Master's Thesis

Automated Cyber Defense Based on Reinforcement Learning Techniques

2310435 NGUYEN Thanh Cong

Supervisor	BEURAN Razvan
Main Examiner	BEURAN Razvan
Examiners	TAN Yasuo
	LIM Yuto
	UDA Satoshi

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)
August 2025

Abstract

In light of increasingly sophisticated and complex cybersecurity threats, developing autonomous cyber defense agents has become a critical and urgent area of research. Traditional human-based defense systems can no longer cope with the speed and scale of modern attacks. Recent studies show that intelligent agents using artificial intelligence can provide flexible defense capabilities by handling key tasks like monitoring, detection, and threat response. However, current defensive agents are trained in nearly perfect environments that do not mirror real-world conditions. Intrusion detection information can be imperfect and often includes specific error rates. In particular, AI-based anomaly detection models frequently misclassify normal user actions as anomalous.

This thesis focuses on developing cyber defense agents that operate in these imperfect environments through a multi-agent reinforcement learning (MARL) approach. Current research often concentrates on training cyber agents for only one side, either attack or defense, causing agents to focus too heavily on single strategies and restricting their adaptability. To address this limitation, we propose competitive training involving two adversarial agents trained simultaneously, allowing them to learn from and counteract each other in dynamic scenarios.

The research contributes a specialized simulation environment extending the Network Attack Simulator (NASim) with competitive agents and realistic IDS integration. We implement and compare Multi-Agent Proximal Policy Optimization (MAPPO) with centralized training and Independent Proximal Policy Optimization (IPPO) with decentralized training across clean baseline and operational noise scenarios simulating real-world sensor uncertainty.

Our evaluation employs exploitability metrics against worst-case adversaries using specialized cybersecurity metrics: True Block Rate (TBR), Hosts Compromised (HC), and Decoy Interaction Ratio (DIR). Testing encompasses simulation and realistic implementations using Snort IDS, iptables, Docker-based decoys, and Metasploit frameworks.

The results demonstrate significant defensive improvements over unprotected baselines. In clean conditions, MAPPO and IPPO achieved approximately 79% reduction in compromised hosts against random attackers (from 3.8 to 0.8 hosts) while maintaining perfect 100% true block rates. Against sophisticated trained adversaries, both approaches limited compromise to a single host, showing robust defensive capabilities. The deception strategy

proved effective with decoy interaction ratios reaching 26-40% against random attackers, successfully redirecting attack attempts from critical assets.

Important algorithmic differences emerged under operational noise conditions simulating realistic IDS false positives. When facing worst-case trained attackers, IPPO demonstrated superior robustness by limiting compromises to approximately two hosts compared to MAPPO's three hosts. IPPO maintained perfect detection accuracy (100% TBR) while MAPPO experienced slight degradation to 96.1% in realistic environments. These findings suggest that IPPO's decentralized training approach provides greater resilience to sensor uncertainty, particularly against sophisticated adversaries trained to exploit observation noise.

Critical validation came through successful simulation-to-real transfer, with performance metrics remaining within 5% between settings, confirming our simulation captures essential cyber defense dynamics. This enables practical training of defensive agents that maintain capabilities when deployed operationally, addressing the significant training-deployment gap challenge in cybersecurity applications.

The comparative analysis reveals that while both approaches perform similarly under ideal conditions, IPPO demonstrates greater robustness under uncertainty with superior stability against sophisticated attacks in noisy conditions. In contrast, MAPPO achieves better coordination through centralized training. These findings demonstrate that competitive training between adversarial agents produces robust defensive capabilities that transfer effectively to realistic environments, providing practical insights for deploying autonomous cyber defense systems that handle real-world complexities while maintaining operational effectiveness against evolving threats.

Keywords: autonomous agents, cyber defense, competitive training.

Acknowledgment

Firstly, I sincerely thank my supervisor, Associate Professor Razvan Beuran. His unchanging support, helpful guidance, and continuous encouragement played a key role throughout my research journey. I want to thank the faculty members and my colleagues in the lab at the Japan Advanced Institute of Science and Technology (JAIST). Their helpful discussions and support during critical times were key in dealing with the more complicated parts of this research. Lastly, I acknowledge JAIST for providing an exceptional research environment and the essential resources that enabled me to complete this thesis.

Contents

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Le Quy Don Technical University.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Cybersecurity Challenges

Though the enhanced global interconnectedness of the digital era has countless benefits, it simultaneously escalates the vulnerability of systems to cyberattacks [3]. Such attacks target a system's confidentiality, integrity, or availability, posing massive threats to individuals and organizations. Defending teams struggle increasingly to cope with the growing volume of cyberattacks. Contributing factors include the rising sophistication of cybercrimes, widespread availability of attack tools, and a considerable shortage of skilled cybersecurity specialists [4].

Critical bottlenecks also emerge from dependence on human intervention. When specialists must investigate events and make decisions—particularly in high-attack-rate environments—the resulting latency enables hackers to inflict substantial damage [5]. To address these challenges, security teams must automate their workflows, including event monitoring, risk identification, and policy enforcement. Note, though, that standard automation tools are typically rule-based and rigid. These pre-specified directives cannot deal effectively with sophisticated, complete, and constantly changing issues. Their inflexibility can cause other problems, like service disruption, primarily if the system must deal with false alerts, like alarms caused by intrusion detection systems (IDS) [6].

1.2 Reinforcement Learning for Cyber Defense

Autonomous cyber defense agents offer promising solutions for reducing defender workloads while enhancing security capabilities. Machine learning (ML) methods have gained significant traction in combating the ever-evolving threat landscape, with reinforcement learning (RL) emerging as a particularly valuable approach within cybersecurity. This learning paradigm involves an agent attempting to develop optimal policies through environmental interaction, maximizing cumulative rewards over time. Such capability

enables intelligent decision-making and appropriate response selection. Beyond cybersecurity, RL has demonstrated revolutionary potential across diverse industries. Notable achievements include DeepMind's AlphaGo [7], which mastered complex strategy games, and autonomous robots executing sophisticated physical tasks without explicit programming [8]. Current applications span energy optimization in data centers, automated financial trading strategies, and intelligent transportation systems [9–11]. The fundamental strength of RL lies in its capacity to develop optimal policies through environmental trial-and-error learning, presenting significant potential for addressing complex operational challenges.

Despite effectiveness in controlled scenarios, Single Agent Reinforcement Learning (SARL) faces significant constraints in real-world cybersecurity contexts. A critical limitation stems from the assumption that environmental changes result solely from the agent's actions. However, cybersecurity environments continuously evolve through the dynamic interplay of defensive measures and adversarial countermeasures [12].

Multi-Agent Reinforcement Learning (MARL) addresses cybersecurity challenges by enabling multiple independent agents to interact within shared environments. This approach enhances both operational flexibility and inter-agent collaboration, supporting coordinated network defense efforts while enabling competitive interactions against adversarial actors in realistic, dynamic scenarios. Within cybersecurity contexts, MARL proves particularly valuable for developing decentralized defense architectures. Here, distributed intelligent agents monitor and protect distinct network segments while sharing critical threat intelligence and coordinating defensive responses. Such coordination strengthens overall system resilience, maintaining operational effectiveness even when individual agents become compromised [13].

1.3 Problem Statement

While recent cybersecurity research has demonstrated RL's effectiveness in developing autonomous defensive agents, training agents exclusively in offensive or defensive roles presents fundamental limitations. Such separation constrains post-training policies, optimizing agents primarily against specific opponent behaviors rather than addressing underlying system vulnerabilities. This narrow focus ultimately diminishes overall defensive strategies.

Competitive multi-agent learning models have emerged to address these limitations by simultaneously training offensive and defensive agents. However, existing competitive training approaches suffer from a critical gap: simulation environment limitations that present idealized scenarios while

omitting the unpredictable elements and noise characteristics of real-world operations. For instance, IDS alerts contain false positives and negatives that simplified simulations typically ignore. This gap becomes problematic when trained agents encounter authentic operational environments, where noise-handling difficulties can compromise system stability and cybersecurity performance.

Furthermore, most current research lacks comprehensive evaluation across simulated and real environments, limiting our understanding of how well these agents transfer from controlled training scenarios to authentic operational contexts. We address these critical gaps by developing competitive training environments incorporating realistic noise characteristics and evaluating agent performance across simulation and real-world testbeds.

This research investigates competitive learning methodologies for developing autonomous cyber defense agents. We propose a MARL framework that concurrently trains defensive and offensive agents within realistic simulation environments. Such simulations enable attack agents to explore innovative intrusion techniques while continuously challenging defensive capabilities in authentic operational contexts. Correspondingly, defense agents must adapt continuously, enhancing their threat detection and response capabilities against evolving attack patterns in realistic settings. By leveraging competitive training within realistic simulations, this approach enables the development of robust defenses capable of handling diverse cyber threats. Through examining attacker-defender interactions, we aim to enhance the effectiveness and resilience of cybersecurity practices.

1.4 Thesis Contributions

The scientific contributions of this thesis to autonomous cyber defense, particularly through reinforcement learning, are as follows:

1. **Development of a Realistic Simulation Environment for Competitive Cyber Training:** This work develops a competitive network simulation with simulated IDS, including realistic false positives. The environment enables direct transfer to real deployments, matching actual network conditions rather than idealized scenarios.
2. **Implementation and Training of a Cyber Defense Agent Through Competitive Learning:** The research establishes defense agent training using multi-agent reinforcement learning against adaptive attackers. Agents learn to counter various attack strategies through competitive training in simulated networks.

3. **Creation of a Real Environment Experiment Setup:** This contribution presents a real testbed using standard cybersecurity tools: Snort IDS, iptables, Docker honeypots, Nmap, and Metasploit. The setup includes configurable false positives to evaluate agent performance under realistic sensor noise.
4. **Evaluation of Agent Performance in Both Simulation and Real Environments:** The study examines trained agents in simulated and real networks using Snort IDS. Results demonstrate agents successfully transfer from simulation to real environments while handling IDS noise and dynamic attacks.

1.5 Thesis Outline

The rest of this thesis is organized as follows:

- **Chapter 2 – Background and Related Work**: Establishes foundational concepts underlying this research, covering Reinforcement Learning (RL), Multi-Agent Reinforcement Learning (MARL), and essential game-theoretic principles. Additionally, this chapter analyzes existing literature on DRL applications within cyber defense, identifying research gaps that this thesis addresses.
- **Chapter 3 – Research Methodology**: Presents the technical framework, formulating cyber defense challenges as Partially Observable Markov Decision Processes (POMDPs), describing competitive agent architectures for autonomous cyber defense, explaining the evaluation methodology, and describing the training and testing flow in detail.
- **Chapter 4 – Experimental Evaluation**: Details the training process for competitive agents, reports research outcomes, and analyzes agent performance across simulated and real-world contexts.
- **Chapter 5 – Conclusion**: Concludes by summarizing key findings, discussing research limitations, and proposing promising directions for future autonomous cyber defense research.

Chapter 2

Background and Related Work

This chapter presents the theoretical foundations essential for autonomous cyber defense development. Beginning with Reinforcement Learning (RL), we examine how agents acquire decision-making capabilities through environmental interaction. The Markov Decision Process (MDP) framework provides a mathematical structure for RL applications, while various reinforcement learning methods offer distinct approaches to complex problems.

Our discussion progresses to Multi-Agent Reinforcement Learning (MARL), where multiple decision-makers operate simultaneously. This complexity requires specialized architectures, each offering particular advantages for different operational scenarios.

We then explore Game-Theoretic Foundations for Competitive Learning, establishing Nash Equilibrium concepts and exploitability metrics that guide optimal policy development in adversarial cybersecurity contexts.

Finally, we review related work in competitive cyber defense training, analyzing existing approaches and identifying critical gaps in simulation realism and sim-to-real transfer that motivate our research contributions.

2.1 Reinforcement Learning

2.1.1 Markov Decision Processes

Markov Decision Processes (MDPs) give us the math to handle decisions over time when outcomes are uncertain. An MDP has five parts: states S (all possible situations), actions A (what the agent can do), transitions $P(s'|s,a)$ (how likely each next state is), rewards $R(s,a)$ (immediate payoff), and discount factor $\gamma \in [0,1]$ (how much future rewards count compared to immediate ones).

MDPs assume the Markov property: what happens next depends only on where you are now and what you do, not on how you got there. As defined in [14]:

$$P(s', r|s, a) = P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$$

This makes problems much easier since we don't need to track history. The goal is to find a policy π^* that gets the most long-term reward.

2.1.2 Partially Observable Markov Decision Processes

Sometimes agents can't see everything in their environment. Partially Observable Markov Decision Processes (POMDPs) handle this problem by adding observations to MDPs. A POMDP uses $(S, A, T, R, \Omega, O, \gamma)$, where Ω is what the agent can observe (often incomplete), $T(s' \mid s, a)$ shows how states change, and $O(o \mid s', a)$ tells us what the agent sees after taking action.

Since agents don't know the exact state, POMDPs use belief states, probability distributions showing how likely each state is. When agents get new observations, they update these beliefs using Bayes' rule [15]:

$$b'(s') = \eta O(o|s', a) \sum_{s \in S} T(s'|s, a)b(s)$$

Here η normalizes so probabilities add up to 1.

Despite their theoretical elegance, modern POMDP algorithms typically avoid explicit belief distribution maintenance due to computational complexity. Large or continuous state spaces render belief distributions computationally prohibitive, creating storage and update challenges that exemplify the curse of dimensionality [16]. Therefore, researchers employ approximate strategies using model-free reinforcement-learning methods based on deep neural networks that directly learn internal representations of relevant history from observation sequences [17].

2.1.3 Reinforcement Learning Paradigm

Reinforcement Learning represents a machine learning paradigm where agents learn through trial-and-error interaction rather than explicit instruction. Unlike supervised algorithms that rely on labeled examples, RL agents receive reward signals indicating decision quality, driving the discovery of effective strategies for objective accomplishment [18].

The learning process centers on an agent that takes actions within an environment, receiving both state transitions and reward feedback for each decision. The agent gradually improves its policy through repeated interactions to maximize cumulative rewards. The goal is to maximize expected cumulative reward [19]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where G_t is the cumulative reward at time t, and γ (discount factor) determines the importance of future rewards.

2.2 Deep Reinforcement Learning Methods

2.2.1 Value-Based Reinforcement Learning

Value-based methods estimate the value of specific states or actions to derive optimal policies. These methods develop a value function that predicts the expected cumulative reward associated with being in a given state or taking a specific action.

Q-learning stands out among value-based methods for its ability to learn action values through experience. The Q-learning update rule [19] is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

where α is the learning rate, s' is the next state after taking action a in state s, and $\max_{a'} Q(s',a')$ is the estimated maximum future reward obtainable from s'.

Deep Q-Networks (DQN) extend this concept by using neural networks to approximate the Q-function, making it practical for high-dimensional state spaces where tabular methods would be infeasible.

2.2.2 Policy-Based Reinforcement Learning

While value-based methods learn to evaluate actions, policy-based approaches directly optimize the decision-making strategy itself. These methods parameterize the policy $\pi_\theta(a \mid s)$ and use gradient ascent to improve the expected return [19, 20]:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \Big[R(\tau) \Big] = \mathbb{E}_{s_0, a_0, \ldots} \Big[\sum_{t=0}^{\infty} \gamma^t \, r(s_t, a_t) \Big].$$

Using the policy gradient theorem [19], the gradient of the objective is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \Big[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \, G_t \Big],$$

Proximal Policy Optimization (PPO) is a prominent algorithm, which restricts each update to a small "trust region." As proposed by Schulman et al. [21], PPO maximizes the following clipped surrogate objective:

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

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}, \quad \hat{A}_t \approx G_t - V_\phi(s_t),$$

and ϵ is a small clipping parameter. This clipping prevents substantial policy updates, effectively balancing exploration and exploitation.

