Canaries and Whistles: Resilient Drone Communication Networks with (or without) Deep Reinforcement Learning

Chris Hicks
c.hicks@turing.ac.uk
The Alan Turing Institute

Vasilis Mavroudis
vmavroudis@turing.ac.uk
The Alan Turing Institute

Myles Foley
m.foley20@imperial.ac.uk
Imperial College London

Thomas Davies
tdavies@turing.ac.uk
The Alan Turing Institute

Kate Highnam
k.highnam19@imperial.ac.uk
The Alan Turing Institute

Tim Watson
tim.watson@turing.ac.uk
The Alan Turing Institute

ABSTRACT

Communication networks able to withstand hostile environments are critically important for disaster relief operations. In this paper, we consider a challenging scenario where drones have been compromised in the supply chain, during their manufacture, and harbour malicious software capable of wide-ranging and infectious disruption. We investigate multi-agent deep reinforcement learning as a tool for learning defensive strategies that maximise communications bandwidth despite continual adversarial interference. Using a public challenge for learning network resilience strategies, we propose a state-of-the-art expert technique and study its superiority over deep reinforcement learning agents. Correspondingly, we identify three specific methods for improving the performance of our learning-based agents: (1) ensuring each observation contains the necessary information, (2) using expert agents to provide a curriculum for learning, and (3) paying close attention to reward. We apply our methods and present a new mixed strategy enabling expert and learning-based agents to work together and improve on all prior results.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; **Distributed systems security**; **Network security**.

KEYWORDS

resilient systems, distributed systems, deep reinforcement learning

1 INTRODUCTION

Drones, and other types of unmanned aerial vehicle, are increasingly utilised for disaster relief efforts where they can help to coordinate efforts on the ground by providing real-time surveillance, ad-hoc communication networks, and the delivery of supplies to remote areas [22]. Drones are low-cost, widely available and can operate in hostile and warlike environments with degraded or inoperable infrastructure. When multiple drones are available, connecting them together wirelessly can provide an ad-hoc communication network with enhanced coverage, resilience and safety. Unfortunately, as with embedded systems at large [14, 65], numerous commercial and military drones are vulnerable to cyber attacks [10, 15, 17, 35, 42, 66]. Even if a drone is secure by design, malware can attack the supply chain to compromise the functionality of a specific component [8]. Drone vulnerabilities have been exploited to provide location tracking [29], botnet attacks [41], sensor tampering [17, 66] and covert data exfiltration [55]. Ultimately, many drone vulnerabilities give complete control to the adversary, allowing them to interfere with the drone as they please.

Given the ubiquity of software vulnerabilities in drone systems, it is essential to anticipate their compromise during operation. If drone malware could be actively impeded during operation, then despite the inevitable attacks, an ad-hoc network of drones with sufficient redundancy could still provide valuable services. Current mitigations for drone vulnerabilities include regular operating system updates, intrusion detection systems (IDS), fine-grained circuit analysis, and remote software attestation [35]. However, these defences are not sufficient in many cases. Operating system updates often take several weeks, at best, to include patches for the latest exploits. IDS can help detect adversaries but, in addition to being computationally expensive, are usually limited to observing network traffic without context and can negatively impact network latency. Fine-grained circuit analysis can be defeated [23] and remote software attestation depends, in addition to being certain that attested software is secure, on managing the multiple drone authorisations for swarm-based solutions. None of the current defences can actively impede malware from affecting drones, maintaining crucial ad-hoc communication networks for disaster relief.

Autonomous cyber defence (ACD) is a class of solution methods, models and intelligent agents that actively respond to cyber attacks without the need for human intervention. Intelligent agents trained using reinforcement learning (RL) have, in particular, shown great potential for autonomously defending computer networks and systems from attack [18, 20, 49]. ACD is particularly valuable

when, as in the case of a disaster relief, there is a lack of human experts available to defend systems against an adversary. People on the ground dealing with the situation, emergency responders and regular civilians, are unlikely to have the skills, resources or time needed to patch vulnerable drones.

In this paper we evaluate multi-agent RL (MARL) for defending ad-hoc communication networks against malware attacks. Using a public challenge and environment for ACD blue teaming, we first propose a state-of-the-art expert agent, *Canary*. Next, we show that RL applied in the standard setting yields unsatisfactory results. We identify a framework for bridging the gap in performance, apply our methods and demonstrate the potential of learning-based policies in actively defending ad-hoc drone networks.

2 BACKGROUND AND MOTIVATION

2.1 Reinforcement Learning (RL)

RL is about learning how to interact with an unknown environment to maximise a numerical reward signal. RL is characterised by *trial-and-error*, in which the learner discovers through repeated interaction which actions lead to success, and *delayed reward*, whereby actions may affect rewards far in the future. An important property of RL is that goals are defined only by specifying the reward function. There is no need to define exactly how to reach the goal. RL provides a mechanism to distribute the long-term rewards of goal accomplishment among the many actions that contributed to success.
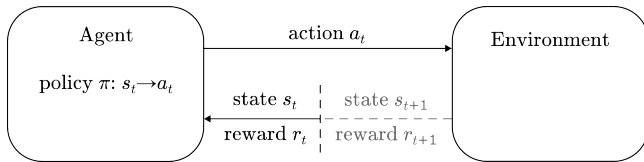

Figure 1: The sequential agent-environment interaction formalisation of RL.

As shown in Figure 1, RL is characterised by a sequential interaction between an *agent* and an *environment*. At each time step t the agent observes the state s_t, the reward r_t, and chooses an action a_t. The environment is formalised as an incompletely-known Markov decision process (MDP) i.e., future states s_{t+1} are independent of past states $s_0, \ldots, s_{t-1}$ given the present state s_t. Learning from interaction takes place over episodes, each comprising multiple time steps. Where H is the maximum number of time steps in an episode, the agent uses trajectories $\tau = \{(s_0, a_0, r_0), \ldots, (s_H, a_H, r_H)\}$ to learn a policy π which is a map from the state space S to the action space A.

Recently, RL algorithms have demonstrated human and super-human level performances including beating world champions at Chess and Go [45, 46], mastering Stratego [39] and discovering faster sorting algorithms [34]. These advances have been made possible by DRL, in which the policy of an RL agent is represented by a deep neural network [36, 43]. DRL enables agents to handle large state spaces by generalising from the trajectories seen during training to rarely or never-before seen states. In addition, neural function approximation relaxes the MDP environment requirement

of early tabular RL algorithms [60] to a partially observable MDP (POMDP) [50]. This means that DRL agents can learn policies in non-Markovian environments by parameterising only the Markovian variables [61]. Even so, performance is diminished in proportion to the lack of Markovian variables and state representations must be carefully constructed to include all of the information needed to properly inform each action.

When RL is extended to include multiple agents it is termed multi-agent RL (MARL). MARL enables a range of cooperative, competitive and mixed strategies to emerge between decentralised autonomous agents [6]. In large problem spaces MARL also permits divide-and-conquer strategies based on specialised agents handling specific sub-problems. Despite the advantages of MARL, it is also considerably more challenging than the single-agent setting. Multiple agents learning simultaneously present a much more difficult non-stationary learning problem [61]. In addition, assigning long-term rewards to specific actions is greatly frustrated when multiple agents contribute to, and detract from, goal accomplishment. The more agents coexist in a MARL environment, the more intractable it is from a learning standpoint [63].

