CyCon 2026: Securing Tomorrow
18th International Conference on Cyber Conflict
C. Kwan, B. M. Åkesson, M. Cvetko,
A. Dollimore, M. Tolppa (Eds.)

Enhancing Cyber Attack Autonomy Through Multi-Agent Reinforcement Learning

Christoph R. Landolt
CISPA Helmholtz Center for Information Security
Saarbruecken, Germany
christoph.landolt@cispa.de

Christoph Würsch
Eastern Switzerland University of Applied Sciences
Rapperswil, Switzerland
christoph.wuersch@ost.ch

Valentin Mulder
Cyber-Defence Campus
Armasuisse
Thun, Switzerland
valentin.mulder@ar.admin.ch

Roland Meier
Cyber-Defence Campus
Armasuisse
Thun, Switzerland
roland.meier@ar.admin.ch

Julian Jang-Jaccard
Cyber-Defence Campus
Armasuisse
Thun, Switzerland
julian.jang-jaccard@ar.admin.ch

Mario Fritz
CISPA Helmholtz Center for Information Security
Saarbruecken, Germany
fritz@cispa.de

Abstract: Advances in artificial intelligence and machine learning are transforming offensive cybersecurity, enabling the creation of autonomous intelligent cyber agents (AICAs) for offensive operations that overcome the limitations of traditional, static red-teaming playbooks. This work leverages deep multi-agent reinforcement learning (MARL) to implement autonomous cyber agents capable of coordinating sophisticated distributed attacks in dynamic environments. Using the network attack simulation (NASim) environment as a testbed, this paper highlights the limitations of single-agent approaches and demonstrates MARL's potential to enable decentralized, collaborative strategies. Multi-agent systems were implemented for collaborative red team exercises, such as lateral movement and capture the flag. The results indicate that MARL-based agents successfully learn coordinated attack patterns in small-scale environments, which improves stealth and adaptation. However, while MARL demonstrates potential for decentralized collaboration, our evaluation reveals

significant scalability challenges as network complexity increases. These findings underscore the importance of AICA in advancing offensive capabilities while identifying the stabilization of multi-agent training in large-scale topologies as a primary bottleneck for future research.

Keywords: *agentic systems in cybersecurity, adaptive strategy learning, autonomous intelligent cyber agents (AICAs), multi-agent reinforcement learning (MARL), offensive cyber operations*

1. INTRODUCTION

The digital domain is now a central battleground in modern warfare, demanding enhanced cyber resilience as threats evolve rapidly. Conventional, reactive defenses struggle against sophisticated, dynamic attacks. Initiatives like the Defense Advanced Research Projects Agency's Cyber Grand Challenge demonstrated autonomous defense systems such as Mayhem, Xandra, and Mechanical Phish, which could identify and patch known vulnerabilities at machine speed [1]. However, these systems often fail to adapt to new or unforeseen threats [2]. Validating robust defenses requires equally sophisticated and adaptive offensive agents.

Traditional intrusion detection systems (IDSs) use static signature-based and dynamic anomaly-based methods, which represent the primary defensive barrier against offensive operations. These systems often struggle to detect sophisticated distributed attacks that mimic benign traffic [3]. For an offensive agent, the IDS defines the environmental constraints and the cost of detection. Reinforcement learning (RL) offers a promising framework for developing dynamic attack and defense systems, enabling agents to learn through trial and error and generate complex attack graphs [4], [5].

Single-agent approaches often struggle with the dynamic and distributed nature of real-world scenarios. Multi-agent reinforcement learning (MARL) provides a solution by enabling collaborative interactions and realistic simulations of attack teams [6]. While MARL has been explored for defense [7], [8], its application in coordinating offensive agents for tasks such as lateral movement remains underexplored [9], [10]. This paper addresses this gap by designing and implementing autonomous intelligent cyber agents (AICAs) for offensive red teaming using MARL [9]. A red team is

an authorized group of security professionals or autonomous agents that simulate adversarial behaviors to identify and exploit vulnerabilities. Our research focuses specifically on how multi-agent collaboration in lateral movement can fragment attack signatures, posing a greater challenge to modern IDSs.

Our main contributions are:

- A comprehensive review of the state of the art in MARL and Cyber Gyms for offensive operations (Section 3).
- A comparison of different neural network policies for offensive cyber agents, specifically evaluating attention-based, recurrent (gated recurrent unit [GRU]), and traditional multilayer perceptron (MLP) architectures (Sections 4.A and 4.B.3).
- A systematic evaluation of MARL scalability across various network topologies, identifying the scalability limits of multi-agent versus single-agent approaches (Section 4.A.4).

This paper is organized as follows: Section 2 summarizes related work. Section 3 details the proposed attack framework and the simulation environment. Section 4 describes the experimental setup and presents the evaluation results. These findings are analyzed in Section 5. Finally, Section 6 provides concluding remarks and outlines directions for future work.

2. BACKGROUND AND RELATED WORK

In this section, we briefly introduce the background to the work described in this paper.

A. Adversarial Machine Learning for Cyber Operations

Machine learning (ML) has dual-use potential in cybersecurity, benefiting both defenders (blue teams) and attackers (red teams) by automating operations across various stages of a cyber kill chain [11]. Offensive autonomous cyber agents capable of sensing, reasoning, and acting without continuous human control represent the next evolution in this domain. While methods exist for automating specific phases, such as reconnaissance or crafting adversarial examples (e.g., IDSGAN [12]), end-to-end autonomy remains challenging due to the complexity of orchestrating multi-stage [13] and distributed attacks [10].

B. Reinforcement Learning for Cybersecurity

RL provides a framework for automating complex sequential attack chains by framing an offensive operation as a Markov decision process (MDP) or partially observable Markov decision process (POMDP). Traditional methods often struggle with the high-dimensional observation and action spaces of real enterprise networks, a challenge known as the *curse of dimensionality* [14]; as the number of hosts, open ports, and potential vulnerabilities grows, the state-action space becomes too vast for exhaustive search.