2.2.3 Actor–Critic Methods

Actor–critic methods combine the complementary strengths of value-based and policy-based approaches by maintaining two parameterized functions: a policy (the "actor") and a value estimator (the "critic"). The actor selects actions according to $\pi_\theta(a \mid s)$, while the critic estimates the state-value function $V_\phi(s)$ (or the action-value function $Q_\phi(s,a)$). At each time step t, the critic computes a temporal-difference (TD) error [19],

$$\delta_t = r_t + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t),$$

which serves both to update the critic by minimizing the squared error,

$$\phi \ \leftarrow\ \phi - \beta\, \nabla_\phi\big(\tfrac{1}{2}\,\delta_t^2\big),$$

and to inform the actor's policy gradient,

$$\theta \ \leftarrow\ \theta + \alpha\, \delta_t\, \nabla_\theta \log \pi_\theta(a_t \mid s_t).$$

One prominent variant, Advantage Actor–Critic (A2C), replaces the raw TD error with an advantage estimate, $\hat{A}_t = r_t + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t)$, thereby reducing variance and improving sample efficiency. In Asynchronous Advantage Actor–Critic (A3C), multiple actor–critic agents run in parallel on independent environment instances; each agent applies its local gradients to a shared set of parameters, stabilizing learning through decorrelated updates and enhancing exploration.

Actor-critic methods work well because the critic helps the actor learn faster than policy methods, while keeping training more stable than pure value methods. They handle mixed action types well—like robots needing continuous motor control and discrete grip decisions, or cars that steer smoothly but change lanes in steps.

But actor-critic has problems too. If the critic gives bad value estimates, training becomes unstable. Getting the learning rates (α, β) right takes careful tuning, and choosing good advantage estimates greatly matters. Running multiple actors in parallel can cause issues with outdated gradients, plus the extra computation costs can be high.

Actor-critic bridges value and policy approaches effectively. These methods work in many areas, from robot control tasks to complex games like StarCraft II. Researchers keep improving them with better variance reduction and new architectures.

2.3 Multi-Agent Reinforcement Learning (MARL)

2.3.1 MARL Overview

Multi-Agent Reinforcement Learning (MARL) generalizes single-agent RL to domains where a population of decision-makers interacts in a shared, stochastic game. As defined in [1], a finite-horizon stochastic game is defined by the tuple $\langle \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, P, \{r^i\}_{i=1}^N, \gamma \rangle$, where N agents jointly influence the transition kernel

$$P\big(s_{t+1} \,\big|\, s_t, \mathbf{a}_t\big), \quad \mathbf{a}_t = (a_t^1, \ldots, a_t^N) \in \mathcal{A}^1 \times \cdots \times \mathcal{A}^N,$$

and each agent i receives a possibly distinct reward $r_t^i = r^i(s_t, \mathbf{a}_t)$. Agents observe $o_t^i = O(s_t, i)$ and select actions according to stochastic policies $\pi_{\theta_i}(a_t^i \,|\, o_t^i)$. The objective of agent i is to maximize its discounted return [1]:

$$J^i(\theta_i) = \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^{\,t}\, r_t^i\Big].$$

When all agents share the same reward function, the game reduces to a Decentralized Partially Observable Markov Decision Process (Dec-POMDP); when rewards sum to zero, the game is competitive. MARL thereby subsumes cooperative control (multi-robot manipulation, formation flight), fully competitive games (poker, Go), and mixed-motive scenarios (autonomous traffic and smart-grid markets) [1].

2.3.2 MARL System Types

Multi-Agent Systems (MASs) can be classified into three main categories based on task characteristics. This classification relies on the reward structure and the agents' motivations within the environment [1].

In a Fully Cooperative setting, all agents share a common goal and must coordinate to maximize a shared reward. The success of the entire system depends on the collective performance of all agents, with the main challenges being coordination and credit assignment. Formally, the reward structure is represented by all agents receiving the same global reward signal [1]:

$$r_t^1 = \cdots = r_t^N \equiv r_t$$

Real-world examples include the *assembly of machine systems*, while a common application in Multi-Agent Reinforcement Learning (MARL) is *warehouse-robot routing* [1].

Conversely, the agents' goals are directly opposed in a Fully Competitive setting. This is often called a zero-sum environment, where one agent's gain is exactly another's loss. An agent's core challenge is outperforming opponents, which emphasizes the importance of opponent modeling. Mathematically, the sum of rewards for all agents at any given time is zero [1]:

$$\sum_{i=1}^{N} r_t^i = 0$$

A sumo wrestling match is a clear example, and in the multi-agent reinforcement learning (MARL) domain, this setting is familiar in two-player games such as capture-the-flag or micromanagement scenarios in real-time strategy games like StarCraft II.

The Mixed-Motive setting is the most complex and general scenario, incorporating cooperative and competitive elements. Agents in this setting must balance their individual goals with the group's objectives. They may need to form alliances to collaborate with specific agents while competing against others. As a result, the rewards for each agent usually include both common components—shared with their allies—and private components unique to them. A good analogy for this dynamic is a soccer match, where teammates work together to score against the opposing team [1].

2.3.3 MARL Training–Execution Paradigms

The training of multiple agents continues to present significant computational challenges, particularly due to the exponential growth of state and action space with the increasing number of agents. As a result, even contemporary deep learning techniques can encounter limitations. Training–Execution Paradigm for developing agent policies in a multi-agent context can be classified into three types: Decentralized Training - Decentralized Execution,

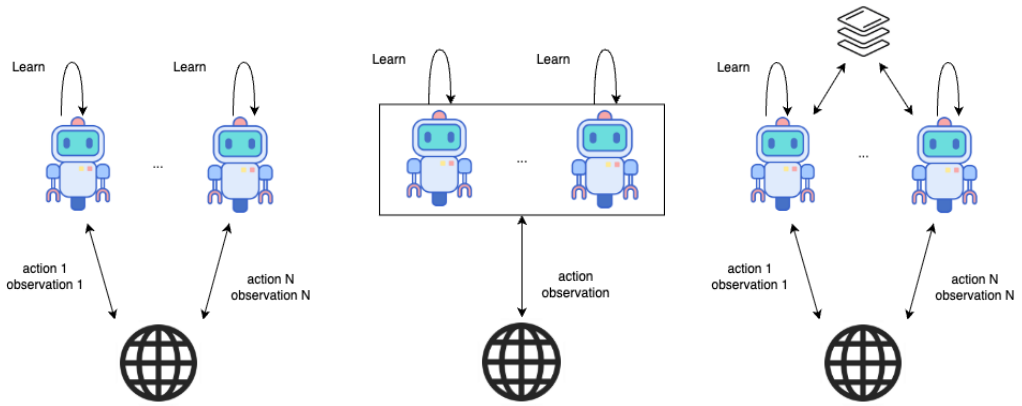
Centralized Training - Centralized Execution, and Centralized Training - Decentralized Execution [1].

Training is the phase when agents collect data to improve their skills and behavior based on the rewards they receive. In contrast, "test time" is when we evaluate the learned policy without making any changes. There are two main types of agent training: centralized and distributed [22].

Centralized training involves agents improving their policies by sharing information during training. However, this shared data is usually ignored during testing. In contrast, distributed training occurs when individual agents update their policies independently, creating unique strategies without depending on external information.

In distributed training scenarios, agents learn independently without exchanging explicit information. Each agent i possesses a policy $\pi_i : O_i \rightarrow P(U_i)$ that translates local observations into a distribution of individual actions, with no shared information between agents.

The Decentralized Training - Decentralized Execution paradigm presents a fundamental challenge, as each agent perceives the environment as non-stationary. This perception arises because agents are unaware of each other's knowledge and joint actions. Limited information in the distributed context exposes independent learners to several challenges. Alongside non-stationarity, environments might present stochastic transitions or rewards, complicating the learning process. Additionally, the pursuit of an optimal policy can influence the decisions of other agents, resulting in action shadowing, thus affecting the exploration-exploitation balance.



(a) Decentralized training - decentralized execution (b) Centralized training - centralized execution (c) Centralized training - decentralized execution

Figure 2.1: MARL training-execution paradigms (based on [1])

Overall, the evolving landscape of multi-agent training methodologies

continues to adapt and refine approaches, addressing challenges and leveraging innovations for improved performance in complex environments.

2.4 Game-Theoretic Foundations for Competitive Learning

The adversarial nature of cyber operations, with an attacker and a defender having opposing objectives. Game theory provides the mathematical foundations to analyze these interactions and define what constitutes an optimal strategy for each agent in such a competitive setting. In this section, we introduce the core concepts that underpin our competitive learning approach. First, we will discuss the Nash Equilibrium idea, representing a pair of stable, optimal policies for both agents. Following that, we will introduce Exploitability, a critical metric that measures how close any given policy is to this optimal solution.

2.4.1 Nash Equilibrium

In game theory, a Nash Equilibrium represents a stable state in a competitive game where no player can improve their outcome by unilaterally changing their strategy. This state is reached when both agents' policies are the best possible response to each other [23]. Formally, for a two-player game, a pair of policies (π_*^1, π_*^2) is a Nash Equilibrium if for every state s and any alteive policies π^1 and π^2, the following conditions hold:

$$v^1(s, \pi_*^1, \pi_*^2) \geq v^1(s, \pi^1, \pi_*^2) \tag{2.1}$$
$$v^2(s, \pi_*^1, \pi_*^2) \geq v^2(s, \pi_*^1, \pi^2) \tag{2.2}$$

In competitive reinforcement learning, the Nash Equilibrium represents the optimal solution the learning process aims to converge upon. Instead of getting trapped in "strategic cycles" where agents constantly exploit temporary weaknesses, algorithms must guide the agents' policies toward this equilibrium [24]. The policy found at this point is the optimal minmax policy, which guarantees the best possible outcome against a worst-case opponent.

2.4.2 Exploitability Metric

In addition to finding the Nash Equilibrium, a key challenge is evaluating how close any given agent's policy is to this optimal state. Exploitability is a formal metric used for this purpose. It is defined as the difference between the

expected return of the optimal (minmax) policy and the expected return of the policy being evaluated, when both are playing against a "worst-case" or "best-response" opponent. In essence, it measures how much potential value an agent loses by using a sub-optimal policy compared to the guaranteed value of the minmax policy.

This can be formally expressed [23] as:

$$\text{expl}(\pi^i) = \mathbb{E}[G|\pi^i_*, \pi^{-i}_*] - \mathbb{E}[G|\pi^i, \pi^{-i}_*] \tag{2.3}$$

where G is the return from a single game, π^i is the policy being evaluated, π^i_* is the optimal minmax policy, and π^{-i}_* represents the opponent's optimal best-response policy. A policy with zero exploitability is considered a perfect minmax policy and thus a component of a Nash Equilibrium. Therefore, competitive training aims to drive the exploitability of an agent's policy as close to zero as possible.

2.5 Related Work

This section examines literature relevant to autonomous cyber defense agent training. Our structured review covers four key areas: broad Applications of Deep Reinforcement Learning in Cyber Defense, demonstrating RL automation of defense tasks; Competitive training for Cyber Defense, analyzing adversarial and multi-agent approaches for enhanced defensive robustness; RL Training Environments for Cyber Defense, evaluating prominent platform capabilities and limitations; and Summary with Research Gap Identification, synthesizing findings to identify current approach limitations that motivate our contributions.

2.5.1 Competitive Training for Cyber Defense

This section examines competitive training approaches for cyber defense, analyzing how adversarial multi-agent frameworks enhance defensive robustness through strategic co-evolution. We review developments from early game-theoretic models to contemporary Nash equilibrium-based training, identifying current limitations and research gaps.

2.5.1.1 Direct Adversarial Training Approaches

The survey by Nguyen and Reddi [25] has a part on Game-Theoretic Approaches, which is directly related to this topic. Among the surveyed works, the research by Zhu et al. [26] stands out by modeling the interactions

between an attacker and a defender as a competitive game between two agents. In their framework, each agent learns to counter the other based on private information; the defender is unaware of the specific attacks being launched, and the attacker does not know the precise defense configuration. An Intrusion Detection System (IDS) provides information to the defender, who computes a corresponding utility value to adapt their strategy. Figure 2.2 illustrates this interaction process between the attacker and defender. However, their IDS assumption provides complete knowledge, yielding deterministic utility feedback that ignores real-world system noise like false negatives and positives. While this idealization simplifies defense problems, it fails to capture operational uncertainties our work addresses. Additionally, their motivating scenario focuses on single vulnerability classes (exemplified by Heartbleed attacks) rather than diverse threat ranges. Since defenders treat attacker strategies as black boxes, learning becomes one-sided adaptation to non-stationary opponents rather than guaranteed competitive co-evolution where both agent policies mutually adapt.

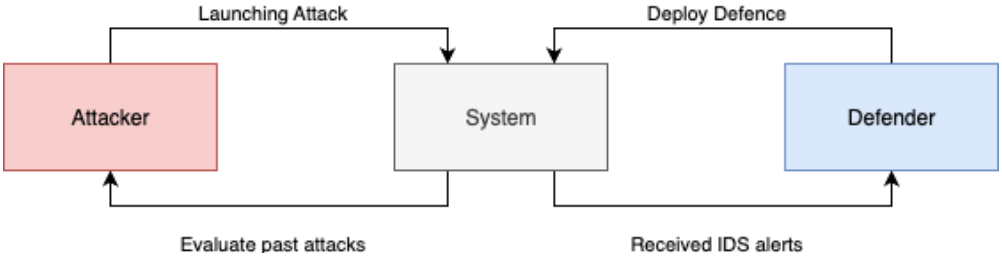


Figure 2.2: Attacker-defender interaction with IDS

Building on these foundations, Kunz et al. [27] also explore competitive agent training by extending the CyberBattleSim environment to support a trainable blue agent and joint red-blue training scenarios. However, similar to the previously discussed works, this framework also operates with an idealized defender model. The defender's observation space provides ground-truth information, such as a definitive list of infected nodes, and its reward is a direct negation of the attacker's success. This setup does not incorporate the challenge of processing imperfect information, such as the false positives (FP) common in real-world Intrusion Detection Systems. Additionally, the authors acknowledge that their choice of the abstract CyberBattleSim environment prioritizes rapid experimentation with RL algorithms over the real-world transferability of the trained agents. While this approach is valuable for exploring agent training methodologies, the significant gap between this simulated environment and the complexities of a live network is a crucial consideration.

More recently, the work of McDonald et al. [2] provides a proof-of-concept for applying competitive Reinforcement Learning (RL) to find game-theoretic optimal policies within an Autonomous Cyber Operations (ACO) environment. Using the CybORG simulator [28], they implement a fictitious play algorithm with opponent sampling, where Red and Blue agents are trained simultaneously against a growing pool of past opponent policies. This training process is illustrated in Figure 2.3. Their work demonstrates that this approach can steer agents from strategic cycles towards a Nash Equilibrium. A key contribution is using exploitability as a metric to validate that the learned policies converge towards optimal play formally.

