Exploring the Efficacy of Multi-Agent Reinforcement Learning for Autonomous Cyber Defence: A CAGE Challenge 4 Perspective

Mitchell Kiely[1], Metin Ahiskali[6], Etienne Borde[9], Benjamin Bowman[12], David Bowman[1], Dirk Van Bruggen[11], KC Cowan[6], Prithviraj Dasgupta[7], Erich Devendorf[8], Ben Edwards[2], Alex Fitts[11], Sunny Fugate[5], Ryan Gabrys [5], Wayne Gould[2], H. Howie Huang[12], Jules Jacobs[10], Ryan Kerr[3], Isaiah J. King[12], Li Li[3], Luis Martinez[5], Christopher Moir[1], Craig Murphy[2], Olivia Naish[2], Claire Owens[2], Miranda Purchase[2], Ahmad Ridley[4], Adrian Taylor[3], Sara Farmer[2], William John Valentine[9], Yiyi Zhang[10]

[1]Defence Science and Technology Group (DSTG), Australia.
[2]Defence Science Technology Laboratory (Dstl), United Kingdom.
[3]Defence Research and Development Canada (DRDC), Canada.
[4]National Security Agency (NSA), USA.
[5]Naval Information Warfare Center (NIWC) Pacific, USA.
[6]Army Combat Capabilities Development Command (DEVCOM), USA.
[7]Naval Research Laboratory (NRL), USA.
[8]Air Force Research Laboratory (AFRL), USA.
[9]University of Canterbury, New Zealand.
[10]Cornell University, USA.
[11]Punch Cyber Analytics, USA.
[12]Cybermonic, USA.

Abstract

As cyber threats become increasingly automated and sophisticated, novel solutions must be introduced to improve defence of enterprise networks. Deep Reinforcement Learning (DRL) has demonstrated potential in mitigating these advanced threats. Single DRL Agents have proven utility toward execution of autonomous cyber defence. Despite the success of employing single DRL Agents, this approach presents significant limitations, especially regarding scalability within large enterprise networks. An attractive alternative to the single agent approach is the use of Multi-Agent Reinforcement Learning (MARL). However, developing MARL agents is costly with few options for examining MARL cyber defence techniques against adversarial agents. This paper presents a MARL network security environment, the fourth iteration of the Cyber Autonomy Gym for Experimentation (CAGE) challenges. This challenge was specifically designed to test the efficacy of MARL algorithms in an enterprise network. Our work aims to evaluate the potential of MARL as a robust and scalable solution for autonomous network defence.

Code —
https://github.com/cage-challenge/cage-challenge-4

Introduction

Cyber threats are escalating concerns for organisations and governments as attacks grow in frequency and sophistication. As the world becomes increasingly interconnected, these threats are expected to rise (McLean 2024), placing a significant burden on cyber security administrators tasked with responding to network incidents. The onslaught of threats has sparked extensive research into Autonomous Cyber Defence (ACD), which is defined as automated decision-making agents for cyber systems to mitigate highly complex cyber attacks (Vyas et al. 2023), to aid human operators in managing the sheer volume of necessary threat response.

Agents are typically categorised into three groups: red, green, and blue. In general, red agents are adversarial, blue agents are allied, and green agents are neutral. Typically, research in this field focuses around adversarial environments, where an attacking red agent is pitted against a defending blue agent and either of them must learn the optimal attack or defence strategy (Macas, Wu, and Fuertes 2023). Deep Reinforcement Learning (DRL), a subfield of Machine Learning (ML), algorithms has shown potential in employing these optimal strategies in complex cyber security environments (Nguyen and Reddi 2021).

Autonomous DRL agents can learn the optimal sequence of actions in a scenario with limited prior information (Li 2017). These algorithms have proven highly effective in single agent adversarial environments such as Chess (Silver et al. 2017), Go (Silver et al. 2018), and the real-time multi-agent adversarial environments such as Starcraft (Vinyals et al. 2019). These successes have led many to investigate the potential of DRL in the field of ACD.

At its core, DRL can be explained through the concepts of states, actions, and rewards. At discrete timesteps, an agent receives an observation based on the state of the environment. For each observation, the agent selects an action to enact, resulting in a new state. Based on this new state, the environment outputs a reward relative to how advantageous the new state is the the learning agent. Different actions yield different states and different rewards. Over numerous interactions with the environment, the agent is incentivized to select actions that yield the greatest reward, implementing a strategy that maximises the total system reward.