2.2 Curriculum Learning

Curriculum learning (CL) [9] is an approach that can help agents learn more effectively by first training on simpler tasks and then gradually increasing the complexity. The main idea is to continuously adjust the difficulty of the task to just beyond the current capabilities of each agent, allowing them to learn new skills gradually. CL is a useful strategy for overcoming the specific difficulties of MARL, including non-stationary learning, and even emerges implicitly in certain cooperative and competitive settings [6, 45, 59].

2.3 Autonomous Cyber Defence

Motivated by the growing shortfall in cyber skills, scale and speed of response that is required to defend modern digital infrastructure from cyber attacks, ACD concerns the development of autonomous agents that actively defend computer networks and systems without the need for human intervention. Once threats have been detected and identified, ACD systems take actions to protect against, respond to, and recover from attacks. ACD systems can also deploy countermeasures including decoys, canaries and honeynets to gain defensive advantages against the adversary [31]. The most recent advances in ACD have emerged from the application of state-of-the-art DRL methods [7, 13, 18, 20, 25, 26].

An important aspect of the ACD landscape are cyber simulator environments, i.e, AI gyms [11], that enable autonomous agents to be trained without the scaling limitations of real systems and networks [12]. At least 16 different ACD simulation environments have been described in the literature, however only Cyber Operations Research Gym (CybORG) [1, 49] is both open source and designed specifically for training defensive RL-based agents [56]. CybORG is an AI gym, and research platform, providing a flexible scenario-driven environment backed by both simulated and emulated (i.e., Amazon Web Services) networks. A number of offensive red agents are included in CybORG to allow for benchmarking defensive strategies. Significantly, CybORG has hosted three competitive public

challenges which have motivated the development of autonomous agents by the wider academic community [5, 18, 20, 62].

Despite the apparent need for MARL approaches to scale ACD solutions to the intractably large observation spaces of real and internet-scale computer networks, only two cyber simulation environments currently support MARL [1, 30]. Of these, only the latest CybORG challenge [24] is defence focused.

3 ATTACKER MODEL

We consider a strong, yet realistic [54], adversary who has compromised the supply chain of a drone manufacturer. The adversary has covertly placed malware on the firmware installed onto every drone. The malware lays dormant until such a time as it is activated by the adversary, whereupon it launches a variety of harmful attacks including passively listening in on communication and actively compromising neighbouring drones. These drones have already been deployed and cannot quickly be replaced. The drones must be used as-is, despite having compromised firmware, to provide a temporary ad-hoc communication network for people on the ground. Whilst being unable to fully remove the malware, we will investigate the possibility of providing an active defence such that the network can still provide communications bandwidth.

4 ENVIRONMENT

To study resilient ad-hoc communication networks from the perspective of malware compromised drones, we use the CybORG AI gym and its latest public competition [24]. The third CAGE challenge, hereafter referred to as "the CAGE challenge" for brevity, tasks competitors with developing an autonomous defensive capability for an ad-hoc network of drones. The drones have been compromised during their manufacturing process and harbour malware that cannot easily be removed. Despite this, a communication network is desperately needed and the drones must be put into service. The CAGE challenge provides a MARL research environment for determining to what extent communication bandwidth can be maintained, by an ad-hoc network of drones, in the face of a disruptive malware attack.

The CAGE challenge uses the CybORG environment [1] to define a specific drone swarm scenario illustrated in Figure 2. The scenario comprises 18 drone hosts that are perpetually at risk from a malware that lies dormant in their firmware. The drones occupy a 100×100 two-dimensional space, have a communication radius of 30, and a maximum bandwidth of 100. At discrete time steps, a defensive blue team, an offensive red team and a neutral green team take turns to accomplish their objectives. The CAGE challenge provides both red and green teams as part of the environment, positioning for researchers the task of controlling the defensive blue team. The green team has one agent for each drone and is used to simulate the demand for communication bandwidth by operatives on the ground. Meanwhile the blue and red teams compete to control the drones, meaning a total of 18 agents are active across both teams at any moment. The motion of each drone, hence the overall network topology, is controlled by a randomised swarming algorithm unaffected by any of the teams. This ensures the task of actively mitigating malware attacks using software command and control (C2) tactics can be researched as an independent variable.

Reward	Event
-1	Comms. blocked
-1	Comms. intercepted
-1	Comms. dropped due to insufficient bandwidth
-1	No comms. route can be established
$-(18 * (t - 500))$	Complete compromise

Table 1: The CAGE challenge reward function.

The development of RL agents for this purpose is explicitly encouraged by the reward function, specified in Table 1, that is returned at every time step through the standard AI gym interface [11]. The reward function is also used to evaluate and benchmark the performance of defensive agents by averaging their scores over 1000 episodes, each up to 500 steps long. The maximum that can be scored in the CAGE challenge is 0, corresponding to flawless message delivery, and the minimum is -9000, equivalent to all messages failing for an entire episode. The results for 12 different approaches from 8 different teams are ranked publicly now that the challenge has officially closed [24]. Our Canary protocol, which we introduce in Section 6, currently ranks as the top performing agent[1].

4.1 Drone Platform

Each drone is a simulated embedded Linux system with a wireless interface that establishes a data link with all neighbouring drones

---

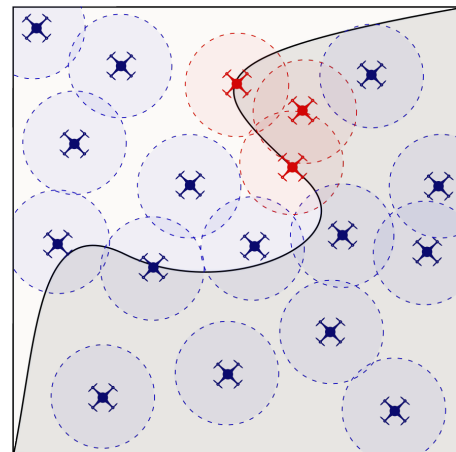[1]Preliminary challenge results correct as of 14th July 2023.

Figure 2: The CAGE Challenge Scenario. Blue and red drones are controlled by the defensive and offensive teams, respectively. Shaded areas illustrate the (not always) overlapping radii of each drone's communication range.

within communication range. All drones have the same firmware-level malware installed which, when activated, will passively and actively disrupt the swarm whilst also spreading to neighboring drones. To help keep drones secure, they include an Internet Protocol (IP) block list and a tool for detecting malicious network activity. The drones have 8 different actions that are selectively available to red, blue and green agents. Actions are either *local*, and act on the agent's host drone, or *remote*, and act on another drone in the swarm. Remote actions consume bandwidth and each drone has a maximum of 100 units to spare, per time step, before all additional actions (and messages) are dropped. To account for stochasticity in real network attack and defence scenarios, several of the actions occasionally fail.

