Reward Shaping for Happier Autonomous Cyber Security Agents

Elizabeth Bates
ebates@turing.ac.uk
The Alan Turing Institute

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

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

ABSTRACT

As machine learning models become more capable, they have exhibited increased potential in solving complex tasks. One of the most promising directions uses deep reinforcement learning to train autonomous agents in computer network defense tasks. This work studies the impact of the reward signal that is provided to the agents when training for this task. Due to the nature of cybersecurity tasks, the reward signal is typically 1) in the form of penalties (e.g., when a compromise occurs), and 2) distributed sparsely across each defense episode. Such reward characteristics are atypical of classic reinforcement learning tasks where the agent is regularly rewarded for progress (cf. to getting occasionally penalized for failures). We investigate reward shaping techniques that could bridge this gap so as to enable agents to train more sample-efficiently and potentially converge to a better performance. We first show that deep reinforcement learning algorithms are sensitive to the magnitude of the penalties and their relative size. Then, we combine penalties with positive external rewards and study their effect compared to penalty-only training. Finally, we evaluate intrinsic curiosity as an internal positive reward mechanism and discuss why it might not be as advantageous for high-level network monitoring tasks.

CCS CONCEPTS

• **Security and privacy** → *Network security*; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Reinforcement Learning, Reward Shaping, PPO, Network Security, Autonomous Network Defense

ACM Reference Format:
Elizabeth Bates, Vasilios Mavroudis, and Chris Hicks. 2023. Reward Shaping for Happier Autonomous Cyber Security Agents. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), November 30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3605764.3623916

1 INTRODUCTION

Advanced Persistent Threats pose a significant challenge for defenders as they involve various attack tactics and vectors over prolonged periods, thus impeding event correlation, detection, and mitigation. Defenders need to monitor the network for abnormal behaviour and take immediate remediation actions when a system gets compromised or traces of an adversary are found. Such actions usually involve a large array of tools for monitoring, scanning and reverting systems in a benign state. Similarly, the adversary needs to combine a variety of tools to perform reconnaissance, gain a foothold, escalate their access and move laterally towards their goal (e.g., exfiltration, impact). In this arms race, the adversary is at an advantage. They can gather information about the target system or network in advance, prepare their steps and rapidly move through the network or remain present stealthily in compromised systems for a long time. In contrast, the defender needs to be able to fend off all attacks quickly and patch the vulnerabilities that enabled the intrusion in the first place. At present most of the incidence response relies on human operators that handle events raised by monitoring software.

Deep reinforcement learning (DRL) is actively researched and has been shown to excel in interactive tasks that cannot easily be solved using analytical solutions. For example, human and even super-human levels of performance have been reported in a range of tasks including board, video and strategy games [29, 30, 32, 42], as well as autonomous driving [39] and robotics [24]. In such RL environments, rewards are typically positive and provide a consistent signal to the agent of how well they are doing during training (e.g., point gathered in a game).

In the context of cybersecurity, DRL has been successfully applied to autonomously defend computer networks [12, 13, 20, 38] which, as discussed above, is a highly interactive task where the defending agent needs to intervene promptly and efficiently. However, cybersecurity environments have certain unusual characteristics that make training more challenging. Similar to human operators, the agent is given only penalties (cf. rewards) when compromises occur or when an adversary impacts a critical system. Consequently, the reward signal is sparsely distributed and focuses on unwanted events (e.g., breaches) rather than proactive defense behaviors (e.g., scanning systems regularly).

This work studies the effect of these reward signal characteristics and investigates various *reward shaping* techniques that could improve the final performance of the agent or/and its sample efficiency (i.e., training speed). Overall, reward shaping for autonomous network defense tasks is an under-explored area; our main contributions are:

- We investigate whether the relative magnitude of the penalties has an effect on sample efficiency or the performance of the agent.
- We introduce positive rewards along the penalties of the environment and study their effect in comparison to a baseline (penalty-driven) agent, as well as the impact of their relative magnitude.
- We study curiosity, a sophisticated internal reward technique that addresses reward sparsity by motivating exploration intrinsically, and compare its performance to a non-curious RL algorithm.

AISec '23, November 30, 2023, Copenhagen, Denmark
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0260-0/23/11.
https://doi.org/10.1145/3605764.3623916

Our trained agents and augmented environments will be available online under an open source license.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning which aims to learn the optimal behaviour (known as the optimal policy) in a given environment through experience and subsequent rewards or penalties for particular actions taken, given their consequence. Sutton and Barto (2018) [48] describe this learning system as "hedonistic" in its approach, as the focus is to maximise a special signal from the environment. Advances in the RL field in the last decade have demonstrated the ability and potential of RL agents, beyond the traditional RL algorithms like Monte-Carlo or tabular Q-Learning. These earlier approaches, whilst able to converge to optimal behaviours, do so in computationally and time intensive ways, and are not scalable to the complexity of many relevant environments. The introduction of deep Learning into the RL space has allowed agents to learn increasingly complex policies in environments with extensive, continuous state spaces and actions [17]. Deep RL has yielded successful algorithms such as Trust Region Policy Optimisation (TRPO) [41] and Proximal Policy Optimisation (PPO) [42] and use of Deep Q-Networks (DQN) [31].

2.2 Proximal Policy Optimisation (PPO)

Proximal Policy optimisation (PPO) is a policy gradient based method which has achieved much success across RL literature [36, 43, 56], and is considered state of the art alongside algorithms like Soft Actor-Critic (SAC) [18] and the use of DQNs [31]. Policy gradient based methods formulate an objective function such that its gradient is an estimator of the policy gradient. The objective function (Equation 1) is defined as the expected rewards over a trajectory τ, which is dependent upon the policy π which is, in turn, dependent upon parameters θ, also known as the network weights. This can be reformulated (Equation 2) and differentiated to be written to use the advantage function (A_{π_θ}), which determines whether a certain action is better to take than another, given the agent is in a particular state.

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} R(\tau) = \sum_{\tau} P(\tau; \theta) R(\tau) \tag{1}$$

$$\begin{gathered}\nabla_\theta J(\theta) = \mathbb{E}_{\pi_0} \left[\nabla_\theta \log \pi_\theta(s,a) A_{\pi_\theta}(s) \right] \\ \text{where } A_{\pi_\theta}(s) = Q_{\pi_\theta(s,a)} - V_{\pi_\theta}(s)\end{gathered} \tag{2}$$

PPO is an extension on the earlier work of Trust Region Policy Optimisation (TRPO) [41], aiming to reduce the complexity of TRPO whilst retaining the reliable performance and data efficiency. TRPO and PPO both go on to maximise the objective function, but in a constrained way such the policy updated are not "destructively large" [42]. TRPO uses hard constraints, whilst PPO uses a clipping method (Equation 3) where the ratio of the old to new policy must not exceed $1 + \epsilon$ or $1 - \epsilon$, depending on the value of the advantage term. The Actor-Critic strategy [42] (Equations 4, 5) can be introduced such that the actor models the policy and the critic models the state-value function in two Deep Neural Networks, and iteratively helps each network improve as training progresses.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{3}$$

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S \left[\pi_\theta \right] (s_t) \right], \tag{4}$$

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{5}$$

Where L_t^{VF} is critic loss, c_1 is the critic coefficient, c_2 is the entropy coefficient, S represents an entropy bonus and γ is the discount factor.

