACO) represents a critical challenge in modern cybersecurity, requiring intelligent agents capable of defending computer networks against sophisticated adversaries. The CybORG framework [1] provides a standardized environment for developing and evaluating such agents through both simulation and emulation modes.

This work focuses on training Blue team (defender) agents against the deterministic B_lineAgent Red team attacker in CybORG's Scenario 1b. We investigate multiple reinforcement learning approaches and their effectiveness in learning state-dependent defensive policies.

1.1 Problem Statement

The primary challenge identified during this research was **action collapse** - where trained agents converge to repeatedly selecting a single action regardless of the environment state. This phenomenon was observed across multiple algorithm configurations and represents a significant barrier to developing effective autonomous defenders.

1.2 Contributions

- Comprehensive comparison of PPO, LSTM-PPO, and DQN for Blue team training
- Identification of the action collapse problem in policy gradient methods
- Demonstration that DQN learns genuine state-dependent policies
- Hierarchical action space design reducing complexity from 54 to 52 actions
- Detailed analysis of learned defensive strategies

2 Background

2.1 CybORG Framework

CybORG [1] is a gym for Autonomous Cyber Operations research featuring:

- **Simulation Mode**: Finite state machine representation for rapid training
- **Emulation Mode**: AWS-based virtual infrastructure for validation
- **OpenAI Gym Interface**: Standard RL interaction protocol
- **Adversarial Scenarios**: Red vs Blue team competitions

The original CybORG paper demonstrated successful training of Red team agents using Deep Q-Networks with LSTM (DRQN), achieving a 66% transfer rate from simulation to emulation. Our work extends this to Blue team defense.

2.2 Scenario 1b Configuration

Table 1: Scenario 1b Network Configuration

Subnet	**Hosts**	**Role**
User Subnet	User0, User1, User2, User3, User4	Entry points
Enterprise Subnet	Enterprise0, Enterprise1, Enterprise2	Mid-tier targets
Operational Subnet	Op_Host0, Op_Host1, Op_Host2, Op_Server0	Critical assets
Defense	Defender	Blue team base

2.3 Action Space

Blue team agents have access to the following action types:

Table 2: Blue Team Action Types

Action	**Description**
Sleep	No operation
Monitor	Network-wide surveillance for threats
Analyse ⟨host⟩	Detailed inspection of specific host
Remove ⟨host⟩	Remove malicious processes/files
Restore ⟨host⟩	Full system restore (expensive)
Misinform ⟨host⟩	Deploy deception/honeypots

2.4 Red Team: B_lineAgent

The B_lineAgent follows a deterministic attack pattern:

1. Discover subnet via scanning
2. Exploit User subnet hosts
3. Pivot to Enterprise subnet
4. Escalate privileges toward Op_Server0

This predictable sequence should theoretically enable Blue agents to learn anticipatory defenses.

3 Methodology

3.1 Hierarchical Action Space

To reduce action space complexity, we implemented a hierarchical encoding:

$$a = t \times |H| + h \tag{1}$$

where $t \in \{0, 1, 2, 3\}$ represents action type (Monitor, Analyse, Remove, Restore), $h \in \{0, ..., 12\}$ represents host index, and $|H| = 13$ is the number of hosts.

This reduces the action space from 54 discrete actions to 52 (4×13), removing Sleep and Misinform.

3.2 Algorithms Evaluated

3.2.1 Proximal Policy Optimization (PPO)

PPO [2] is a policy gradient method that optimizes:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right] \tag{2}$$

Configuration:

- Learning rate: 3×10^{-4}
- Entropy coefficient: 0.05
- Network: MLP [256, 256]
- Training steps: 500,000

3.2.2 LSTM-PPO (Recurrent PPO)

RecurrentPPO extends PPO with LSTM memory to capture temporal patterns:

Configuration:

- LSTM hidden size: 128
- LSTM layers: 1
- Shared LSTM for policy and value
- Entropy coefficient: 0.01
- Training steps: 500,000

3.2.3 Deep Q-Network (DQN)

DQN [3] learns action-value function $Q(s, a)$ directly:

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a') \tag{3}$$

Configuration:

- Learning rate: 1×10^{-4}
- Replay buffer: 100,000 transitions
- Exploration: ϵ from 1.0 to 0.05 over 30% of training

- Network: MLP [256, 256, 128]
- Target update interval: 1,000 steps
- Training steps: 500,000

3.3 Evaluation Metrics

- **Mean Reward**: Average episode return (native CybORG reward)
- **Unique Actions**: Number of distinct actions per episode
- **Top Action %**: Percentage of most common action
- **Action Changes**: Number of times action changes within episode
- **Deterministic vs Stochastic**: Comparing $\arg\max$ vs sampled action selection

4 Results

4.1 Overall Performance Comparison

Table 3: Deterministic Evaluation Results

Algorithm	Mean Reward	Unique Actions	Top Action %	State-Dependent
PPO (Flat)	-20.0	1	100.0%	No
PPO (Hierarchical)	-291.4	1	100.0%	No
LSTM-PPO	-1115.8	1	100.0%	No
DQN	**-228.1**	**6**	**56.7%**	**Yes**

4.2 Stochastic vs Deterministic Performance

Table 4: PPO Hierarchical: Stochastic vs Deterministic

Metric	Deterministic	Stochastic
Mean Reward	-291.4	-163.5
Unique Actions/Episode	1.0	26.8
Total Unique Actions	1	39
Most Common Action %	100.0%	26.1%

This reveals that PPO learns a **flat probability distribution** - stochastic sampling shows diversity, but deterministic $\arg\max$ collapses to a single action.

4.3 Action Type Distribution

Table 5: Action Type Usage (Deterministic Mode)

Algorithm	Monitor	Analyse	Remove	Restore
PPO Flat	0%	0%	0%	100%
PPO Hierarchical	0%	0%	0%	100%
LSTM-PPO	0%	100%	0%	0%
DQN	**2.0%**	**74.8%**	**7.9%**	**15.3%**

DQN is the only algorithm utilizing multiple action types, with a preference for Analyse operations.

4.4 State-Dependency Analysis

Table 6: Action Changes per 50-Step Episode

Algorithm	Action Changes	Classification
PPO Flat	0	Static
PPO Hierarchical	0	Static
LSTM-PPO	0	Static
DQN	**39**	**State-Dependent**

4.5 DQN Learned Policy

Analysis of Q-values reveals DQN learned meaningful preferences:

Table 7: DQN Q-Values by Action Type (Initial State)

Action Type	Mean Q	Max Q	Min Q
Monitor	-2.97	-2.77	-3.59
Analyse	-2.92	-2.79	-3.30
Remove	-2.95	-2.81	-3.30
Restore	-3.91	-3.60	-4.39

Key insight: DQN learned to **avoid Restore** (lowest Q-values) and prefer Analyse/Remove/Monitor.

4.5.1 Learned Defense Cycle

DQN learned a repeating defensive pattern:

1. **Step 0**: Monitor (reconnaissance)
2. **Steps 1-2**: Analyse Op_Server0 (check critical server)
3. **Step 3**: Remove Enterprise1 (clean compromised host)
4. **Steps 4+**: Alternate between Analyse and Remove

Table 8: DQN Top 6 Actions (Deterministic Evaluation)

Action	Count	Percentage
Analyse Op_Server0	567	56.7%
Analyse Op_Host0	156	15.6%
Restore Op_Server0	153	15.3%
Remove Enterprise1	79	7.9%
Analyse User0	25	2.5%
Monitor	20	2.0%

5 Discussion

5.1 Why Policy Gradient Methods Failed

PPO and LSTM-PPO learn a policy $\pi(a|s)$ that produces a probability distribution over actions. Our experiments show these methods converge to near-uniform distributions:

$$\pi(a|s) \approx \frac{1}{|A|} \quad \forall a \in A \tag{4}$$

When using deterministic inference (arg max), tiny numerical differences determine the selected action, resulting in the same action regardless of state.