In addition to the communication network provided by the drone network at large, an explicit low-bandwidth broadcast channel is also provided. At every time step, each drone can broadcast a 16 bit message to all neighbouring systems within range. Since each drone only benefits from a partial observation of the environment, inter-drone communication is essential to effectively coordinate a defensive response.

4.2 Green Team

The CybORG environment uses a green team to simulate the demand for communication bandwidth. At every time step, the green agent on each drone chooses a random recipient drone from the swarm and an arbitrary message is sent, consuming 1 unit of bandwidth from each drone along the shortest route (as determined by Dijkstras algorithm [16]). Importantly, noting the reward function in Table 1, a negative reward is expected regardless of the disruption caused by malware. Green agents cause negative reward by choosing target drones that are unreachable owing to the randomised network topology.

4.3 Malware

There is a firmware-level malware on each drone that cannot easily be removed. Once activated, occurring on a random drone with a 5% independent probability at each time step, the malware immediately removes the blue team's agent and replaces it with a high-privilege red team agent (controlled by the malware). The activated malware is able to fully control the drone beginning the same time step. Until the drone is reclaimed by the blue team, the malware behaves according to one of 6 different strategies which are chosen, and switched between, at random.

ExploitRandom. Exploit as many neighbouring drones as possible. The malware first remotely exploits a neighbouring drone at random yielding a low-privilege red agent presence on the target. Remote exploitation is flagged as malicious 50% of the time, independently by each host along the route), and also fails 50% of the time. If the low-privilege red agent session has not been removed by the next time step (i.e., by the blue agent), then the agent will escalate privileges to high (and cannot fail). A high privilege red agent represents complete activation of malware on the target.

ExploitFarthest. This strategy is the same as ExploitRandom. However, drones are targeted from farthest to nearest.

FloodRandom. Consumes all available bandwidth on the target drone which is chosen at random from the swarm. This action will never fail, provided the target is reachable using the drone network (i.e., is routable and the relevant IP addresses are not blocked). This action is always flagged as malicious by drones along the route.

FloodFarthest. This strategy is as FloodRandom however target drones are chosen from farthest to nearest.

BlockRandom. A neighbouring drone within direct communication range is selected at random and added to the block list. All subsequent attempts to reach the target via the infected host will be dropped until either 1. the end of the episode, or 2. a blue agent resumes control and unblocks the relevant IP address.

Intercept. The malware silently intercepts messages on behalf of the red team. This strategy is never flagged as malicious and does not fail.

Every turn that the malware executes one of these strategies, there is also an approximately 10% chance that the strategy will be swapped.

4.4 Blue Team

The blue team begins each episode with high privilege control of all 18 drones. The CybORG environment provides a standard AI gym interface for training RL agents which, as in Figure 1, includes an action space, a state space and a reward signal. An additional PettingZoo [53] wrapper is provided which provides a standard API for MARL environments.

Action Space. Defensive blue agents can use the following actions. The RetakeControl, BlockTraffic and AllowTraffic actions are each parameterised by the IP address of the target ($n = 18$), in total providing an action space of 56 (i.e., $1 + 18 + 18 + 18 + 1$).

- **RemoveOtherSessions.** Remove locally any low privilege red agent session from the host drone, succeeding 90% of the time. The agent can observe whether the action succeeded on the next time step.

- **RetakeControl.** Removes any red agents on the remote target, including high privilege, and returns control to the blue team. This action succeeds with a 75% probability, consumes 10 units of bandwidth from each drone along the network route, and is falsely identified as malicious activity 15% of the time.

- **BlockTraffic.** Adds the target drone to the IP block list, blocking all subsequent traffic until either 1. the end of the episode, or 2. a blue agent unblocks the relevant IP address.

- **AllowTraffic.** Removes a specific drone IP address from the block list.

- **Sleep.** Wait silently until this turn is over.

In addition to these actions, each drone can also explicitly broadcast a 16 bit message to all neighbouring drones within immediate communication range. It is possible to include the full message space in the action space, although this would result in an extremely large action space of 65592 (i.e., $56 + 2^{16}$).

Index	Description	Values
0	Was the last action successful?	True: 0 Unknown: 1 False: 2
1 : 18	Block list. Index $i + 1 \rightarrow$ drone i	Unblocked: 0 Blocked: 1
19	Malicious process on host	False: 0 True: 1
20 : 38	Malicious network events list Index $20 + i \rightarrow$ drone i	None: 0 1 event: 1 2 events: 2
39	Host x position	$0 \leq x \leq 100$
40	Host y position	$0 \leq x \leq 100$
41 : 108	For each neighbouring drone, a tuple (id_i, x_i, y_i, s_i) specifying the drone ID id_i, coordinates (x_i, y_i) and number of sessions s_i.	$0 \leq x \leq 100$
109 : 381	(Optional) A sequence of 16 bit messages from neighbouring drones.	$\{0, 1\}^{16 \times 17}$

Table 2: The observation space provided to blue agents by the standard CAGE challenge gym wrapper.

State Space. Blue agents partially observe the state of the environment with decreasing perception as distance from the host increases. Locally, blue agents can observe whether the previous action was successful, the contents of the block list, the position of the drone and whether any malicious activity has been detected. Within their communication range, blue agents can also observe the location of neighbouring drones and how many sessions (agents) their host has. Full details of the state space provided by the standard AI gym wrapper are shown in Table 2.

5 BASELINE

Before presenting our expert and MARL based agents for resilient communication networks, we motivate the need with a baseline evaluation of the CAGE challenge problem. In other words,

How difficult is the CAGE challenge?

Green Agent Noise. The reward function shown in Table 1 only includes negatives, so the maximum possible score is trivially zero. However, because communicating green agents choose recipient drones at random, these choices include unroutable destinations and incur a negative reward. To improve the upper bound on maximum score we evaluated the green agent in the CAGE challenge scenario but with malware completely disabled. Over 1000 episodes, the standard evaluation size, green agents alone induce an average score of -355.3 with a standard deviation of 253.2.

A Sleepy Defence. The trivial worst-case score is -9000. To improve our bound on the minimum score we should deem interesting,

we evaluate a team of blue agents who simply use the Sleep action. This provides no defence whatsoever and allows us to quantify the damage an unchecked malware is capable of. Over 1000 episodes, red and green agents with no defence result in an average score of -8292.1 with standard deviation 432.2.

Basic Strategies To further improve our bounds for a noteworthy defence, we evaluate the four basic expert agents included in the CAGE challenge source code. Once again we report the average score μ and standard deviation σ over 1000 episodes.

- **RemoveBlueDrone,** $\mu = -7396.8, \sigma = 599.0$.
  This agent always uses the RemoveOtherSessions action, trying to remove low-privilege red agents before they can escalate further.

- **RetakeBlueDrone,** $\mu = -8309.7, \sigma = 409.0$.
  This agent always uses the RetakeControl action on a random target from the swarm. This will restore blue agents on some compromised drones.

- **AdvancedRetakeBlueDrone,** $\mu = -8087.4, \sigma = 592.8$.
  Uses RetakeControl if malicious activity is detected on a neighbour, otherwise uses RemoveOtherSessions. This combines the previous two approaches and conserves bandwidth.

