Learning Cyber Defence Tactics from Scratch with Multi-Agent Reinforcement Learning

Jacob Wiebe[1] , **Ranwa Al Mallah**[1] and **Li Li**[2] ,
[1]Royal Military College of Canada
[2]Defence Research and Development Canada
{jacob.wiebe, ranwa.al-mallah}@rmc-cmr.ca, li.li@ecn.forces.gc.ca

Abstract

Recent advancements in deep learning techniques have opened new possibilities for designing solutions for autonomous cyber defence. Teams of intelligent agents in computer network defence roles may reveal promising avenues to safeguard cyber and kinetic assets. In a simulated game environment, agents are evaluated on their ability to jointly mitigate attacker activity in host-based defence scenarios. Defender systems are evaluated against heuristic attackers with the goals of compromising network confidentiality, integrity, and availability. Value-based Independent Learning and Centralized Training Decentralized Execution (CTDE) cooperative Multi-Agent Reinforcement Learning (MARL) methods are compared revealing that both approaches outperform a simple multi-agent heuristic defender. This work demonstrates the ability of cooperative MARL to learn effective cyber defence tactics against varied threats.

1 Introduction

The pace of technological advancements in modern network infrastructure has levied demand for highly skilled and experienced cybersecurity professionals in roles such as penetration testing, threat hunting, and incident response. Meanwhile, machine learning and deep learning have risen as dominant technologies in the space of complex problem-solving. The emerging field of autonomous cyber operations has sought to harness the potential of Reinforcement Learning (RL), deep learning, and multi-agent systems to model the complexity of the cyber battlespace. Autonomous agents operating in cyber defence environments enable rapid tactical-level response to threats given complex observational inputs [Vyas *et al.*, 2023].

Cyber defence analysts typically employ a chain of decisions leading them from the first discovery of a threat to incident response and mitigation. Human experts are the best tool available for this role. However, human ability in cyber defence tasks faces the challenges of, among others, attention allocation, cognitive load, lack of measurable impact, and reaction time [Gutzwiller *et al.*, 2015]. Furthermore, cyber defence is characterized by a large search space of computer network features with which vulnerabilities may be exposed and exploited. To provide coverage over this space, teams of analysts typically coordinate actions. Autonomous RL agents can derive tactical policies in concert to achieve a coordinated effect, mitigating some of the limitations of human actors. This multi-agent approach has yet to be shown in a complex network defence setting.

This work demonstrates the applicability of cooperative Multi-Agent Reinforcement Learning (MARL) for tactical-level decision-making in cyber defence. The results of this work show that cooperative MARL can learn to defend a simulated network environment against three heuristic attack patterns with an emphasis on host-based identification and mitigation of lateral movement.

There are numerous advantages to a cooperative MARL approach over single-agent RL for ACD: (1) input-output spaces can be constrained, avoiding the curse of dimensionality while enabling a high resolution representation of the environment, (2) agents can learn specialized roles relating to a specific function or area of a network, and (3) segregating agents into local areas of control can allow for improved robustness (e.g., if a particular agent is compromised an attacker cannot leak observations from other agents/network segments).

This work makes the following contributions:

1. To the best of our knowledge, CyMARL (Cyber MARL) is the first cooperative MARL training environment for enterprise network defence tasks. It extends the CybORG simulator with additional game types, actions, and network topologies and it provides a PyMARL environment interface with which it is possible to train tens of open-source algorithms.
2. A comparison of cooperative MARL approaches to learn independent and centrally-learned policies in a ACD context.
3. A demonstration of the adaptability of cooperative MARL training when encountering multiple attacker types and action sets.

The remainder of this paper is organized as follows. Section 2 reviews related work in autonomous and multi-agent applications for cyber defence and advancements in cooperative MARL. Section 3 discusses the background of RL, cooperative MARL, and autonomous cyber defence. Section 4

describes the evaluation of the two cooperative MARL approaches tested. Section 6 discusses the results. Section 7 suggests future work and concludes.

2 Related Work

RL for specific cybersecurity tasks has been studied in the context of DDoS protection [Xu *et al.*, 2007], anomaly-based intrusion detection [Utic and Ramachandran, 2022], and penetration testing [Chowdhary *et al.*, 2020; Hu *et al.*, 2020a], among other specific use cases [Nguyen and Reddi, 2019]. RL has been applied to a variety of decision-making tasks relating to cyber defence [Wang *et al.*, 2022]. Although ACD techniques have been evaluated using multiple game environments and solving methods [Vyas *et al.*, 2023], to the best of our knowledge, this work is the first to consider multi-agent coordination of tactical decision-making using state-of-the-art RL techniques.

2.1 ACD Training Environments

Game environments are commonly modelled as graph-based abstractions of computer networks. Simple defensive decision-making games can be learned using tabular RL methods [Hu *et al.*, 2020b; Applebaum *et al.*, 2022]. In more complex game environments, deep RL is preferred for its generalizability and representational capacity. Methods such as DDQN [Van Hasselt *et al.*, 2016], A3C [Mnih *et al.*, 2016], and PPO[Schulman *et al.*, 2017] have been shown to learn to minimize attacker propagation in graph-based network games modelled as Partially-Observable Markov Decision Processes (POMDP) [Han *et al.*, 2020; Nyberg and Johnson, 2023].