2.3 Reward Shaping

When applying reinforcement learning techniques to solve a problem that has a sparse reward signal, it can often take the agent a long time to learn the optimal behaviour. This can be seen in tasks that are episodic in nature e.g., a reward is only given to the agent on reaching the final goal state. Reward shaping is the notion of adding small rewards along the agent's trajectory to encourage faster learning and convergence to the optimal policy [11]. Deciding how to shape these intermediate rewards so that they support learning the best behaviour is critical and non-straighforward. If incorrectly applied, the agent could learn sub-optimal policies, and become effectively 'distracted' from the real objective of the task. There are a variety of reward shaping techniques used throughout the literature, such as potential-based [53], count-based [7, 16, 50], curiosity-based [9, 22, 37] and distance-based [34, 52]. The effects of reward shaping can help improve the efficiency of RL agents during training and improve sample efficiency [16]. Sample efficiency in this context refers to "the number of time steps in which the algorithm does not select near-optimal actions" [49]. Altering the external rewards of an environment is a form of extrinsic reward shaping. However, there is very little discussion (if any) on augmenting reward signal in specialized cyber defence environments. The environment used in this paper has a sparse and penalty-driven reward signal, and is thus suitable for exploring the use of intrinsic and extrinsic reward shaping strategies and their effects.

2.4 Intrinsic Curiosity

Intrinsic Curiosity is a mechanism for encouraging an RL agent to explore novel states, or states that it is less certain about. An Intrinsic Curiosity Module (ICM) was first introduced by Pathak et al. (2017) and involves adding a curiosity-driven intrinsic reward ($r_t^{(i)}$) to the agent's reward signal alongside extrinsic ($r_t^{(e)}$) environmental rewards [37]. The sum of these two rewards ($r_t = \eta_e r_t^{(e)} + \eta_i r_t^{(i)}$) are to be maximised as the policy improves, and the weighting of either reward can be altered using η_e and η_i, where η_i=1-η_e and η_i, η_e are between 0 and 1 [27]. The ICM consists of a two neural networks, the inverse model and the forward model. First, state s and s_{t+1} are fed into a feature embedding layer. The inverse model takes in feature representations of state s and s_{t+1}, $\phi(s_t)$ and $\phi(s_{t+1})$ and outputs the predicted action given a state s_t and subsequent state s_{t+1}. The feature encoded state is provided as input for the forward model, as well as the action a_t to predict the feature representation of the next state $\hat{\phi}(s_{t+1})$. The difference between this prediction and the actual value of $\phi(s_{t+1})$ is the internal reward signal [37] added to the RL algorithm, in the case of this paper, PPO.

3 THREAT MODEL

We assume an adversary who aims to breach critical infrastructure systems in a enterprise or production computer network. For example, such a scenario could involve the computer network of a manufacturing facility with both employee computers and critical systems controlling the production line. We assume a sophisticated persistent adversary that moves *laterally* across the systems of the network by progressively compromising hosts, and can potentially have prior information about the topology of the network.

This modus operandi is compatible with advanced persistent threats (APTs) where adversaries with a foothold in the network, discover, exploit, access, and escalate their privileges in neighbouring systems. Such adversaries typically gain the initial foothold through a non-critical host of the network by employing an indirect strategy (e.g., phishing emails, a spear-phishing text, voice deepfakes, or a watering hole attack) [25, 28]. The scope of the defences we consider in this work begins after the adversary gains a foothold. Thereafter the adversary employs various tools to perform reconnaissance, gain unauthorized access to systems, escalate their privileges, bypass security controls and launch attacks against the liveness of critical servers.

The adversary is not able to bypass network traffic routing limitations and cannot access systems that are not directly linked to the system they currently reside in. For example, if the network is compartmentalized into subnets A, B and C and A and C not directly connected, the adversary is not able to attacks C's systems from systems in A. Moreover, the adversary can gain a foothold only through systems that are connected to the Internet and are either vulnerable or used in such a way that an indirect attack might be applicable (e.g., receiving emails). In contrast, the adversary cannot gain a foothold through systems that are offline or non-vulnerable.

4 CYBORG ENVIRONMENT

For our experiments, we use the CybORG environment that simulates (and emulates) an enterprise computer network [2, 4]. CybORG is one of the most commonly used environments for training cyber defense agents [6, 13, 35, 55] and provides a simulator paired with an emulator. This is to address the reality gap, a generalization problem that occurs when training RL agents in a simulated environment. CybORG's emulator runs on Amazon Web Services (AWS) and was used to validate that all the actions, observations and state transitions are consistent between the simulator and the emulator [2].

A CybORG *scenario* defines the network topology, the subnets and firewalls of the network along with the systems included in each subnet, their type (e.g., user host, enterprise server) and the services each of them exposes. Each episode is played between two actors: a defensive "blue" agent and an attacker "red agent". The environment makes available a number of actions to each agent and they take turns in interacting with network. Table 1 summarizes the individual actions. Each action can be applied on a host of the network. The "Decoy" action sets up a honeypot for the adversary in the selected host and has seven variants (DecoyApache, DecoyFemitter, DecoyHarakaSMPT, DecoySmss, DecoySSHD, DecoySvchost, DecoyTomcat) depending on the service it mimics. To increase realism, each action has some likelihood to fail even when applied in a valid case (e.g., restore action applied on a compromised system). The red agents can perform an exploit action, and must specify which service by which to exploit the host. These decoy services set up by the blue agent can reveal red agent behaviour as any attempt to exploit a host through a decoy-service immediately fails.

Table 1: The available actions that can be taken by the Blue and Red agents in a CybORG scenario.

Actions	
Blue Defensive Agent	**Red Adversarial Agent**
1. Monitor a host	1. Scan a subnet for hosts
2. Analyse processes on a given host	2. Discover Network Services of a host
3. Remove Red access, given the red agent has not escalated their privileges	3. Exploit a service on a port
4. Restore a host to its initial configuration	4. Escalate privilege on a host
5. Set up decoy services	5. Disrupt the services on the operational server

The observation space available to the defender is probabilistic in nature and is a vector consisting of 52 bits, with 4 bits corresponding to each host on the network. The first 2 bits indicate if the host has been scanned or exploited by the adversary, and the remaining two bits represent what level of access the adversary has on that host, either none, user or administrator [3, 13].

To measure the success of the defenders, the CybORG environment uses a scoring function. The scores are all negative, hence sometimes referred to as penalties. The defender receives these penalties every time the red agent gains administrator access to a system, with the magnitude varying depending on how critical that host is to the enterprise network. The possible negative scores range from -0.1 to -1. The blue agent also receives a penalty of -10 if the red agent successfully impacts the "Operational Server" (green system in the operational subnet in Figure 1). Overall, the penalties reflect the importance of the system in the enterprise network. Due to the disruption to benign user operations, the defender receives a penalty of -1 when they perform the "restore" action on any host. The relationship between scores and rewards is discussed further in Section 6.1.