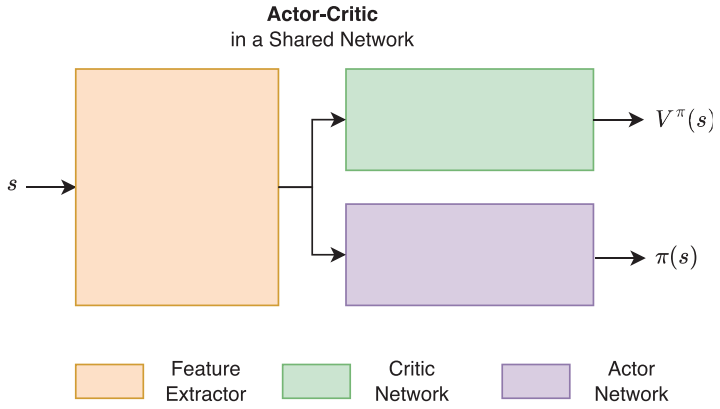
To handle the high-dimensional action spaces inherent in cyber operations (e.g., selecting specific exploits across multiple subnets), this work utilizes deep reinforcement learning (DRL) with policy gradient methods. We especially employ the proximal policy optimization (PPO) algorithm [15] due to its low variance, high sample efficiency, and stable updates compared to baseline methods like REINFORCE (see Table I for a comparison).

TABLE I: COMPARISON OF POLICY GRADIENT ALGORITHMS

Algorithm	Mechanism	Pros and Cons
REINFORCE (Monte Carlo)	Updates policy using the total return G_t at the end of a complete episode.	**Pros:** unbiased gradient estimates. **Cons:** high variance; low sample efficiency (must wait for episode end).
Actor-Critic / PPO (temporal difference)	Uses a Critic (V function) to estimate returns at every step (bootstrapping).	**Pros:** low variance; high sample efficiency; stable updates via clipping. **Cons:** introduces bias due to approximation.

As illustrated in Figure 1, we use a shared network structure within the PPO Actor-Critic architecture. This utilizes a shared encoder backbone to extract features from the observation O and the actor's network state. These features are then passed to separate "heads" for the actor and critic, allowing the critic to shape the actor's representation of network topology and connectivity, facilitating faster convergence in complex environments.

FIGURE 1: ACTOR-CRITIC SHARED ARCHITECTURE: A COMMON FEATURE EXTRACTOR WITH SEPARATE HEADS FOR V AND π



C. Multi-Agent Reinforcement Learning for Cybersecurity

Multi-agent solutions are necessary to capture the complexity and scale of enterprise-scale offensive cyber operations, where agents can specialize in different areas of the attack chain. MARL extends traditional RL by modeling interactions between multiple agents in dynamic, game-theoretic environments where training objectives are typically formulated as one of three setups:

1. **Cooperative learning:** Agents collaborate to maximize a shared global reward, aligning their strategies toward a common objective. This setup encourages joint performance and often converges to Pareto-efficient strategies. A typical application is distributed intrusion detection [16], [17] or coordination of a red team.

2. **Competitive learning:** Agents compete in a zero-sum game where one agent's gain is the other's loss. In cybersecurity, these models are attacker–defender scenarios in which the attacker maximizes a payoff while the defender minimizes it. Training aims to reach a Nash equilibrium where neither side can unilaterally improve its outcome [16].

3. **Mixed-interest learning:** This framework combines cooperative and competitive elements, involving agents with partially aligned and partially conflicting objectives. For example, a team of attackers may cooperate to maximize a collective reward while simultaneously competing against a team of distributed defenders. This setup can be applied in large-scale cyber warfare simulations [18].

For our work, cooperative learning is most relevant for coordinating a red team. The MARL algorithms evaluated in this study include independent learners (independent proximal policy optimization [IPPO], independent soft actor-critic [ISAC]) and centralized critic architectures for training stability (multi-agent proximal policy optimization [MAPPO], multi-agent soft actor-critic [MASAC]), which facilitate cooperation while maintaining decentralized execution. Table II summarizes the MARL algorithms used in this paper.

3. GAME DESIGN AND CYBER GYMS

This section discusses the definition of a MARL training environment for autonomous offensive cyber agents and the process of training these agents.

A. Multi-Agent Execution Paradigms

To allow multiple agents to interact with a single environment, mechanisms must be defined for triggering agents and resolving conflicts. Race conditions can occur when actions are executed in parallel on a shared object [19]. Real-world time is modeled as a sequence of discrete time steps t, where each step corresponds to a single decision. This simplifies complex, asynchronous network interactions into a sequence of state transitions $s_t \rightarrow s_{t+1}$ that an agent can process. To ensure consistent state transitions, two paradigms from the PettingZoo framework can be used [19]:

- **Parallel cycle:** This scenario, visualized in Figure 2, illustrates a case where all agents act simultaneously. Since multiple agents may modify the same resource attribute, the environment requires predefined rules known as tie-breaking mechanisms to determine which action takes priority [19].

  In this scenario, the transition from state s_t to s_{t+1} depends on n agents and their corresponding n actions. As a result, assigning a unique individual reward to a single agent is often not feasible. Instead, all agents typically receive the same reward, regardless of whether an individual agent's action was optimal or counterproductive. This reward structure is well-suited for collaborative tasks but poses challenges in competitive or mixed-interest scenarios.