More realistic behaviours can be trained using more detailed simulated data. Simulators such as CybORG [Standen *et al.*, 2021], CyberBattleSim [Team, 2021], and FARLAND [Molina-Markham *et al.*, 2021] emphasize host-based features from which agents learn. Host-based simulations improve task realism by providing more possible states, represented by host processes, vulnerabilities, sessions, and opportunities for Red and Blue agents to interact with these features. In contrast, Yawning Titan [Andrew *et al.*, 2022] enables greater detail for agents to observe and affect network traffic in simulation.

Foley *et al.* demonstrate how a hierarchical PPO algorithm can learn to adapt to different attacker types to defend a high value server from a heuristic attacker in a simulated environment [2022], winning CAGE Challenge 2 [2022]. Building from this foundation using the CybORG simulator, Wolk *et al.* analyze the generalizability of various RL methods used in CAGE Challenge 2 [Wolk *et al.*, 2022].

Emulation has been used to bridge the sim-to-real gap between RL systems that can be trained in defence games and real network operations [Li *et al.*, 2023; Molina-Markham *et al.*, 2021]. Emulated environments are impractical for training agents in many research settings due to higher requirements for compute and clock-time than comparable simulations, instead providing a valuable validation component for agents trained in simulation. Red versus Blue asymmetric attack-defence games have used self-play to train defenders against RL-trained attackers [Gabirondo-López *et al.*, 2021; Kunz *et al.*, 2023]. RL self-play and adversarial attacks on RL cyber defenders [Standen *et al.*, 2023] are outside the scope of this research.

2.2 Multi-Agent Systems for ACD

Communication between heuristic defender agents has been shown to improve their ability to protect a simulated network from DDoS attacks [Kotenko, 2007]. Heuristic defender agents programmed to communicate filter configurations between teams had improved reaction time and effectiveness than those that did not share information. MARL systems have defended a simulated network against DDoS attacks by selectively throttling network traffic without the use of communication [Malialis and Kudenko, 2015]. MARL systems have successfully performed anomaly-based intrusion detection when framed as a classification task [Servin and Kudenko, 2008; Shi and He, 2021].

One of the most commonly used cooperative MARL environments for research is the StarCraft Multi-Agent Challenge (SMAC) [Samvelyan *et al.*, 2019]. SMAC provides a variety of scenarios ranging from easy to very hard in which virtual units learn to defeat each other in a battle video game. A comparative study of cooperative MARL algorithms across a variety of environments, including SMAC, showed that CTDE methods generally perform better than independent learning [Papoudakis *et al.*, 2020]. SMAC tasks are assumed to have transferability for general tactical-level decision-making as a result of agents needing to target actions in a coordinated way toward individual enemies in order to achieve success. The cyber defence problem is modelled similarly, using actions to selectively affect host characteristics.

In practice, the QMIX MARL algorithm [Rashid *et al.*, 2018] has outperformed state-of-the-art value-based and policy-based methods at a range of tasks [Papoudakis *et al.*, 2020; Samvelyan *et al.*, 2019]. Most notably, QMIX has been shown, with the help of specific implementation tricks, to discover optimal or near-optimal policies on all SMAC [Samvelyan *et al.*, 2019] scenarios [Hu *et al.*, 2021].

3 Background

A useful ACD agent must be able to interpret its environment and derive a logical chain of decisions. The RL learner, an autonomous agent, takes sequential actions given inputs from its environment. The agent's goal is to choose actions that maximize its cumulative reward, referred to as *return*, received from the environment. To learn which series of actions will produce favourable results, an agent must explore the environment, receive rewards, and update its understanding. This process occurs iteratively, allowing the agent to refine its knowledge over many steps and episodes of play. To provide an analysis of how learning architectures compare when used to learn cyber defence tasks, this work constrains the approaches studied to model-free, value-based algorithms.

3.1 The RL Framework

In a POMDP, an agent will take action a_t causing the environment to step to the next state s_{t+1} based on a transition probability p, where $p(s_{t+1}|s_t, a_t)$ is the transition function.

The environment generates two outputs from the state transition: the reward $r(s, a)$ as a function of the state and action at time $t-1$, and the observation $o(s)$ as a function of the state at time t. The observation is some subset of state information determined by the observation function. A *game* in this context generalizes the POMDP and refers to an environment with a consistent set of rules in which one or more agents interact.

An agent's *policy* $\pi(a|o)$ is a mapping of actions to the observation space. The policy, therefore, determines a series of actions for an agent to take. An agent adjusts its policy over many episodes of a game to optimize for its cumulative reward in an episode, its *return* [Sutton and Barto, 2018].