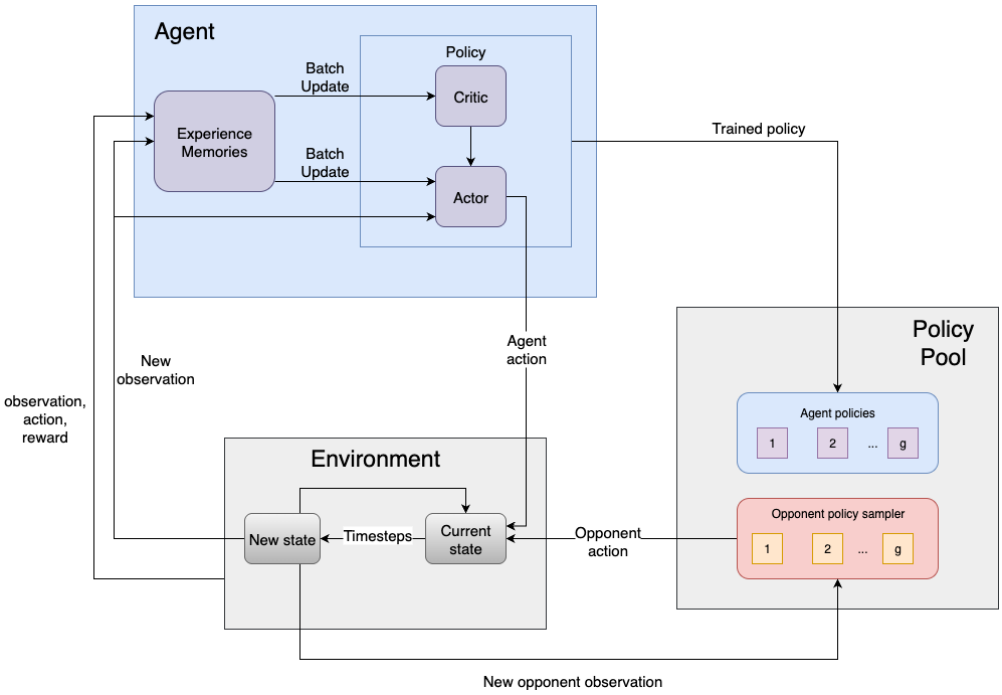


Figure 2.3: Fictitious play training with opponent sampling (based on [2])

Despite training methodology advances, environmental idealizations persist. Endpoint monitoring device notifications reach Blue agents as deterministic facts, ignoring noise and false alarms characteristic of real Intrusion Detection Systems. Moreover, real-world agent testing remains unimplemented. Authors acknowledge that CybORG environments, despite simulator improvements, lack sufficient realism for practical trained model deployment. This highlights ongoing challenges in bridging simulated training environments with operational network defense complexities.

2.5.1.2 Strategic Frameworks Based on Uncertainty

A complementary line of research explores game-theoretic frameworks that explicitly model uncertainty about adversary capabilities. Sengupta and Kambhampati [29] propose Bayesian Stackelberg Markov Games (BSMGs) for Moving Target Defense scenarios, where defenders must adapt their configurations under incomplete information about different attacker types. Their framework addresses a key limitation in prior work by modeling attacker uncertainty through probability distributions over adversary capabilities, while maintaining a strong threat model where all attacker types know the defender's policy. The proposed BSS-Q learning algorithm converges to Strong Stackelberg Equilibrium and demonstrates superior performance compared to single-agent RL methods that ignore adversarial strategic behavior. Though this work focuses on proactive defense mechanisms like configuration switching rather than reactive intrusion detection, it establishes important theoretical foundations for competitive learning under uncertainty.

Another approach to multi-agent coordination focuses on distributed response mechanisms for large-scale network threats. Malialis and Kudenko [30] address DDoS defense through Coordinated Team Learning (CTL), where multiple reinforcement learning agents collaborate hierarchically to throttle malicious traffic at different network points. Their framework demonstrates scalability to over 100 learning agents through task decomposition and team-based rewards, showing significant performance improvements over traditional rate-limiting approaches like AIMD (Additive Increase Multiplicative Decrease). The hierarchical communication structure enables decentralized response without single points of control, though the approach assumes attackers with sending rates distinguishable from legitimate users. While not employing competitive training between attacker and defender agents, this work illustrates the potential for coordinated multi-agent defense strategies in realistic network scenarios.

In the domain of cyber-physical systems, game-theoretic actor-critic approaches have shown promise for handling unknown adversarial behaviors. Feng and Xu [31] formulate cyber defense as a continuous-time zero-sum game between defender and attacker policies, developing a game-theoretical actor-critic neural network architecture that learns optimal defense strategies online. Their framework employs two actor networks—one approximating the worst-case attack policy and another learning the optimal defense response—trained simultaneously through deep reinforcement learning. The approach demonstrates convergence to stable defense policies even against completely unknown attack strategies, though evaluation remains limited to cyber-physical system simulations rather than network security scenarios.

This work contributes to the theoretical understanding of simultaneous policy learning in adversarial settings, though without our approach's explicit competitive training dynamics.

Our work directly addresses these gaps through CMARL-ACD (Competitive Multi-Agent Reinforcement Learning for Autonomous Cyber Defense), a novel framework that combines competitive training between adversarial agents with realistic operational constraints. We introduce a competitive training framework that not only pits attacker and defender agents against each other but also does so in an environment with realistic sensor imperfections, explicitly modeling IDS false positives. Moreover, we also test our agents on an emulated environment that reflects the real one using operational tools like Snort and Metasploit, demonstrating our approach's practical effectiveness and transferability.

2.5.2 Reinforcement Learning Training Environment for Cyber Defense

Training environments prove essential for all cybersecurity Reinforcement Learning methods. Recent environmental advances trade realism against experimentation speed. Microsoft's *CyberBattleSim* provides high-level computer system and security principle abstractions, enabling extensive attack and defense tactic experimentation while limiting focus on real-world agent transferability.

Another significant training platform is the Network Attack Simulator (NASim) [32], which provides a lightweight, fast simulation environment specifically designed for network penetration testing scenarios. NASim models networks as directed graphs where nodes represent hosts with varying operating systems, services, and vulnerabilities, while edges define network connectivity and access permissions. The environment supports flat network topologies and more complex multi-subnet configurations, making it suitable for evaluating diverse attack strategies and defense mechanisms. Unlike more complex simulators, NASim prioritizes computational efficiency and simplicity, enabling rapid experimentation with different network configurations and attack scenarios. The platform implements a state-action framework where attackers perform reconnaissance, exploitation, and lateral movement actions, while defenders can monitor network activity and deploy countermeasures. This makes NASim particularly valuable for developing and testing multi-agent reinforcement learning approaches in cybersecurity, as it provides a controlled yet realistic enough environment for competitive training between attacker and defender agents.

Conversely, platforms like CybORG [28] prioritize higher fidelity for successful real network deployment. CybORG provides standardized interfaces for simulated and emulated environments, facilitating sim-to-real transfer. However, environmental observation mechanisms often abstract real-world security operations. CybORG provides defending agents with consolidated observation objects per timestep, generated by filtering and merging endpoint monitoring tool data like Velociraptor. While such tools see practical use, many defenders primarily respond to continuous alert streams from Intrusion Detection Systems (IDS) or SIEM systems. Critically, these practical alerts frequently contain false alarms, creating challenges our work addresses.

2.5.3 Summary and Research Gap Identification

Recent advances in applying Deep Reinforcement Learning (DRL) to cyber defense have demonstrated the potential of autonomous agents to detect, respond to, and adapt against sophisticated threats in dynamic environments. Despite these advances, a critical gap persists in the realism of training and evaluation environments. Many prior studies, including those leveraging competitive multi-agent training, rely on idealized simulation settings where defenders receive deterministic, noise-free information from sensors such as Intrusion Detection Systems (IDS). This assumption overlooks the prevalence of false positives and negatives in real-world IDS deployments, which can significantly impact the effectiveness and robustness of learned defense strategies. Furthermore, most competitive training frameworks are validated exclusively in abstract or highly simplified environments, leaving the sim-to-real transfer of agent policies largely unaddressed. Another limitation is the lack of a comprehensive evaluation in operational or emulated environments. While platforms like CybORG [28] have improved simulation fidelity, they still abstract many complexities of real-world network defense, and few works have demonstrated the deployment or effectiveness of trained agents in practical enterprise settings.

There is a pressing need for competitive training frameworks that explicitly model sensor imperfections—such as IDS alert noise—and for systematic evaluation of agent performance in both simulated and realistic (emulated) environments. Addressing these gaps is essential for developing autonomous cyber defense agents that are robust, transferable, and effective in real-world operational contexts.

Our work addresses this gap through a two-stage process. First, we model an imperfect Intrusion Detection System directly within our simulation environment by embedding alert noise with false-positive IDS rules. After training our agents under these conditions, we conduct comprehensive

evaluations within the simulation and in an emulated environment to evaluate the sim-to-real transfer. This evaluation integrates an operational Snort IDS, allowing us to demonstrate that the policies learned by our agents remain effective in a practical enterprise setting.

Table 2.1 compares our research contributions with existing works in automated cyber defense and reinforcement learning. This comparison highlights the key differentiators and advancements our work introduces to address the research gaps mentioned above.

Table 2.1: Comparison of research approaches in automated cyber defense

Criteria	**[2]**	**[27]**	**[30]**	**[29]**	**[31]**	**[26]**	**CMARL-ACD**
IDS Integration			✓	✓		✓	✓
Real/Emulated Evaluation			✓				✓
Sim-to-Real Transfer							✓
Industry Tools							✓
Competitive Training	✓	✓		✓	✓		✓
Noise/Uncertainty				✓	✓		✓
Multi-stage Attacks	✓	✓		✓			✓
Large Scale Experiment			✓				
Multi-agent Learning	✓	✓	✓	✓	✓		✓
Action Space Realism	✓	✓	✓	✓			✓
Environment Realism	✓	✓	✓	✓			✓

Chapter 3

Research Methodology

This chapter presents our methodology for training and evaluating autonomous cyber defense agents. Our approach centers on competitive environments where attacker and defender agents co-evolve through continuous interaction across simulated and real environments.

For simulation training, we extend Network Attack Simulator (NASim), an open-source cybersecurity simulation platform designed for reinforcement learning research [32], with competitive multi-agent capabilities. Our enhanced platform incorporates attacking agents, defensive agents, and realistic benign traffic patterns. We implement a testbed for real-world validation using industry-standard tools, including Snort IDS, Metasploit, and iptables firewall rules. Central to both environments is an integrated Intrusion Detection System (IDS) model that generates alerts with false positives, mirroring real-world security operation uncertainties.

We examine two Multi-Agent Reinforcement Learning (MARL) frameworks: MAPPO for centralized training and IPPO for fully decentralized learning, with tailored self-play implementation. Experimental methodology tests these approaches across clean baseline and noisy sensor uncertainty environments, evaluating performance in simulation and realistic conditions. Our metrics capture agent performance and robustness, particularly defensive agent handling of sensor noise and sim-to-real adaptation.

3.1 Problem Formulation for Cyber Defense

We formulate cyber defense as a competitive multi-agent Partially Observable Markov Decision Process (POMDP), capturing fundamental network security characteristics: adversarial attacker-defender dynamics and incomplete information security decision-making.

Figure 3.1 illustrates how we model this cyber defense environment as a POMDP. The following sections detail each component of this environment: state space, action space, reward functions, observation space, and learning objectives.

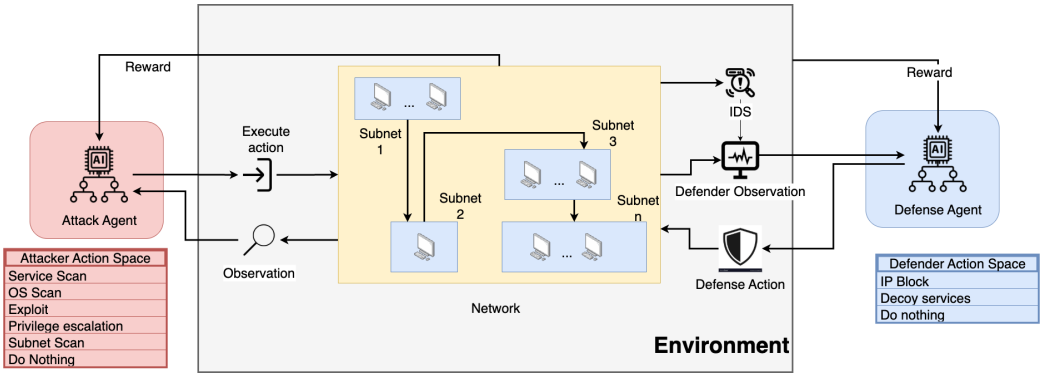


Figure 3.1: POMDP components for cyber defense

3.1.1 State Space

The full state space $\mathcal{S}$ captures the complete network topology and security state, including:

- Network topology (hosts, subnets, connections)
- Service configurations on each host
- Operating system on each host
- Attacker's discovery state on each host
- Privilege levels achieved by the attacker on each host
- Active decoy services on each host
- Alerts generated by the IDS on each host

While the environment maintains complete state information, agents observe only partial state aspects, mirroring information asymmetries characteristic of real-world cybersecurity operations.

Network Topology captures the structural organization of the network: hosts, subnets, and their interconnections. This topology essentially defines available attack paths and establishes defensive boundaries.

Attackers begin from external hosts beyond network perimeters, penetrating systems through subnet progression. Subnet 0 serves as the internet-facing entry point, requiring initial attacker targeting of contained hosts. Distinct host values assigned through scenario configuration files drive attacker and defender reward calculations. Figure 3.2 illustrates this attack progression.

Service Configurations specifies which services operate on each host—web servers, SSH daemons, and similar applications. Running services poten-

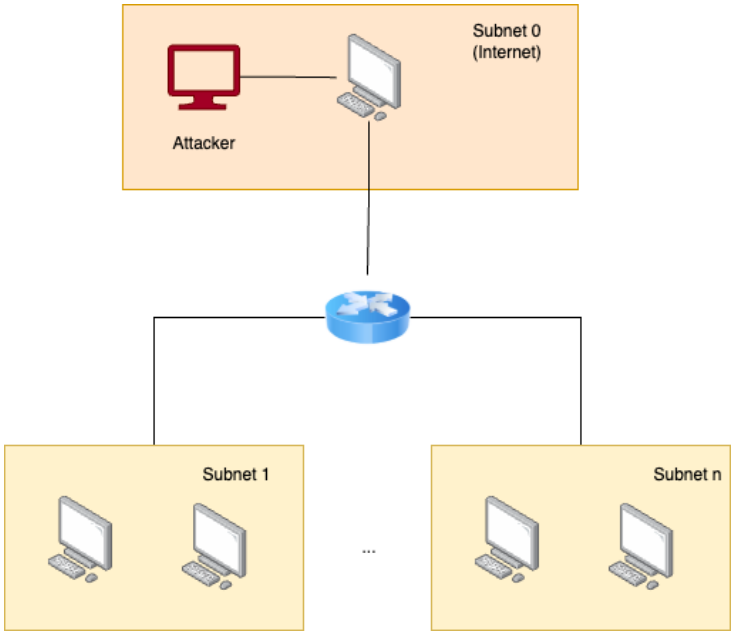


Figure 3.2: Attack progression through network topology

tially introduce vulnerabilities, enabling unauthorized attackers to access. Vulnerable web servers might allow remote code execution, while unpatched SSH services could permit unauthorized authentication attempts. Defenders must prioritize protection efforts based on service criticality and vulnerability exposure.