Root causes:

1. High entropy coefficients encourage exploration but prevent commitment
2. Observations may lack sufficient discriminative information
3. Action space too large relative to observation signal

5.2 Why DQN Succeeded

DQN learns $Q(s, a)$ values directly, where each action's expected return is estimated independently. This architecture naturally produces **state-dependent** action selection:

$$a^* = \arg\max_a Q(s, a) \tag{5}$$

Key advantages:

1. Q-values differentiate actions based on state features
2. Replay buffer provides stable, decorrelated training
3. ϵ-greedy exploration is simpler than entropy regularization
4. No distribution collapse - each Q-value is learned independently

5.3 Comparison with Original CybORG Paper

Table 9: Comparison with Standen et al. (2021)

Aspect	Original Paper	This Work
Agent Type	Red (attacker)	Blue (defender)
Algorithm	DRQN (DQN + LSTM)	DQN, PPO, LSTM-PPO
Scenario	3-host penetration	Scenario 1b (13 hosts)
Action Space	8 actions	52 actions (hierarchical)
Training Steps	2,500 iterations	500,000 steps
Success Metric	System access	Reward maximization
Transfer Rate	66% (sim $\rightarrow$ emu)	N/A (simulation only)

The original CybORG paper demonstrated DQN with LSTM (DRQN) for Red team training. Our results confirm that value-based methods (DQN) are more effective than policy gradient methods (PPO) for this domain, extending the finding to Blue team defense.

5.4 Limitations

1. **Simulation Only**: No emulation validation performed
2. **Single Red Agent**: Only tested against B_lineAgent
3. **Fixed Scenario**: Scenario 1b-vulnerable configuration only
4. **Observation Quality**: FixedFlatWrapper may lose information

6 Conclusion

This study demonstrates that **DQN significantly outperforms PPO and LSTM-PPO** for training Blue team agents in CybORG. The key findings are:

1. Policy gradient methods suffer from **action collapse** - converging to single-action policies regardless of state
2. DQN learns **state-dependent policies** with 39 action changes per episode
3. DQN achieves the best reward (-228.1) with 6 unique actions used deterministically
4. The learned DQN policy implements a **cyclic defense strategy**: Monitor $\rightarrow$ Analyse $\rightarrow$ Remove

These results align with the original CybORG paper's use of DQN-based methods and suggest that value-based RL is more suitable for autonomous cyber defense than policy gradient approaches.

7 Future Work

1. **Emulation Validation**: Test trained DQN agent on AWS virtual infrastructure
2. **Multiple Red Agents**: Evaluate against diverse adversary strategies
3. **Observation Engineering**: Investigate alternative state representations

4. **Action Masking**: Only allow actions relevant to compromised hosts
5. **Multi-Agent Training**: Co-evolve Red and Blue agents
6. **Transfer Learning**: Apply trained agents to different scenarios

References

[1] Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian Marriott. *CybORG: A Gym for the Development of Autonomous Cyber Agents*. IJCAI-21 1st International Workshop on Adaptive Cyber Defense, 2021. arXiv:2108.09118.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. arXiv preprint arXiv:1707.06347, 2017.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. *Human-level control through deep reinforcement learning*. Nature, 518(7540):529–533, 2015.

[4] Matthew Hausknecht and Peter Stone. *Deep Recurrent Q-Learning for Partially Observable MDPs*. AAAI Fall Symposium Series, 2015.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. arXiv preprint arXiv:1606.01540, 2016.

A Hyperparameters

Table 10: Complete Hyperparameter Configuration

Parameter	**PPO**	**LSTM-PPO**	**DQN**
Learning Rate	3×10^{-4}	3×10^{-4}	1×10^{-4}
Batch Size	64	64	64
Discount (γ)	0.99	0.99	0.99
Network	[256, 256]	LSTM(128) + [128, 64]	[256, 256, 128]
Entropy Coef.	0.05	0.01	N/A
Exploration	Entropy	Entropy	ϵ-greedy
Replay Buffer	N/A	N/A	100,000
Training Steps	500,000	500,000	500,000

B Action Space Mapping

Table 11: Hierarchical Action Encoding

Action Type	**Type Index**	**Action Range**
Monitor	0	0–12
Analyse	1	13–25
Remove	2	26–38
Restore	3	39–51