Historically, ACD environments have been single agent or single adversarial environments, where one blue agent defends the network against one attacking red agent. Previous CAGE Challenges (Kiely et al. 2023) have resulted in highly capable single-agent based autonomous defensive agents. However, these agents struggle to scale efficiently when dealing with large enterprise networks common in real-world scenarios. Moreover, a single agent becomes a central point of failure for the entire network's defence. MARL algorithms can mitigate many of these drawbacks.

By applying MARL to ACD, the defence of a large enterprise network can be distributed across individual subnets. Each defensive agent is tasked with defending a single subnet, reducing state space while contributing to the defence of the entire network. This distributed approach is inherently more robust (Buşoniu, Babuška, and De Schutter 2010), as the failure of an agent to defend a single subnet is typically less impactful than the agent failing to defend the entire network. Furthermore, cooperative agents can learn to maximise their collective reward (Tan 1993) by providing assistance to other subnets in more critical situations. In the single agent scenario, the state-action space is larger as it comprises all subnets. In contrast, for the multi-agent case, the state-action space can be significantly reduced to include only those within and between subnets. Despite these benefits, very few environments have been developed to explore the effectiveness of MARL algorithms for autonomous cyber defence

In this paper, we introduce a novel environment, Cage Challenge 4 (CC4), which simulates a relatively large enterprise network. This tool has been employed across academia, industry and government to examine the effectiveness of multi-agent systems in executing autonomous cyber operations. The environment acts as a foundational platform, allowing other researchers and industry experts to compare and extend existing agents in this specialized field. Moreover, the environment facilitates the reproducibility of results for future world implementations.

Related Works

The number of adversarial cyber-security environments dedicated to developing defensive blue agents has seen a steady rise over recent years. These environments encompass a variety of adversarial scenarios designed to foster research into implementation of autonomous agents for cyber-defence. A common feature of these environments is the adversarial 1-on-1 set up, where one attacking agent attempts to gain further network access, and one defending agent protects the system. To our knowledge, apart from environments using CybORG, no enterprise-based environments exist where multiple blue and red actors execute actions simultaneously on the same network. A summary of existing environments and their place in this context is provided in Table 1.

We categorise single agent environments as those where only one red or one blue agent can learn at a time. These environments are stationary, meaning, nothing within the environment changes over time except the learning agent. Red and blue environments are those where one red agent and one blue agent can learn concurrently. These environments are non-stationary. Both agents learn and adapt to the adversary's actions as the training of each agent progresses. Lastly, n blue or n red environments are those where multiple blue agents, or multiple red agents can learn simultaneously. Within this last category, CybORG stands alone in being the only code base that supports training of multiple blue agents.

CC4 extends the CybORG code base to include an environment that replicates a large enterprise network, composed of numerous subnets, where each subnet contains a unique

Environment	**Red or Blue**	**Red and Blue**	***n* blue or *n* Red**
CyGIL (Li, Fayad, and Taylor 2021)	✓		
PrimAITE (Dstl 2023)	✓		
CSLE (Hammar and Stadler 2022)	✓		
Gym-IDS game (Hammar and Stadler 2020)	✓	✓	
CyberBattle Sim (Microsoft Threat Intelligence 2021)	✓		
MARLon (Kunz et al. 2022)	✓	✓	
Gym-Threat-defence (Miehling, Rasouli, and Teneketzis 2015)	✓		
Gym-Optimal-Intrusion-Response (Hammar and Stadler 2021)	✓		
AtMOS (Akbari et al. 2020)	✓		
Yawning Titan (Collyer, Andrew, and Hodges 2022)	✓		
Farland (Molina-Markham et al. 2021)	✓		
CYST (Drašar et al. 2020)	✓		
CybORG (Standen et al. 2021)	✓		✓

Table 1: A list of open-source Autonomous Cyber Defensive environments and which category of agent training they facilitate

defender agent. These autonomous agents range from advanced reinforcement learning algorithms to simple heuristics, offering flexibility to the user. The only prerequisite for agents on each subnet is that they must be able to receive an input and output an action. This challenge offered insights into the nature of the interactions between multiple blue agents that defend a large enterprise network.