To make training RL agents more straightforward, the environment provides a wrapper that realises an OpenAI Gym interface (www.gymlibrary.dev, gymnasium.farama.org) enabling agents to act as attackers, defenders or both [8, 10]. This interface is compatible with all major RL frameworks.

5 THE CAGE CHALLENGE

This CybORG environment is released and maintained by the TTCP CAGE Working Group and was used in all three of the "Cyber Autonomy Gym for Experimentation" (CAGE) challenges [1, 3, 14]. Each CAGE challenge introduced a (gradually more complex) cyber defense scenario implemented on CybORG and lasted for a period of three months each. During this period teams train and submit defense agents competing for the best score. The goal of the competition is to incentivise further research in autonomous defense agents. The CAGE environments and scenarios are have been used in several past work on autonomous decision making in a computer network environment [13, 35, 54, 55].

In this work, we use the second CAGE scenario as it is the most recent challenge with a fixed network topology. In contrast, the third CAGE challenge uses a mesh network where transmission links are constantly changing. This version of the problem is more difficult and requires a multi-agent RL solution. So far no efficient RL agents have been published (for CAGE 3) to be used as a baseline for our experiments and the top-performing solutions for the challenge were based on heuristics.

In the CAGE 2 scenario, the CybORG network is compartmentalized in three subnets, separated by firewalls (Figure 1): Subnet 1 consists of 5 user hosts, subnet 2 of three enterprise servers and subnet 3 of an operational server and 3 operational hosts. The firewalls prevent direct movement from subnet 1 to subnet 3, and thus an adversary can only get to subnet 3 via subnet 2. This topology is identical to the one used in the first of the iteration of the CAGE challenge.

Each episode begins with the adversary (red agent) gaining a foothold on one of the user hosts in subnet 1. Subsequently, they can choose specific actions to move laterally through the network towards their goal which is the Operational Server (green system in operational subnet in Figure 1). This server maintains a service critical to the manufacturing facility that uses the enterprise network. Once the adversary has reached subnet 3, they seek to disrupt the operation of the Operational Server for as long as possible [3] (Denial of Service attack). This setup and the goals of the adversary is compatible with the threat model introduced in Section 3.

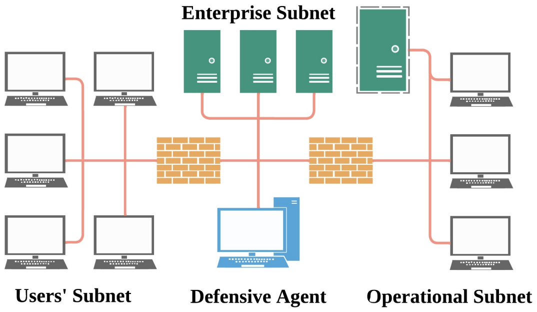

Figure 1: A visualisation of the CybORG Network topology [10], the network used in the CAGE challenge 1 and 2, featuring 3 subnets separated by firewalls. The green system in the operational subnet is the *Operational Server* which is a critical system.

The challenge introduces also two scripted adversaries (with the same end-goal but different strategies) that each defender has to protect the network against: the "meander" and the "Bline" adversary. *Meander* has no prior knowledge of the network's topology, and thus thoroughly explores each subnet before moving to the next. In contrast, *Bline* has prior information of the network's layout and follows an almost optimal (but randomized) trajectory towards the operational server. The defender (blue agent) has to take remediation (e.g., Remove action) and preventative (e.g., setting up decoy services) actions to contain the adversary. Note that full removal of the adversary from the foothold user host is not possible.

5.1 CybORG Versions

The CAGE Challenge 2 code base has been updated since the submission deadline of the competition on the 18th of July 2022. The updates included two separate bug fixes in August 2022 fixing: 1) an exploit failing on User Hosts 3 and 4, and 2) a bug in the initial observation of the agent that indicated an adversary having scanned on all hosts. The top-performing (state-of-the-art) solution [19] was able to achieve their best scores using the version of the CAGE 2 challenge available at the time of submission (SHA: f12ff493cd3b8327cab93f645d71330e7b282377). However, for our experiments we use the newest version (commit on the 19th of August 2022) with the two bug fixes (SHA: 9421c8e985627810c5cac2abf0bcb62dfa6749fc). Section 6.2 discusses further the impact of these patches to the pre-existing solutions and the steps we have taken to establish a reliable baseline.

6 CONSIDERATIONS & METHODOLOGY

Before we move on with our reward shaping experiments, we establish a baseline using a state-of-the-art solution from the literature. We use the winning agent of the CAGE 2 Challenge that combines the PPO Algorithm with greedy decoys [19]. As we discuss in Sections 5.1 and 6.2, we retrain this agent to use the latest version of the environment. Thereafter, we refer to this retrained agent as *baseline*.

6.1 Score Terminology & Augmentation

In the context of this paper and the CAGE 2 Challenge environment, rewards refer to the individual numerical value given to the agent from the environment after taking an action, moving from one state to the next. The rewards available in the CAGE 2 environment were 0.0, -0.1, -1.0 and -10.0, depending on the action taken and importance of the servers or hosts compromised. Since all the rewards in the CAGE 2 Challenge environment are negative, we also refer to them as "penalties". These baseline extrinsic rewards are altered in Experiment sets 1 and 2, and are therefore referred to as the "augmented rewards". Moreover, "score" refers to the cumulative rewards the defending agent acquires after a given amount of steps defending the network. In the CAGE challenge 2, this score was evaluated on episodes with 30, 50 and 100 steps.

For our experiments, we introduced minor adjustments in the scoring function of CybORG. In particular, we exposed a method that alter the structure of the extrinsic rewards whilst also retaining the outputs of the original scoring function. In other words, at each step we output both the original rewards and the augmented reward from the reward shaping experiment. Tracking both of these values means that the agent can use the altered rewards to learn, whilst also being able to track what the original rewards would have been for the same defense strategy. For example, this allows us to plot the original rewards for the behaviour learnt from the altered rewards, so that the learning curves and the best average model scores can be easily compared to the baseline PPO model in the unmodified CAGE 2 challenge environment.

6.2 State of the Art Defender

The baseline used in our experiments combines an Actor-Critic PPO agent and greedy decoys to achieve the best model scores (Table 2). Prior works had begun using a Dueling Deep Q-Network (DDQN), but replaced this with a PPO agent due to concerns of stability during training. However, the plain PPO implementation struggled to target decoys for specific hosts, in that potentially unhelpful decoy actions could be taken for that specific host (e.g., twice placing the same decoy, selecting a less effective decoy). This implementation included having a single decoy action selected greedily from all the possible decoys. Thus, an adjustment was made to change how decoy actions were chosen. The final solution included a greedy decoy placement strategy which greedily used only the most effective decoy from the nine available decoys for each host [19]. The hyperparameters used for the baseline Actor-Critic PPO algorithm can be seen in Table 3 and these are used for all the experiments throughout this paper.