Operating Systems information determines exploit compatibility per host. Vulnerability existence alone doesn't guarantee exploitation success—attackers require OS-specific exploits. Windows-targeted exploits prove ineffective against Linux systems running identical vulnerable services.

Attacker's Discovery State tracks the attacker's identification of network hosts. Reconnaissance marks discovered hosts as "on" while undiscovered hosts remain "off." Binary tracking provides attackers with network exploration progress overviews, identifying surveyed areas versus unexplored territories.

Privilege Levels measures attacker control extent over compromised hosts, ranging from no access through user privileges to root/administrator rights. Enhanced privileges enable broader attacker actions and potential host leveraging as lateral movement pivot points.

Active Decoy Services monitors defender honeypot deployments for attacker mislead and detection. While decoys mimic legitimate services, they generate high-confidence alerts when accessed, providing reliable attacker presence indicators.

IDS Alerts encompasses Intrusion Detection System notifications identifying potentially malicious behavior. Alerts inevitably include false positives—benign activities mistakenly flagged as threats—reflecting imperfect real-world security system detection capabilities.

3.1.2 Action Space

We partition the action space between attackers (Red team) and defenders (Blue team), with each side possessing specialized capabilities. Attackers conduct reconnaissance to identify hosts and services, exploit discovered vulnerabilities, and escalate their privileges. Defenders deploy decoy services and implement traffic blocking measures. Table 3.1 details these action spaces.

Table 3.1: Attacker and defender action spaces

Red Action Space (Attacker)	**Blue Action Space (Defender)**
Service Scan Probes a specific host to discover open ports and running services.	**Deploy Decoy Service** Sets up a honeypot service on a host to deceive and lure the attacker.
OS Scan Scans a host to identify its operating system.	**Block IP Address** Deploys a firewall rule to block traffic from a specific IP address identified as malicious.
Subnet Scan Scans a subnet to discover active hosts and their IP addresses.	**Do Nothing (NoOp)** Takes no action for one timestep, which can be a strategic choice to conserve resources.
Exploit Attempts to use a known vulnerability to gain initial access to a host.	
Privilege Escalation Attempts to gain root/administrator privileges on a previously compromised machine.	
Do Nothing (NoOp) Takes no offensive action for one timestep.	

Attackers follow structured progressions to compromise internal network

hosts, beginning with reconnaissance activities (service scans, OS scans) targeting Subnet 0 perimeter hosts, then exploiting system vulnerabilities before conducting subnet scans identifying additional connected network targets. Lateral movement to other subnet hosts requires successful compromise of the current subnet system. Figure 3.3 demonstrates this attack sequence.

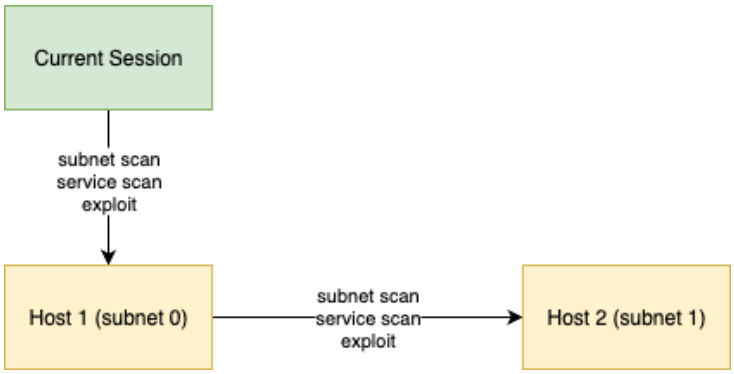


Figure 3.3: Attack sequence to compromise internal hosts

3.1.3 Reward Functions

Reward functions establish zero-sum competitive environments between attackers and defenders while maintaining asymmetric objectives.

The attacker's reward function, shown in Equation 3.1, comprises two primary elements: the value obtained from successful operations and the costs associated with action execution.

$$R_R(s, a_R, s') = V(a_R, s') - C(a_R) \tag{3.1}$$

where the following notations are used:

- $V(a_R, s')$ is the value gained from the action, which includes:

$$V(a_R, s') = \sum_{h \in \mathcal{H}_{\text{new}}} v_h^{\text{disc}} + \sum_{h \in \mathcal{H}_{\text{comp}}} v_h \cdot (1 - \mathbb{I}_{\text{decoy}}(h)) \tag{3.2}$$

  where:

  - $\mathcal{H}_{\text{new}}$ is the set of newly discovered hosts
  - v_h^{disc} is the discovery value of host h
  - $\mathcal{H}_{\text{comp}}$ is the set of successfully compromised hosts
  - v_h is the value of host h

- $\mathbb{I}_{\text{decoy}}(h)$ is an indicator function that equals 1 if h is a decoy service, 0 otherwise

- $C(a_R)$ is the cost associated with executing action a_R

The defender's reward function (Equation 3.3) exhibits greater complexity, incorporating defensive action values, action costs, IP blocking decision rewards/penalties, time-based compromised system penalties, and attacker-decoy interaction rewards.

$$R_B(s, a_B, s') = V(a_B, s') - C(a_B) + R^{\text{block}} + R^{\text{time}} + R^{\text{decoy}} \tag{3.3}$$

where the following notations are used:

- $V(a_B, s')$ is the value gained from the defender's action
- $C(a_B)$ is the cost of executing the defensive action a_B
- R^{block} represents the reward/penalty from IP blocking:

$$R^{\text{block}} = \begin{cases} p_{\text{correct}} \cdot r_{\text{correct}} & \text{if block targets attacker} \\ -p_{\text{wrong}} & \text{if block targets innocent} \end{cases} \tag{3.4}$$

  where p_{correct} is the reward factor for correct blocks, r_{correct} is the reward for correct blocks, and p_{wrong} is the penalty for incorrect blocks
- R^{time} represents the time-based penalty for compromised hosts:

$$R^{\text{time}} = -\sum_{h \in \mathcal{H}_{\text{comp}}} \min(v_h \cdot \alpha \cdot e^{-\beta \Delta t_h}, r_{\max}) \tag{3.5}$$

  where:

  - $\mathcal{H}_{\text{comp}}$ is the set of compromised hosts
  - v_h is the value of host h
  - Δt_h is the number of timesteps host h has been compromised
  - α and β are adjustment parameters (default: $\alpha = 1.0$, $\beta = 0.1$)
  - $r_{\max}$ is the maximum penalty per host

- R^{decoy} represents the reward when the attacker interacts with decoy services:

$$R^{\text{decoy}} = r_{\text{decoy}} \cdot \mathbb{I}_{\text{triggered}} \tag{3.6}$$

  where r_{decoy} is the reward value and $\mathbb{I}_{\text{triggered}}$ equals 1 if a decoy is triggered, 0 otherwise

3.1.4 Observation Space

The observation function $\mathcal{O}$ models incomplete information accessible to both agents, establishing realistic attacker-defender information asymmetries mirroring actual cybersecurity operation visibility limitations.

Defender's Observation Space The defender's observational capabilities include:

- Complete visibility of system configuration (services, operating systems, processes)
- Knowledge of host values and discovery values
- Awareness of monitoring status and deployed decoy services
- Access to alerts for all IP monitoring (primary detection mechanism)
- Cannot directly observe which hosts are compromised
- Cannot determine which hosts the attacker has discovered or can reach
- Cannot see the attacker's privilege levels on hosts

Attacker's Observation Space The attacker's observational capabilities include:

- Knowledge of which hosts they have compromised
- Awareness of which hosts they've discovered and can reach
- Visibility of their access level on hosts
- Information about services, operating systems, and processes, but *only after* performing appropriate scans
- Cannot see host values unless they've successfully exploited the host
- Cannot determine monitoring status or identify which services are decoys
- No visibility into generated alerts

This observation model captures key real-world cybersecurity dynamics:

1. Attackers possess direct operational progress knowledge while lacking defensive countermeasure insight
2. Defenders maintain comprehensive infrastructure knowledge but depend on alert systems for attacker presence identification
3. Effective defense requires alert correlation with potentially compromised systems for appropriate response implementation

Rather than providing unrealistic defender omniscience regarding attacker activities, our simulation enforces alert-based detection reliance, creating challenging, authentic detection scenarios.

3.1.5 Learning Objective

Within our competitive multi-agent framework, both attackers and defenders strive to optimize their cumulative discounted rewards across time [19]:

$$\max_{\pi_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_t^i, s_{t+1})\right] \tag{3.7}$$

Here, π_i denotes the policy for agent i (attacker or defender), a_t^i represents agent i's action at timestep t, and γ serves as the discount factor balancing immediate against future rewards.

While both agents pursue reward maximization, our research emphasizes robust defensive policy development. Defenders must master several capabilities:

1. Accurately detect attacker activities in the presence of sensor noise
2. Deploy decoys strategically to divert and identify attackers
3. Block malicious traffic with minimal impact on benign users
4. Balance immediate defensive responses with long-term security objectives

Conventional reinforcement learning typically optimizes for average-case scenarios. Cybersecurity demands preparation for worst-case situations and sophisticated adversaries. Consequently, we assess defensive policies on average performance and exploitability—vulnerability to optimal attack strategies.

Adversarial contexts necessitate multi-agent reinforcement learning algorithms explicitly modeling competitive dynamics. We train defensive agents against increasingly capable attackers, developing policies resisting diverse attack strategies, including sophisticated adaptive attacks approximating worst-case scenarios. This methodology ensures defensive policy effectiveness against previously unencountered attack patterns.

3.2 Competitive Agent Design

We construct competitive agents using Independent Proximal Policy Optimization (IPPO) and Multi-Agent Proximal Policy Optimization (MAPPO) architectures. This dual approach enables comparisons between decentralized and centralized training strategies for attacker and defender agents.

Agent architectures utilize deep neural networks with shared structural foundations for attacker and defender agents. Principal IPPO-MAPPO distinctions involve critic design and training information sharing protocols.

3.2.1 Network Structure

Both policy and value networks follow the same architectural pattern:

- **Policy Networks**: Input → [256] → [512] → [512] → [256] → Action Logits
- **Value Networks**: Input → [256] → [512] → [512] → [256] → State Value
- **Activation Function**: Hyperbolic tangent (tanh) activation function for all hidden layers

The detailed architectures of our actor and critic networks are illustrated in Figures 3.4 and 3.5, respectively.

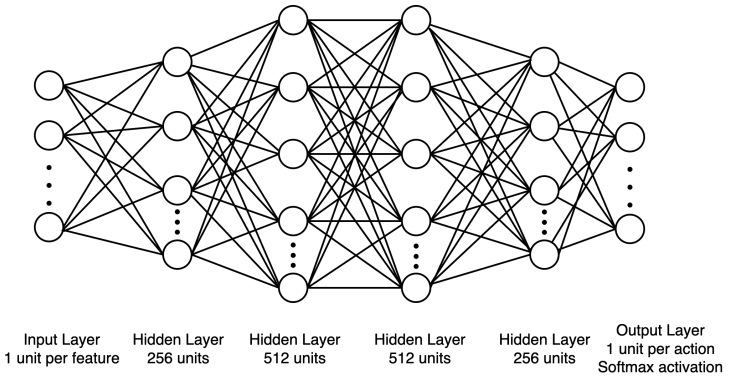


Figure 3.4: Actor network architecture

Actor networks process state observations through fully-connected layers with tanh activation functions, transforming high-dimensional observation spaces into action probabilities for stochastic policy selection during training and evaluation phases. The final layers output action logits processed through softmax functions, generating valid action probability distributions.

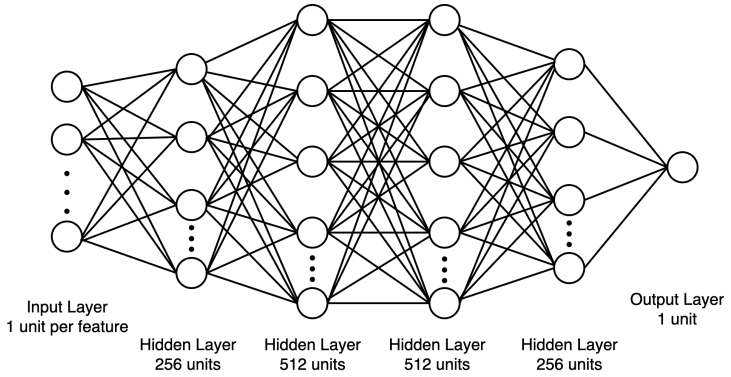


Figure 3.5: Critic network architecture

Critic networks employ similar architectural foundations while serving fundamentally different purposes—estimating state values for policy gradient computation. Unlike actor network probabilistic outputs, critics produce single scalar values representing expected returns from current states. Such value estimation proves essential for PPO algorithm advantage calculation.

3.2.2 Key Architectural Decisions

Our design prioritizes independence and specialization, with each agent maintaining distinct policy networks without attacker-defender parameter sharing. IPPO employs independent critics observing only individual agent state information, while MAPPO uses centralized critics accessing both agent information during training. Agents receive specialized observations filtered and customized for specific roles and capabilities.

3.2.3 Action Processing

Action masking guarantees agent selection of valid actions only, preventing impossible operation attempts. This proves crucial in cybersecurity contexts where actions become available only under specific conditions (host exploitation requiring prior discovery and vulnerability identification).

3.2.4 Learning Algorithm

Our training process incorporates several essential components:

Table 3.2: Learning algorithm parameters

Component	**Configuration**
Optimization	PPO with clipped objective
Advantage Estimation	GAE with $\gamma = 0.99$, $\lambda = 0.95$
Exploration	Entropy bonus (coefficient = 0.1)
Value Loss	Smooth L1 Loss
Optimizer	Adam (learning rate = 3×10^{-4})

3.2.5 Training Setup

Agent training uses IPPO and MAPPO algorithms with network infrastructure designed to provide sufficiently complex environments for realistic attack simulation while maintaining computational efficiency. The competitive training process enables agents to develop sophisticated strategies through adversarial interaction, with specific experimental configurations and scenario descriptions detailed in Chapter 4.

We implement competitive training paradigms where attacker and defender agents learn concurrently through repeated interactions, generating natural curricula that progressively intensify task difficulty:

1. Both agents initiate training with random policies, facilitating initial exploration.
2. As attacker capabilities advance, defenders must evolve more sophisticated defensive strategies.
3. Conversely, attackers must uncover increasingly complex attack patterns as defensive capabilities improve.
4. This adversarial co-evolution persists until convergence, producing progressively more robust policies.

During training, models are periodically saved to create comprehensive agent pools for evaluation purposes. We employ exploitability metrics to assess defensive model effectiveness, using worst-case performance analysis to identify robust defensive strategies capable of handling sophisticated adversaries. This methodology proves essential in adversarial environments where preparation for challenging scenarios remains critical.

The experimental evaluation chapter provides the specific implementation details of model saving intervals, evaluation episodes, and exploitability calculations.

3.3 System Flow and Architecture

3.3.1 Simulation Environment

The training process in our simulator follows the architecture illustrated in Figure 3.6. Our simulation environment extends the Network Attack Simulator (NASim) with Multi-Agent Reinforcement Learning (MARL) capabilities for competitive training between attacker and defender agents.