A value-based agent will predict the value of states or state-action pairs using its value function and decide on an action that will maximize its expected future return $\pi(o) = \arg\max_{a \in A} Q(o, a)$. The value function is calculated from the expected discounted future return:

$$Q(o,a) = \alpha[r + \gamma \max_{a_{t+1}} Q(o_{t+1}, a_{t+1})] + (1-\alpha)Q(o,a) \quad (1)$$

Where α is the learning rate and the discount rate ($\gamma \in (0, 1)$) weights rewards less as the agent looks to further k states. To encourage exploration, rather than to continue to revisit known high-value states, an ϵ-greedy strategy is commonly employed which sets a probability ϵ that an agent will randomly select an action rather than taking the action with the highest expected reward.

With deep value-based methods, sampling from experience replay [Lin, 1992] often produces more favourable results, as demonstrated by Mnih *et. Al* with Deep Q-Networks [2013]. A replay buffer can improve sample efficiency by exposing the agent to more learning experience without generating each sample through actions. Recurrent Neural Networks (RNN) allow RL models to learn long-term dependencies between data samples which is useful for accurate predictions in partially observable environments [Bakker, 2001].

3.2 Multi-Agent Reinforcement Learning

The cooperative multi-agent game builds upon the POMDP defined previously by allowing multiple agents to take actions and receive rewards simultaneously within a single timestep [Littman, 1994; Buşoniu *et al.*, 2010]. Although each agent, denoted by i, receives individual observations o^i from the environment, a joint reward r is determined based on the state and actions of all agents. Agents collectively seek to optimize the joint reward. Figure 1 provides a cooperative MARL representation of agent-environment interaction. The game's global state depends on the vector of actions $\mathbf{a}_t = \{a_t^1, ..., a_t^n\}$, where $i = 1, 2, ...n$ for n agents. At each timestep, the environment will output a joint reward as a scalar and an observation vector $\mathbf{o}_{t+1} = \{o_{t+1}^1, ..., o_{t+1}^n\}$ that includes individual observations for each agent.

Simultaneous actions in a cooperative MARL setting allow for greater exploration. Cooperative MARL also has the advantage of dividing large tasks into manageable goals, thus handling greater task complexity. However, since the game's current state depends on the joint actions of all agents, an agent's actions can affect other agents' observations. Since both agents are learning and updating their policies, this interaction creates *multi-agent*-non-stationarity in the environment; the optimal policy is a moving target. In a single-agent game, the fixed probability of a stochastic transition function can be incorporated into the learned policy, for example, with the use of a replay buffer. However, in the multi-agent case, actions of "external agents" affect observations in a way that cannot be explained by changes in the observing agent's policy [Lowe *et al.*, 2017]. It can therefore be challenging for an agent to learn a stable policy when the actions of its peers are silently affecting its reward and observations.

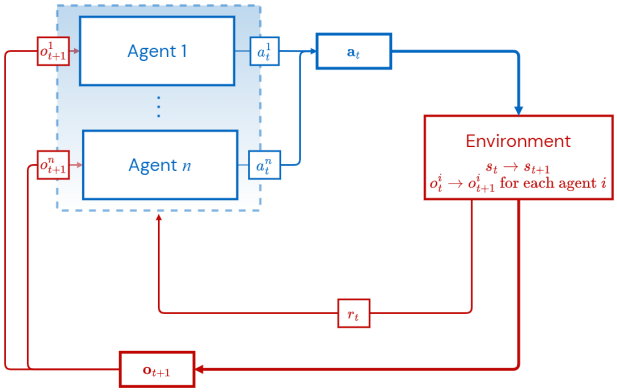

Figure 1: Multi-Agent RL Agent-Environment Framework.

Independent Q-Learning

A naive approach to cooperative multi-agent learning systems is to decentralize prediction and control, referred to as *independent learning*. This approach employs self-contained RL agents that act out separate policies based on independent observations while seeking to maximize a joint reward. Each agent is responsible for learning a policy from their individual observations. A decentralized approach allows for greater scalability, at the cost of high variance due to the unmitigated non-stationarity. Independent Q-Learning (IQL), as implemented in this work, is equivalent to training independent DQN-style agents[1] to learn to take simultaneous actions to maximize their joint reward.

QMIX

QMIX [Rashid *et al.*, 2018] is a Centralized Training with Decentralized Execution (CTDE) [Lowe *et al.*, 2017] algorithm that uses a central mixing architecture to perform *value decomposition* [Sunehag *et al.*, 2017]. With CTDE, semi-independent agents follow separate policies and receive updates periodically from a central learner. The central learner trains on information from all agents and provides a learning update, allowing each agent's policy to condition on the policies of their peers, mitigating the non-stationarity problem. The foundational difference between QMIX and IQL

[1]This work utilizes an implementation of IQL from PyMARL2 [Hu *et al.*, 2021] in which each DQN-style agent uses an RNN in addition to other implementation tricks to optimize for cooperative play.

implementations in this work is that QMIX performs learning updates centrally.

3.3 Autonomous Cyber Defence

ACD does not seek to replace tools that are successful in the field, such as anomaly-based intrusion detection. It instead aims to build a tactical-level decision-making framework to integrate with existing technologies. In this research, the decision space is modelled as a simulation on top of a tool-based abstraction of the network state. An RL simulation for ACD must model the elements of the environment to the level of detail that will allow the agent to learn tactical-level action chains when presented with a series of alerts about the underlying network status. A multi-agent framework further allows for the decentralization of the decision-making processes while maintaining an input-output space that is constrained to a level that can be learned effectively by current RL methods.