As discussed in Section 5.1 the CAGE challenge solutions were trained on a flaw version of the environment. However, the performance of the baseline agent was not reproducible in the patched version. To confirm that the discrepancy was due to the bug fixes in the code, we cloned a the competition version of the CAGE 2 Challenge repository and trained the baseline agent there. As seen in Table 2, the reported baseline results where then reproduced confirming that changes in the environment affected the agent's performance. In our experiments, we use the most recent version of the CAGE 2 Challenge code base with the bugs fixed, and hence the new scores in the second row of Table 2 are used as the baseline (SOTA) to compare reward shaping model scores against.

6.3 Hardware, Algorithms & Hyperparameters

As discussed, we use the top-performing agent from CAGE 2 (i.e., baseline) as a starting point for our rewards shaping experiments. For training (actor-critic PPO) we use the Adam optimiser to update the actor and critic networks' weights. PPO is a policy gradient method that is typically sample efficient and often matches or even outperforms other state of the art (SOTA) methods [29, 42, 47]. The specific hyperparameters used for PPO in all our experiments are listed in Table 3. We used 5 Azure Standard D48s v3 (48 vcpus, 192 GiB memory) virtual machines to train our agents in parallel.

7 EXPERIMENTAL EVALUATION

Three sets of experiments were conducted, covering both intrinsic and extrinsic reward shaping ideas. The first two cover extrinsic reward shaping ideas by exploring scaling up the magnitude of the rewards and adding a mix of positive and negative rewards to the environment. The third focuses on the results of applying an Intrinsic Curiosity Module (ICM) to explore how it affects the PPO learning agent in the CAGE 2 context. From these experiments, we aim to: 1) gain a deeper understanding of the effect of extrinsic rewards in a cyber environment, 2) make comparisons with the impact of intrinsic rewards, and 3) investigate whether reward shaping alterations can lead to improved sample complexity or a faster convergence to optimal behaviour.

Due to the inherent variance in training of RL models, quantitative results are more reliable when averaged over several training sessions [21]. In each experiment, the same model is trained from scratch 15 times for 75,000 episodes each. We then evaluate the scores and standard deviations of the trained models for 1000 episodes of 30, 50 and 100 steps. Although this is a large number of iterations, it matches the iterations used in the evaluation script of the CAGE 2 Challenge. There was no signficant improvement in models after 50,000 episodes of training, thus all the figures are plotted up to the 50,000 episode-mark.

Each experiment was conducted twice, against the Red Bline agent and then the Red Meander agent. The third sleep agent was also evaluated and the scores were consistently 0, as expected, and thus we do not include this agent's results in any tables or figures. In all experiments a larger reward indicates better performance. Moreover, to examine the statistical significance between our experimental results we use the P-value [5] between the performance means, and report accordingly on each experiment. Each experiment was run in parallel (Section 6), taking approximately 24 hours each. Our trained agents and augmented environments will be publicly available as open source.

7.1 Exp. 1: Extrinsic - Magnitude Change

In this experiment, we study how the magnitude of the extrinsic penalties applied to a blue agent affects how sample-efficiently the agent learns to protect the network. The baseline penalties are the initial values set in the CAGE environment (see Section 6.1). We alter these and generate a new tuple of rewards with 1) the manually augmented penalties, and 2) the baseline penalties that the agent would receive for the same behaviour. Keeping both of these values means that the agent can learn from the augmented rewards, but the performance of the agents can be fairly compared on the basis of the same scoring function.

We perform three scaled-reward experiments: one with normalised rewards, one with rewards scaled up by one order of magnitude and one scaled up disproportionately to the importance of the system (any red action/presence on the operational server) rewards. In the context of this experiment, the normalised set of augmented rewards means that the reward values are between 0 and -1, as opposed to that of the baseline CAGE environment which are between 0 and -10. See Table 4 for the augmented rewards in comparison to the CAGE baseline.

The main incentive for exploring adjusting the rewards in this manner are that preliminary experiments in the OpenAI Gym's Mountain Car environment demonstrated changes in the learning curve when similar experiments were trialled. The Mountain Car environment is relevant as it too is a sparse reward environment with mostly negative rewards, much like the CAGE 2 Environment.

From our review of the literature, there are few works that perform a systematic investigation of the effects of reward magnitude. Reward normalisation is a strategy used in other RL experiments,

Table 2: Scores of the top-performing defense agent in the competition and the patched versions of CAGE 2 Challenge for episodes of 30, 50 and 100 steps. There are notable differences in the Bline Adversary scores, but much less of a difference for the Meander Red Agent versions. The standard deviations were not included on the competition results page [10] and are denoted as N/A.

CybORG Env. Version	Average scores and standard deviation											
	Bline Red Agent						Meander Red Agent					
	30		50		100		30		50		100	
	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ
Competition (f12ff49)	-3.47	N/A	-6.41	N/A	-13.76	N/A	-5.64	N/A	-8.69	N/A	-16.6	N/A
Latest (9421c8e)	-4.232	2.247	-7.596	3.190	-15.993	5.491	-5.624	1.345	-8.894	2.224	-16.996	4.285

Table 3: Hyperparameters for the Actor-Critic PPO algorithm used in all our experiments.

Hyperparameters	Values
Learning rate (α)	0.002
Epochs (K)	6
Minibatch Size (in timesteps)	20,000
Discount (γ)	0.99
GAE parameters (λ)	1.0
Betas	[0.0, 0.990]
Clipping Coefficent (ϵ)	0.2
C_1	0.5
C_2	0.01

such as in the initial Deep Q-Learning (DQN) paper [31] which trained an RL agent to successfully play a multitude of Atari games. The rationale in [31] was to try to maintain the same hyperparameters of the agent when training across all Atari games (some games had much larger rewards than others). Another benefit is that clipping the rewards this way limited the scale of the error derivatives, which could potentially lead to more efficient training. This concept of large rewards leading to large error derivatives and potentially slowing the agent's training was also the reason that the largest negative reward for both experiment 2 (Scaled-up by one order of magnitude) and experiment 3 (disproportionately scaled-up) was capped at -100.

Figure 2 and Table 5 show that the two experiments with scaled up the rewards performed the best for the first 10,000 episodes of training. Both then converged to average scores of -16.132 and -15.680 for 100 episodes for the scaled-up by one order of magnitude rewards and the disproportionately scaled-up rewards experiments respectively. The disproportionately scaled-up rewards experiment achieved a marginally better average score (-15.993) than the baseline PPO agent. As seen in Figure 2, both experiments that augmented the rewards by scaling them up were able to achieve the same performance (the performance difference in both cases was not statistically significant, they had P-values of >0.95 for both experiments in comparison to the baseline score) with the baseline agent but in a more sample efficient manner.

Table 4: Reward intervals for the set of extrinsic reward shaping experiments on penalty magnitude change. Baseline rewards are the rewards used in the original CAGE 2 environment while the rest of the columns include augmented rewards.