Actions taken by a defender on one subnet may positively or negatively impact another defender on a different subnet. This reflects real-world incident response teams which are composed of numerous individuals, each with their own assigned job. Most existing cyber defence environments focus solely on the capacity of a single agent to defend a network. This fails to reflect real world scenarios where individuals typically work together to accomplish a goal.

Given the unique nature of this environment, no studies have yet explored how multiple autonomous defensive agents interact to defend the same network. Understanding these interactions is crucial for future real system deployments, where the network is too complex for a single agent to defend. CC4 provides the preliminary environment for individuals to investigate these interactions. Furthermore, CC4 serves as an initial benchmark for future developers of multiple autonomous agents to measure the effectiveness of their AI model in a complex cyber security scenario.

CAGE Challenge 4

CAGE Challenge 4 (CC4) is an open-source MARL environment that facilitates the study of multi-agent autonomous cyber defence. This foundational platform enables researchers to repeatedly build, initialize, and test multi-agent AI models across various cyber security scenarios. Detailed documentation that outlines the environment's inner workings is publicly accessible on the Github page (TTCP 2024). This documentation provides a high-level overview of the challenge, along with low-level specifics explaining the classes implemented and functions' called. Additionally, a suite of user-friendly tutorials with an interdisciplinary design are available to assist both cybersecurity and AI experts in developing and evaluating agents in this environment.

Network Structure

The simulated network for this challenge is designed to mimic an operational military network. It is divided into eight individual subnets, each representing either an operational or 'back-office' support capability. A visual representation of this network can be viewed on the Github repository.

Critical services are present on the servers in the subnets: 'Operational Zone A' and 'Operational Zone B'. Both of these subnets have network links with their respective 'Restricted Zones', which are then linked together via the Internet node. This collection of subnets make up the operational capability of the network and are contained in the 'MISSIONNET'.

The 'SIMNET' segment holds the supporting network functions for the mission. The 'Contractor Network' subnet is connected directly to the Internet node and is not under the control of the blue defenders in this scenario, instead being the responsibility of the contracting company. The inclusion of this subnet helps represent the impact of supply chains on cyber-security and is the point of origin for the attacker in this scenario. Due to this separation in control, the remaining military subnets in this section are commonly referred to and acted upon as one large subnet, 'HQ Network'. The internal structure of the HQ segment is made up of the 'Public Access Zone' subnet, which acts as a boundary between the Internet node and the independently connected subnets 'Admin Network' and 'Office Network'.

Each subnet in this simulated network contains several computer systems or hosts. A host can be of type 'user' or 'server'. User represents a desktop computer associated with a green agent while a server represents a server hosting services green agents connect to. To increase scenario complexity, the number of hosts and servers vary per episode. All hosts contain services which are used by green agents producing neutral actions but introduce vectors for red 'attacker' agents to exploit allowing access to a host.

Agent Host Allocation

All active agents are linked to one or more hosts by a session, which is recorded in the environment. Green agents are permanently linked to one host and are present on every user host in the environment. As the generation of hosts is dynamic, this can result in a varying number of green agents between episodes.

Blue and red agents however, are allocated up to one per subnet. A blue agent has a session on all the hosts in its subnet to allow it to act on them, and these links do not change over the course of the episode. The Contractor Network subnet stands out as the only subnet that has red presence but not blue. The red agent in the Contractor Network is the only active red agent in the environment at the start of the episode, with one privileged session. It explores the subnet, gaining more sessions on different hosts until it discovers another subnet. When the red agent succeeds at getting a session in a different subnet, the red agent allocated to that subnet is activated and given ownership of the newly created session. This is how the red presence expands without a singular red agent being stretched too thin over a larger network.

Reward

The rewards for CC4 are scaled to reflect mission priorities that are outlined in the instructional materials of the challenge. Thus, rewards for actions that adversely affect critical mission priorities are more impactful than actions that target non-critical systems. For example, exploiting a host with escalated privileges has a more impactful reward score than simply discovering a host on the network. The dynamic reward function employed in CC4 primarily focuses on the effect blue and red actions in the network have on the green agents. This focus was preferred over a purely red agent action focus in order to limit state leakage to the blue agents because, as in reality, the blue defenders do not instantly know the movements of every attacker on the network. Green agents can perform three types of actions: 1) Sleep - the equivalent of performing no action that step; 2)

Local Work - the green user "uses" a service on the host they have a session on; and 3) Access Service - the green user accesses a service on another host. Green agents randomly select one of these three actions every turn.

