Knowledge Retention for Generic Reinforcement Learning Policies in Autonomous Cyber Defence

Joshua Sylvester and Rogério de Lemos

University of Kent, Canterbury, UK
{jrs71,r.delemos}@kent.ac.uk

Abstract. Within autonomous cyber defence, building scalable agents that can generalise across attack behaviours is crucial to developing a truly autonomous system. These generic agents are pivotal, as over time, attackers will inevitably change their behaviour, requiring the defence mechanisms to adapt accordingly. Current approaches for generic agents use deep reinforcement learning policies to learn multiple attack behaviours and mitigate them. When a new attack behaviour is introduced, the generic policy is retrained to incorporate this behaviour and not forget previous attack behaviours. In this paper, we propose a novel solution based on a modified version of the Proximal Policy Optimisation (PPO) reinforcement learning algorithm that retains previously acquired knowledge, enabling a scalable and generic framework in which new attack behaviours can be incorporated modularly. The modified PPO algorithm demonstrates a 22.11% performance improvement compared to standard PPO when trained to sequentially learn two distinct attack behaviours. These results show a step towards building more scalable autonomous cyber defence systems capable of incorporating evolving cyber threats.

1 Introduction

Reinforcement learning (RL) has shown great results in areas such as robotics, healthcare and finance [5, 13, 16]. However, when applied to autonomous cyber defence (ACD) a new challenge emerges, attack behaviours change rapidly over time. This challenge requires policies that can operate reliably in ever changing network environments, and this presents a significant obstacle for standard RL approaches, which typically require complete retraining on new adversarial tactics [6]. Moreover, retraining an RL policy from scratch is computationally expensive and delays the ability to respond quickly to a new threat.

Solutions for developing generic reinforcement learning policies in autonomous cyber defence remain limited. Farooq and Kunz [6] propose one such approach by first training dedicated policies for each attack, then collecting observation-action pairs to train a supervised model capable of generalising across behaviours. While effective, this method requires training a specific policy and collecting data for each new attack behaviour in order to develop an updated generic policy. Alternative methods such as policy distillation, actor-mimic networks, and experience replay aim to address these challenges by transferring knowledge across tasks and

preserving performance on previous behaviours [8, 15, 17]. However, these methods either require storing large replay buffers, accessing previous environments, or retraining from scratch, which hinders their scalability. This highlights the need for an approach that, in addition to learning from new behaviours, also supports knowledge retention.

To address the challenges of scalability and adaptability in reinforcement learning for autonomous cyber defence, we propose a modified Proximal Policy Optimisation (PPO) algorithm that incorporates a KL divergence based regularisation term [18]. This encourages the new policy to remain close to a previously learned reference policy during training, enabling the agent to retain prior knowledge while acquiring new behaviours. Unlike traditional PPO, which optimises the policy with no explicit mechanism for preserving past capabilities, our KL regularised variant supports sequential learning across attack behaviours without requiring full retraining or replay buffers.

Using a modified PPO algorithm, we analyse how policies trained against one class of behavioural attacks can be extended to handle additional adversaries while preserving competence on earlier ones. Through empirical evaluation using the TTCP CAGE Challenge 2 simulation environment, we demonstrate that this approach enables incremental expansion of defensive capabilities without full retraining, and we explore the role of the KL-divergence in managing the trade-off between knowledge retention and plasticity.

In the context of whether knowledge retention can be used to build scalable, generic reinforcement learning policies in autonomous cyber defence, a key contribution of this paper is a modified PPO algorithm that supports knowledge retention. We have demonstrated that our modified PPO algorithm can be used to develop scalable and generic reinforcement learning policies for autonomous cyber defence. However, initial observations suggest that the final policy is influenced by both the characteristics of the initial attack behaviours and the sequence in which behaviours are learned.

The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 outlines the proposed approach, including the modified PPO algorithm. Section 5 presents the experiments, along with their results and analysis, comparing standard PPO with the modified PPO, and also shows the impact that the initial policy has when generating a generic policy. The paper concludes with findings and future directions.

2 Related Work

Developing agents for autonomous cyber defence requires policies that generalise across different attacker behaviours. This section reviews two key areas: generic policies, which aim to operate across varied network structures and threats, and continual multi-task learning, which focuses on learning new tasks without forgetting previous ones. While existing approaches show promise, many still rely on retraining or revisiting past environments, limiting scalability. Our work addresses these gaps by enabling continual learning without such dependencies.