The high-level actions of the defender in this game represent the choices for a defender to selectively understand or act against the threat. The tactical decision-making competency of agents then can be inferred from their ability to minimize the impact of the attack. Many thousands or millions of iterations are often required for RL to learn reasonable policies. This problem is exasperated by a lower sample efficiency of partially-observable environments [Papadimitriou and Tsitsiklis, 1987]. By simulating ACD, RL systems can be trained over many iterations in an abstraction of the underlying computer network environment. Simulation requires less compute than the alternative emulated approach while still offering enough complexity to challenge current state-of-the-art designs.

Cyber Operations Research Gym (CybORG) is a framework that provides a simulated cyber operations environment for training and evaluating RL agents [Standen *et al.*, 2021]. CybORG simulates information technology systems related to network security that a cybersecurity professional may use in network defence or penetration testing. The simulated environment presented in this work, CyMARL, adapts CybORG to create a multi-agent host-based monitoring game for training tactical cyber defence. CyMARL includes a PyMARL[2] environment allowing for integration with the tens of open-source cooperative MARL algorithms built on the PyMARL framework.

4 Experimental Design

To demonstrate the applicability of cooperative MARL for tactical-level decision-making for ACD, three game scenarios are evaluated against three attacker types. CAGE Challenge 2 [CAGE, 2022] is a central contributor to many aspects of this experiment, including the attacker models, network topology, and agent actions. CyMARL extends the complexity of CAGE Challenge 2 by employing new scenarios (with diverse objectives, attacker behaviours, and reward functions), modified observation and action spaces, and a cooperative multi-agent framework.

4.1 Simulated Network Topology

The simulated game environment is composed of nine hosts, five in a user subnet and four in an operational subnet. Hosts are a mix of Linux and Windows, with unique exploits for each OS. Hosts simulate processes, users, active sessions, and network interfaces. Rewards are calculated from the importance score of hosts. Hosts in the user subnet have a score of 0.1, operational subnet hosts, 1, and the operational server (located in the operational subnet), 10. Each host in a *compromised* state incurs a negative reward for the defender team equal to its importance score.

A heuristic attacker agent begins each episode with an initial foothold on a user host and moves between hosts by scanning for and exploiting vulnerable services. The attacker can move laterally to a target host if one of its sessions' hosts has visibility of the target host through its network interface. The attacker's actions follow a hierarchy: it must first discover a host, then a port, before attempting to move laterally on the network using an exploit. Escalating session privilege can then be performed. The attacker's session cannot be removed from the initially compromised host allowing for episodic gameplay where both sides have opportunities to influence the reward. RL defender agents are segregated by subnet, one agent responsible for the user subnet and the other controlling the operational subnet. An example of the agent-network interface is shown in Figure 2.

4.2 Attack Patterns

A real-world attacker is often not solely motivated to establish communication channels on a network. The heuristic attacker's behaviour in this game is differentiated by its goal, which can be *confidentiality*, *integrity*, or *availability*. In each case, the attacker takes different actions upon establishing a session on a host, though the heuristic for lateral movement is the same across all scenarios. In the confidentiality scenario, the attacker's intent is to spy on privileged information within the network but not take actions to modify it directly. The confidentiality attacker achieves "compromise" for each host it is connected to at each timestep. The integrity attacker seeks to modify files stored on a host. At each timestep that modified data exists, a host is considered compromised. The availability attacker will spawn a Denial of Service (DoS) malware process that represents a consumption of CPU cycles, affecting user availability. Similarly, each host in this scenario with a malware process running is compromised. This does not affect the defender's ability to monitor the victim host. Although the attacker's behaviour is a simplification of real attack patterns, the difference in objective affects how the threat appears to the defender and which actions it should take.

To compromise a host in the integrity and availability scenarios, an additional action must be taken once a session is established on a victim host using the `exploit` action. The `tamper` action is taken by the integrity attacker to simulate the modification of important files. The `deny` action is taken

[2]PyMARL is an open-source MARL research project that allows algorithms to be built from existing deep RL components such as agents and trainers. It is included as part of the SMAC environment and paper [Samvelyan *et al.*, 2019]. The repository for PyMARL can be found at https://github.com/oxwhirl/pymarl.

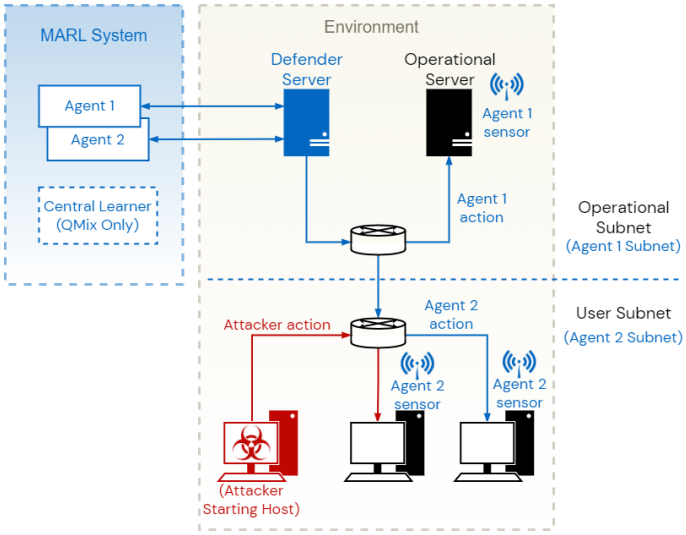

Figure 2: Simple network diagram with defender sensors and targets.

by the availability attacker to simulate the creation of a DoS process. The two-step scoring process in the integrity and availability scenarios decouples the observation of attacker presence on a host from the signal of a host compromise via the negative reward.