- **AdvancedBlockBlueDrone,** $\mu = -8312.2, \sigma = 419.0$.
  As AdvancedRetakeBlueDrone except BlockTraffic is used on the target with suspected malicious activity instead of Retake-Control.

Proximal Policy Optimisation (PPO) As our final baseline, we straightforwardly (i.e., with no changes to the environment) apply PPO to the CAGE challenge. PPO [43] is an efficient policy gradient [51] algorithm for DRL. PPO offers a comparatively simple implementation, helping to solve a broad range of difficult problems including competitive e-sports [38], emergent tool use [6] and microprocessor side-channel attack discovery [33]. In ACD, PPO has been used to successfully defend enterprise networks including top performances in the first two CAGE challenges [18, 27]. Furthermore, multi-agent PPO (MAPPO) achieves state-of-the-art performances in cooperative multi-agent environments [64].

Taking standard parameters from the literature [4], we use a fully connected neural architecture of width 256 and depth 2 with ReLU [21] activation between each layer. We train two models, first using the standard standard state space from Table 2 and, second using the explicit 16 bit communication channel to broadcast the action selected by each agent[2]. The second experiment significantly increases the model size, increasing the state space from 109 to 381 and the action space from 56 to 896. We evaluate the best checkpoints from both models after training each for 25 million time steps. The average model scores over 1000 episodes, alongside the other baselines, are shown in Table 3.

6 CANARIES AND WHISTLES

The results in the previous section highlight the importance of explicit communication and the favourable performance of even simplistic expert agents. When explicit communication is neglected,

---

[2]Using the `AgentCommsPettingZooParallelWrapper` included in CybORG

Baseline method	Avg. reward	Std. dev.
Sleep	−8292.1	432.1
RemoveBlueDrone	−7396.8	599.0
RetakeBlueDrone	−8309.7	409.0
AdvancedRetakeBlueDrone	−8087.4	592.8
AdvancedBlockBlueDrone	−8312.2	419.0
PPO	−7617.8	651.3
PPO with explicit comms.	−6745.3	945.4

Table 3: Baseline scores in the CAGE challenge.

PPO fails to outperform a basic approach based on greedily preventing privilege escalation on the local host (i.e., the RemoveBlueDrone method). Building on this insight, and to provide a state-of-the-art benchmark for the CAGE challenge, we developed a new expert blue agent that utilises a system of canaries and whistles to actively mitigate drone malware. Our Canaries and Whistles (CW) agent, described in Algorithm 1, is based on an explicit communication protocol that allows blue agents to keep track of neighbouring drones. Every time step that a host drone is not compromised, the blue agent broadcasts a canary message containing a unique identity number (UID). Meanwhile, blue agents also keep track of the canaries they receive at each step. Once a drone becomes compromised, its blue agent will be displaced by the red team and it will cease broadcasting canaries. Immediately, neighbouring drones still controlled by the blue team notice the missing Canary and apply a strategy to mitigate the malware. In addition, blue agents that detect a compromised neighbour become "whistleblowers" and broadcast the infected drone's UID alongside the host's own. Whistle messages are subsequently spread throughout the swarm by way of re-broadcasting.

Our CW agent was designed by domain experts to tackle this challenge specifically. A major determining factor in the design is the 16 bit limit applied to explicit drone messages. However, this is enough to encode two drone UIDs and a bit to distinguish between original and re-broadcast whistles. Indeed, 5 bits remain unused and could be used to enhance the algorithm further. The PAD and UNPAD algorithms used to encode and decode messages, respectively, can be found in Appendix A. Over 1000 episodes, the CW agent averages a score of −1577.7 with a standard deviation of 800.4. Based on the CAGE challenge competition phase [24], which is now closed, the CW agent provides a state-of-the-art performance.

7 JOINING THE OPERA

Inspired by the performance of our CW agent we experimentally investigate the gap between straightforward PPO, PPO with explicit communication, and the CW agent. We identify three improvements that, when combined, outperform the CW agent alone and establish a new state-of-the-art result. Specifically 1) addressing limitations in the CybORG observation space, 2) a CL method determined by the finding that it's easier to learn from "joining the opera" (i.e., join

---

Algorithm 1: Canaries and Whistles Agent

Args : d_{UID}, position
$t \leftarrow -1$, neighbours $\leftarrow \emptyset$, to-fix $\leftarrow \emptyset$
while *episode is not done* **do**
  /* For each time step in the episode */
  $t \leftarrow t + 1$
  if *this drone is infected* **then**
    | /* Blow whistle on self */
    | msg $\leftarrow$ PAD$(0, 0, d_{\mathrm{UID}})$
    | action $\leftarrow$ RemoveOtherSessions
    | **return** msg, action
  if $t > 1$ *and position has changed* **then**
    | position $\leftarrow$ new position
    | neighbours $\leftarrow \emptyset$
  else
    | **for** *each m in* {*received messages*} **do**
      | /* Parse explicit messages */
      | $m_{\mathrm{canary}}, m_{\mathrm{ovheard}}, m_{\mathrm{whistle}} \leftarrow$ UNPAD(m)
      | neighbours$[m_{\mathrm{canary}}] = t$
      | **if** $d_{UID} \neq m_{canary} \wedge m_{ovheard} = 1$ **then**
        | /* Notified of infected drone */
        | **if** *to-fix*$[m_{whistle}] = \emptyset$ **then**
          | to-fix$[m_{\mathrm{whistle}}] \leftarrow$ "block"
  overheard $\leftarrow 0$
  for *each id in* {*neighbours*} **do**
    | /* Check canaries */
    | **if** *neighbours*$[id] = t - 1$ **then**
      | /* Infected neighbour detected */
      | to-fix$[id] \leftarrow$ "block"
      | whistle $\leftarrow$ id
      | overheard $\leftarrow 1$
  if *to-fix* $\neq \emptyset$ **then**
    | /* Try to repair neighbouring drone */
    | $t_{ID} \xleftarrow{\$}$ {to-fix}
    | **if** *to-fix*$[t_{ID}] =$ "*block*" **then**
      | rnd $\leftarrow \{0.0 \leq \mathbb{R} \leq 1.0\}$
      | **if** $rnd \leq 0.225$ **then**
        | to-fix$[canary] \leftarrow$ "allow"
        | action $\leftarrow$ RetakeControl
      | **else**
        | to-fix$[canary] \leftarrow$ "retake"
        | action $\leftarrow$ BlockTraffic
    | **if** *to-fix*$[t_{ID}] =$ "*retake*" **then**
      | to-fix$[canary] \leftarrow$ "allow"
      | action $\leftarrow$ RetakeControl
    | **if** *to-fix*$[t_{ID}] =$ "*allow*" **then**
      | to-fix$[canary] \leftarrow \emptyset$
      | action $\leftarrow$ AllowTraffic
    | msg $\leftarrow$ PAD$(d_{\mathrm{UID}},$ overheard, whistle$)$
    | **return** action, msg
  /* If no neighbours are infected then greedily
    prevent local privilege escalation */
  action $\leftarrow$ RemoveOtherSessions
  return action, msg

the team of CW agents) than starting from scratch, and 3) defining a denoised reward function for optimised learning.