Reward Intervals			
Baseline	**Normalised**	**Scaled-up by 1 order of magnitude**	**Scaled-up disproportionately**
0.0	0.0	0.0	0.0
-0.1	-0.01	-1.0	-0.1
-1.0	-0.1	-10.0	-10.0
-10.0	-1.0	-100.0	-100.0

In comparison, the learning curve of the normalised rewards experiment took significantly longer to achieve an average score (-15.946) that was not significantly different statistically than that of the baseline -15.993 (for P=0.98). The slower convergence is likely due to the very small (and sparse) rewards that provide only an attenuated signal to the agent that is not significant enough to promote learning efficiently.

However, it can be seen in Figure 3 and the same table that these trends were not consistent against the Red Meander agent. Both the normalised and the disproportionately scaled-up rewards experiments converged at scores much worse than the baseline result, at -43.695 and -42.210 for a 100 time step episode respectively compared to the baseline's -16.996. In contrast, the scaled-up rewards experiment was able to achieve a much better score of -17.237. All the experiments, except the normalised one, performed similarly up until approximately 6,000 episodes of training, which is where the disproportionately scaled-up rewards experiment began to plateau, whilst the baseline and scaled-up to one order of magnitude rewards continued to improve, but at a slower rate. As with the Bline agent, the normalised rewards agents took the longest to converge to its best average score.

7.2 Exp. 2 - Extrinsic - Positive Rewards

We now explore introducing positive rewards in an otherwise entirely penalty-driven learning problem. Two sets of models were trained, one with a small positive reward of 0.1 added, and the second with a larger positive reward of 1.0. The positive reward was introduced by adding a reward for all the times the agent would usually receive a reward signal of 0.0, i.e. when the defensive blue

Table 5: Average scores and standard deviations for Experiment set 1 for 30, 50 and 100 steps of the trained blue agent defending against either the Bline red agent or the Meander red agent. Average baseline scores for models trained only with PPO and no augmented rewards are included for comparison. The PPO baseline remains the best average score for attack from the Meander Red Agent, however the differences in average scores were not statistically significant to the baseline scores for any experiments, except for the normalised and disproportionately scaled-up rewards experiments.

Experiment	Average scores and standard deviation for each set of steps											
	Bline Red Agent						Meander Red Agent					
	30		50		100		30		50		100	
	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ
Baseline	-4.232	2.247	-7.596	3.190	-15.993	5.491	-5.624	1.345	-8.894	2.224	-16.996	4.285
Normalised rewards	-4.253	2.177	-7.591	3.088	-15.946	5.271	-7.077	1.458	-16.995	2.642	-43.695	4.760
Scaled-up rewards	-4.247	2.255	-7.652	3.254	-16.132	5.690	-5.630	1.359	-8.893	2.231	-17.237	8.202
Disproportionately scaled-up rewards	-4.179	2.135	-7.473	3.009	-15.680	4.875	-6.814	1.450	-16.267	2.681	-42.210	6.544

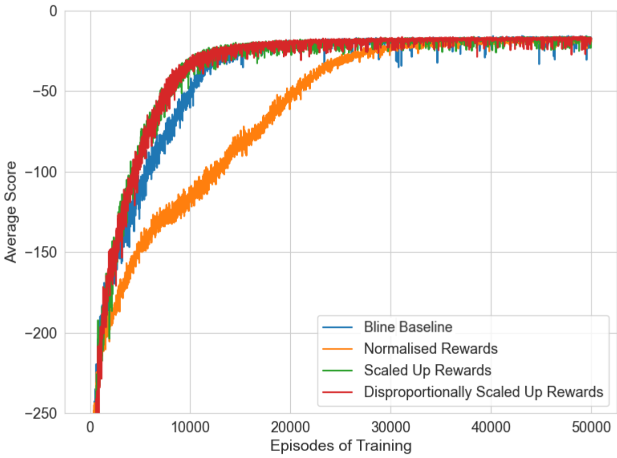

Figure 2: The average training curves for the Experiment set 1, where the attacker is the Bline Red agent. We conduct three magnitude-altering experiments using normalised, scaled-up by one order of magnitude and disproportionately scaled-up rewards. Each curve is the average of 15 models trained for 50,000 episodes. The two sets of scaled up rewards seem to achieve better scores than the baseline and the normalised rewards throughout the start of the training, then they all plateau at similar scores. Score in this context refers to the cumulative sum of rewards for 100 timesteps.

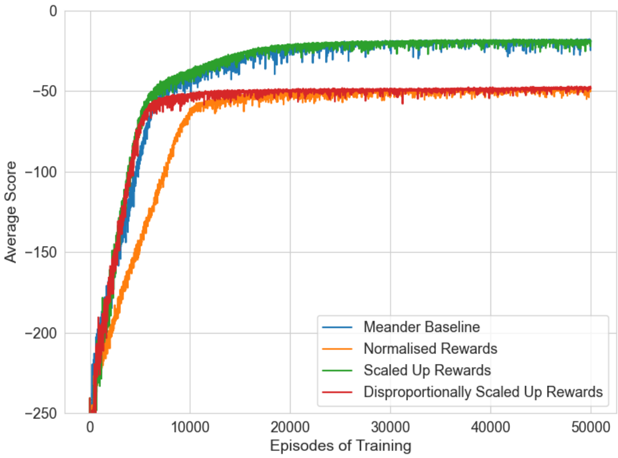

Figure 3: The average training curves for the Experiment set 1, where the attacker is the Meander Red agent. We examine three magnitude-altering strategies using normalised, scaled-up by one order of magnitude and disproportionately scaled-up rewards. Each curve is the average of 15 models trained with each set of altered rewards (for 50,000 episodes). In this case, only the scaled-up by one order of magnitude experiment performed equally to the baseline, and the rest of the experiments plateaued at much lower scores of approximately -43. Score in this context refers to the cumulative sum of rewards for 100 timesteps.

agent is performing actions proactively but those reveal no adversarial presence. See Table 6 for the positive augmented rewards in comparison to the CAGE baseline. Similarly with Experiment 1, our motivation was our preliminary experiments with positive and negative rewards in the simpler mountain car environment as well as the reward shaping techniques introduced in [23]. More specifically, [23] investigated limiting their rewards between 0 and 8, and between -8 and 8 such that both successes and failures were rewarded/penalised proportionally. This notion of including both positive and negative rewards was an interesting idea and although applied in a different environment, it seemed like a relevant avenue to explore, especially since these negative rewards are set in the CAGE 2 environment and have not been studied previously.

The results of these two experiments in Table 7 show that the addition of both small and large positive rewards does minorly

Table 6: Reward Intervals for the set of extrinsic reward shaping experiments on adding positive rewards. Both the experiments augment the reward value that is usually 0.0 in the baseline CAGE 2 challenge environment, to a small positive reward of 0.1 or a larger reward of 1.0.

Reward Intervals		
Baseline	**Small positive reward**	**Large positive reward**
0.0	0.1	1.0
-0.1	-0.1	-0.1
-1.0	-1.0	-1.0
-10.0	-10.0	-10.0

improve upon baseline results for the Bline agent, though this improvement is not statistically significant. The addition of the small rewards does achieve a score for 30 timesteps of -4.072 which is better than the baseline's -4.232. Moreover, the experiments outperform the baseline at 50 and 100 time steps; the baseline at 50 was -7.596 and the agent's was -7.352 and for 100 the baseline was -15.993 against the agent's -15.849. All the standard deviations were also smaller than the baselines across all the Bline agent at 30, 50 and especially at 100, indicating more reliable performance when small positive rewards are added to this environment.