The cumulative reward for each strategy serves to model the risk to the network while malicious artifacts exist on network hosts. As the number of timesteps increases, there is a greater probability that the attacker will have collected privileged information or disrupted normal processes on the network. Likewise, the severity of the risk is proportional to the importance of the host that becomes compromised.

4.3 Observations

Host-based monitoring systems are a common and effecti tool for cyber defence operations, particularly in situations where an attacker has made an initial breach and is attempting to gain Command and Control (C2) channels within a network. In CyMARL, each defender agent's observations are modelled as an abstraction of information that may be provided from a host-based monitoring service. The defender receives a set of six flags at each turn for each host (denoted by the variable x) within its area of control:

1. Host x was scanned last turn.
2. An exploit was run on a Host x service last turn.
3. There are no malicious sessions on Host x.
4. Host x status is unknown.
5. A malicious session on Host x exists with user privileges.
6. A malicious session on Host x exists with root/admin privileges.

The flags are one-hot encoded into a vector that is assigned to an agent ID for interface with PyMARL. These observations do not specifically alert of malicious files or processes so the defender must learn patterns between the information encoded within its observation space and its rewards.

4.4 Defender Actions

To mitigate attacker activity, the defender agents select actions to monitor, remove suspicious processes or files, restore hosts, or gather file information. `monitor` returns the observation vector and checks all hosts within an agent's vision for new processes that are then added to a suspicious process list. An agent can then choose to `remove` suspected malicious processes from a specific host. As a more aggressive action, an agent can re-image a host using `restore`. Restoring is the only way to remove an attacker that has escalated its privilege on a host. It will also remove any malicious processes or files. However, the `restore` action has an associated penalty that scales with the importance score of the target host.

In the integrity scenario, agents have additional actions to manage the specific threat type. Agents can take an `analyze` action to determine the security characteristics of files on a host, allowing it to identify any malicious files. To remove the discovered files, the `data_repair`, is required. In the availability scenario, the `remove` action can be used to target DoS processes and simulated TCP sessions. In the second set of trials, the `misinform` action is introduced which generates a service on a host that can be misinterpreted by an attacker as vulnerable. The attacker may choose to exploit this process which will always fail. A defender agent may choose to perform any action on any of its connected hosts. However, some actions are invalid (e.g., if a host is targetted with the remove action but does not have a malicious process running) and are penalized with a reward of -0.1.

4.5 Heuristic Defender

To validate the performance of MARL systems, a heuristic multi-agent defender is used as a benchmark for each game scenario. The heuristic defender's design assumes it has access to a tailored expert system that provides a precise signal of malicious process creation. It relies on selected information from the true state and does not utilize the observation

space of the MARL systems. In the confidentiality scenarios, this policy monitors all hosts within each agent's subnet, and restores hosts indicated by the process creation alert. The `restore` action was chosen because it scored higher in testing than using combinations of other actions. In the availability scenario, detection occurs due to a DoS process being created, not an attacker session. In the integrity scenarios, the heuristic defender will attempt to remove malware on a host that it has discovered to have a tampered file with a 50% probability and restore otherwise. In trials using the `misinform` action, the defender system creates a decoy process on a random host for the first five turns and then proceeds with its standard policy for the remainder of the episode. Since using `monitor` to detect and `restore` to re-image affected hosts takes two turns, the attacker has the opportunity to stay one step ahead if it continues to exploit new hosts. Although the heuristic defender has perfect game information, its policy is restrictive enough to allow the attacker opportunities to overcome its defences.

4.6 Evaluation Metrics

For each experiment, two algorithms are evaluated: IQL and QMIX, implementing fully-decentralized and CTDE architectures, respectively. Agents do not directly share information with each other, but in the case of QMIX, a joint policy representation is learned centrally before being decomposed into separate behaviour policies. The architecture of cooperative MARL systems has been shown to significantly affect the learning ability of models at a range of tasks [Papoudakis *et al.*, 2020; Rashid *et al.*, 2018]. Learning ability is the capacity of an RL model to generate a policy from its initial randomized state. A trained model that achieves a higher evaluation score (i.e., mean return) at a task has a higher learning ability than a lower-performing model given the same training opportunity.