TABLE II: MARL ALGORITHMS, GAMES, AND TRAINING FRAMEWORKS

Name / Type of Game	Algorithm Description	Key Characteristics
IPPO Independent stochastic games	IPPO adapts the PPO algorithm [15] for independent agents who maximize individual rewards based on local observations, treating them as self-interested entities [20].	• Independent policy optimization for each agent • Often results in non-cooperative dynamics and limited coordination • Simple but risks inefficiency in collaborative scenarios [24]
ISAC Independent stochastic games	ISAC applies the off-policy soft actor-critic [SAC] algorithm [21] to multi-agent settings. Agents independently maximize a trade-off between expected return and entropy using only local observations [22].	• Off-policy learning improves sample efficiency compared to on-policy methods • Entropy maximization encourages robust exploration • Fully decentralized training and execution (no global state usage)
MAPPO Cooperative games	MAPPO builds on PPO [15] with a centralized critic to support cooperation among agents during training, while execution remains decentralized [23].	• Centralized critic during training for global value sharing • Balances cooperation during training and scalability in decentralized execution • Effective for cooperative multi-agent tasks
MASAC Cooperative games	MASAC extends SAC [21] into a centralized training, decentralized execution (CTDE) framework. It utilizes a centralized critic that conditions on the global state to stabilize learning [22].	• Centralized critic mitigates non-stationarity in multi-agent environments • Supports continuous and discrete action spaces • Combines high sample efficiency (off-policy) with stability

- **AEC cycle:** The agent environment cycle (AEC) is the second training paradigm, visualized in Figure 3. In this scenario, agents execute actions sequentially within the environment, enabling the assignment of individual rewards to each action. While this resolves reward attribution issues, it can significantly increase simulation time, especially in environments with many agents. This is because observations, actions, and rewards must be processed sequentially rather than in parallel, making efficient GPU-based simulation more challenging.

FIGURE 2: PARALLEL CYCLE: ALL AGENTS PERFORM THEIR ACTIONS IN PARALLEL, AND ALL RECEIVE THE SAME REWARD

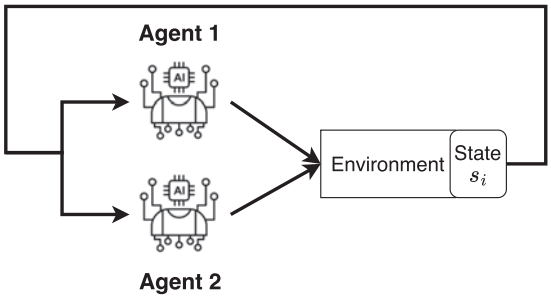
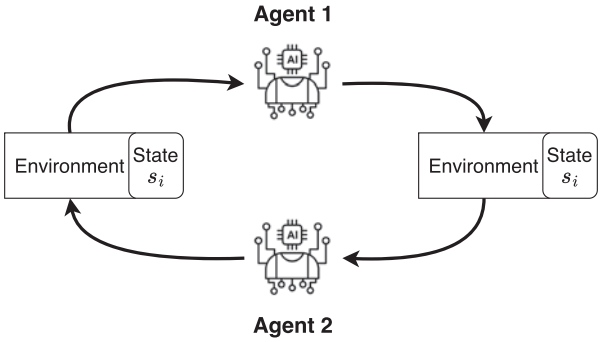


FIGURE 3: AGENT ENVIRONMENT CYCLE: THE AGENTS PERFORM THEIR ACTIONS SEQUENTIALLY AND RECEIVE AN INDIVIDUAL REWARD

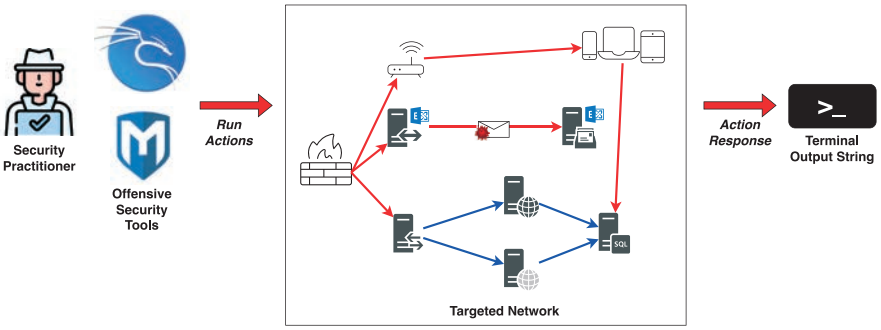


Mathematical Formalization of Spaces

A critical challenge in designing a CyberGym is bridging the gap between real-world cyber operations and the mathematical constraints of RL, or the so-called sim-to-real gap.

In the context of offensive cyber operations, a security practitioner utilizes a wide range of tools, such as Nmap, OpenVAS, Wireshark, or the Metasploit Framework. As illustrated in Figure 4, these tools serve as the interface to the network, allowing the attacker to gather information, scan for vulnerabilities, and exploit them. The results of these actions are typically returned as unstructured terminal strings, which may contain lists of active nodes, open ports, or vulnerability reports.

FIGURE 4: VISUALIZATION OF THE MODELED PROBLEM: THE SECURITY PRACTITIONER USES DIFFERENT SOFTWARE AS AN ACTION INTERFACE FOR THE ATTACKED NETWORK, WITH RESULTS DISPLAYED AS TERMINAL STRINGS



To train autonomous agents effectively, these unstructured interactions must be encoded into structured mathematical sets. We address this using the Gymnasium Framework [25], a standard in the RL community, and its multi-agent extension, PettingZoo [19].

In this framework, the environment is defined by two key attributes: the *action space*, which defines all valid operations, and the *observation space*, which encodes environment feedback. These are constructed using *fundamental spaces* for basic sets and *composite spaces* for complex structures:

- Discrete: Represents a finite set of n mutually exclusive options, suitable for selecting specific tools or targets:

$$S_{\text{Discrete}} = \{0, 1, \ldots, n-1\}, \quad n \in \mathbb{N}^{+} \tag{1}$$

- Box: Represents a d-dimensional space of continuous values within defined bounds:

$$S_{\text{Box}} = \{x \in \mathbb{R}^{d} \mid x_{\min} \leq x \leq x_{\max}\} \tag{2}$$

- MultiBinary: Models a vector of d independent binary flags:

$$S_{\text{MultiBinary}} = \{0, 1\}^{d} \tag{3}$$

- MultiDiscrete: Extends the discrete space to multiple independent dimensions:

$$S_{\text{MultiDiscrete}} = S_{\text{Discrete},n1} \times \cdots \times S_{\text{Discrete,nd}} \quad (4)$$

- Text: Encodes variable-length strings, essential for processing raw terminal output:

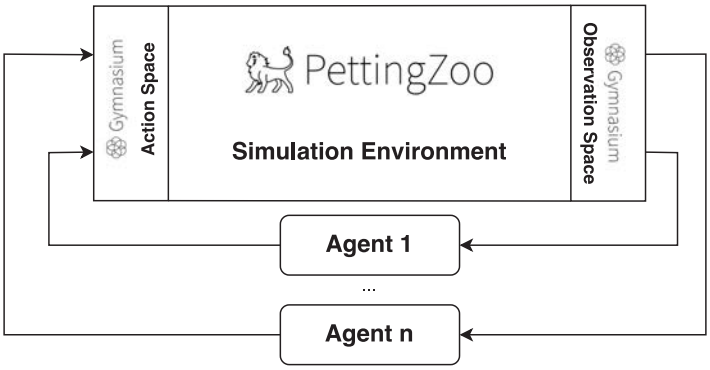
$$S_{\text{Text}} = \{s \in C^L \mid L_{\min} \leq |s| \leq L_{\max}\} \quad (5)$$

- Graph: A composite space representing network topologies via nodes N, edges E, and an adjacency matrix L:

$$S_{\text{Graph}} = (N, E, L) \quad (6)$$

As summarized in Figure 5, these definitions form the interface of the simulation. Each cyber agent interacts with the environment by selecting actions from the defined action space and receiving updates through the observation space.

FIGURE 5: THE ARCHITECTURE OF THE PETTINGZOO FRAMEWORK, WHERE AGENTS INTERACT WITH A SIMULATION ENVIRONMENT USING A GYMNASIUM-BASED ACTION SPACE AND RECEIVE OBSERVATIONS THROUGH A GYMNASIUM OBSERVATION SPACE



B. Overview of Cyber Simulation Environments

The development of autonomous cyber agents for offensive operations relies heavily on simulation environments, often called Cyber Gyms. Table III outlines some of the most important Cyber Gyms in the current landscape, comparing their primary focus, key characteristics, and supported agent architectures, including single-agent reinforcement learning (SARL) and MARL.

TABLE III: COMPARISON OF CYBER SIMULATION ENVIRONMENTS

Environment	Primary Focus	Agent Support	Key Strengths	Limitations
CyberBattleSim (Seifert et al. [26])	Lateral Movement	SARL (ext. MARL)	Simple integration of SARL; effective for basic adversarial actions.	Abstracted simulations lack real-world applicability; limited scenario complexity.
CybORG++ (Emerson et al. [27])	Red vs. Blue (Adversarial)	MARL	Flexible adversarial modeling; used in cyber autonomy gym for experimentation (CAGE) challenges.	Scalability issues in large networks; significant simulation-to-reality gap.
VINE (Eskridge et al. [28])	Moving Target Defense (MTD)	Exp. Platform	Scalable; supports dynamic background traffic and detailed instrumentation.	Primarily focused on experimentation rather than agent training benchmarks.
NASimEmu (Janisch et al. [29])	Realistic Offensive Operations	SARL	High realism via emulation; integrates real-world tools; supports generalization.	Resource-intensive setup; emulation maintenance is costly; primarily restricted to SARL.

C. The NASim Implementation

The simulation environment used in this work builds upon the single-agent environment NASim [30], which was improved for real-world applications with an emulation component in the NASimEmu environment [29]. It enables agents to learn optimal strategies for automated offensive operations in complex network structures by exploring and exploiting network vulnerabilities under uncertainty and partial observability.

1) Action Space Encoding

In the NASim environment, the action space consists of various cyber operations an agent can perform to interact with the network. These actions are categorized into scanning, exploitation, and privilege escalation.

- **Scanning actions** collect information about hosts (e.g., identifying open services, determining the operating system, enumerating network subnets, and detecting active processes).
- **Exploit actions** attempt to utilize vulnerabilities in identified services to gain access to hosts.
- **Privilege escalation actions** allow the agent to increase access rights on a compromised host, thereby gaining administrative control.

Action masking is used to filter out illegal actions automatically. This ensures that the agent selects only logically possible actions (e.g., preventing the exploitation of a service that has not yet been discovered).

2) Reward Function

The reward function in the NASim environment [30] is designed to incentivize efficient attack strategies while penalizing unnecessary actions. The total reward at time step t is given by:

$$R_t = V_h + V_d - C_a \tag{7}$$

where:

- V_h = Value of the compromised host. Sensitive hosts yield higher rewards.
- V_d = Discovery reward for identifying a new host.
- C_a = Cost of performing an action (negative reward).

For the scenarios examined in this paper, the specific reward values and action costs are defined in Table IV.

TABLE IV: REWARDS AND ACTION COSTS CONFIGURATION

Rewards		**Action Costs**	
Type	Value	Action Type	Value
Host reward (V_h) Host discovery reward (V_d)	10 1	Exploit cost Privilege escalation cost Service scan cost Operating system scan cost Subnet scan cost Process scan cost	1 1 1 1 1 1

The episode terminates when the agent either successfully compromises all hosts or reaches the maximum step limit. This reward function balances exploration (scanning) and exploitation (attacking) to encourage efficient, optimal attack strategies.

4. EXPERIMENTS

In this section, we first conduct a comprehensive study of different policy-network implementations for SARL setups, then train and benchmark MARL agents across various baseline scenarios.

A. Single-Agent Policy Evaluation

To evaluate SARL for complex, sequential tasks, policies trained via PPO were tested in a fixed, challenging network with a non-trivial lateral movement task under partial observability.

1) Hyperparameter Tuning

We identified entropy regularization and latent-space dimensionality as the critical hyperparameters governing agent performance [31].

Using a baseline MLP with approximately 3.4 million parameters, we observed that fixed entropy coefficients led to either early convergence or excessive variance. We successfully addressed this by implementing an entropy scheduler during training that linearly decays αH, transitioning the agent from exploration to exploitation.

Additionally, we compared embedding dimensions for policy regularization: 128 versus 64. Results indicated that the smaller 64-dimensional space facilitated better generalization by forcing the network to learn essential features, whereas larger embeddings led to overfitting.