Red and blue actions can both cause Local Work and Access Service to fail, resulting in a negative reward. The red agent's Impact action can also cause a negative reward, as this signifies an attack on the Operational Technology (OT) Service. The success or failure of an action, and in which subnet, is used to calculate a reward score for the given timestep. The cumulative reward for each time-step is calculated into the final reward.

Changing mission priorities are also captured in the reward function with the use of mission phases. The simulation's episode length is divided into three phases, with the first being the mission pre-planning phase, and the last being two different active missions. The rewards change with each phase to reflect their shifting priorities and requirements.

Network Communication Policy

Another dynamic aspect of the simulation is the network communication policy. The policy is not enforced by the environment but is obeyed by the green agents. When the green agents are not using a specific route, having it open only benefits red agent movement; conversely, if the green agents plan on using that route and it is blocked, blue will receive negative rewards due to the green Access Service action failing. This communication policy changes with the mission phase to reflect changing mission priorities.

Blue Agent Space & Communications

The blue action space consists of eight actions with varying time durations in steps, as shown in brackets next to the action:

1. Monitor (1) – collects all the 'events' generated by the environment during that step, for the subnet the agent is in. This mimics the event logs produced by an intrusion detection system (IDS).
2. Analyse (2) – examines a host for files left behind by attacker activity, such as utilising an exploit or escalating privileges. This mimics threat hunting behaviour.
3. Deploy Decoy (2) – creates a honeypot service on a host, which always generates events and cannot be used to gain access to the host.
4. Remove (3) – removes any non-privileged, user-level sessions from a specified host. This action is limited due to the assumption of a persistence mechanism for privileged sessions.
5. Restore (5) – re-images a host, resulting in the removal of all malicious activity; including privileged red sessions.
6. Block Traffic Zone (1) – blocks traffic between two subnets. This affects all traffic apart from blue agent messaging and red sessions resulting from a phishing email.
7. Allow Traffic Zone (1) – reverses the effect of a block traffic action.
8. Sleep (1) – equivalent of performing no action.

The time durations penalise the use of more powerful actions to balance the action's overall effectiveness, and to mirror the time requirements of more resource intensive actions, such as re-imaging a host. Actions that are submitted by agents but cannot be performed, such as trying to run an action on a host that does not exist in the environment, will result in an Invalid Action.

As well as choosing an action each turn, each blue agent can also send eight bit messages to the other blue agents. This is unaffected by Block/Allow Traffic Zone actions and is an opportunity for collaborative agent decision-making. How this communication method is encoded is defined by the user, however it will default to zeros if not specified.

Red Agent

The red agent action space consists of actions that allow the agent to discover information about the area of the network they are in, gain sessions of different privileges, and negatively affect the ability for green to successfully complete their actions. The type of red agent that the challenge evaluates the blue agents against uses a finite state machine. The agent internally records the hosts that they are aware of and assigns a state to each host. The states represent how much knowledge the red agent has about the host and the stage they are at of taking control over that host. The agent contains a probability matrix that is used to choose the next action based on a chosen host's current state. For the challenge evaluation, the host chosen to be acted upon is random.

Evaluation Method

The platform we chose to host this challenge is Codalabs. On submission of an entry, Codalabs automatically evaluates the agent in the CC4 environment. This produces data used for later analysis and extracts the average reward which is used for the challenge leaderboard. The usage of this platform significantly reduced the difficulty and time needed to successfully run the challenge. To test the generalisability of the agents, we evaluated them in an additional four scenarios that are slight modifications to CC4, these being: 1) Constant Network Size; 2) Improved Decoy Detection; 3) Increased Phishing; 4) Stealthy Red; 5) Aggressive Red. All submitted blue agents were developed without any knowledge that these additional tests were to be carried out. Details of these modifications are in the following section.