2.1 Generic Policies in Autonomous Cyber Defence

In the field of autonomous cyber defence (ACD), research on generic policies has primarily focused on two key dimensions of generalisation: generalisation across network structures and generalisation across varying attack behaviours [14]. The former concerns the ability of a defensive policy to operate effectively across different network topologies or configurations without retraining. The latter addresses the challenge of enabling a policy to detect and respond to a wide range of attacker strategies.

Research into generic policies over network structures has progressed from handling minor topological variations, such as changes in connectivity with a fixed number of nodes, to more advanced techniques that work on dynamic scenarios [1, 22]. These accommodate changes in both the number of nodes and overall network configuration, without the need for retraining [20]. However, a different challenge emerges when considering attack behaviours.

Generalising over attack behaviours introduces constraints unique to reinforcement learning. Generic policies outperform specialised ones when responding to previously unseen attack strategies [22]. However, as attackers develop novel tactics or exploit new vectors, even these general policies can become outdated [4]. This underscores the need for policies capable of incorporating new threats into their defensive repertoire. Over time, the cost, both in computation and effort, of retraining policies from scratch becomes increasingly prohibitive, highlighting the importance of mechanisms for continual learning and efficient policy updating.

Solutions for policies that generalise across attack behaviours remain scarce, with Farooq and Kunz [6] presenting one of the few approaches for developing generic policies using both supervised and reinforcement learning. By training a policy on each attack behaviour, then collecting observation and action pairs from the collection of specific policies to create a dataset that will be used to train a supervised multi-layer perceptron (MLP). Although this model performs well from a scalability point of view, it has some drawbacks. Not only does the specific policy need to be trained on the new attack behaviour, but the observation and action pairs need to be collected to add them to the dataset to be used in retraining a new supervised model.

Loevenich et al. [12] present a framework that combines Proximal Policy Optimisation (PPO) with a fine-tuned Large Language Model (LLM) to develop a defensive agent. While the approach demonstrates strong performance, one of its purported advantages is scalability. However, when considering evolving attack behaviours, this scalability is limited: the PPO model must be retrained on all attack behaviours each time a new attack behaviour is introduced, hindering the agent's ability to adapt efficiently in dynamic threat environments.

Both Dehghantanha et al. [4] and Kott [11] emphasise the need for adaptive frameworks that can evolve in tandem with the shifting threat landscape by continuously learning and responding to novel challenges. Central to achieving this is the development of policies capable of incremental learning without forgetting previously acquired knowledge.

2.2 Continual Multi-Task Learning for Reinforcement Learning

Continual Multi-Task Learning (CMTL) in Reinforcement Learning (RL) seeks to develop agents capable of learning multiple tasks over time without catastrophic forgetting. Unlike traditional multi-task learning, which assumes simultaneous access to all tasks, CMTL addresses the sequential nature of task exposure, aiming to preserve performance on previously learned tasks whilst acquiring new behaviours.

One of the central challenges in CMTL is catastrophic forgetting, where learning new tasks interferes destructively with previously acquired knowledge. Several approaches have been proposed to mitigate this, such as Elastic Weight Consolidation (EWC) [10], Progress & Compress [19], and Memory Aware Synapses (MAS) [2], which regularise parameter updates based on their importance to prior tasks. Although originally proposed for supervised settings, these techniques have been used in RL with varying degrees of success [10, 19]. For example, Elastic Weight Consolidation (EWC) and Progress & Compress, like other weight regularisation methods, require clearly defined and distinct task boundaries to function effectively, which is not the case in autonomous cyber defence [21].

Policy distillation and actor-mimic networks offer a strategy for transferring knowledge from previously trained policies into a single student model capable of handling multiple tasks [15, 17]. These techniques can be used to initialise or regularise learning when adapting to new tasks.

Experience replay buffers have also been widely used in lifelong learning to help agents retain and revisit past experiences [8]. For example, Xu et al. [23] combines experience replay with policy distillation, using teacher policies to support continual learning without forgetting previously learned tasks.

However, a common limitation of these approaches is that they still require interaction with the environments of previous tasks, either to regenerate data through the original policies or to refresh stored experiences. As the number of tasks increases, the need to repeatedly access and run older environments can become a major scalability bottleneck. Such dependence on past environments also raises concerns around long-term maintenance, especially when older environments become deprecated or harder to support.

To overcome these limitations, our approach enables continual learning without requiring further interaction with previously encountered environments. By eliminating this dependency, our method not only reduces computational overhead but also improves scalability, making it more suitable for real-world applications.