2) Architecture Comparison

We evaluated three architectures to handle sequential dependencies: feed-forward (MLP), GRU, and self-attention. The choice of these architectures represents a systematic evaluation of how an agent maintains an internal representation of the environment, a concept central to both the World Model framework [32] and the NATO AICA reference architecture [33]. Our specialized architecture comprises three distinct elements that correspond directly to these theoretical functional components:

- **Encoder/sensing:** The vision model in a World Model encodes high-dimensional observations into low-dimensional latent vectors [32]. This aligns with the AICA sensing function, which captures, formats, and normalizes external and internal world state descriptors [33].

- **Memory module / world state identification:** The memory module integrates historical observations to create a representation that approximates the true state of the environment and can predict future states [32]. Similarly,

the AICA world state identification function compares current observations with learned patterns in a world knowledge database to reconstruct the current world state [33].

- **Controller / action selector:** The controller uses representations from both the vision and memory components to select good actions [32], which mirrors the AICA action selector, whose task it is to produce an executable response plan [33].

The MLP baseline combines sensing and action selection but lacks the memory component required to store a representation of past states [32]. In contrast, the GRU and self-attention policies represent two distinct choices for the memory component [34], enabling the agent to leverage a predictive model of the future to improve its decision-making in complex, stochastic network environments [32], [33].

The training reward trajectories for the three architectures are visualized in Figure 6. While the MLP and GRU baselines plateau at a suboptimal reward level, the self-attention model demonstrates a distinct phase transition (grokking) between 45k and 50k steps, rapidly converging to the maximum possible reward.

Table V presents a statistical comparison of the learned policies' action distributions.

TABLE V: COMPARISON OF POLICY METRICS

Metric	MLP	GRU	Self-Attention
Entropy	2.33	2.70	**2.96**
Kurtosis	8.54	6.16	**5.17**

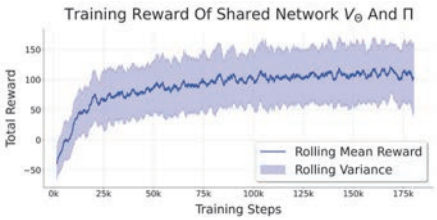
The self-attention policy exhibits the highest entropy (2.96), paired with the lowest kurtosis (5.17). This statistical profile indicates a policy that maintains a broad, adaptive exploration of the action space, avoiding the "peaked" action selection distribution observed in the MLP (kurtosis: 8.54).

The superior performance of the attention mechanism suggests that an optimal AICA for offensive operations requires a specialized architecture comprising three distinct elements: (1) an **encoder** for representation learning, (2) an **attention-based memory module** to prioritize local information [35], and (3) a **controller** for precise action selection. However, it is important to note that the attention policy showed the highest

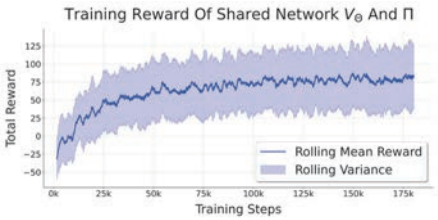
hyperparameter sensitivity, requiring significantly more tuning effort to achieve convergence than the MLP baseline.

FIGURE 6: TRAINING REWARD COMPARISON ACROSS ARCHITECTURES

(a) MLP Policy



(b) GRU Policy



(c) Self-Attention Policy

Note: The MLP (a) and GRU (b) policies converge early to suboptimal solutions. In contrast, the self-attention policy (c) experiences a "grokking" phase after 45k steps, overcoming the local minimum to solve the environment.

B. Multi-Agent Evaluation

To evaluate the efficacy of MARL in coordinated cyber operations, we benchmarked our agents in a series of generated network scenarios. These experiments were designed to test the agents' ability to collaborate, share information, and execute synchronized attacks under varying levels of complexity.

1) Experimental Setup and Scenario Generation

The training environments were generated using the NASim benchmark generator, which creates realistic network topologies based on the methodologies of Sarraute et al. [36] and Speicher et al. [37]. The generated networks are segmented into functional zones (e.g., demilitarized zone (DMZ), sensitive, and user) separated by stochastic firewall rules that ensure at least one viable attack path to the target exists. To emulate enterprise realism, host configurations are not uniformly random; instead, the NASim benchmark generator uses a nested Dirichlet process to introduce correlations in software stacks across the network. Furthermore, exploit success probabilities are modeled using a "mixed" distribution based on Common

Vulnerability Scoring System (CVSS) attack complexity, ranging from 0.3 (high complexity) to 0.9 (low complexity).

To evaluate agent performance across varying degrees of difficulty, we employed a standardized suite of benchmark scenarios ranging from tiny (3 hosts) to large (23 hosts). The detailed specifications for each environment are provided in Table VI.

TABLE VI: BENCHMARK SCENARIO SPECIFICATIONS

Name	Type	Subnets	Hosts	OS	Services	Processes	Exploits	Privilege Escalation Actions	Actions	Observation Space Dimensions	States	Step Limit
Tiny	Static	4	3	1	1	1	1	1	18	4 × 14	576	1,000
Small	Static	5	8	2	3	2	3	2	72	9 × 32	4,576	1,000
Medium	Static	6	16	2	5	3	5	3	192	17 × 27	3.9×10^5	2,000
Large	Generated	8	23	3	7	3	7	3	322	24 × 32	4.5×10^6	5,000

These predefined benchmark scenarios provide standardized simulations of networks with increasing complexity. For example, the tiny scenario simulates three hosts distributed across three internal subnets reachable from an internet-facing entry subnet. All hosts run Linux with a secure shell protocol (SSH) service and a Tomcat process, enabling exploitation via an SSH vulnerability followed by privilege escalation through Tomcat. Two hosts are designated as sensitive targets and yield a reward value of 100 upon compromise.

The small and medium scenarios increase complexity by introducing additional hosts, heterogeneous operating systems (Linux and Windows), multiple services (e.g., SSH, file transfer protocol [FTP], and hypertext transfer protocol [HTTP]), and additional privilege escalation paths. Firewall rules between subnets restrict the services reachable from one subnet to another, requiring agents to perform reconnaissance actions such as subnet scans before executing lateral movement.

Through these increasingly complex scenarios, agents perform common penetration testing tasks such as service discovery, exploitation, privilege escalation, and multi-stage lateral movement.