For scores against the red Meander agent, the addition of the small positive rewards produced scores identical to the baseline agent (statistically non-significant difference), while the addition of the larger positive rewards performed worse than the baseline, especially for the 100 timesteps evaluation.

By adding a positive reward for states where no red adversaries are encountered, an additional positive incentive is provided to keep the network healthy (in addition to the environment's penalties). Moreover, this positive addition makes the environment's rewards less sparse i.e., the agent is now receiving relevant rewards much more frequently. Finally, this shows that rewarding the absence and penalising the presence of the adversary *did not* incentivise the agent to game the rewards by e.g., avoiding scanning potentially compromised systems.

7.3 Exp. 3 - Intrinsic - ICM

This experiment studies intrinsic reward shaping, thus all the extrinsic rewards are reset back to their initial values used in the baseline models, ranging from 0.0, -0.1, -1.0 and -10.0. An additional model was created using an identical architecture to the Actor-Critic PPO model used in all the previous experiments with the addition of an Intrinsic Curiosity Module (ICM) incorporated into it. We retained the hyperparameters as reported in Section 3 and set five additional ones specific ICM (Table 8). β was set to the value recommended in the initial Intrinsic Curiosity paper [37] while η, α_i, η_e and η_i were set to match the configuration of mainstream implementations of a PPO agent with ICM [1] [2] [27].

As seen in Figure 6, the inclusion of ICM does not improve the PPO agent's training speed or final performance, it in fact performs worse than the baseline agents. The standard deviation for each set of scores was larger for both the red Bline and Meander agent for almost all the sets of steps. This can be seen again in Figure 6 in the ICM learning curves that show much more variance throughout training. This larger variance could be explained by the greater emphasis on exploration that the ICM agents have, especially in the early stages of training. This larger variance indicates that baseline PPO agent can achieve scores close to the average more reliably.

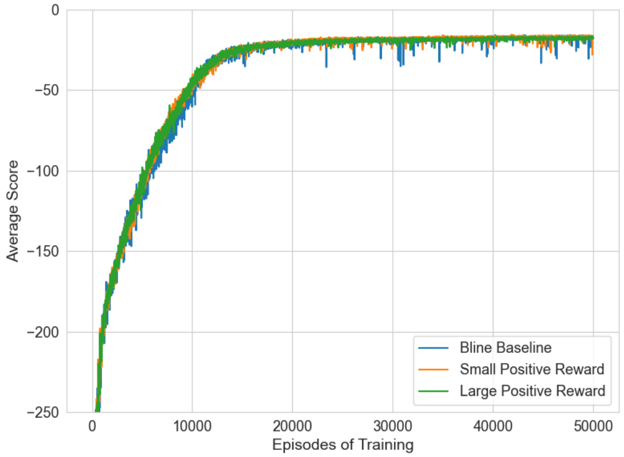

Figure 4: The average training curves for the Experiment set 2, where the attacker is the Bline Red agent. Each curve is the average of 15 models trained with each set of altered rewards (for 50,000 episodes). The learning curves for both the small and large addition of positive rewards experiments are very similar in rate of score improvement as the baseline curve, though both experiments converge to slightly better values (see Table 7). Score in this context refers to the cumulative sum of rewards for 100 timesteps.

The lack of positive effect from introducing internal rewards through ICM is not entirely unexpected. It is likely that the baseline agent has been able to learn all defense strategies that were within its capabilities (given its observation space and the capacity of the policy neural network) and thus ICM could not help uncover new strategies. Moreover, curiosity has reportedly aided the Bline PPO agent in the CAGE Challenge 1 environment [12] and in the CAGE 2 environment prior to the submission deadline [13]. The subsequent bug fixes mentioned in 5.1 and the addition of the greedy decoys could have made curiosity redundant. In particular, minimising the need for more thorough exploration (in CAGE 2) and ruling out potentially unorthodox defense strategies that took advantage of the implementation bugs (in CAGE 1). Interestingly, [13] also found that curiosity-based internal rewards did not benefit the defender's performance against the Meander agent.

Finally, to ensure that the magnitude of the rewards (η) was not the impeding ICM from assisting, we ran a small scale experiment (single model, 30,000 steps) with $\eta = 1$. In comparison, an η of 0.01 is commonly used in RL frameworks[3] and when applying curiosity in environments of similar reward sparsity with exclusively negative

[1] https://github.com/adik993/ppo-pytorch
[2] https://github.com/chagmgang/pytorch_ppo_rl
[3] https://github.com/adik993/ppo-pytorch

Table 7: Average scores and standard deviations for Experiment set 2 for 30, 50 and 100 steps of the trained blue agent defending against either the Bline red agent and the Meander red agent. Average baseline scores for models trained only with PPO and no augmented rewards are included for comparison. The scores that are statistically significantly better are in bold.

Experiment	Average scores and standard deviation for each set of steps											
	Bline Red Agent						Meander Red Agent					
	30		50		100		30		50		100	
	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ
Baseline	-4.232	2.247	-7.596	3.190	-15.993	5.491	-5.624	1.345	-8.894	2.224	-16.996	4.285
Small positive rewards	**-4.072**	2.033	**-7.352**	2.967	**-15.457**	4.955	-5.640	1.354	-8.890	2.210	-17.005	5.776
Large positive rewards	-4.210	2.210	-7.475	3.128	-15.849	5.111	-5.700	1.369	-8.963	2.270	-17.205	6.429

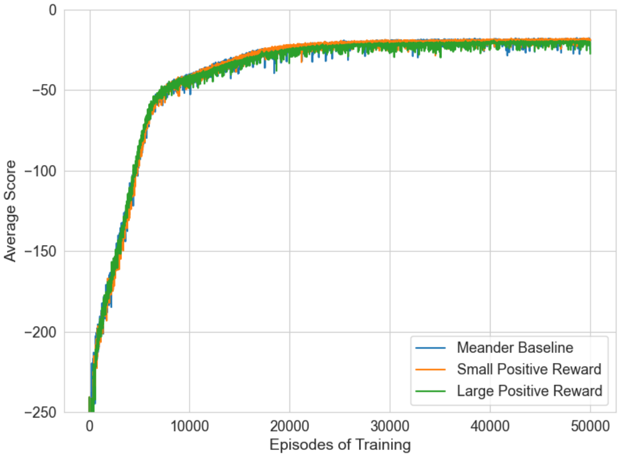

Figure 5: The average training curves for the Experiment set 2, where the attacker is the Meander Red agent. Each curve is the average of 15 models trained with each set of altered rewards (for 50,000 episodes). The two sets of scaled up rewards seem to achieve better scores than the baseline and the normalised rewards throughout the start of the training, then they all converge to similar scores. Again, the learning curves for both the small and large addition of positive rewards experiments are very similar in rate of score improvement as the baseline curve, though they do converge to perform slightly worse than the baseline.

rewards e.g., Mountain Car [4] [8]. Even after this substantial amplification of the rewards, ICM did not improve neither the sample efficiency nor the final performance of the baseline model.