7.1 Observation Space

The CAGE challenge observation space provided by the CybORG environment is performance limiting when considering the behaviour of our expert CW agents. In particular, CW agents keep track of neighbouring drones over multiple time steps. This allows blue agents to keep track of which drones in the swarm could be compromised and respond accordingly. Considering the observation space in Table 2, it is not possible for the standard PPO agent we evaluated in Section 5 to learn the CW strategy. Despite the ability of deep RL algorithms to tolerate POMDP environments [50], they cannot parameterise non-Markovian variables. In other words, whether a drone is currently compromised (and needs blocking or retaking) is not independent of past observations given the present state. We address the limitations of the standard observation space with a new space, shown in Table 4, designed to incorporate the minimum features needed to learn a CW-like policy. We deliberately remove redundant variables (e.g., drone x, y positions) to improve learning efficiency, and include new stateful variables that provide a more Markovian observation. The most significant changes include providing an estimation of whether a neighbouring drone needs fixing (based on CW messages which are processed transparently in the background) and reporting the last action performed on each specific neighbour.

7.2 An Expert Curriculum

CL is an approach that has been successfully used to master a number of MARL environments [59], including in combination with

Index	Description	Values
0	Host Drone UID	$\{0, \ldots, 17\}$
1	Was the last action successful?	False: 0 True: 1
2	Last action number	$\{0, \ldots, 55\}$
3 : 20	Last action type on neighbour Index $i + 2 \rightarrow$ drone i UID	RetakeControl: 0 BlockTraffic: 1 AllowTraffic 2
21	Malicious process on host	False: 0 True: 1
22 : 39	Block list. Index $i + 22 \rightarrow$ drone i	Unblocked: 0 Blocked: 1
40 : 57	Neighbour needs fixing Index $i + 22 \rightarrow$ drone i UID	False: 0 True: 1
58 : 330	(Optional) A sequence of 16 bit messages from neighbouring drones.	$\{0, 1\}^{16 \times 17}$

Table 4: The revised observation space designed to allow learning CW-like policies.

PPO [6, 64]. We identify a CL method, Opera, that allows for a gradual increase in the complexity of the CAGE challenge environment, helping to mitigate the difficulties (e.g., non-stationary learning and combinatorial explosion of the action space) associated with MARL environments. The essential observation is that, with a high-performing expert agent to hand, DRL agents can be incrementally introduced to the environment. We begin training just a single DRL agent, amongst a swarm of CW agents, and then gradually increase the number of learning-based agents until maximum performance is reached. For this purpose we create a series of wrappers for the CybORG environment that allow us to specify an arbitrary mixture of CW and PPO agents. To prevent bias in our models we ensure they are hosted randomly every episode, placing different agents on different drone hosts.

7.3 A denoised reward

The CAGE challenge scoring function includes a negative penalty for communications that are unroutable. As discussed in Section 5, this stochastically decreases the maximum agent score based on the ad-hoc network topology and randomised choosing of message recipients. Unfortunately, this directly antagonises DRL-based approaches which are optimised exclusively on the maximisation of cumulative reward. This is further exacerbated by the partial observability of each agent, which additionally obfuscates the relationship between state, action and reward. To train our new agents we modify the CybORG environment to distinguish between messages that are unroutable and those that are blocked, dropped or intercepted. We then remove the negative reward for unroutable messages during training (i.e., we still evaluate our models using the reward in Table 1), providing improved learning.

8 EVALUATION

Here we evaluate the performance of our learning-based PPO policies, investigate our proposed improvements to the observation space, scrutinise our mixed Opera method of CL (i.e., combining CW and PPO agents) and compare the behaviour of our learning-based agents with their expert collaborators. Our best performing model overall utilises a 7:11 split of CW:PPO agents. The learning-based agents are trained using an expert curriculum ranging from 1:17 to 14:4. We also use our improved observation space from Table 4 and the denoised reward strategy. Averaged over 1000 episodes, our best mixed Opera scores -1487.9 with a standard deviation of 626.1.

8.1 Observation Space

Our proposed observation space allows learning-based agents to keep track of whether neighbouring drones are likely to be infected with malware. In Figure 3, we compare the learning curves of the baseline PPO result from Section 5 and our new observation space from Table 4. In both cases we train a single policy (with width 256 and depth 2) for all 18 agents simultaneously. Each policy took approximately 3.5 hours to train on a single machine with a 24-core Intel i9-13900K, 64 GB of memory and a 24GB RTX 4090 GPU. Table 5 shows the corresponding policy scores, indicating that our changes do indeed improve performance. We believe that
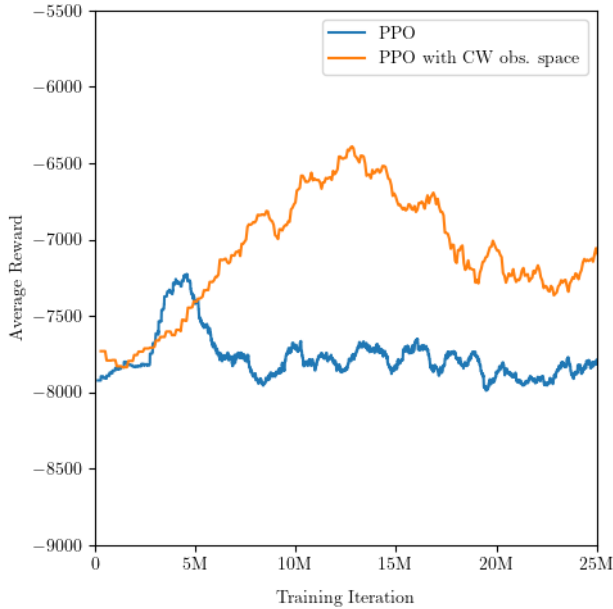
Figure 3: Comparing the learning performance of baseline PPO with and without our improved observation space.

significant further score improvements in this setting (i.e., training 18 learning-based agents simultaneously) are unlikely without utilising methods for centralised coordination such as centralised training decentralised execution (CTDE) [64].

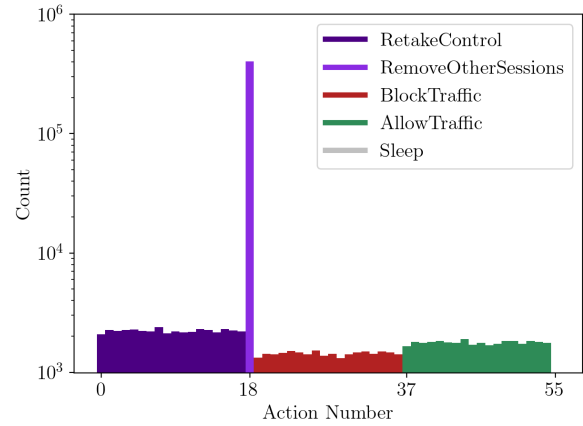| Observation Space	Avg. reward	Std. dev.
As in Table 2	−7617.8	651.3
As in Table 4	−6884.4	845.4