2) Algorithm Selection

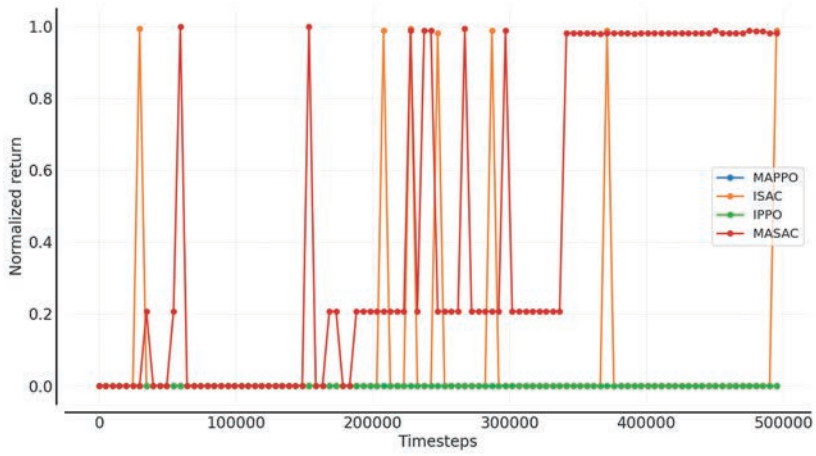
To determine the optimal learning strategy, we benchmarked the four candidate algorithms listed in Table II within the tiny environment. This evaluation was

conducted using BenchMARL [22], a training library created to enable standardized benchmarking across different algorithms, models, and environments [22]. By leveraging the BenchMARL framework and its high-performance TorchRL backend, we ensure our results are situated within a reproducible framework that facilitates fair and systematic comparison. The objective was to evaluate convergence speed and stability before scaling to larger network topologies.

The evaluation results, shown in Figure 7, reveal a substantial distinction in performance. The on-policy PPO variants (MAPPO and IPPO) failed to learn a successful policy within the allocated budget, likely due to the sample inefficiency inherent to on-policy learning in sparse-reward stochastic environments. While the independent off-policy learner, ISAC, achieved sporadic successes, it exhibited significant variance and failed to stabilize, resulting in inconsistent performance during evaluation.

In contrast, MASAC significantly outperformed all other algorithms. It demonstrated the fastest convergence and, critically, the ability to maintain a stable policy. We attribute this superior performance to three factors: (1) *sample efficiency* from off-policy updates, (2) *robust exploration* driven by the maximum entropy objective, and (3) *the CTDE framework*, which effectively stabilizes the non-stationarity caused by simultaneous agent updates. Consequently, MASAC was selected as the primary algorithm for all subsequent large-scale experiments.

FIGURE 7: BENCHMARK OF MARL ALGORITHMS IN THE TINY ENVIRONMENT



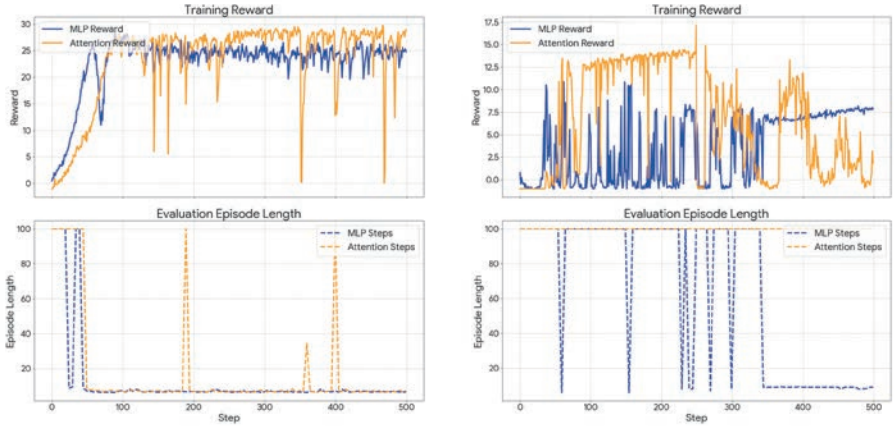
Note: The plot shows the normalized return over 500,000 timesteps in the evaluation. PPO-based methods (MAPPO, IPPO) fail to learn a successful policy. While ISAC shows sporadic success, MASAC demonstrates the fastest convergence and highest stability, eventually achieving consistent maximum returns.

3) Policy-Network-Selection

Following the selection of MASAC (Section 4.B.2), we evaluated the optimal policy architecture by comparing a standard MLP against an attention-based network. The results are presented in Figure 8. In the SARL baseline (Figure 8a), both architectures demonstrated rapid learning. However, the attention network showed greater instability and hyperparameter sensitivity, and yet achieved a slightly higher training reward.

This difference was amplified in the multi-agent MASAC setting (Figure 8b). Training duration increased significantly due to the complexity of learning three policies and a centralized Q-function simultaneously. In this environment, the attention policy suffered from performance collapse and failed to solve the evaluation task. The MLP policy established a robust equilibrium with minimal variance. Ultimately, the MLP-MASAC configuration solved the task in 8.82 ± 0.38 steps. The slight performance gap compared to the single-agent baseline (6.78 ± 0.36 steps) is attributed to the non-stationarity inherent in multi-agent coordination. Given its superior stability and convergence reliability, the MLP architecture was selected for all subsequent experiments.

FIGURE 8: COMPARISON OF THE MLP AND ATTENTION ARCHITECTURES



(a) Single PPO agent policy evaluation for a tiny environment

(b) Multi-agent (3 agents) MASAC policy evaluation for a tiny environment

Note: While the attention network shows initial promise, it exhibits high instability in the multi-agent setting. The MLP architecture demonstrates robust convergence and was selected for large-scale experiments.

4) Scalability and Performance

Building on the selection of the MASAC algorithm and the MLP policy architecture,

we evaluated the agents across increasingly complex network topologies. Table VII presents the quantitative results across the benchmark scenarios introduced in Table VI. While the SARL baseline maintains robust performance up to the medium scenario, the MARL agents (MASAC) exhibit a sharp performance degradation beyond the tiny environment. In the small and medium scenarios, the multi-agent system failed to reach a stable reward plateau.