3 Methodology

This section introduces a method to train reinforcement learning (RL) agents that can learn new attack behaviours without forgetting how to respond to previously seen ones. In real-world autonomous cyber defence (ACD), the threat

landscape is constantly changing. New attack strategies emerge, old ones are modified, and defenders must remain effective against both.

Our proposed approach enables knowledge retention across attack types by modifying the Proximal Policy Optimisation (PPO) algorithm to support continual learning. Crucially, this method avoids the need to store or retrain on data from prior tasks, addressing scalability issues.

At a high level, our method proceeds as follows:

1. We begin by training an agent to defend against one type of attacker (Task $\mathcal{T}_i$).
2. When a new attacker behaviour (Task $\mathcal{T}_j$) is introduced, we fine-tune the existing agent rather than starting from scratch.
3. During fine-tuning, we employ a knowledge retention mechanism that constrains learning, ensuring the agent remains effective on the previous task without needing direct access to it.

The following subsections present this method in detail. First, we explain its mathematical foundations and then describe our modified PPO algorithm, which integrates a lightweight regularisation term to preserve prior knowledge.

3.1 Overview

Autonomous cyber defence systems must operate in dynamic threat environments where attacker tactics change over time. To enable long-term adaptability, our objective is to train RL agents that generalise across attack behaviours while preserving knowledge of previously encountered attack behaviours.

Assume there are two tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ where both tasks have different attack behaviours [3]. An RL task $\mathcal{T}_x$ is typically defined as a Markov Decision Process (MDP).

$$\mathcal{T}_x := (\mathcal{S}, \mathcal{A}, P_x, R)$$

where:

- $\mathcal{S}$ is the *state space*
- $\mathcal{A}$ is the *action space*
- $P_x : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the *transition probability kernel*, where $P(s' \mid s, a)$ denotes the probability of transitioning to state s' after taking action a in state s
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *reward function* specifying the immediate reward received after taking action a in state s.

In this setup, the state and action spaces are consistent across tasks, while the differences between tasks arise due to variations in attack behaviour shown in P_i and P_j. These differences can correspond to changes in movement patterns, target selection strategies, or stealth mechanisms used by adversaries in the simulation.

The objective of this method is to obtain a generic policy $\pi_{\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}}$ such that it performs well on task $\mathcal{T}_j$, it retains performance on task $\mathcal{T}_i$, and no samples from task $\mathcal{T}_i$ are needed to retain knowledge.

$$\pi_{\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}} \approx \arg\max_{\theta} J(\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}; \mathcal{T}_j) \quad \text{s.t.} \quad J(\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}; \mathcal{T}_i) \approx J(\theta_{\mathcal{T}_i}; \mathcal{T}_i)$$

where:

- $J(\theta_{\mathcal{T}_x}; \mathcal{T}_x)$ is the expected return of policy π_θ on task $\mathcal{T}_x$
- $\theta_{\mathcal{T}_x}$ are the parameters of a policy trained on $\mathcal{T}_x$.

This setup allows the agent to adapt to new threats (task $\mathcal{T}_j$) while preserving its effectiveness against previously seen threats (task $\mathcal{T}_i$), thereby addressing a key challenge in sequential learning for cyber defence.

3.2 Modified PPO Algorithm

To achieve the above objective, we propose a modified variant of the PPO algorithm that incorporates an additional term to promote knowledge retention. The standard PPO algorithm is a widely used on-policy actor-critic algorithm that balances policy improvement with stability through a clipped surrogate objective.

Standard PPO Loss Function. The standard PPO loss function ($\mathcal{L}(\theta)$) combines three components: clipped policy loss ($\mathcal{L}_{\text{clip}}(\theta)$), value function loss ($\mathcal{L}_{\text{V}}(\theta)$), and entropy regulariser ($\mathcal{L}_{\text{entropy}}(\theta)$) to promote exploration.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{clip}}(\theta) + c_1 \mathcal{L}_{\text{V}}(\theta) - c_2 \mathcal{L}_{\text{entropy}}(\theta) \tag{1}$$

where:

- c_1 and c_2 are hyperparameters controlling the relative importance of the value loss and entropy regularisation.