Table 5: Observation-space related score changes.
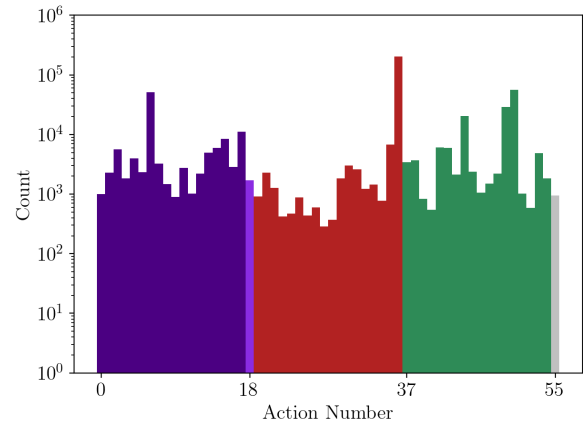
8.2 Mixed Opera

Our Opera method incrementally introduces learning-based agents to a population of expert influencers until they learn all of the skills they need (or are capable of). We evaluate our approach in Figure 5 by calculating the average score as CW agents are incrementally substituted for their learning-based contemporaries. The policy is trained iteratively using the expert curriculum method, beginning with 1 agent and gradually increasing the number to 14. Training was stopped at this point as beyond 14 performance declined rather than improved. As a baseline, we also substitute an inactive agent that takes no actions whatsoever (i.e., the Sleep Agent).
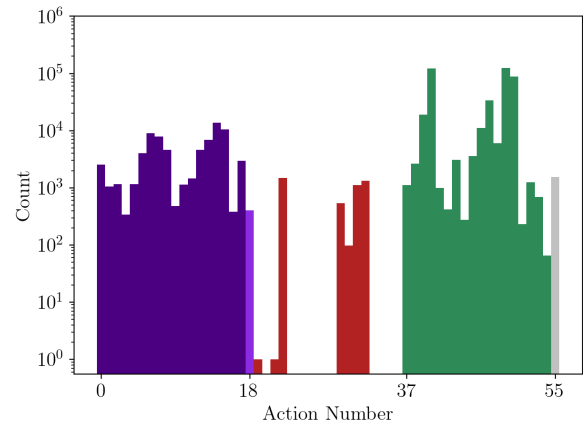
8.3 Policy Analysis

As shown in Figure 4, we investigate the coarse-grained policy differences between the expert CW agent, the baseline PPO agent and our mixed Opera agent by counting the actions chosen over

(a) Expert CW agent.

(b) Baseline PPO agent.

(c) Mixed Opera PPO agent.

Figure 4: Action distributions for the CW, baseline PPO and mixed Opera PPO agents. For normalisation, actions are sampled from a single agent chosen randomly each episode.
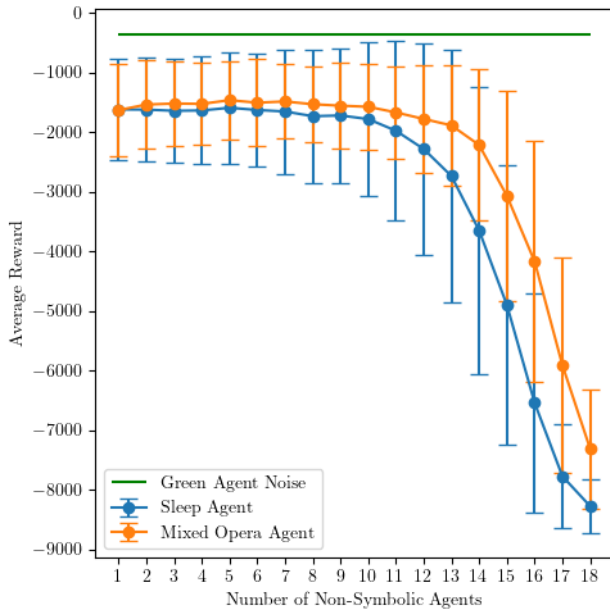
Figure 5: The change in average score and standard deviation as the number of learning-based agents is increased from 1/18 to 18/18.

Reward	Avg. reward	Std. dev.
Noisy	−6770.5	1064.0
Denoised	−6884.4	845.4

Table 6: Denoised reward related score changes.

1000 randomised episodes. The CW agent distinctively chooses RemoveOtherSessions, locally removing low-privilege red agents placed by neighbouring infected drones, just over 80% of the time. The remaining actions are distributed more evenly between Retake-Control, BlockTraffic and AllowTraffic. As the red team's trojan maliciously blocks traffic to disrupt communication, AllowTraffic is chosen more often than BlockTraffic. In comparison, the baseline PPO agent chooses from the available actions (including the sleep action which does nothing) with a more equal probability.

9   DISCUSSION

Compared with previous ACD gyms, the third CAGE challenge represents a substantially more challenging learning environment. This is evident, in addition to the results presented in this paper, from the challenge leaderboard which currently scores expert agents from multiple teams significantly higher than any of the learning-based agents. Nevertheless, multi-agent algorithms are well motivated for ACD because they can potentially divide and conquer impractically large action spaces into more manageable sub tasks attended by specialised agents. It is remarkable however, that the CAGE challenge has a large number of agents relative to the wider literature where 20 is usually the maximum (e.g., [47, 58]) and 2-6 is

far more common (e.g., [6]). The only other MARL ACD environment currently supports a maximum of two agents [30] trained competitively. Regarding our approaches to MARL in the CybORG environment, it is notable that we have not applied any approaches based on a centralised critic. The success of PPO in multi-agent environments has generally otherwise been based on the CTDE paradigm [64].

As in the previous two CAGE challenges, the issue of determining the optimal score is a difficult one. There currently is no principled approach to knowing how close various approaches are, leaving the possibility that significant performance improvements are yet to be made. Certainly, although it can be understood as resulting from the impact of drone malware, there is a gap between the ceiling generated by green agents (as discussed in Section 5) and the state-of-the-art presented here. Finally, we note that the design of the CybORG gym could be improved regarding parallelisation. A major bottleneck in the training and evaluation of our agents is the need to wait for a single thread to simulate the environment before returning new observations and rewards to each agent. Multi-agent AI gyms should ideally be designed with parallelisation to match the algorithms that will train agents within them.

10   RELATED WORK

To the best of the authors knowledge no other publications have applied MARL to the problem of resilient ad-hoc communications in the presence of malware infected hosts [37]. Indeed, there is only one other open source MARL environment for ACD beyond CybORG [30, 56] and both were released only in the last 12 months.

Concerning single-agent RL for ACD, the literature is considerably more established. Foley et al. [18] showed the effectiveness of hierarchical PPO, based on choosing a specialised sub-agent every time step, in tackling two differentiated red agents in the first CAGE challenge. In later work, Foley et al [20] further determine that the hierarchical DRL supervisor can be replaced with a more effective bandit-based algorithm for choosing the best agent. In the same work, the authors also study the explainability of DRL agents for ACD, apply an ablation study, and calculate SHapley Additive exPlanations (SHAP) values [32] to determine the importance of features in the CybORG environment. There are several recent surveys of autonomous and automated cyber defence [2, 28, 37, 44] including by Vyas et al. [56] who comprehensively survey the current landscape of autonomous cyber operations gyms and develop a set of requirements they use to evaluate each environment. Beyond CybORG [1], CyberBattleSim [52] and Yawning Titan (YT) [3] are notable for also providing open source simulation based environments for attacking and defending computer networks, respectively. Beyond ACD, the RL paradigm is increasingly accomplishing remarkable results in a range of systems security environments including evading hardware trojan detection [23], finding new injection attacks [19, 57], closed-box malware generation [48], and microprocessor cache-timing vulnerability discovery [33].