MARL systems are trained for two million timesteps. A timestep is represented in the game as a turn in which each agent takes one action. Each trained model is evaluated over 1000 timesteps of play without learning. The evaluation score is the average of the mean return of five evaluation runs using separately trained static policies for each trial. Evaluation scores are compared using percent difference. The variance of models is expressed in the standard deviation of the mean score. Learning speed is an implicit evaluation criterion to mitigate the impracticality of long training times. Although there may be greater performance gains outside of the bounds of the training time for the model, training time is constrained to conserve computational resources. This allows for greater ease of reproducibility of these results and a reasonable expectation of performance given a fixed amount of model experience.

5 Experiments

Two architectures are trained and compared at controlling two defender agents in three scenario types each with different attacker behaviour (confidentiality, integrity, and availability) in two sets of experiments. The first set uses the standard action set described in Section 4 and the second adds the `misinform` action to each scenario. The addition of a proactive action to the set of reactive actions sets the misinform experiment apart and allows for an evaluation of how the two MARL architectures perform under varied conditions.

To set initial conditions, a selection of hyperparameters was varied in a grid search in which MARL systems were trained to 500,000 timesteps at the confidentiality scenario evaluating four parameters: batch size (128, 256), buffer size (5,000, 10,000), learning rate (0.005, 0.01), and TD-λ trace decay (None, 0.6). For each set of hyperparameter values, three random seeds were trained to provide a representative result. The simplest scenario is confidentiality scenario due to the attacker's exploit and the reward for compromise occurring simultaneously. It is assumed that these hyperparameter values transfer reasonably well to the other scenarios. The hyperparameter values chosen were based on initial trials with the environment and the findings in Hu *et al.* and Hessel *et al.* that evaluated the effect of RL implementation designs in QMIX and DQN, respectively [Hu *et al.*, 2021; Hessel *et al.*, 2018].

The training curves of IQL and QMIX are shown in Figure 3 for the three baseline scenarios and Figure 4 for the misinform scenarios. The shaded area of each curve is the standard deviation of the average training return. The dashed line is the mean heuristic defender score for each scenario. Both IQL and QMIX outperform the heuristic multi-agent defender model in all scenarios. QMIX has a slight learning speed advantage in the confidentiality and integrity scenarios but otherwise, both architectures tend towards similar return scores. Table 1 presents the evaluation scores of IQL, QMIX, and the multi-agent heuristic defender at each scenario.

The option to generate decoy processes generally improved the performance of both architectures with a 19.8% and 8.3% average improvement in score for IQL and QMIX, respectively. However, QMIX had reduced performance in the confidentiality scenario and a negligible difference in the integrity scenario. The heuristic model using the `misinform` action at the start of each episode performed 2.7% better on average.

6 Discussion

The availability and integrity scenario types employ an attacker that must perform an additional action upon establishing a session to score points. The attacker in these scenario types is therefore slower and will tend to score less in a fixed-timestep game. On average the MARL systems trained at the integrity and availability scenarios respectively scored 19.1% and 36.7% higher than the confidentiality scenario. This effect is more pronounced in the heuristic, scoring 51.4% and 66.7% higher in the integrity and availability scenarios, respectively, than in the confidentiality scenario. Moreover, the addition of the `misinform` action narrowed the performance gap between IQL and QMIX from a 11.2% QMIX advantage to 0.2%. These results suggest that there is an attractive local minimum for each scenario that both architectures tend to settle in.

The multi-agent host-monitoring game presented does not require explicit coordination for a defender team to be suc-

Figure 3: Learning curves of IQL and QMIX compared to heuristic at three scenarios.

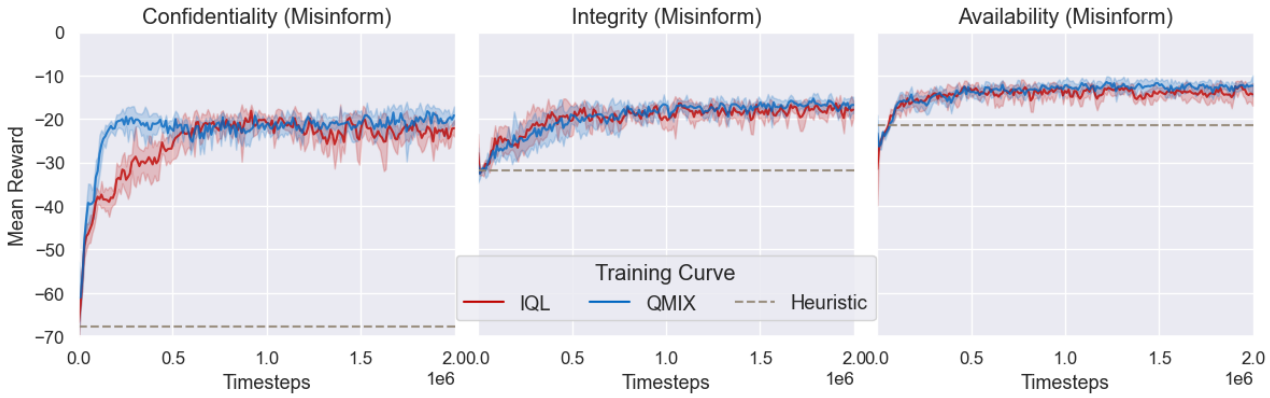

Figure 4: Learning curves of IQL and QMIX compared to heuristic with the addition of the proactive misinform action.

cessful. Moreover, the segregation of agents into subnets limits their ability to influence the state of other agents. As a result, IQL is able to discover policies that score closely to those of QMIX despite it's disadvantages of lacking coordination and suffering from multi-agent non-stationarity. IQL also has the advantage of scalability. QMIX requires that Q-functions are learned centrally, thus limiting the number of possible agents since the central learner will eventually create a bottleneck. IQL does not require any central learning and therefore the number of agents in a game does not pose a constraint.