Results

The Best Agent from the Top Four Teams

CC4 was designed and developed to study MARL algorithms that coordinated a defensive strategy to achieve a goal. However, it is not limited to only evaluating MARL solutions. Across the fifteen CC4 participants, there were sixty-five unique submissions, composing both MARL-based and non MARL-based agents. For brevity, we provide an analysis on the best performing agent of the top four teams. *The following sections describe how these agents work, and were authored by the developers of the agents.*

Team Name	Agent Type	CC4	Constant Network Size	Improved Decoy Detection	Increased Phishing	Stealthy Red	Aggressive Red
Team UC	Heuristic	-113 ± 35	-101 ± 36	-138 ± 77	-534 ± 347	-153 ± 112	-508 ± 277
Team Lancer	Heuristic	-118 ± 40	-71 ± 23	-151 ± 52	-232 ± 79	-871 ± 627	-801 ± 236
Team Punch	Heuristic	-142 ± 44	-94 ± 20	-171 ± 44	-205 ± 78	-158 ± 46	-837 ± 203
Team Cybermonic	MARL	-193 ± 84	-176 ± 51	-198 ± 53	-350 ± 182	-996 ± 482	-654 ± 244

Table 2: Scores for the highest performing CC4 agents from the top four teams submitted to the public CC4 competition and variations of the evaluation environment considered throughout the discussion chapter: absence of invalid actions and generalisations. Each score indicates the average cumulative reward per episode, taken over 100 episodes of 500 steps each.

Team UC We developed a rule-based heuristic based on the following principles: (i) events filtering to improve malicious activities detection, and (ii) combining reactive and proactive defence. We revised the observation vector to use one-hot encoded vectors that included additional useful information such as malicious processes, connections, and file flags remain until either Restore or Remove has been used on the host. In doing so, we were able to better understand the threats being faced in each subnet, allowing us to implement an effective defensive protocol. Our agent reactively uses the Restore action on hosts flagged for connection events, while process events and malicious files are dealt with by using either Restore or Remove. If there are no malicious flags, then agent will implement proactive defensive actions such as Allow Traffic Zone to subnets according to the communication policies. After the necessary subnets are unblocked, the agent will Block Traffic Zone to any subnets according to the communication policies of the current phase. If there are no malicious flags, and no requirement to block or unblock any subnets, the agent will select either Analyse, Monitor, or DeployDecoy. If DeployDecoy is selected, it will be used on a random valid host. If Analyse is selected, it will be used on any host that has longest since been analysed or restored, and that has not been subject to the Remove action.

Team Lancer Our highest-scoring agent is heuristic and does not use machine learning. Our agents work independently but use identical logic. The agents start by placing one decoy on each host. They then keep performing Analyse actions on the hosts in a roughly round-robin order. Depending on the result of Analyse, the agent immediately performs Restore (if escalate.sh is found), or Remove (if cmd.sh is found), or continues with the next host. Each agent maintains a floating point priority value for each host, and always chooses the host with the highest priority. The priority of each host is initialised to a configurable value, decreased after each Analyse/Restore/Remove action on the host, and increased in response to Monitor observations.

Team Punch Our top performing agent, which performed much better than our RL agents, applied an identical strategy in each subnet. This followed a simple pattern of analysing and subsequently restoring infected machines in a round-robin fashion. To implement this heuristic policy, an additional wrapper was needed that included the results from analyse actions in the observation vector. The new wrapper queried CybORG for each agent after stepping the environment and checked for files present in the raw CybORG observation. If files were present and any had a density of greater than 0.9, a feature for the host the files were from was set to True, else in all other instances, it was set to False. With this new information, we were able to implement the described heuristic as a state machine by tracking the step count and current state.

Team Cybermonic We created a wrapper to model the environment as a temporal, attributed graph. We track five types of entities: Hosts (users and servers), Routers, Open Ports, Files, and the Internet. The relationship between entities, and their associated features, can be found on our GitHub page (King 2024). Each agent independently maintains an approximation of the true environment graph using the partial observations emitted by the environment. Because we represent the environment as a graph, actions are realised as graph edits. When an agent successfully completes an action, it updates its internal approximation of the environment graph. By representing actions as functions upon nodes, we are free to increase or decrease the action space by changing the size of set of actionable nodes. Agents process states using a graph neural network (GNN) (Hamilton, Ying, and Leskovec 2017), which encodes node features and topological features of the network. We implement a topological embedder proposed by (Janisch, Pevný, and Lisý 2020), and also create a global graph vector that encodes the observation as a whole. We use these embeddings to calculate the probability of taking an action on a node or edge. The actor module outputs the probability of taking an action, and the critic uses the global graph vector $\mathbf{g}$ to estimate the value of the current state. Each agent is trained independently using PPO (Schulman et al. 2017).