TABLE VII: COMPARISON OF MEAN TOTAL REWARD (± STD DEV) ACROSS NASIM BENCHMARK SCENARIOS

Scenario	SARL PPO		MASAC (3 Agents)		
	Reward	Steps	Reward	Steps	Status
Tiny	193.23 ± 0.68	6.78 ± 0.36	58.51 ± 0.05	8.82 ± 0.38	Converged
Small	184.83 ± 1.41	8.48 ± 0.19	−100 ± 0	100 ± 0	DNC
Medium	181.87 ± 2.64	9.15 ± 0.47	−100 ± 0	100 ± 0	DNC
Large (Gen)	−89.93 ± 1.25	100 ± 0	−100 ± 0	100 ± 0	DNC

Note: Scores are averaged over the last 100 evaluation runs in the training. DNC – did not converge within the training budget.

5. DISCUSSION AND LIMITATIONS

Our investigation of SARL in cybersecurity provides insight into the architectural components required for effective sequential decision-making in computer networks. Across different network sizes and network topologies, successful SARL agents relied on three elements: an *encoder* for efficient representation learning, a *memory module* to combine current local observations with the current state of the offensive operation encoded in the history of the agent, and a *controller* for action selection. Crucially, we found that balancing local and global information helps agents generalize, whereas prioritizing local information prevents overfitting to specific network topologies.

This finding underscores the need for MARL approaches, which naturally leverage local information, as each agent operates with a partial view of the environment. This aligns with our observation that balancing local and global information is central to training sophisticated offensive agents. However, our evaluation of MARL algorithms highlights two critical architectural requirements: First, a *centralized critic* proved essential for stable training; and second, in parallel environments, the *memory module* reduced agents' performance due to the non-stationarity as agents conditioned their

behavior on historical observations that no longer reflected the current state of the environment and policies of their peers.

Our investigation into MARL highlights both its promise for coordinated autonomy and its current limitations in scaling to realistic cyber environments.

A. Scalability and Convergence Challenges

While coordinated behavior emerged reliably in small scenarios, the agents failed to converge in an equilibrium as network size increased. These results indicate that, under current algorithmic assumptions, the benefits of MARL do not scale proportionally with environmental complexity.

The most noticeable limitation observed in this study is the scalability gap between SARL and MARL approaches. While single-agent policies reliably solved up to the medium environment with 16 hosts and 6 subnets, MARL agents failed to generalize beyond the tiny scenario.

We identify three primary factors underlying this limitation. First, the joint action space grows in combination with the number of agents, making exploration more difficult in sparse-reward settings. Second, non-stationarity introduced by parallel interacting agents within the environment destabilizes training dynamics. Third, credit assignment becomes increasingly ambiguous as agents' parallel actions interact with shared network resources, weakening the learning signal available to individual policies.

These challenges are well-documented in current MARL research and are not unique to cybersecurity [24]. In cybersecurity environments, where rewards are inherently sparse (e.g., successful privilege escalation occurring only after extended reconnaissance), coordinating multiple independent actors presents a severe credit-assignment challenge. Although scalability remains a central open problem, the successful coordination observed in tiny scenarios nevertheless serves as a proof-of-concept for coordinated AICAs for offensive operations. Achieving industrial-scale deployment will require further advances in MARL algorithms and training paradigms.

B. Challenges and Open Questions

Despite scalability, several challenges remain that hinder immediate deployment in the real world.

- **The sim-to-real gap:** Existing training environments, so-called CyberGyms, are abstractions of real computer networks. They show a

significant gap regarding the encoding of states and actions, which are currently incompatible with real-world offensive security tools. These abstract representations do not reflect the true complexity of offensive cyber operations, where observations are encoded as terminal output, and actions often require creating custom payloads or writing customized exploits. Therefore, a transition from highly abstract CyberGyms toward RL approaches within industry-standard capture the flag infrastructures, such as Hack the Box, is required to achieve greater alignment with real-world cyber operations. Furthermore, these environments are often implemented in native Python, which creates central processing unit (CPU) bottlenecks that restrict parallelization and compute efficiency.

- **Adversarial instability:** As identified in our experiments on equilibrium learning, the interplay between ML and game theory is inherently affected by instabilities. Adversarial policies can exploit these instabilities and sensitive equilibrium points, making it difficult for agents to interact optimally against out-of-distribution and adversarial strategies. The question remains of how to stabilize training to ensure that agents stay robust when encountering network configurations not present during training or when facing malicious adversarial behavior.

- **State representation:** Our experiments with latent-space dimensions demonstrated that while feature extraction is vital to prevent overfitting, traditional vector-based representations may be fundamentally insufficient for complex network environments. A flat vector cannot easily encode hierarchical relationships in a modern enterprise network, such as those among subnets, hosts, and active processes. A transition toward graph neural networks and graph-based observation spaces is likely better suited to capture the structural topologies inherent in these environments.

6. CONCLUSION

In this work, we examined RL's capabilities for training AICAs for offensive operations. We established that while single-agent approaches can function robustly in medium-sized networks when using highly tuned hyperparameters, such as entropy regularization for exploration and precise latent-space dimensions to prevent overfitting, they lack the inherent coordination potential of multi-agent systems.

Our research demonstrates that MARL can successfully learn coordinated attack patterns, although it currently faces significant scalability barriers in larger network

topologies. This performance drop-off highlights a critical trade-off between the depth of individual agent capability and the complexity of group coordination.

Future work should focus on bridging the sim-to-real gap by developing modular, scalable simulation environments that support standard MARL frameworks such as PettingZoo or MeltingPot. This step is essential for creating robust agents that generalize across complex, real-world scenarios. By addressing these challenges, MARL can evolve from a theoretical capability into a powerful and adaptive tool for automated offensive cybersecurity operations.

ACKNOWLEDGMENTS

This research is supported by Armasuisse Science and Technology, and partially supported by the German Federal Ministry of Education and Research (BMBF) under grant AIgenCY (16KIS2012).