7 Conclusion and Future Work

Tactical decision-making in cyber defence contends with an expansive problem space necessitating expert knowledge and, in many cases, decentralization of effort to provide essential coverage. This work presented the initial evidence of the success of cooperative MARL to generate of decentralized, tactical-level control policies for a variety of host-based ACD scenarios. Independent and CTDE value-based cooperative MARL architectures can learn policies that outperform a basic heuristic model in this game. In particular, the performance of IQL at these tasks suggests that this approach may scale well to larger networks with more agents. Future gains in performance are expected to be possible both in terms of the learning ability of MARL systems and in the realism of training games. We suggest the following areas for the future development of cooperative MARL in the domain of ACD:

1. Techniques to approximate greater environmental realism, including the use of generative programs to increase the sample size of experimentation [Ellis *et al.*, 2022; Molina-Markham *et al.*, 2021],
2. Games offering more sophisticated and varied threat types using self-play [Hammar and Stadler, 2020] and adversarial machine learning attacks [Standen *et al.*, 2023; Molina-Markham *et al.*, 2021; Han *et al.*, 2020], and
3. Leveraging promising MARL techniques such as multi-agent policy-gradient methods [Yu *et al.*, 2021] or hierarchical role assignment [Wang *et al.*, 2020a], transfer learning [Wang *et al.*, 2020b], attention mechanisms [Yang *et al.*, 2020] and transformers for value-decomposition [Khan *et al.*, 2022].

References

[Andrew *et al.*, 2022] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. Developing optimal causal cyber-

Table 1: Mean return +/- standard deviation of trained iql and qmix policies relative to the heuristic model.

Scenario	IQL	QMIX	Heuristic
Confidentiality	-26.19 ±5.5	**-23.01 ±2.47**	-65.96 ±24.33
Integrity	-24.38 ±5.44	**-16.8 ±2.73**	-32.97 ±20.82
Availability	**-15.94 ±3.78**	-17.5 ±4.68	-23.11 ±18.88
Confidentiality (Misinform)	**-19.76 ±4.18**	-25.07 ±7.1	-67.64 ±24.03
Integrity (Misinform)	-18.17 ±3.86	**-16.75 ±3.48**	-31.9 ±20.73
Availability (Misinform)	-14.47 ±2.98	**-11.63 ±2.84**	-21.39 ±16.95

*The highest scores for each scenario are given in bold.

defence agents via cyber security simulation. *arXiv preprint arXiv:2207.12355*, 2022.

[Applebaum *et al.*, 2022] Andy Applebaum, Camron Dennler, Patrick Dwyer, Marina Moskowitz, Harold Nguyen, Nicole Nichols, Nicole Park, Paul Rachwalski, Frank Rau, Adrian Webster, et al. Bridging automated to autonomous cyber defense: Foundational analysis of tabular q-learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pages 149–159, 2022.

[Bakker, 2001] Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14, 2001.

[Buşoniu *et al.*, 2010] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[CAGE, 2022] CAGE. Ttcp cage challenge 2. In *AAAI-22 Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2022.

[Chowdhary *et al.*, 2020] Ankur Chowdhary, Dijiang Huang, Jayasurya Sevalur Mahendran, Daniel Romo, Yuli Deng, and Abdulhakim Sabur. Autonomous security analysis and penetration testing. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pages 508–515. IEEE, 2020.

[Ellis *et al.*, 2022] Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022.

[Foley *et al.*, 2022] Myles Foley, Chris Hicks, Kate Highnam, and Vasilios Mavroudis. Autonomous network defence using reinforcement learning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 1252–1254, 2022.

[Gabirondo-López *et al.*, 2021] Jon Gabirondo-López, Jon Egana, Jose Miguel-Alonso, and Raul Orduna Urrutia. Towards autonomous defense of sdn networks using muzero based intelligent agents. *IEEE Access*, 9:107184–107199, 2021.

[Gutzwiller *et al.*, 2015] Robert S Gutzwiller, Sunny Fugate, Benjamin D Sawyer, and PA Hancock. The human factors of cyber network defense. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 59, pages 322–326, 2015.

[Hammar and Stadler, 2020] Kim Hammar and Rolf Stadler. Finding effective security strategies through reinforcement learning and self-play. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE, 2020.

[Han *et al.*, 2020] Yi Han, David Hubczenko, Paul Montague, Olivier De Vel, Tamas Abraham, Benjamin IP Rubinstein, Christopher Leckie, Tansu Alpcan, and Sarah Erfani. Adversarial reinforcement learning under partial observability in autonomous computer network defence. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[Hessel *et al.*, 2018] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Hu *et al.*, 2020a] Zhenguo Hu, Razvan Beuran, and Yasuo Tan. Automated penetration testing using deep reinforcement learning. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 2–10. IEEE, 2020.