8 RELATED WORK

The application of ML for autonomous cyber operations is still an emerging field. The capabilities and potential of autonomous agents trained using techniques such as RL enable defending networks autonomously at scale and speed, thus providing new efficient methods to preserve network security.

Table 8: Additional hyperparameters table for the ICM used in Experiment 3.

Hyperparameters	Values
ICM Learning rate (α_i)	0.001
ICM beta (β_i)	0.2
Reward scale (η)	0.01
External Reward Factor (η_e)	0.9
Internal Reward Factor (η_i)	0.1

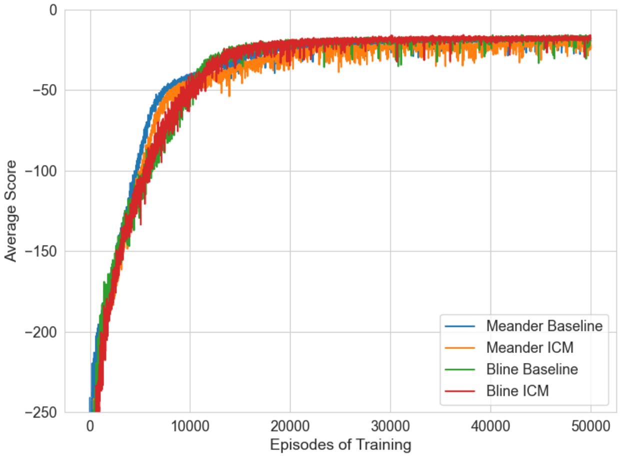

Figure 6: The average training curves for the Experiment 3, with both the Red Meander and Bline agent curves plotted. Each curve is the average of 15 models trained with each set of altered rewards (for 50,000 episodes). Score in this context refers to the cumulative sum of rewards for 100 timesteps. The inclusion of ICM has not significantly affected the learning speed and has lower average scores throughout training than the baseline PPO agent could achieve against either the red Bline or Meander agent.

To train such agents, various network simulation and emulation environments have been proposed in the literature e.g., CyberBattleSim [51], FARLAND [33], CANDLES [38]and Galaxy [40].

[4] https://www.gymlibrary.dev/environments/classic_control/mountain_car

Table 9: Average scores and standard deviations for Experiment 3 ICM for 30, 50 and 100 steps of the trained blue agent defending against either the Bline red agent or the Meander red agent. Average baseline scores for models trained only with PPO and no augmented rewards are included for comparison.

Experiment	Average Scores and standard deviation for each set of steps											
	Bline Red Agent						Meander Red Agent					
	30		50		100		30		50		100	
	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ	Score	σ
Baseline	-4.232	2.247	-7.596	3.190	-15.993	5.491	-5.624	1.345	-8.894	2.224	-16.996	4.285
ICM	-4.483	2.576	-8.297	4.368	-18.127	9.990	-6.030	1.503	-9.742	2.843	-18.967	7.807

With the exception of Galaxy [40], all these environments allow for parallelisation which significantly speeds training up. Candles and CyberBattleSim are strictly simulations of cyber security environments and use a finite state machine [2], while the others are emulations that use virtual machines (e.g., on the cloud). CyberBattleSim and FARLAND are the most recently published (2021) alongside CybORG. CyberBattleSim, much like CybORG, is based on the Open AI Gym environment and provides a high-level simulated abstraction of computer networks [51]. FARLAND is a framework designed for training autonomous agents for network defense by gradually increasing complexity.

Galaxy is similar framework, offering a modular environment to experiment and tailor to the requirements and constraints of different real-world systems. Unfortunately, Galaxy is limited in terms of scaling their network emulation; currently each host computer can support only one visualised network due to emulation being computationally expensive, restricting its ability to scale up. The developers of this framework are planning to containerise the infrastructure in future work to allow for higher-fidelity emulation and scalability [40].

CybORG is one of the most commonly used environments in the literature. This is primarily due to the CAGE Challenges. Researchers, practitioners and participants have been publishing solutions and algorithm combinations demonstrating a variety of successful RL uses specifically to explore the creation of autonomous network defenders, both from a single-agent (CAGE 1,2 [1, 3]) and a multi-agent (CAGE 3) perspective. In particular, there is relevant work in the winning submission for the CAGE 1 Challenge [12] and 3rd place submission for the CAGE 2 Challenge. Both submissions included curiosity as an additional reward shaping strategy, which improved the scores of their CAGE 1 Bline PPO agent "by nearly double" [12] and significantly improved performance of their Bline agent in CAGE 2 [13] as well. Their CAGE 2 implementation showed that the solely PPO agent was able to achieve maximum best scores of approximately -10, and the two alternative agents trained were able to achieve much better scores between approximately -2 and -4 [13].

Similarly, CAGE scenarios (running on CybORG) have been used in various works within the field of autonomous cyber defense, such as that of Wolk et al. (2022) [54] which evaluates a set of different RL approaches, such as ensemble RL, action masking, hierarchical RL and custom training in cyberdefense scenarios. They selected CAGE due to its minimal reality gap when emulating an attacker on an enterprise network [54]. Furthermore, CAGE has been used in exploring automatic penetration testing [55] where it was used as an environment to test the RL framework 'CLAP' alongside another SOTA penetration testing environment for networks, NASim [44, 45].

Reward shaping for a reinforcement learning is a well-studied area, especially in environments with typically sparse rewards [15, 26, 34]. Janssen and Grey's (2012) [23] work explored different magnitudes of reward shaping in a cognition-based task called 'Blocks World'. The magnitude of rewards (based on accuracy and time taken to complete the task) were altered to either be between 0 and 8, or -8 and 8. The results in their particular environment show that using either of these two reward bounds did not result in significant difference for their model in terms of accuracy [23]. Another example of deliberately altered reward values is [46] that simulated two agent environments and repeatedly varies the reward values within the range of -1 and 1. This work was not specifically focused on the effect of varying extrinsic rewards but instead discussed rewards in an evolutionary context [46]. Besides these, there is very limited research (especially in the cybersecurity context) on the impact of reward magnitude, combinations of negative and positive rewards and how they compare to baseline results and intrinsic reward strategies (such as ICM).

9 CONCLUSION

Cybersecurity environments differ significantly from typical RL environments as the objective of the agent is to preserve the initial (i.e., non-compromised) state of the system/network (i.e., any deviation from that results in a negative reward). This work took a first step in better understanding how this characteristic affects learning and what techniques can be used to train a performant agent more efficiently. Our findings show that reward shaping can be effective in increasing sample efficiency and performance. However, depending on the technique used and the magnitude of the rewards, the improvement can vary and may even result in performance decrease in some cases. In the future, such techniques could be combined with curriculum learning to quickly bootstrap learning (on simpler but relevant reward-augmented tasks) and then gradually fall back to non-augmented rewards on the original task.