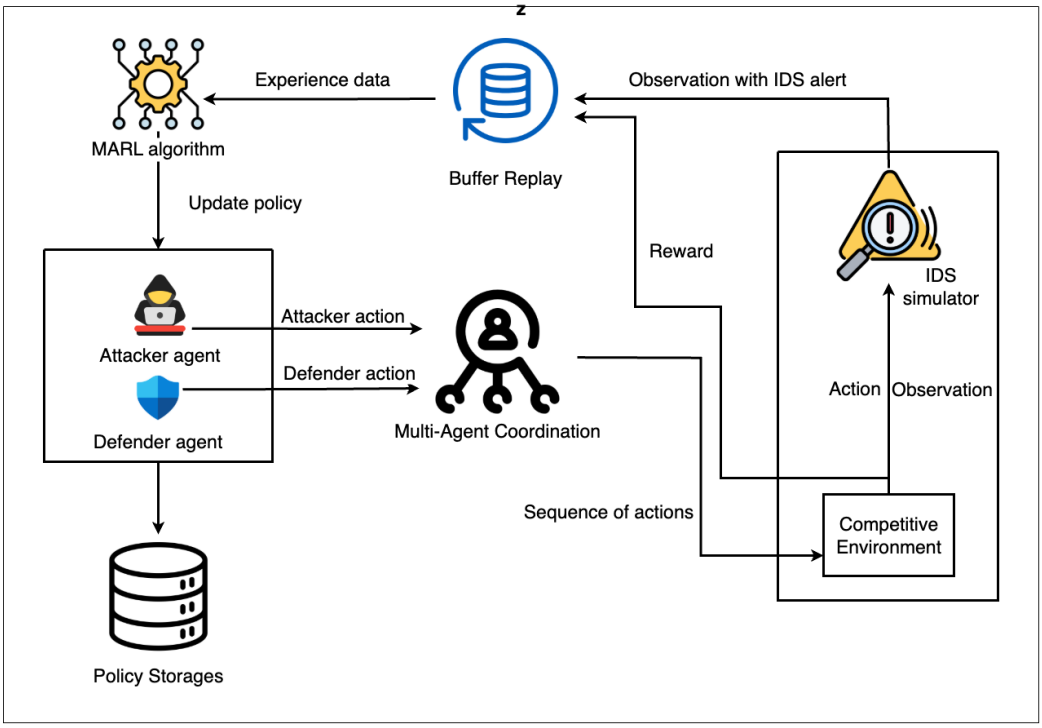


Figure 3.6: Simulation training architecture

The system consists of several key components working in coordination. Initially, attacker and defender policies are initialized to begin training. During action execution, agent actions are collected simultaneously by the Multi-agent coordination component, which determines the execution order for each agent. To prevent logical errors in action sequences, we establish the rule that the defender executes first, followed by the attacker's turn. These actions are then executed sequentially in the competitive environment.

The execution results are subsequently forwarded to the IDS simulator component, which determines whether to generate IDS alerts for the defender and constructs complete observations for both attacker and defender

agents. This information is finally stored in the replay buffer to support training through multi-agent reinforcement learning algorithms. Models are periodically saved into policy pools throughout the training process for later evaluation.

3.3.2 Real Environment

Infrastructure Setup We implemented real environments using virtual machines managed through Virsh (Virtual Shell) for realistic setting evaluation. All virtual machines ran Ubuntu 20.04 LTS, providing consistent experimental platforms. This setup created controlled yet realistic network environments where attack and defense components could be deployed and evaluated.

Attack Implementation Attacker implementations utilized industry-standard penetration testing tools, executing realistic attack scenarios. Network reconnaissance used Nmap, providing detailed host availability and open service information. Actual exploit attempts were executed through the Metasploit Framework, a comprehensive penetration testing platform enabling realistic target system vulnerability exploitation.

Attack methodology follows systematic approaches mirroring real-world adversarial behavior. Modern cyber attacks typically begin with extensive reconnaissance to understand target environments before exploitation. This enables attacker identification of vulnerable services, network topology understanding, and appropriate exploitation vector selection based on discovered systems and configurations.

Our implementation captures realistic attack progression through careful tool selection and configuration. Nmap, widely regarded as the industry standard for network discovery, provides comprehensive scanning capabilities including port enumeration, service detection, and operating system fingerprinting. Such capabilities enable attackers to build accurate target environment pictures and identify potential attack vectors.

The specific attack actions implemented in CMARL-ACD can be categorized into reconnaissance and exploitation phases, as shown in the following tables. The reconnaissance phase, detailed in Table 3.3, encompasses the initial discovery activities that form the foundation of any successful attack campaign.

Reconnaissance activities serve multiple critical attack lifecycle functions. Service scanning provides essential running service and version information, enabling an attacker to identify known vulnerabilities and select appropriate exploits. Operating system detection allows targeted exploitation strategies

Table 3.3: Reconnaissance actions and tools

Action	**Tool**	**Description**
Service Scan	Nmap	Discovers open ports and services.
OS Scan	Nmap	Identifies operating system.
Subnet Scan	Nmap	Map network topology.

leveraging OS-specific vulnerabilities and behaviors. Subnet scanning reveals overall network architecture, helping attackers understand potential target scope and plan lateral movement strategies.

Following successful reconnaissance, attackers transition to exploitation phases, attempting to compromise identified targets using gathered reconnaissance intelligence. This phase requires sophisticated tools and techniques as attackers must exploit vulnerabilities while potentially evading defensive measures. The exploitation actions, presented in Table 3.4, represent common attack vectors encountered in real-world scenarios.

Table 3.4: Exploitation actions and tools

Action	**Tool**	**Description**
e_ssh	Metasploit	SSH service exploitation.
e_ftp	Metasploit	FTP service exploitation.
e_http	Metasploit	HTTP service exploitation.
pe_tomcat	Metasploit	Tomcat privilege escalation.
pe_daclsvc	Metasploit	DACLSVC privilege escalation.

Exploitation actions represent carefully selected common attack vector subsets, providing a realistic representation of attacker capability while maintaining computational efficiency in the simulation environment. SSH exploitation targets commonly exposed enterprise services, often providing initial target system foothold access. FTP exploitation leverages file transfer service vulnerabilities, frequently containing misconfigurations or outdated software versions. HTTP exploitation targets web applications and services, representing prevalent attack vectors in modern cybersecurity incidents.

Privilege escalation actions complete attack chains by enabling the attacker to gain elevated access following initial compromise. Tomcat privilege escalation targets widely deployed Java application servers, while DACLSVC exploitation targets Windows service vulnerabilities. Such privilege escalation techniques prove essential for attackers to achieve their ultimate objectives, whether data exfiltration, system control, or lateral movement.

Tool and technique combinations enabled accurate, sophisticated attack pattern simulation similar to real-world environments, providing realistic testing grounds for our defensive approaches.

Defense Implementation Defender components used production-grade security tools, focusing on two primary defensive strategies: traffic filtering and deception. Such strategies represent fundamental modern cybersecurity operation approaches, providing reactive and proactive defense capabilities.

Traffic filtering through IP blocking provides immediate threat mitigation by preventing further communication from identified malicious sources, serving as a critical first-line defense, enabling rapid detection of threats, and limiting attacker network resource access. IP blocking effectiveness depends heavily on accurate threat detection and timely response, making it well-suited for machine learning-based defensive system integration.

Complementing traffic filtering, decoy deployment creates attractive targets, redirecting attackers from genuine assets while providing early compromise attempt warnings. Deception technologies gain significant cybersecurity attention due to sophisticated attack detection capabilities that might bypass traditional security measures. Creating realistic but non-critical targets enables defenders to observe attacker behavior, gather attack technique intelligence, and buy valuable time for additional protective measures implementation.

Integrating defensive strategies in our environment realistically represents modern security operations. Table 3.5 summarizes specific defensive actions available to our agents and production tools used for implementation.

Table 3.5: Defender actions and tools

Action	**Tool**	**Description**
Deploy Decoy	Docker	Containerized honeypot services.
Block IP	iptables	Firewall rules for IP blocking.

Specific tool choices for defensive actions reflect industry best practices, ensuring realistic performance characteristics. IP blocking actions executed through iptables, the standard Linux firewall framework, allow precise network traffic filtering control. Iptables implementation provides robust packet filtering capabilities with minimal performance overhead, suiting real-time defensive operations. Implementation supports individual IP blocking and subnet-based filtering, enabling defender threat response at various scales, from single compromised hosts to entire malicious networks.

We leveraged Docker containers with specific images tailored to simulate vulnerable services for decoy deployment. Containerization approaches enabled rapid decoy deployment and reconfiguration across networks, providing agile deception capabilities that adapt to changing attack patterns. Docker's lightweight containerization ensures rapid decoy service deployment without significant resource consumption, allowing defenders to create extensive deception networks when needed.

Each decoy service is configured to appear authentic while logging all interaction attempts for analysis. Decoys were designed to closely mimic real services to attract the attacker's attention while providing clear malicious activity indicators. This approach enables defenders to distinguish between legitimate traffic and potential attacks, providing valuable attacker tactic intelligence and enabling informed defensive decisions.

Intrusion Detection System The Intrusion Detection System (IDS) was a critical real-environment implementation component. We deployed Snort, an industry-standard open-source IDS, to monitor network traffic and detect potentially malicious activities. Snort was configured with custom rules identifying common attack patterns, including port scanning, vulnerable service connection attempts, and exploitation signatures. Snort-generated alerts fed into defensive agent observation spaces, providing real-time potential threat information.

In the Operational Noise Scenario, we introduced realistic false positives by configuring Snort with strict detection rules. We implemented rules that generate alerts when three SSH connection attempts occur within 10-second windows. This threshold was intentionally selected, creating ambiguity and potentially representing normal user behavior (connection retries during legitimate access) or potential attacker brute force attempts. This configuration created challenging environments where defenders must distinguish between benign and malicious activities despite sensor uncertainty, mirroring real-world security operation challenges.

The complete evaluation flow in the real environment is illustrated in Figure 3.7. Similar to the simulation environment, attacker and defender actions are collected by the coordination component and executed within the competitive environment. However, instead of simulating action logic against the environment, the competitive environment utilizes tool executors containing essential tools for direct interaction with the real environment, including Nmap, Metasploit, Docker scripts, and iptables. The competitive environment employs these tools to interact with the real environment and receive accurate action results.

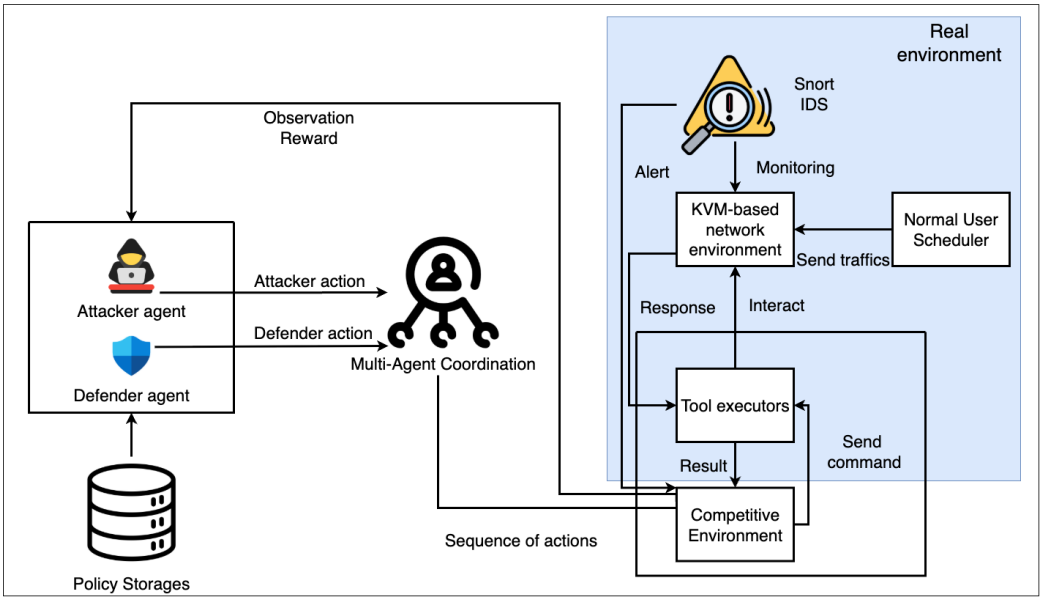


Figure 3.7: Real environment evaluation flow architecture

Additionally, the real environment is continuously monitored by Snort to detect attacker activities, with alert information from the Snort IDS updated in real-time to the environment. Final results are sent back to the agents to inform subsequent actions. During execution, a Normal user scheduler component regularly performs connections and utilizes services on the real network system, creating noise to evaluate the defender's robustness against environmental disturbances.

Chapter 4

Experimental Evaluation

This chapter evaluates our multi-agent reinforcement learning approaches for autonomous cyber defense. Building upon Chapter 3 methodology, we assess IPPO and MAPPO architecture performance across clean baseline environments and operational noise scenarios simulating real-world sensor uncertainty. Evaluations compare the algorithm's defensive capabilities against various attacker profiles, including random and worst-case trained adversaries, measuring effectiveness in simulated and real network environments.

Primary evaluation goals determine which defensive approaches demonstrate superior robustness against sophisticated attackers while maintaining effectiveness under sensor noise and false positives. We analyze agent capabilities in protecting critical network assets, deploying strategic decoys, and accurately identifying malicious traffic. Evaluations provide insights into practical applicability for real-world cybersecurity operations while highlighting learning architecture trade-offs in adversarial settings.

4.1 Experiment Setup

This section details the experimental configurations used to evaluate our multi-agent reinforcement learning approaches. The experimental setup builds upon the methodology outlined in Chapter 3, providing concrete implementation parameters and evaluation protocols.

4.1.1 Experiment Scenarios

Agent training and evaluation are conducted across two contrasting scenarios designed to assess defensive capabilities under different operational conditions:

1. **Baseline Scenario**: A pristine environment featuring perfect observations and noise-free detection, enabling agents to develop optimal

strategies under ideal conditions. IDS rules deliver completely accurate detections without false positives or negatives.

2. **Operational Noise Scenario**: A realistic environment incorporating imperfect observations and intrusion detection system noise. SSH-related IDS rules specifically generate false positives, introducing uncertainty for defenders about whether detected SSH activities represent genuine attacks or legitimate system operations.

4.1.2 Network Environment Configuration

Our experimental evaluation uses a network infrastructure for a compact yet sufficiently complex realistic attack simulation. The network comprises four internal subnets plus one internet-facing subnet (subnet 0), with simulated environments including five hosts distributed across subnets, each featuring distinct operating systems, services, and assigned values. Table 4.1 details this network configuration, while Figure 4.1 illustrates the overall topology and connectivity patterns.