[Hu *et al.*, 2020b] Zhisheng Hu, Minghui Zhu, and Peng Liu. Adaptive cyber defense against multi-stage attacks using learning-based pomdp. *ACM Transactions on Privacy and Security (TOPS)*, 24(1):1–25, 2020.

[Hu *et al.*, 2021] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv e-prints*, pages arXiv–2102, 2021.

[Khan *et al.*, 2022] Muhammad Junaid Khan, Syed Hammad Ahmed, and Gita Sukthankar. Transformer-based value function decomposition for cooperative multi-agent reinforcement learning in starcraft. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, pages 113–119, 2022.

[Kotenko, 2007] Igor Kotenko. Multi-agent modelling and simulation of cyber-attacks and cyber-defense for homeland security. In *2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 614–619. IEEE, 2007.

[Kunz *et al.*, 2023] Thomas Kunz, Christian Fisher, La Novara-Gsell, Christopher Nguyen, Li Li, et al. A multiagent cyberbattlesim for rl cyber operation agents. *arXiv preprint arXiv:2304.11052*, 2023.

[Li *et al.*, 2023] Li Li, Jean-Pierre S El Rami, Adrian Taylor, James Hailing Rao, and Thomas Kunz. Enabling a network ai gym for autonomous cyber agents. *arXiv preprint arXiv:2304.01366*, 2023.

[Lin, 1992] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

[Littman, 1994] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[Lowe *et al.*, 2017] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[Malialis and Kudenko, 2015] Kleanthis Malialis and Daniel Kudenko. Distributed response to network intrusions using multiagent reinforcement learning. *Engineering Applications of Artificial Intelligence*, 41:270–284, 2015.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[Molina-Markham *et al.*, 2021] Andres Molina-Markham, Cory Miniter, Becky Powell, and Ahmad Ridley. Network environment design for autonomous cyberdefense. *arXiv preprint arXiv:2103.07583*, 2021.

[Nguyen and Reddi, 2019] Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[Nyberg and Johnson, 2023] Jakob Nyberg and Pontus Johnson. Training automated defense strategies using graph-based cyber attack simulations. *arXiv preprint arXiv:2304.11084*, 2023.

[Papadimitriou and Tsitsiklis, 1987] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[Papoudakis *et al.*, 2020] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Servin and Kudenko, 2008] Arturo Servin and Daniel Kudenko. Multi-agent reinforcement learning for intrusion detection: A case study and evaluation. In *German Conference on Multiagent System Technologies*, pages 159–170. Springer, 2008.

[Shi and He, 2021] Guochen Shi and Gang He. Collaborative multi-agent reinforcement learning for intrusion detection. In *2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*, pages 245–249. IEEE, 2021.

[Standen *et al.*, 2021] Maxwell Standen, Martin Lucas, David Bowman, Toby J Richer, Junae Kim, and Damian Marriott. Cyborg: A gym for the development of autonomous cyber agents. *arXiv preprint arXiv:2108.09118*, 2021.

[Standen *et al.*, 2023] Maxwell Standen, Junae Kim, and Claudia Szabo. Sok: Adversarial machine learning attacks and defences in multi-agent reinforcement learning. *arXiv preprint arXiv:2301.04299*, 2023.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Team, 2021] Microsoft Defender Research Team. Cyberbattlesim. https://github.com/microsoft/cyberbattlesim, 2021.

[Utic and Ramachandran, 2022] Zheni Utic and Kandethody Ramachandran. A survey of reinforcement learning in intrusion detection. In *2022 1st International Conference on AI in Cybersecurity (ICAIC)*, pages 1–8. IEEE, 2022.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[Vyas *et al.*, 2023] Sanyam Vyas, John Hannay, Andrew Bolton, and Professor Pete Burnap. Automated cyber defence: A review. *arXiv preprint arXiv:2303.04926*, 2023.

[Wang *et al.*, 2020a] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020.

[Wang *et al.*, 2020b] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7293–7300, 2020.

[Wang *et al.*, 2022] Wenhao Wang, Dingyuanhao Sun, Feng Jiang, Xingguo Chen, and Cheng Zhu. Research and challenges of reinforcement learning in cyber defense decision-making for intranet security. *Algorithms*, 15(4):134, 2022.

[Wolk *et al.*, 2022] Melody Wolk, Andy Applebaum, Camron Denver, Patrick Dwyer, Marina Moskowitz, Harold Nguyen, Nicole Nichols, Nicole Park, Paul Rachwalski, Frank Rau, et al. Beyond cage: Investigating generalization of learned autonomous network defense policies. *arXiv preprint arXiv:2211.15557*, 2022.

[Xu *et al.*, 2007] Xin Xu, Yongqiang Sun, and Zunguo Huang. Defending ddos attacks using hidden markov models and cooperative reinforcement learning. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 196–207. Springer, 2007.

[Yang *et al.*, 2020] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.

[Yu *et al.*, 2021] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.