Discussion

Constant Network Size

One challenge facing participants was the randomisation of the number of users and servers for each episode. With 100 episodes all agents faced a wide variety of users and servers in each zone, making it extremely challenging for the agents on their own to learn and adapt to this. As such, when investigating the strategies of the submissions it became apparent that several agents were attempting to take actions on users and servers that did not exist within an episode. In doing so, the environment was unable to process the action, and thus deemed the action to be invalid.

Team UC's submission accounted for this behaviour by expanding the observation vector to include information on present users and servers in each episode. This produced a more efficient agent by eliminating the inadvertent return of invalid actions. Almost all other agents did not account for this, so we sought to investigate the utility of each agent strategy where the users and servers were known for each episode. This was achieved by fixing the number of users and servers to their maximum value. The change in agent rankings in this experiment, as shown in Table 2, indicate that an agent's strategy and the information given to it are vital for successfully defending a network. Notably, all submitted MARL solutions failed to learn how to determine the network configuration at the beginning of each episode. This modifiable parameter highlighted a weakness that future MARL approaches will have to overcome to become effective autonomous agents.

Generalisability of Agents

A common criticism of RL often centres around the challenges of applying RL into OT (e.g., a networked system with industrial control hardware) where the environment is often much more complex and the adversarial actions/strategies less defined. The challenge for RL is often the balance between generalisation, by keeping the training limited but the agent adaptable, and over-fitting, whereby the agent is trained too well on a specific scenario and cannot react to new situations. The goal is to find the sweet-spot where an RL agent is well-suited to the challenge it is presented whilst being able to handle changes to the environment.

CC4 has many parameters, all of which can be easily modified to produce a starkly different scenario, enabling future researchers to readily tune the environment to test novel scenarios of interest. To investigate generalisability, numerous scenario parameters were modified and the best performing submissions of the top four teams were evaluated under the new conditions. The participants, as well as their agent submissions, were never exposed to these scenarios. Four scenarios were explored: Improved Decoy Detection, Increased Phishing, Stealthy Red, and Aggressive Red:

1. Improved Decoy Detection. DiscoverDeception red action was increased from 0.5 to 1, the false positive rate from 0.1 to 0, and the duration reduced from 2 steps to 1, making decoys an ineffective means of red detection.
2. Increased Phishing. The probability that when a green agent chose the EnterpriseGreenAgent action that it would trigger a phishing email attack was increased from 0.01 to 0.75, significantly increasing the rate at which green agents spawned red agents.
3. Stealthy Red. The false positive detection rate of the EnterpriseGreenAgent is changed from 0.01 to 0.75. Additionally, within the DiscoverNetworkServices red action, the detection rate in each of the three types was changed from 1, 0.75, and 0.25 to 0.1.
4. Aggressive Red. All red agent actions are reduced to a duration of one timestep, forcing blue agents to react more quickly to all red agent behaviours whilst being slower to complete their actions.

The Improved Decoy Detection experiment suggests that the red agent gains a slight advantage in comparison to the original scenario. However, the red agent may reveal its intentions by determining if a decoy is a decoy, which the blue agent can exploit if it acts swiftly. All three other generalisations (Improved Phishing, Stealthy Red, and Aggressive Red) detrimentally affect the defenders' cumulative scores. When the conditions of the attack are altered, the implemented blue agent, whether it was a heuristic or MARL-based agent, is unable to effectively counter the attack.

Similar results were found for other MARL based submissions that were submitted to CC4. Across all experiments, and all the top submissions from the fifteen participants, we found that the heuristic-based agents outperformed all MARL-based submissions. This suggests that MARL-based agents require further research to more effectively respond to both known and unknown threats.

Impact & Future Works

CC4 significantly reduces both the financial cost and time needed for future studies by offering a reusable open-source environment. This foundational platform empowers future researchers and developers to rapidly extend this field by eliminating the need to build environments and agents from scratch. Further, the comprehensive documentation enables a diverse group of individuals to quickly comprehend this challenging area, potentially fostering growth in this emerging field. Nevertheless, a range of additional case studies must be carried out in order to deploy MARL-based technologies on real world systems. Crucially, a comparison between single agent and multi-agent DRL algorithms, across different network sizes, must be done to determine if and when MARL based algorithms become more effective. Lastly, MARL based algorithms must be tested in an emulated environment prior to deployment in real-world networks. CC4 can serve as the foundational platform for building these emulated environments, reducing time required for development and expediting agent training and evaluation.