REFERENCES

[1] DARPA, "Cyber grand challenge," 2016. [Online]. Available: https://www.darpa.mil/research/programs/cyber-grand-challenge

[2] D. Brumley, "The future of cyber-autonomy," in *Enigma 2018*, Santa Clara, CA, USA, 2018. [Online]. Available: https://www.usenix.org/node/208122

[3] Federal Office for Information Security, "Orientation guide to using intrusion detection systems (IDS)," Bonn, Germany, 2022. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KRITIS/oh_sza_en.pdf

[4] Y. Guo, "A review of machine learning-based zero-day attack detection: Challenges and future directions," *Computer Communications*, vol. 198, pp. 175–185, 2023, doi: 10.1016/j.comcom.2022.11.001.

[5] Z. Hu, P. Chen, M. Zhu, and P. Liu, "Reinforcement learning for adaptive cyber defense against zero-day attacks," in *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense*, S. Jajodia et al., Eds., Cham: Springer, 2019, pp. 54–93, doi: 10.1007/978-3-030-30719-6_4.

[6] T. Kunz, C. Fisher, J. La Novara-Gsell, C. Nguyen, and L. Li, "A multiagent CyberBattleSim for RL cyber operation agents," in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2022, pp. 897–903, doi: 10.1109/CSCI58124.2022.00161.

[7] M. Kiely et al., "CAGE challenge 4: A scalable multi-agent reinforcement learning gym for autonomous cyber defence," *AI Magazine*, vol. 46, no. 3, art. no. e70021, 2025.

[8] F. Contractor, L. Li, and R. Al Mallah, "Learning to communicate in multi-agent reinforcement learning for autonomous cyber defence," 2025, *arXiv:2507.14658*.

[9] A. Kott et al., "Autonomous intelligent cyber-defense agent (AICA) reference architecture. Release 2.0," 2023, *arXiv:1803.10664*.

[10] C. R. Landolt, C. Würsch, R. Meier, A. Mermoud, and J. Jang-Jaccard, "Multi-agent reinforcement learning in cybersecurity: From fundamentals to applications," 2025, *arXiv:2505.19837*.

[11] E. Hutchins, M. Cloppert, and R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, pp. 80–106, Jan. 2011.

[12] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," in *Advances in Knowledge Discovery and Data Mining*, Cham: Springer International Publishing, 2022, pp. 79–91, doi: 10.1007/978-3-031-05981-0_7.

[13] B. Singer, K. Lucas, L. Adiga, M. Jain, L. Bauer, and V. Sekar, "Incalmo: An autonomous LLM-assisted system for red teaming multi-host networks," 2025, *arXiv:2501.16466*.
[14] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013, doi: 10.1177/0278364913495721.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[16] A. Tampuu et al., "Multiagent cooperation and competition with deep reinforcement learning," 2015, *arXiv:1511.08779*.
[17] F. Christianos, G. Papoudakis, and S. V. Albrecht, "Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning," 2022, arXiv:2209.14344v3.
[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, vol. 30, pp. 6379–6390, 2017.
[19] J. Terry et al., "PettingZoo: Gym for multi-agent reinforcement learning," Advances in Neural Information Processing Systems, vol. 34, pp. 15032–15043, 2021.
[20] C. Schroeder de Witt et al., "Is independent learning all you need in the StarCraft Multi-Agent Challenge?," 2020, *arXiv:2011.09533*.
[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*.
[22] M. Bettini, A. Prorok, and V. Moens, "BenchMARL: Benchmarking multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 217, pp. 1–10, 2024. [Online]. Available: http://jmlr.org/papers/v25/23-1612.html
[23] C. Yu et al., "The surprising effectiveness of PPO in cooperative, multi-agent games," 2022, *arXiv:2103.01955*.
[24] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-agent Reinforcement Learning: Foundations and Modern Approaches*, MIT Press, 2024.
[25] M. Towers et al., "Gymnasium: A standard interface for reinforcement learning environments," 2024, *arXiv:2407.17032*.
[26] C. Seifert, M. Betser, and W. Blum, "CyberBattleSim," GitHub, 2021. [Online]. Available: https://github.com/microsoft/cyberbattlesim
[27] H. Emerson, L. Bates, C. Hicks, and V. Mavroudis, "CybORG++: An enhanced gym for the development of autonomous cyber agents," 2024, *arXiv:2410.16324*.
[28] T. C. Eskridge, M. M. Carvalho, E. Stoner, T. Toggweiler, and A. Granados, "VINE: A cyber emulation environment for MTD experimentation," in *Proceedings of the Second ACM Workshop on Moving Target Defense (MTD '15)*, Denver, Colorado, USA, 2015, pp. 43–47, doi: 10.1145/2808475.2808486.
[29] J. Janisch, T. Pevný, and V. Lisý, "NASimEmu: Network attack simulator & emulator for training agents generalizing to novel scenarios," 2023, *arXiv:2305.17246*.
[30] J. Schwartz, "Network attack simulator: Installation guide," 2020. [Online]. Available: https://networkattacksimulator.readthedocs.io/en/latest/tutorials/installation.html
[31] T. Eimer, M. Lindauer, and R. Raileanu, "Hyperparameters in reinforcement learning and how to tune them," 2023, *arXiv:2306.01324*.
[32] D. Ha and J. Schmidhuber, "World models," 2018, *arXiv:1803.10122*.
[33] P. Theron et al., "Towards an active, autonomous and intelligent cyber defense of military systems: The NATO AICA reference architecture," in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, 2018, pp. 1–9.
[34] J. Janisch, "Applications of deep reinforcement learning in practical sequential information acquisition problems," Ph.D. dissertation, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic, 2024. [Online]. Available: https://dspace.cvut.cz/handle/10467/114377
[35] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013, doi: 10.1613/jair.3987.
[36] C. Sarraute, O. Buffet, and J. Hoffmann, "POMDPs make better hackers: Accounting for uncertainty in penetration testing," 2013, *arXiv:1307.8182*.
[37] P. Speicher, M. Steinmetz, J. Hoffmann, M. Backes, and R. Künemann, "Towards automated network mitigation analysis (extended)," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*, Limassol, Cyprus, Apr. 2019, pp. 1971–1978. doi: 10.1145/3297280.3297473.