ACKNOWLEDGMENTS

Research funded by the Defence Science and Technology Laboratory (Dstl) which is an executive agency of the UK Ministry of Defence providing world class expertise and delivering cutting-edge science and technology for the benefit of the nation and allies. The research supports the Autonomous Resilient Cyber Defence (ARCD) project within the Dstl Cyber Defence Enhancement programme.

REFERENCES

[1] 2021. Cyber Autonomy Gym for Experimentation Challenge 1. https://github.com/cage-challenge/cage-challenge-1. Created by Maxwell Standen, David Bowman, Son Hoang, Toby Richer, Martin Lucas, Richard Van Tassel.

[2] 2021. *CybORG: A Gym for the Development of Autonomous Cyber Agents*. arXiv.

[3] 2022. Cyber Autonomy Gym for Experimentation Challenge 2. https://github.com/cage-challenge/cage-challenge-2. Created by Maxwell Standen, David Bowman, Son Hoang, Toby Richer, Martin Lucas, Richard Van Tassel, Phillip Vu, Mitchell Kiely.

[4] 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.

[5] Chittaranjan Andrade. 2019. The P value and statistical significance: misunderstandings, explanations, challenges, and alternatives. *Indian journal of psychological medicine* 41, 3 (2019), 210–215.

[6] 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* (Los Angeles, CA, USA) *(AISec'22)*. Association for Computing Machinery, New York, NY, USA, 149–159. https://doi.org/10.1145/3560830.3563732

[7] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

[9] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. 2018. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355* (2018).

[10] CAGE. 2022. TTCP CAGE Challenge 2. https://github.com/cage-challenge/cage-challenge-2.

[11] Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. 2022. Exploration-Guided Reward Shaping for Reinforcement Learning under Sparse Rewards. *Advances in Neural Information Processing Systems* 35 (2022), 5829–5842.

[12] 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* (Nagasaki, Japan) *(ASIA CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1252–1254. https://doi.org/10.1145/3488932.3527286

[13] Myles Foley, Mia Wang, Zoe M, Chris Hicks, and Vasilios Mavroudis. 2023. Inroads into Autonomous Network Defence using Explained Reinforcement Learning. arXiv:2306.09318 [cs.CR]

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

[15] Marek Grzes. 2017. Reward shaping in episodic reinforcement learning. (2017).

[16] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. 2022. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems* 35 (2022), 15281–15295.

[17] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement Learning with Deep Energy-Based Policies. *CoRR* abs/1702.08165 (2017). arXiv:1702.08165 http://arxiv.org/abs/1702.08165

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG]

[19] John Hannay. 2022. Cyborg Cage 2 Solution. https://github.com/john-cardiff/-cyborg-cage-2.

[20] Chris Hicks, Vasilios Mavroudis, Myles Foley, Thomas Davies, Kate Highnam, and Tim Watson. 2023. Canaries and Whistles: Resilient Drone Communication Networks with (or without) Deep Reinforcement Learning. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), Copenhagen, Denmark* (2023).

[21] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

[22] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. *Advances in neural information processing systems* 29 (2016).

[23] Christian P. Janssen and Wayne D. Gray. 2012. When, What, and How Much to Reward in Reinforcement Learning-Based Models of Cognition. *Cognitive Science* 36, 2 (2012), 333–358. https://doi.org/10.1111/j.1551-6709.2011.01222.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1551-6709.2011.01222.x

[24] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research* 32 (Sept. 2013), 1238–1274. https://doi.org/10.1177/0278364913495721

[25] Meicong Li, Wei Huang, Yongbin Wang, Wenqing Fan, and Jianfang Li. 2016. The study of APT attack stage model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE, 1–5.

[26] Ofir Marom and Benjamin Rosman. 2018. Belief reward shaping in reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[27] Pietro Mazzaglia, Ozan Catal, Tim Verbelen, and Bart Dhoedt. 2022. Curiosity-Driven Exploration via Latent Bayesian Surprise. arXiv:2104.07495 [cs.LG]

[28] Yisroel Mirsky and Wenke Lee. 2021. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)* 54, 1 (2021), 1–41.

[29] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG]

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG]

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A A. Rusu, J. Veness, M G. Bellemare, A. Graves, M. Riedmiller, A K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* (2015).

[33] Andres Molina-Markham, Cory Miniter, Becky Powell, and Ahmad Ridley. 2021. Network Environment Design for Autonomous Cyberdefense. arXiv:2103.07583 [cs.CR]

[34] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, Vol. 99. Citeseer, 278–287.

[35] Jakob Nyberg and Pontus Johnson. 2023. Training Automated Defense Strategies Using Graph-based Cyber Attack Simulations. arXiv:2304.11084 [cs.CR]

[36] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:1912.06680 [cs.LG]

[37] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*. PMLR, 2778–2787.

[38] George Rush, Daniel R. Tauritz, and Alexander D. Kent. 2015. Coevolutionary Agent-Based Network Defense Lightweight Event System (CANDLES). In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO Companion '15)*. Association for Computing Machinery, New York, NY, USA, 859–866. https://doi.org/10.1145/2739482.2768429

[39] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep Reinforcement Learning framework for Autonomous Driving. *Electronic Imaging* 29, 19 (Jan. 2017), 70–76. https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023 arXiv:1704.02532 [cs, stat].

[40] Kevin Schoonover, Eric Michalak, Sean Harris, Adam Gausmann, Hannah Reinbolt, Daniel R. Tauritz, Chris Rawlings, and Aaron Scott Pope. 2018. Galaxy: A Network Emulation Framework for Cybersecurity. In *11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18)*. USENIX Association, Baltimore, MD. https://www.usenix.org/conference/cset18/presentation/schoonover

[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

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

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

[44] Jonathon Schwartz. [n. d.]. Network Attack Simulator. https://github.com/Jjschwartz/NetworkAttackSimulator, Year = 2020.

[45] Jonathon Schwartz and Hanna Kurniawati. 2019. Autonomous Penetration Testing using Reinforcement Learning. arXiv:1905.05965 [cs.CR]

[46] Satinder Singh, R. Lewis, and A. Barto. 2009. Where Do Rewards Come From? *Proceedings of the 31st Annual Meeting of the Cognitive Science Society.*

[47] Dr. Surjit. 2020. Deep reinforcement learning using proximal policy optimization. https://medium.com/analytics-vidhya/deep-reinforcement-learning-using-proximal-policy-optimization-7555280ef941

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

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

[50] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[51] 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..

[52] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. 2019. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems* 32 (2019).

[53] E. Wiewiora. 2003. Potential-Based Shaping and Q-Value Initialization are Equivalent. *Journal of Artificial Intelligence Research* 19 (sep 2003), 205–208. https://doi.org/10.1613/jair.1190

[54] 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. arXiv:2211.15557 [cs.LG]

[55] Yizhou Yang and Xin Liu. 2022. Behaviour-Diverse Automatic Penetration Testing: A Curiosity-Driven Multi-Objective Deep Reinforcement Learning Approach. arXiv:2202.10630 [cs.LG]

[56] 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. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.