Related to our mixed Opera method of CL, Campbell et al. [13] propose a curriculum framework for autonomous network defense using MARL. Yu et al. [64] study the performance of multi-agent PPO in cooperative environments and discover surprisingly strong performance that is competitive to conventional off-policy methods.

Piterbarg et al. [40] study the NeurIPS 2021 NetHack Challenge and also discover a gap between expert and learning-based approaches. The authors use different methods based on a hierarchical action space, neural architecture improvements and imitation learning to bridge the gap for their agents.

11 CONCLUSION

In this paper we present the full details of our state-of-the-art expert CW agent for decentralised autonomous malware resilience in an ad-hoc drone network. We use CW to address some of the deficiencies in prior learning-based agents, identifying three specific methods for doing so: (1) Providing a (more) Markovian observation space, (2) implementing CL by gradually increasing the proportion of learning-based agents, and (3) creating a noise-free reward function, allowing us to considerably close the gap in performance between expert and learning-based agents. Finally, we present a new state-of-the-art result in the third CAGE challenge based on a mixed Opera of expert and learning-based Canary agents.

In future work we will consider emergent communication and whether protocols similar to CW could be learned automatically without the need for an explicit specification. In addition, since red agents will inevitably try to use our protocols against us, exploring communication learned (e.g., using autocurricula [6]) under competitive adversarial pressure is a compelling direction. More broadly in ACD, many open problems remain including improved methods for coordinating shared situational awareness in decentralised agents; scalable observation spaces able to deal with an arbitrary number of neighbouring drones, algorithms for lifelong learning, and robustness for autonomous agents.

ACKNOWLEDGMENTS

REFERENCES

[1] 2022. Cyber Operations Research Gym. https://github.com/cage-challenge/CybORG. Created by Maxwell Standen, David Bowman, Son Hoang, Toby Richer, Martin Lucas, Richard Van Tassel, Phillip Vu, Mitchell Kiely, KC C., Natalie Konschnik, Joshua Collyer.

[2] Amrin Maria Khan Adawadkar and Nilima Kulkarni. 2022. Cyber-Security and Reinforcement Learning – A Brief Survey. *Engineering Applications of Artificial Intelligence* (2022).

[3] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. 2022. Developing Optimal Causal Cyber-Defence Agents via Cyber Security Simulation. In *Workshop on Machine Learning for Cybersecurity as part of the Proceedings of the 39th International Conference on Machine Learning (ML4Cyber '22)*.

[4] Marcin Andrychowicz, Anton Raichuk, and Others. 2021. What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. In *International Conference on Learning Representations (ICLR '21)*.

[5] Andy Applebaum, Camron Dennler, Patrick Dwyer, Marina Moskowitz, Harold Nguyen, Nicole Nichols, Nicole Park, Paul Rachwalski, Frank Rau, Adrian Webster, and Melody Wolk. 2022. Bridging Automated to Autonomous Cyber Defense: Foundational Analysis of Tabular Q-Learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security (AISec'22)*.

[6] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2020. Emergent Tool Use From Multi-Agent Autocurricula. In *International Conference on Learning Representations (ICLR '20)*.

[7] Liz Bates, Chris Hicks, and Vasilios Mavroudis. 2023. Reward Shaping for Happier Autonomous Cyber Security Agents. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), Copenhagen, Denmark* (2023).

[8] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. 2017. dr0wned – Cyber-Physical Attack with Additive Manufacturing. In *11th USENIX Workshop on Offensive Technologies (WOOT '17)*.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning (ICML '09)*.

[10] Katharina L. Best, Jon Schmid, Shane Tierney, Jalal Awan, Nahom M. Beyene, Maynard A. Holliday, Raza Khan, and Karen Lee. 2020. *How to Analyze the Cyber Threat from Drones: Background, Analysis Frameworks, and Analysis Tools.* Online. RAND Corporation.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym.

[12] Anthony Burke. 2020. *Robust artificial intelligence for active cyber defence.* Online. The Alan Turing Institute.

[13] Robert G. Campbell, Magdalini Eirinaki, and Younghee Park. 2023. A Curriculum Framework for Autonomous Network Defense using Multi-agent Reinforcement Learning. In *2023 Silicon Valley Cybersecurity Conference (SVCC)*.

[14] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. 2014. A Large-Scale Analysis of the Security of Embedded Firmwares. In *23rd USENIX Security Symposium (USENIX Security '14)*.

[15] Vishal Dey, Vikramkumar Pudi, Anupam Chattopadhyay, and Yuval Elovici. 2018. Security Vulnerabilities of Unmanned Aerial Vehicles and Countermeasures: An Experimental Study. In *17th International Conference on Embedded Systems (VLSID '18)*.

[16] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* (1959).

[17] Benjamin C. Fernando, Greg Beams, and Reece Rivera. 2019. *Security Analysis of DJI Phantom 3 Standard.* Online. Massachusetts Institute of Technology. https://courses.csail.mit.edu/6.857/2016/files/9.pdf

[18] Myles Foley, Chris Hicks, Kate Highnam, and Vasilios Mavroudis. 2022. Autonomous Network Defence Using Reinforcement Learning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*.

[19] Myles Foley and Sergio Maffeis. 2022. Haxss: Hierarchical Reinforcement Learning for XSS Payload Generation. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 147–158. https://doi.org/10.1109/TrustCom56396.2022.00031 ISSN: 2324-9013.

[20] Myles Foley, Mia Wang, Zoe M, Chris Hicks, and Vasilios Mavroudis. 2022. Inroads into Autonomous Network Defence using Explained Reinforcement Learning. In *Proceedings of the Conference on Applied Machine Learning in Information Security (CAMLIS '22)*.

[21] Kunihiko Fukushima. 1975. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics* (1975).

[22] Edward J Glantz, Frank E Ritter, Don Gilbreath, Sarah J Stager, Alexandra Anton, and Rahul Emani. 2020. UAV use in disaster management. In *Proceedings of the 17th ISCRAM Conference*.

[23] Vasudev Gohil, Hao Guo, Satwik Patnaik, and Jeyavijayan Rajendran. 2022. ATTRITION: Attacking Static Hardware Trojan Detection Techniques Using Reinforcement Learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*.

[24] TTCP CAGE Working Group. 2022. TTCP CAGE Challenge 3. https://github.com/cage-challenge/cage-challenge-3.

[25] Kim Hammar and Rolf Stadler. 2020. Finding Effective Security Strategies through Reinforcement Learning and Self-Play. In *2020 16th International Conference on Network and Service Management (CNSM '20)*.

[26] Yi Han, Benjamin I. P. Rubinstein, Tamas Abraham, et al. 2018. Reinforcement Learning for Autonomous Defence in Software-Defined Networking. In *Decision and Game Theory for Security*.