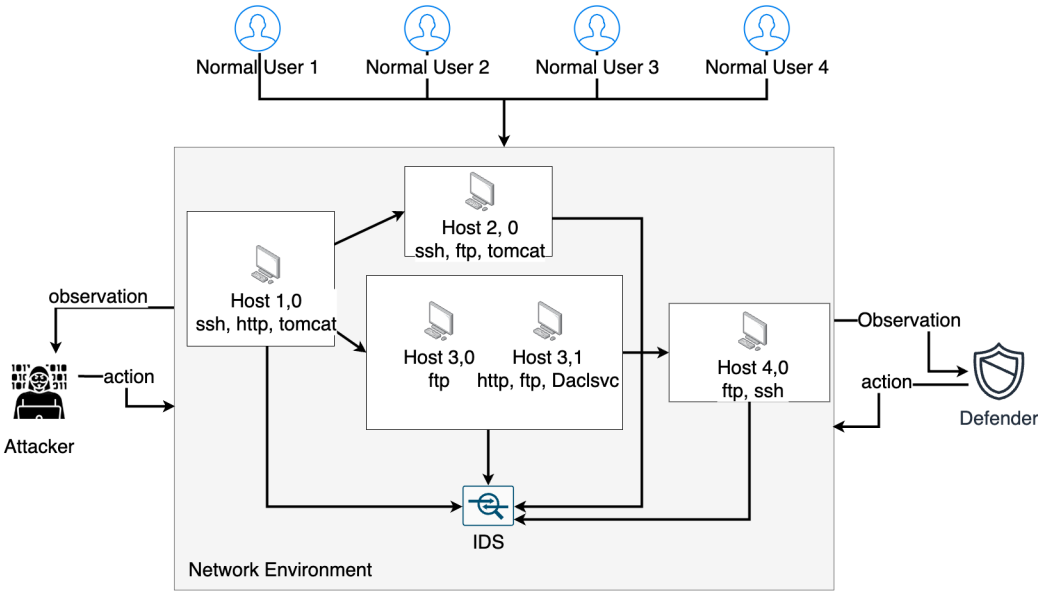


Figure 4.1: Network topology overview

Network structures implement hierarchical connectivity patterns with subnet one connecting to the Internet and remaining subnets arranged in tiered connections. This architecture compels attacker network pivoting, replicating lateral movement patterns characteristic of real-world attacks.

Table 4.1: Network configuration and host details

Host ID	OS		Services	Subnet	Reward
	Simulation	Real			
(1,0)	Linux	Linux	HTTP, SSH, Tomcat	1	10
(2,0)	Linux	Linux	SSH, FTP, Tomcat	2	40
(3,0)	Windows	Linux	FTP	3	70
(3,1)	Windows	Linux	FTP, HTTP, DACLSVC	3	70
(4,0)	Windows	Linux	SSH, FTP	4	100

4.1.3 Attack Tools and Capabilities

The attacker agent can access several exploits and privilege escalation techniques, as detailed in Table 4.2.

Table 4.2: Available attack tools

Tool Type	Name	Cost	Privilege
Exploits	e_ssh	3	user
	e_ftp	1	root
	e_http	2	user
Privilege Escalation	pe_tomcat	1	root
	pe_daclsvc	1	root

For scenario simplification, attack tools target any operating system running corresponding services or processes, irrespective of Linux or Windows distinctions in the simulation environment. In the real environment deployment, all hosts uniformly run Linux operating systems to simplify infrastructure management while preserving the essential attack-defense dynamics. This abstraction enables focus on fundamental attack-defense dynamics rather than OS-specific vulnerability details.

4.1.4 Training and Evaluation Method

Episodes are limited to 150 timesteps, providing adequate time for complex attack sequences while preserving manageable training durations. During training, defender and attacker agent models are saved every 100,000 timesteps, creating comprehensive agent pools at various development stages. This approach enables evaluation of capability evolution throughout training.

For exploitability analysis, each defender model is evaluated against a pool of attackers across 50 episodes, extracting worst-case performance scores. Defenders with the highest worst-case performance demonstrate the greatest robustness and superior ability to handle challenging adversarial scenarios. Following worst-case score determination, exploitability values are calculated by computing relative distances between the best defender's worst-case values and other defender values, quantifying defender vulnerability compared to the most robust defender.

4.2 Training Results and Analysis

4.2.1 Training Results

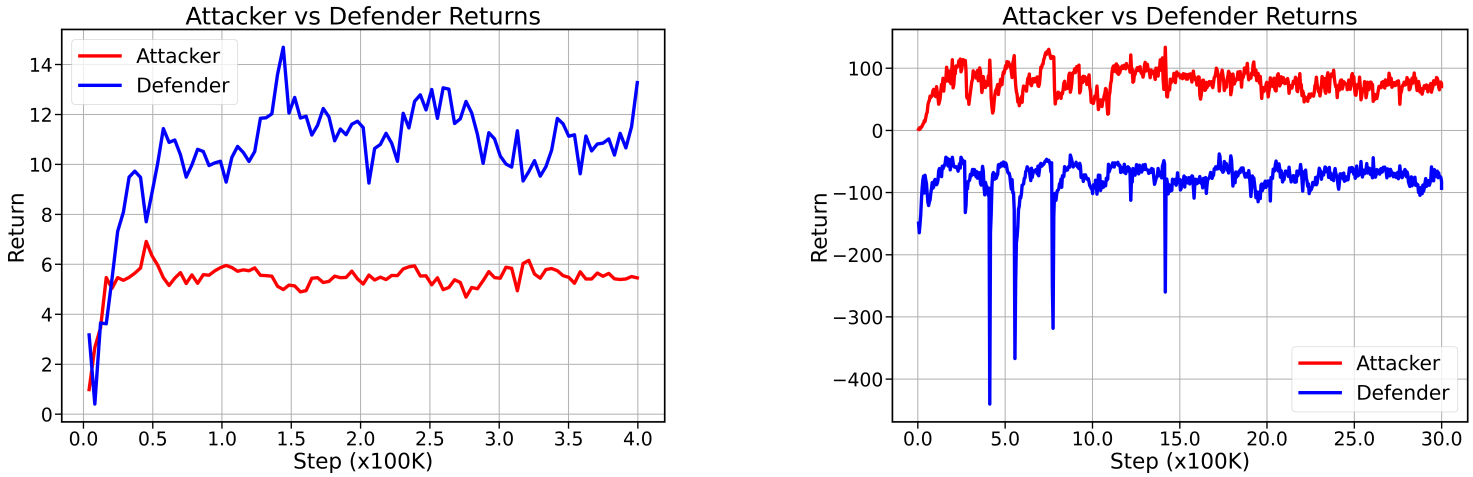
Figures 4.2 and 4.3 present training curves demonstrating agent performance evolution over time, revealing competitive attacker-defender dynamics with shifting advantage periods as agents develop strategies and counter-strategies. Both attacker and defender agents eventually achieve convergence and stabilization, though the required training duration varies by scenario. The Baseline Scenario required 400,000 training timesteps, while the Operational Noise Scenario required 3 million timesteps for proper convergence.

Training curves reveal MAPPO's superior stability compared to IPPO, exhibiting minimal amplitude fluctuations during training. IPPO displays considerably larger training oscillations by contrast. This disparity reflects MAPPO's theoretical stability advantages over IPPO given our specific problem formulation and environmental conditions.

Particularly interesting observations emerged during Operational Noise Scenario training regarding attacker behavioral evolution. Initially, attackers attempted diverse exploits targeting various services. After experiencing repeated defensive blocks, they gradually focused on vulnerabilities associated with false positives, especially SSH services. We configured IDS to generate SSH connection false positives, incorrectly classifying legitimate user SSH activities as malicious attacks. Attackers learned to exploit this confusion by targeting SSH vulnerabilities more frequently, capitalizing on defender uncertainty, and distinguishing legitimate from malicious SSH traffic.

4.2.2 Exploitability and Worst-Case Analysis

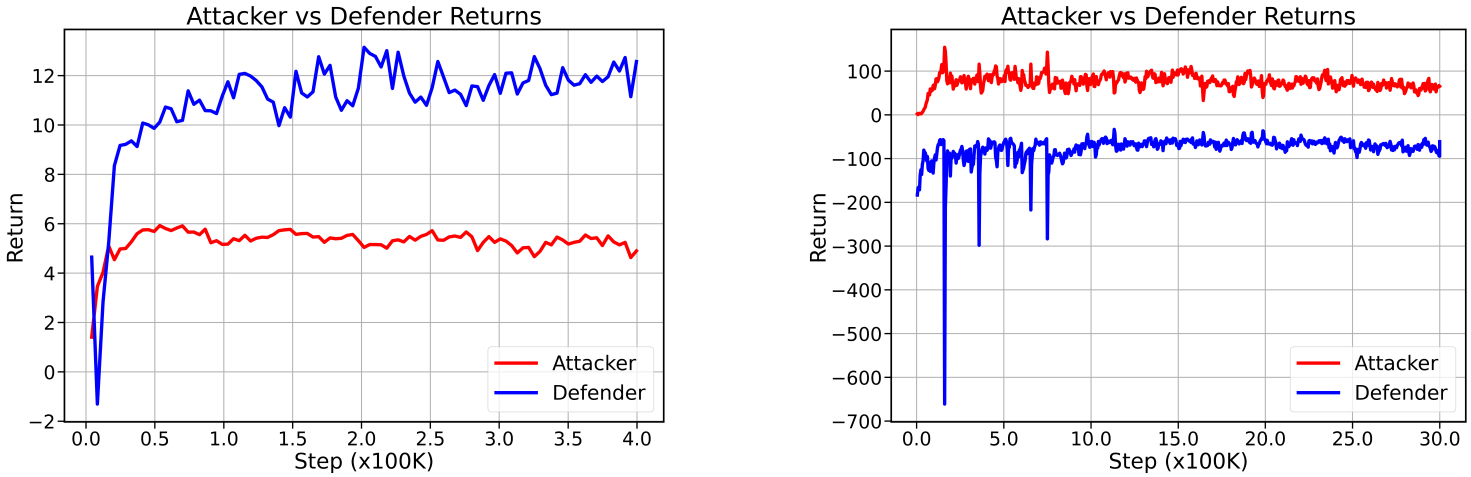
Figures 4.4, 4.5, 4.6, and 4.7 present the worst-case performance and exploitability results for our defensive agents across both scenarios. These figures illustrate how the defensive capabilities evolved throughout the training



(a) Baseline Scenario

(b) Operational Noise Scenario

Figure 4.2: IPPO training curves



(a) Baseline Scenario

(b) Operational Noise Scenario

Figure 4.3: MAPPO training curves

process and provide insight into the robustness of each approach.

Baseline scenario results (Figures 4.4 and 4.5) show worst-case performance progression as training advances, with both IPPO and MAPPO defenders improving over time. Exploitability curves demonstrate defender checkpoint vulnerability compared to the most robust model, with lower values indicating stronger defensive capabilities against worst-case attackers.

Figures 4.6 and 4.7 show identical metrics for the Operational Noise Scenario, revealing sensor uncertainty effects on defensive approach learning and robustness. Results enable IPPO and MAPPO architecture resilience comparisons when facing additional intrusion detection system false positive challenges. MAPPO algorithm values show minimal variation after reaching optimal points, demonstrating approach stability. IPPO exhibits significant fluctuations with higher amplitude, indicating less stability than MAPPO methods. MAPPO stability advantages prove vital in cybersecurity applications where consistent defense performance remains critical for operational deployment. Centralized MAPPO training provides more reliable convergence properties under noisy conditions, potentially suiting real-world security environments where sensor reliability cannot be guaranteed and defensive consistency remains paramount.

4.3 Experiment Results and Analysis

4.3.1 Performance Metrics

We employ specialized metrics explicitly designed for cyber defense evaluation to assess actual model effectiveness in network defense domains. Metrics focus on defensive capabilities against attackers while maintaining normal network operations without disrupting legitimate users. Domain-specific evaluation criteria rather than general reinforcement learning metrics enable a better understanding of CMARL-ACD's performance in operational security environments.

We report three complementary metrics computed per episode and averaged over the evaluation set:

Hosts Compromised (HC) This metric counts the number of unique hosts on which the attacker executed at least one successful exploit. A lower value is better, indicating a more effective defense.

Decoy Interaction Ratio (DIR) This metric measures the effectiveness of the defender's decoys by calculating the proportion of the attacker's exploit actions that were successfully diverted to a decoy instead of a