Conclusion

CC4 is a novel open-source multi-agent cyber defence environment, designed to facilitate research into Multi-Agent Reinforcement Learning applications for autonomous cyber defence. Its easily configurable nature offers a variety of scenarios to evaluate the effectiveness of MARL based algorithms, as well as general multi-agent systems. Additionally, we have established agent performance benchmarks that provide a comparative baseline for future agent development. Preliminary studies indicate that while Deep Reinforcement Learning shows high potential in single-agent environments, its application in multi-agent scenarios necessitates further consideration and additional features for effective, coordinated cyber defence.

References

Akbari, I.; Tahoun, E.; Salahuddin, M. A.; Limam, N.; and Boutaba, R. 2020. ATMoS: Autonomous threat mitigation in SDN using reinforcement learning. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, 1–9. IEEE.

Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183–221.

Collyer, J.; Andrew, A.; and Hodges, D. 2022. ACD-G: Enhancing autonomous cyber defense agent generalization through graph embedded network representation. In *Proceedings of the 39th International Conference on Machine Learning (ML4Cyber workshop)*. International Conference on Machine Learning.

Drašar, M.; Moskal, S.; Yang, S.; and Zat'ko, P. 2020. Session-level adversary intent-driven cyberattack simulator. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 1–9. IEEE.

Dstl. 2023. PrimAITE. https://github.com/Autonomous-Resilient-Cyber-Defence/PrimAITE. Accessed: 2018-07-03.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

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

Hammar, K.; and Stadler, R. 2021. Learning intrusion prevention policies through optimal stopping. In *2021 17th International Conference on Network and Service Management (CNSM)*, 509–517. IEEE.

Hammar, K.; and Stadler, R. 2022. Intrusion prevention through optimal stopping. *IEEE Transactions on Network and Service Management*, 19(3): 2333–2348.

Janisch, J.; Pevnỳ, T.; and Lisỳ, V. 2020. Symbolic Relational Deep Reinforcement Learning based on Graph Neural Networks and Autoregressive Policy Decomposition. *arXiv preprint arXiv:2009.12462*.

Kiely, M.; Bowman, D.; Standen, M.; and Moir, C. 2023. On Autonomous Agents in a Cyber Defence Environment. *arXiv preprint arXiv:2309.07388*.

King, I. 2024. CAGE Challenge 4 Submission. https://github.com/cybermonic/cage-4-submission. Accessed: 2024-07-08.

Kunz, T.; Fisher, C.; La Novara-Gsell, J.; Nguyen, C.; and Li, L. 2022. A Multiagent CyberBattleSim for RL Cyber Operation Agents. In *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, 897–903. IEEE.

Li, L.; Fayad, R.; and Taylor, A. 2021. Cygil: A cyber gym for training autonomous agents over emulated network systems. *arXiv preprint arXiv:2109.03331*.

Li, Y. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Macas, M.; Wu, C.; and Fuertes, W. 2023. Adversarial examples: A survey of attacks and defenses in deep learning-enabled cybersecurity systems. *Expert Systems with Applications*, 122223.

McLean, M. 2024. 2024 Must-Know Cyber Attack Statistics and Trends. https://www.embroker.com/blog/cyber-attack-statis. Accessed: 2024-05-14.

Microsoft Threat Intelligence. 2021. Gamifying machine learning for stronger security and AI models. https://www.microsoft.com/en-us/security/blog/2021/04/08/gamifying-machine-learning-for-stronger-security-and-ai-models/. Accessed: 2018-07-03.

Miehling, E.; Rasouli, M.; and Teneketzis, D. 2015. Optimal defense policies for partially observable spreading processes on Bayesian attack graphs. In *Proceedings of the second ACM workshop on moving target defense*, 67–76.

Molina-Markham, A.; Miniter, C.; Powell, B.; and Ridley, A. 2021. Network environment design for autonomous cyberdefense. *arXiv preprint arXiv:2103.07583*.

Nguyen, T. T.; and Reddi, V. J. 2021. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8): 3779–3795.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

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

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

TTCP. 2024. TTCP CAGE Challenge 4. https://github.com/cage-challenge/cage-challenge-4. Accessed: 2025-01-27.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature*, 575(7782): 350–354.

Vyas, S.; Hannay, J.; Bolton, A.; and Burnap, P. P. 2023. Automated cyber defence: A review. *arXiv preprint arXiv:2303.04926*.