[27] John Hannay. 2022. CAGE Challenge 2 Winning Submission. PPO + Greedy Decoys. https://github.com/john-cardiff/-cyborg-cage-2.

[28] Yunhan Huang, Linan Huang, and Quanyan Zhu. 2022. Reinforcement Learning for feedback-enabled cyber resilience. *Annual Reviews in Control* (2022).

[29] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. 2014. Unmanned Aircraft Capture and Control Via GPS Spoofing. *Journal of Field Robotics* (2014).

[30] Thomas Kunz, Christian Fisher, James La Novara-Gsell, Christopher Nguyen, and Li Li. 2023. A Multiagent CyberBattleSim for RL Cyber Operation Agents. In *9th International Conference on Computational Science and Computational Intelligence (CSCI '22)*.

[31] Andrew Lohn, Anna Knack, Ant Burke, and Krystal Jackson. 2023. *Autonomous Cyber Defence. A roadmap from lab to ops.* Online. Centre for Emerging Technology and Security (CETaS) at The Alan Turing Institute.

[32] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS'17)*.

[33] Mulong Luo, Wenjie Xiong, et al. 2023. AutoCAT: Reinforcement Learning for Automated Exploration of Cache-Timing Attacks. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[34] Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, et al. 2023. Faster sorting algorithms discovered using deep reinforcement learning. *Nature* (2023).

[35] Yassine Mekdad, Ahmet Aris, Leonardo Babun, Abdeslam El Fergougui, Mauro Conti, Riccardo Lazzeretti, and A. Selcuk Uluagac. 2023. A survey on security and privacy issues of UAVs. *Computer Networks* (2023).

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2015. Human-level control through deep reinforcement learning. *Nature* (2015).

[37] Thanh Thi Nguyen and Vijay Janapa Reddi. 2021. Deep Reinforcement Learning for Cyber Security. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[38] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, and Others. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:1912.06680

[39] Julien Perolat, Bart De Vylder, et al. 2022. Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning. *Science* (2022).

[40] Ulyana Piterbarg, Lerrel Pinto, and Rob Fergus. 2023. NetHack is Hard to Hack. arXiv:2305.19240

[41] Theodore Reed, Joseph Geis, and Sven Dietrich. 2011. SkyNET: A 3G-Enabled Mobile Attack Drone and Stealth Botmaster. In *5th USENIX Workshop on Offensive Technologies ((WOOT '11))*.

[42] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347

[44] Mohit Sewak, Sanjay K. Sahay, and Hemant Rathore. 2022. Deep Reinforcement Learning in the Advanced Cybersecurity Threat Detection and Protection. *Information Systems Frontiers* (2022).

[45] David Silver, Aja Huang, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* (2016).

[46] David Silver, Thomas Hubert, Julian Schrittwieser, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* (2018).

[47] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. Individualized Controlled Continuous Communication Model for Multiagent Cooperative and Competitive Tasks. In *International Conference on Learning Representations (ICLR '19)*.

[48] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. 2022. MAB-Malware: A Reinforcement Learning Framework for Blackbox Generation of Adversarial Malware. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*.

[49] Maxwell Standen, Martin Lucas, David Bowman, Toby JRicher, Junae Kim, and Damian Marriott. 2021. CybORG: A Gym for the Development of Autonomous Cyber Agents. In *1st International Workshop on Adaptive Cyber Defense (IJCAI '21)*.

[50] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*.

[51] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NeurIPS '99)*.

[52] Microsoft Defender Research Team. 2021. CyberBattleSim. https://github.com/microsoft/cyberbattlesim. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei.

[53] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. 2021. PettingZoo: Gym for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS '21)*.

[54] U.S. DoD 2022. *Securing Defense-Critical Supply Chains. An action plan developed in response to President Biden's Executive Order 14017*. Online. U.S. DoD.

[55] Junia Valente and Alvaro A. Cardenas. 2017. Understanding Security Threats in Consumer Drones Through the Lens of the Discovery Quadcopter Family. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy (IoTS&P '17)*.

[56] Sanyam Vyas, John Hannay, Andrew Bolton, and Pete Burnap. 2023. Automated Cyber Defence: A Review. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2023).

[57] Salim Al Wahaibi, Myles Foley, and Sergio Maffeis. 2023. SQIRL: Grey-Box Detection of SQL Injection Vulnerabilities Using Reinforcement Learning. In *32nd USENIX Security Symposium (USENIX Security '23)*.

[58] Guihong Wang and Jinglun Shi. 2019. Actor-Critic for Multi-agent System with Variable Quantity of Agents. In *IoT as a Service*.

[59] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. 2020. From Few to More: Large-Scale Dynamic Multiagent Curriculum Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* (2020).

[60] Christopher J.C.H. Watkins and Peter Dayan. 1992. Technical Note: Q-Learning. *Machine Learning* (1992).

[61] Steven D. Whitehead and Long-Ji Lin. 1995. Reinforcement learning of non-Markov decision processes. *Artificial Intelligence* (1995).

[62] Melody Wolk, Andy Applebaum, Camron Dennler, Patrick Dwyer, Marina Moskowitz, Harold Nguyen, Nicole Nichols, Nicole Park, Paul Rachwalski, Frank Rau, and Adrian Webster. 2022. Beyond CAGE: Investigating Generalization of Learned Autonomous Network Defense Policies. In *Reinforcement Learning for Real Life Workshop in the 36th Conference on Neural Information Processing Systems (RL4RealLife '22)*.

[63] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*.

[64] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS '22)*.

[65] Ruotong Yu, Francesca Del Nin, Yuchen Zhang, Shan Huang, Pallavi Kaliyar, Sarah Zakto, Mauro Conti, Georgios Portokalidis, and Jun Xu. 2022. Building Embedded Systems Like It's 1996. In *29th Annual Network and Distributed System Security Symposium (NDSS '22)*.

[66] Ce Zhou, Qiben Yan, Yan Shi, and Lichao Sun. 2022. DoubleStar: Long-Range Attack Towards Depth Estimation based Obstacle Avoidance in Autonomous Systems. In *31st USENIX Security Symposium (USENIX Security '22)*.

APPENDIX

Supporting information.

A   CW AGENT DEPENDENCIES

The CW agent algorithm (Algorithm 1) depends on two subroutines to encode and decode explicit communication messages. These are provided by Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: PAD

Args : canary, overheard, whistle
/* Pad canary, whistle and overheard into a
   16 bit binary string */
1 Convert canary to binary and pad to 5 bits.
2 Convert overheard to 1 bit binary.
3 Convert whistle to binary and pad to 5 bits.
4 **return** (whistle $<<$ 11) $\vee$ (overheard $<<$ 6) $\vee$ canary

---

Algorithm 3: UNPAD

Args : $m \in \{0, 1\}^{16}$
/* Recover the canary, whistle and overheard
   values from a 16 bit binary string */
1 canary $\leftarrow$ $m \wedge$ 0x3F
2 overheard $\leftarrow$ ($m >> 6$) $\wedge$ 0x1
3 whistle $\leftarrow$ ($m >> 11$) $\wedge$ 0x3F
4 **return** (canary, overheard, whistle)