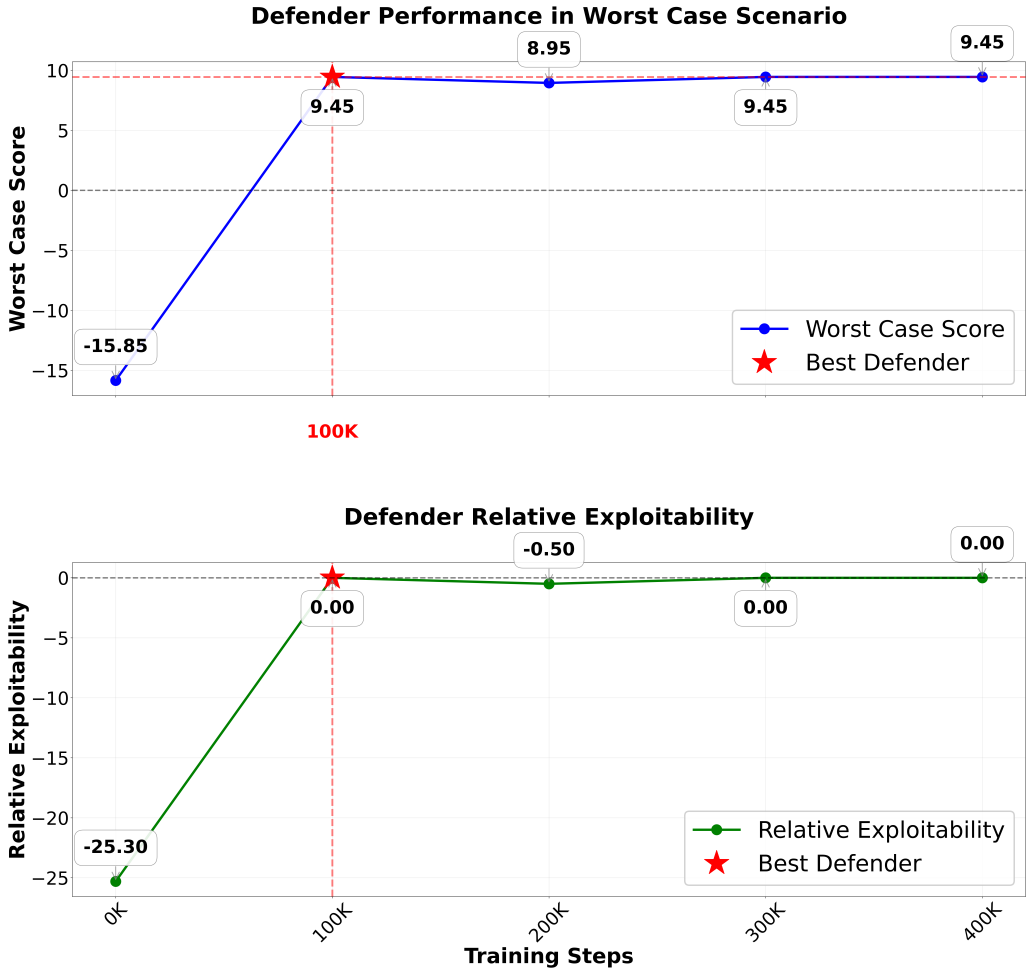


Figure 4.4: Worst case analysis for IPPO - Baseline Scenario

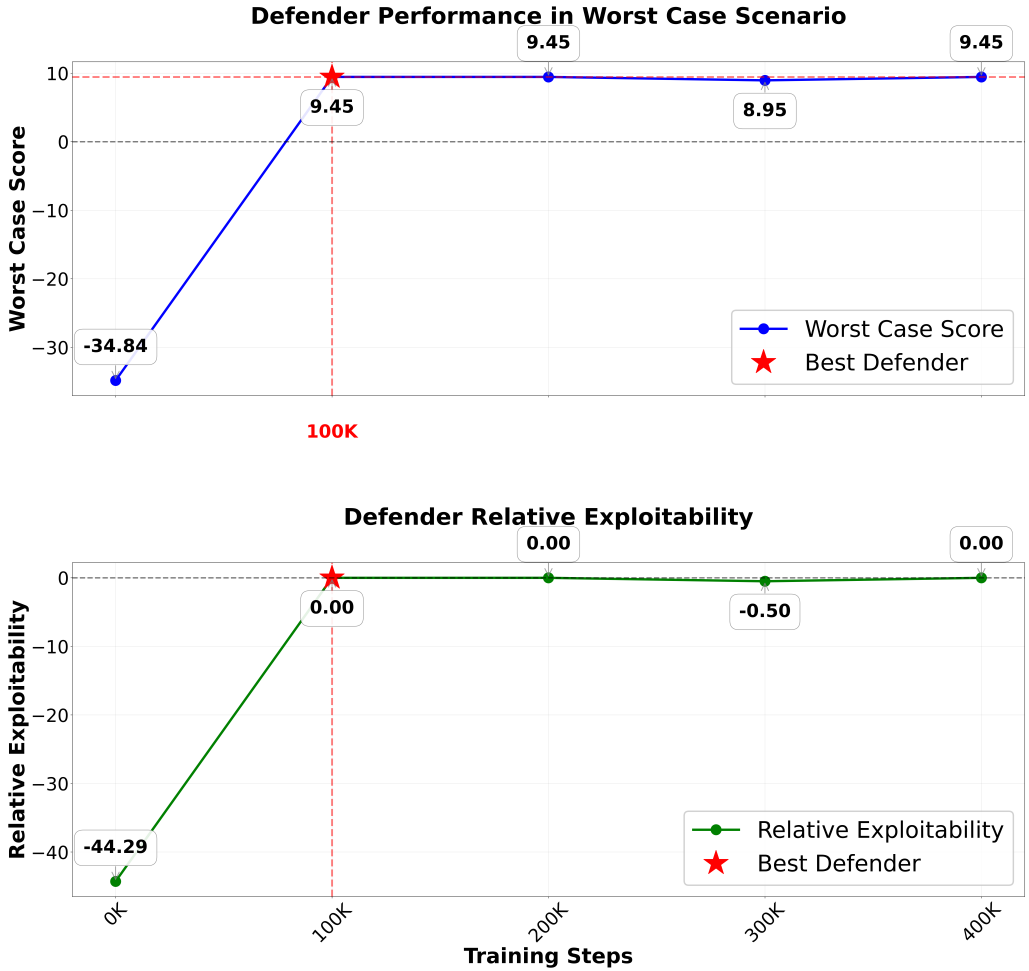


Figure 4.5: Worst case analysis for MAPPO - Baseline Scenario

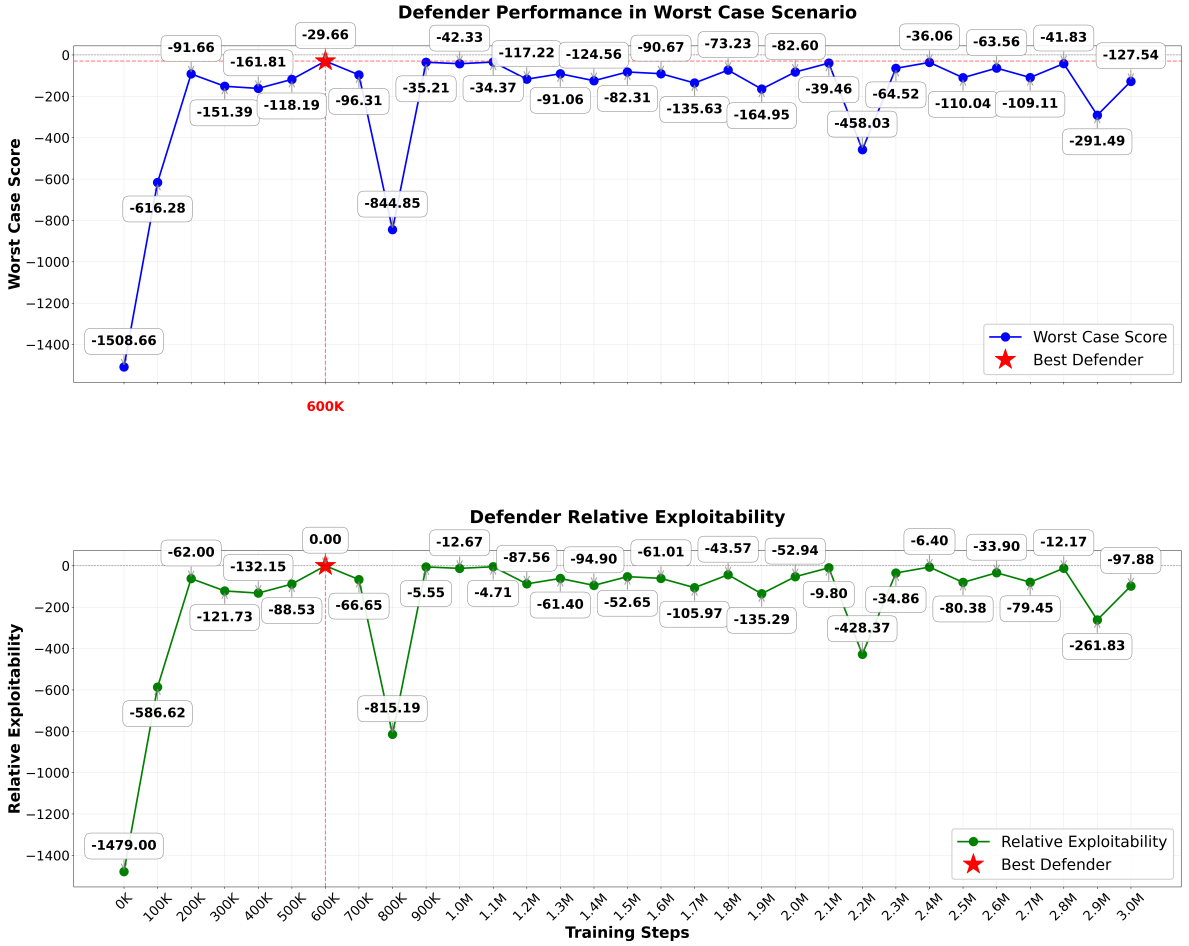


Figure 4.6: Worst case analysis for Operational Noise Scenario - IPPO

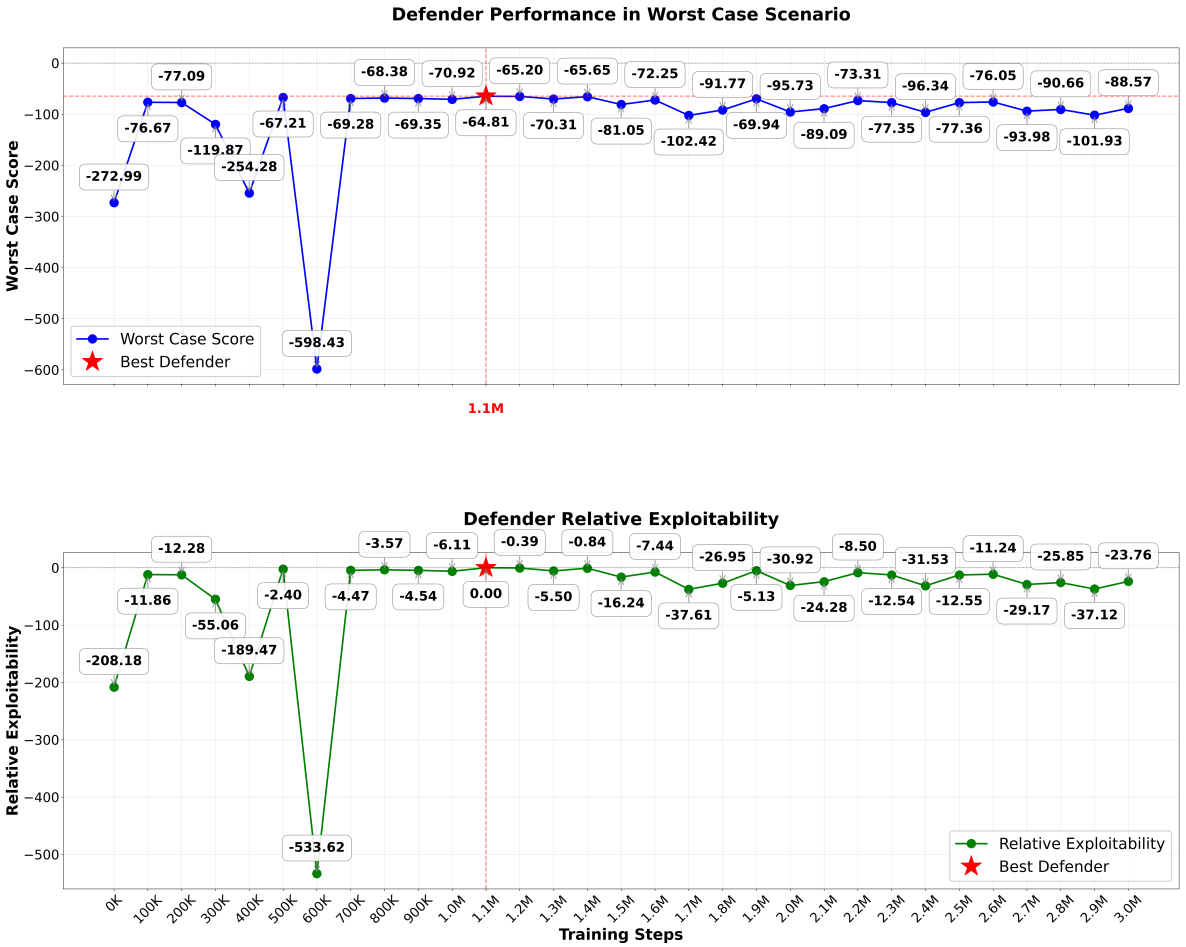


Figure 4.7: Worst case analysis for Operational Noise Scenario - MAPPO

real asset. It is calculated as:

$$\text{DIR} = \frac{\text{attacker exploits on decoys}}{\text{total attacker exploits performed}}$$

True Blocking Rate (TBR) This metric measures the precision of the defender's blocking actions, reflecting how often a block action correctly targets the attacker. It is calculated as:

$$\text{TBR} = \frac{\text{malicious IPs correctly blocked}}{\text{total block actions performed}}$$

These metrics provide comprehensive defensive agent performance views, enabling tracking of system protection effectiveness while maintaining normal system functionality without disrupting legitimate users. True Block Rate measures defensive action precision, ensuring accurate agent distinction between malicious and benign traffic. Hosts Compromised metrics directly quantify the defender's primary objectives of preventing successful attacks. Decoy Interaction Ratio evaluates the effectiveness of the deception strategy, which is designed to redirect attacker efforts from critical assets. This balanced metric set ensures the determination of defensive approach security effectiveness and operational impact.

4.3.2 Experiment Results

The results of our experiments are displayed in Figures 4.8, 4.9 and 4.10, which provide key insights into defensive approach effectiveness across different scenarios and against various attacker models. In addition, the numerical results are presented in Table 4.3.

4.3.2.1 Baseline Scenario Analysis

In clean baseline scenarios (Table 4.3 and Figure 4.8), IPPO and MAPPO defensive agents demonstrated strong capabilities. Without defenders, attackers successfully compromised significant host numbers—3.8 average for random attackers, and precisely one host for trained attackers. Deploying our defensive agents, both MAPPO and IPPO, substantially reduced compromised host numbers against random attackers to 0.64 hosts, representing 83% successful attack reductions, against worst-case trained attackers. Both defensive approaches limited compromise to single hosts, indicating robust defensive capabilities even against sophisticated adversaries.

Deception strategy effectiveness appeared evident in decoy interaction ratios (DIR), reaching 36% against random attackers, as illustrated in

Table 4.3: Performance evaluation overview

Defender	**Attacker**	**Environment**	**Baseline Scenario**			**Operational Noise Scenario**		
			HC↓	**DIR↑**	**TBR↑**	**HC↓**	**DIR↑**	**TBR↑**
None	Random	Simulation	3.802	0%	N/A	3.802	0%	N/A
		Real	3.81	0%	N/A	3.81	0%	N/A
	MAPPO	Simulation	1	0%	N/A	3	0%	N/A
		Real	1	0%	N/A	3	0%	N/A
	IPPO	Simulation	1	0%	N/A	3	0%	N/A
		Real	1	0%	N/A	3	0%	N/A
MAPPO	Random	Simulation	0.64	36%	100%	0.72	36.71%	100%
		Real	0.64	36%	100%	0.94	31.85%	100%
	Worst-case	Simulation	1	0%	100%	3	0%	100%
		Real	1	0%	100%	3	0%	100%
IPPO	Random	Simulation	0.64	36%	100%	0.68	36.99%	100%
		Real	0.64	36%	100%	0.9	26.7%	98.0%
	Worst-case	Simulation	1	0%	100%	2	0%	100%
		Real	1	0%	100%	2	0%	100%

Note: HC = Hosts Compromised (average number), DIR = Decoy Interaction Ratio, TBR = True Blocking Rate; the arrow symbol ↓ indicates that lower is better, and ↑ indicates that higher is better.

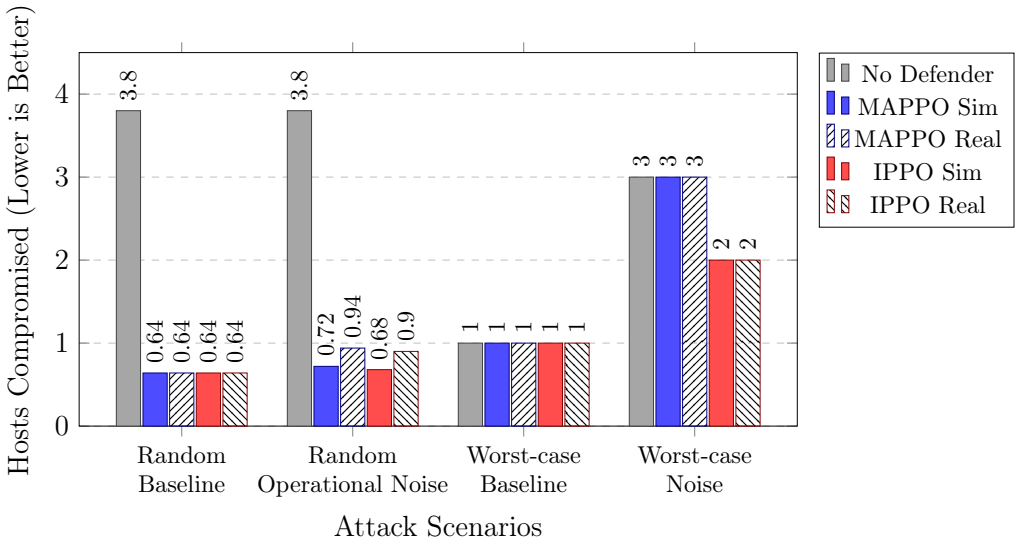


Figure 4.8: Defensive performance across attack scenarios showing robust capabilities with minimal sim-to-real degradation.

Figure 4.9. This shows effective attack attempt redirection, with both IPPO and MAPPO achieving similar deception success rates in baseline scenarios. Both algorithms achieved perfect true block rates (TBR) in this scenario, with 100% malicious traffic identification accuracy, as shown in Figure 4.10. Results confirm that under ideal conditions with accurate sensor data, IPPO and MAPPO can protect critical network assets and correctly identify attack patterns.

4.3.2.2 Operational Noise Scenario Analysis

Operational noise scenarios (Table 4.3 and Figure 4.8) reveal more significant algorithm differences when facing imperfect sensor data. Baseline vulnerability without defenders increased to 3 compromised hosts for trained attackers, highlighting increased scenario challenges. Against random attackers, both MAPPO and IPPO continued performing well, maintaining low compromise rates (0.68-0.94 hosts) and high decoy interaction ratios (26.7-36.99%), as detailed in Figure 4.9. However, IPPO demonstrated superior robustness when facing worst-case trained attackers, limiting compromises to 2 hosts in both simulation and real environments, while MAPPO allowed three host compromises. Regarding detection accuracy, both algorithms maintained high true block rates across scenarios, with IPPO showing slight degradation to 98.0% in real operational noise scenarios while MAPPO maintained

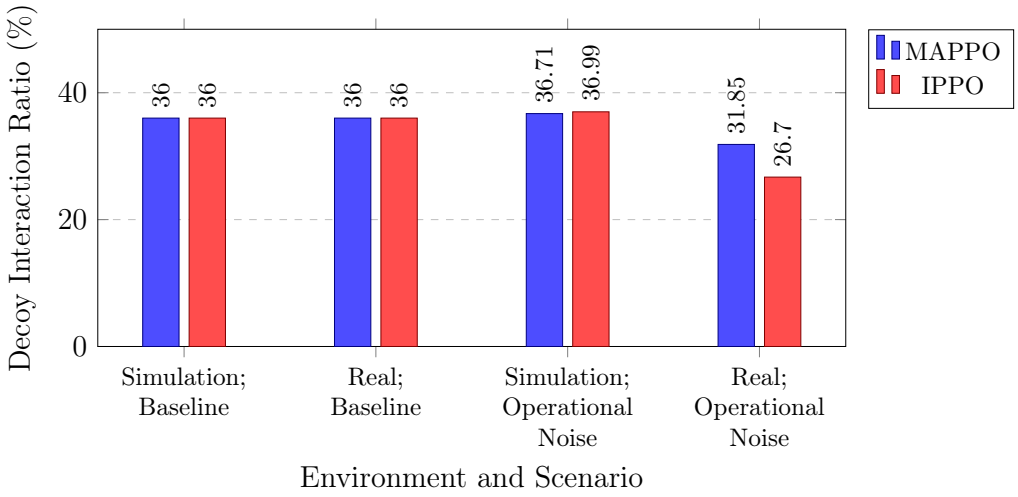


Figure 4.9: Decoy interaction effectiveness showing MAPPO's superior real-environment deception performance.

perfect 100% accuracy, as illustrated in Figure 4.10. These findings suggest both approaches provide robust sensor uncertainty resilience, with IPPO showing slightly better performance against sophisticated attacks despite minor accuracy trade-offs.

4.4 Comparative Analysis

4.4.1 Simulation-to-Real Transfer

Critical evaluation aspects involved determining simulation-trained policy transfer to realistic environments. Both algorithms demonstrated excellent sim-to-real transfer, with minimal performance differences between simulation and real-world results. In baseline scenarios, performance metrics remained identical between simulation and real environments, showing strong consistency. More significant differences emerged in operational noise scenarios, but overall trends remained consistent, validating our simulation approach. Close simulation-real environment result alignment proves particularly noteworthy for baseline scenarios. For instance, hosts compromised (HC) metrics averaged across both algorithms against random attackers show a 0.25 difference between simulation (0.7 average) and real environments (0.92 average) in the Operational Noise Scenario. Decoy interaction ratios (DIR) show some variation, with an average decrease from 36.85% in simulation to 29.28% in real environments under operational noise conditions,

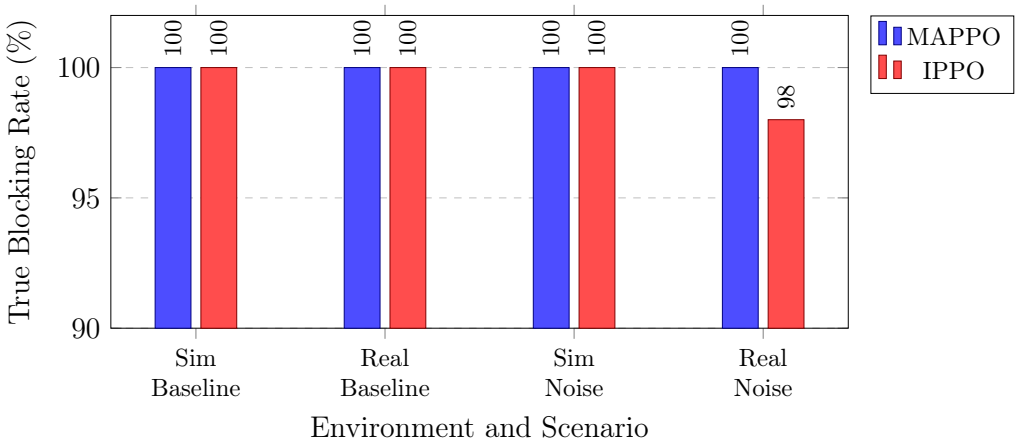


Figure 4.10: True blocking rate showing MAPPO maintains perfect 100% accuracy across all scenarios, while IPPO experiences slight degradation to 98.0% under operational noise in real environments.

indicating some degradation in deception strategy effectiveness when transferred to real environments.

Successful transfer indicates that CMARL-ACD's simulation environment captures essential cyber defense problem dynamics, enabling practical defensive agent training in realistic settings. Performance metric consistency between simulation and real-world deployment suggests defenders trained using our approach can maintain protective capabilities when deployed in operational environments. This proves crucial for practical cybersecurity applications, where training-deployment environment gaps often lead to significant performance degradation. Our results demonstrate that when properly implemented, IPPO and MAPPO approaches can overcome sim-to-real gaps and provide effective defense against various real-world attack strategies.

4.4.2 IPPO and MAPPO Performance Comparison

IPPO and MAPPO comparisons across both scenarios yield several important insights. Both approaches performed identically well in baseline scenarios with perfect observation data, without any performance differences. However, differences became apparent when introducing operational noise. IPPO demonstrated greater robustness against worst-case trained attackers in operational noise scenarios, limiting compromises to 2 hosts compared to MAPPO's 3 hosts, suggesting better uncertainty generalization capabilities. However, MAPPO maintained higher decoy interaction ratios in simulation

under noise conditions (36.71% vs. 36.99%), though both experienced degradation in real environments. Both algorithms maintained high true block rates across scenarios, with IPPO achieving 98.0% accuracy in real operational noise environments while MAPPO maintained perfect 100% accuracy, demonstrating robust precision in identifying malicious traffic. Results partially align with theoretical expectations: decentralized approaches like IPPO show slightly better robustness against sophisticated attackers under noise, while both approaches demonstrate similar coordination capabilities.

Chapter 5

Conclusion

This research developed and evaluated autonomous cyber defense agents using multi-agent reinforcement learning (MARL). We formulated cyber defense challenges as competitive, partially observable Markov Decision Processes (POMDPs) reflecting real-world network security's adversarial and information-asymmetric nature. Primary contributions involved creating simulation environments and training defensive agents using decentralized (IPPO) and centralized (MAPPO) architectures. Agent testing across clean baseline and noisy sensor uncertainty scenarios enabled rigorous robustness and practical applicability assessment.

5.1 Summary of Findings

Experimental evaluation yielded several key findings contributing to autonomous cyber defense. In clean baseline scenarios without sensor noise, IPPO and MAPPO demonstrated high effectiveness, reducing host compromises by approximately 83% while proving capability against worst-case adversaries. Introducing operational noise revealed important performance differences: MAPPO maintained perfect 100% blocking precision across all scenarios, while IPPO experienced slight degradation to 98% in real operational environments, though IPPO demonstrated superior robustness against sophisticated worst-case attackers. Competitive training paradigms successfully fostered arms races, producing intelligent attackers adapting strategies to exploit environmental weaknesses like false positive patterns. Critical outcomes included successful sim-to-real policy transfer, where simulation-trained agents performed consistently in realistic environments, validating our simulation methodology for deployable agent creation.

Research confirms MARL as a viable, robust approach for training autonomous cyber defense agents. Findings highlight clear trade-offs between evaluated centralized and decentralized methods. Centralized MAPPO demonstrated advantages in maintaining perfect blocking precision (100

5.2 Limitations of the Study

Despite promising results, this study presents several limitations, offering future work avenues. Key limitations involve environmental scale and complexity; our network topology simplifies large-scale enterprise networks, with agent scalability untested in such settings.

Additionally, while representative, both attacker and defender action spaces remained discrete, failing to capture the full nuanced range of real-world cyber operations. The study assumed static environments, whereas real networks prove dynamic with emerging devices, patches, and vulnerabilities over time. Benign user traffic models remained simplistic, where more complex behavior models could introduce new defensive agent challenges.

5.3 Directions for Future Research

Building on findings and limitations, we propose several future research directions. Primary directions involve improving scalability through Hierarchical Reinforcement Learning (HRL) techniques for managing larger, more complex network environments. Critical research avenues include agents handling continuous and expanded action spaces, enabling more nuanced defensive maneuvers beyond discrete actions.

Incorporating lifelong or continual learning could address static environment limitations, allowing agent adaptation to new threats and network changes without complete retraining. For practical deployment, exploring human-in-the-loop frameworks and explainable AI (XAI) proves critical for building trust and ensuring human analysts understand and validate agent decisions. Extensions could involve developing advanced deception strategies, such as dynamically managed decoys or honeynets, actively manipulating attacker beliefs and actions.

To facilitate future research and enable reproducibility, we plan to make the CMARL-ACD framework and all experimental code publicly available as open-source software. This will include the extended NASim simulation environment, competitive training implementations, real-world evaluation testbed configurations, and comprehensive documentation. Open-sourcing this work aims to accelerate research progress in autonomous cyber defense and provide the cybersecurity research community with validated tools for developing and evaluating multi-agent reinforcement learning approaches in realistic adversarial environments.

References

[1] L. Yuan, Z. Zhang, L. Li, C. Guan, and Y. Yu, "A survey of progress on cooperative multi-agent reinforcement learning in open environment," 2023. [Online]. Available: https://arxiv.org/abs/2312.01058

[2] G. Mcdonald, L. Li, and R. Al Mallah, "Finding the optimal security policies for autonomous cyber operations with competitive reinforcement learning," IEEE Access, 2024.

[3] W. Alhasan, M. Wannous, A. Abualkishik, M. R. Al Nasar, L. Ali, and H. Al-Zoubi, "An in-depth examination of cybersecurity: Unveiling contemporary trends and recent advancements in the world of cyber threats," in 2024 2nd International Conference on Cyber Resilience (ICCR), 2024, pp. 1–11.

[4] P. Rathod, N. Polemi, M. Lehto, K. Kioskli, J. Wessels, and R. Lugo, "Leveraging the european cybersecurity skills framework(ecsf) in eu innovation projects: Workforce development through skilling, upskilling, and reskilling," in 2024 IEEE Global Engineering Education Conference (EDUCON), 2024, pp. 1–9.

[5] A. Babar, L. Li, A. Taylor, and M. Zulkernine, "Towards autonomous network defense: Reinforcement learning environment for a defense agent," in 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2024, pp. 168–169.

[6] V. Khatri, G. Agarwal, A. K. Gupta, and A. Sanghi, "Machine learning and artificial intelligence in cybersecurity: Innovations and challenges," in 2024 Second International Conference on Advanced Computing & Communication Technologies (ICACCTech), 2024, pp. 732–737.

[7] A. Shaheen, A. Badr, A. Abohendy, H. Alsaadawy, and N. Alsayad, "Reinforcement learning in strategy-based and atari games: A review of google deepminds innovations," 2025. [Online]. Available: https://arxiv.org/abs/2502.10303

[8] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet et al., "Google research football: A novel reinforcement learning environment," in

Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 04, 2020, pp. 4501–4510.

[9] Y. Ran, H. Hu, X. Zhou, and Y. Wen, "Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 645–655.

[10] D. Katsikas, N. Passalis, and A. Tefas, "Bi-directional knowledge transfer for continual deep reinforcement learning in financial trading," in 2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP), 2024, pp. 1–6.

[11] M. Li, M. Ma, L. Wang, Z. Pei, J. Ren, and B. Yang, "Multiagent deep reinforcement learning based incentive mechanism for mobile crowdsensing in intelligent transportation systems," IEEE Systems Journal, vol. 18, no. 1, pp. 527–538, 2024.

[12] V. Mavroudis, G. Palmer, S. Farmer, K. S. Whitehead, D. Foster, A. Price, I. Miles, A. Caron, and S. Pasteris, "Guidelines for applying rl and marl in cybersecurity applications," 2025. [Online]. Available: https://arxiv.org/abs/2503.04262

[13] Y. Tang, J. Sun, H. Wang, J. Deng, L. Tong, and W. Xu, "A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning," Computers & Security, vol. 142, p. 103871, 2024.

[14] Artificial Intelligence School, "Introduction to q-learning," https://artificialintelligenceschool.com/introduction-to-q-learning/, accessed: 2025-01-27.

[15] Wikipedia, "Partially observable markov decision process," https://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process, accessed: 2025-01-27.

[16] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "Plgrim: Hierarchical value learning for large-scale exploration in unknown environments," in Proceedings of the international conference on automated planning and scheduling, vol. 31, 2021, pp. 652–662.

[17] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," IEEE transactions on cybernetics, vol. 50, no. 9, pp. 3826–3839, 2020.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of artificial intelligence research, vol. 4, pp. 237–285, 1996.

[19] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, 2018.

[20] H. Zhang and T. Yu, "Taxonomy of reinforcement learning algorithms," Deep reinforcement learning: Fundamentals, research and applications, pp. 125–133, 2020.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[22] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," Artificial Intelligence Review, vol. 55, no. 2, pp. 895–943, 2022.

[23] J. F. Nash Jr, "Equilibrium points in n-person games," Proceedings of the national academy of sciences, vol. 36, no. 1, pp. 48–49, 1950.

[24] D. Fudenberg and D. K. Levine, The theory of learning in games. MIT press, 1998, vol. 2.

[25] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 8, pp. 3779–3795, 2023.

[26] M. Zhu, Z. Hu, and P. Liu, "Reinforcement learning algorithms for adaptive cyber defense against heartbleed," in Proceedings of the first ACM workshop on moving target defense, 2014, pp. 51–58.

[27] 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). IEEE, 2022, pp. 897–903.

[28] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "Cyborg: A gym for the development of autonomous cyber agents," arXiv preprint arXiv:2108.09118, 2021.

[29] S. Sengupta and S. Kambhampati, "Multi-agent reinforcement learning in bayesian stackelberg markov games for adaptive moving target defense," arXiv preprint arXiv:2007.10457, 2020.

[30] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," Engineering Applications of Artificial Intelligence, vol. 41, pp. 270–284, 2015.

[31] M. Feng and H. Xu, "Deep reinforecement learning based optimal defense for cyber-physical system in presence of unknown cyber-attack," in 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017, pp. 1–8.

[32] J. Schwartz and H. Kurniawatti, "Network attack simulator (NASim)," https://networkattacksimulator.readthedocs.io/, 2019, a lightweight, fast simulation environment for network penetration testing scenarios.