The clipped policy loss ($\mathcal{L}_{\text{clip}}(\theta)$) is defined as follows:

$$\mathcal{L}_{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \tag{2}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3}$$

$$\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & \text{if } r_t(\theta) < 1-\epsilon \\ r_t(\theta) & \text{if } 1-\epsilon \leq r_t(\theta) \leq 1+\epsilon \\ 1+\epsilon & \text{if } r_t(\theta) > 1+\epsilon \end{cases} \tag{4}$$

where:

- $r_t(\theta)$ is the probability ratio of the new and old policies at time t
- $\hat{A}_t$ is the estimated advantage at time t

- ϵ is the clipping parameter
- $\pi_\theta(a_t \mid s_t)$ is the probability of taking action a_t given the state s_t at time step t, under the current policy π_θ
- $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ clips the value of $r_t(\theta)$ to be between $[1-\epsilon, 1+\epsilon]$

The value function loss ($\mathcal{L}_{\text{V}}(\theta)$) is defined as follows:

$$\mathcal{L}_{\text{V}}(\theta) = \hat{\mathbb{E}}_t \left[(V_\theta(s_t) - R_t)^2 \right] \tag{5}$$

where:

- $V_\theta(s_t)$ is the predicted value of state s_t
- R_t is the empirical return at time step t

The entropy regulariser ($\mathcal{L}_{\text{entropy}}(\theta)$) is defined as follows:

$$\mathcal{L}_{\text{entropy}}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \right] \tag{6}$$

Modified PPO Loss Function. To retain knowledge of the original task $\mathcal{T}_i$, we introduce a regularisation term ($\mathcal{L}_{\text{KL}}(\theta)$) that penalises divergence from a reference policy ($\pi_{\theta_{\text{ref}}}$), which is the policy trained on $\mathcal{T}_i$ and frozen during subsequent learning. This term constrains updates to remain close to the original policy on the shared state-action distribution.

This results in the modified loss function ($\mathcal{L}'(\theta)$):

$$\mathcal{L}'(\theta) = \mathcal{L}_{\text{clip}}(\theta) + c_1\mathcal{L}_{\text{V}}(\theta) - c_2\mathcal{L}_{\text{entropy}}(\theta) + c_3\mathcal{L}_{\text{KL}}(\theta) \tag{7}$$

where:

- c_3 is a hyperparameter controlling the strength of knowledge retention.

The regularisation term ($\mathcal{L}_{\text{KL}}(\theta)$) is defined as follows:

$$\text{KL}[\pi_{\theta_{\text{ref}}} \,\|\, \pi_\theta] \approx \mathcal{L}_{\text{KL}}(\theta) = (r'_t(\theta) - 1) - \log(r'_t(\theta)) \tag{8}$$

$$r'_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{ref}}}(a_t \mid s_t)} \tag{9}$$

where:

- $r'_t(\theta)$ is the probability ratio of the new and old policies at time t
- $\pi_{\theta_{\text{ref}}}(a_t \mid s_t)$ is the probability of taking action a_t given the state s_t at time step t, under the reference policy $\pi_{\theta_{\text{ref}}}$

The regularisation term $\mathcal{L}_{\text{KL}}(\theta)$ is used to estimate the reverse KL divergence between the reference policy and the current policy, we use a Monte Carlo approximation based on samples from the reference policy [1]. This approach evaluates how much the current policy shifts away from the actions chosen by the

[1] `http://joschu.net/blog/kl-approx.html`

reference. By relying on samples rather than exact computation, this method makes the reverse KL practical to compute during training while still encouraging the new policy to stay close to the reference in terms of its action choices.

By incorporating the regularisation term, the policy update is guided to preserve past behaviour, preventing catastrophic forgetting during the addition of task $\mathcal{T}_j$.

3.3 Training Procedure

In this section, we describe the training procedure used to accomplish knowledge retention. Our approach extends the standard PPO algorithm by incorporating a knowledge retention mechanism through a reverse KL divergence penalty. This allows the policy to adapt to a new attack behaviour while mitigating the risk of catastrophic forgetting on the source task.

Algorithm 1 Online Transfer PPO with Knowledge Retention

Input: Pretrained policy $\pi_{\theta_{\mathcal{T}_i}}$ on task $\mathcal{T}_i$, target task $\mathcal{T}_j$, KL penalty coefficient c_3, number of training iterations N
Output: Updated policy $\pi_{\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}}$

1: Initialize $\pi_\theta \leftarrow \pi_{\theta_{\mathcal{T}_i}}$
2: **for** iteration $= 1$ to N **do**
3: Collect trajectories $\tau_j = \{(s_t, a_t, r_t, s_{t+1})\}$ by interacting with task $\mathcal{T}_j$ using π_θ
4: Compute advantages $\hat{A}_t$ and value targets from τ_j
5: **for** each mini-batch in τ_j **do**
6: Compute clipped surrogate loss $\mathcal{L}_{\text{clip}}(\theta)$
7: Compute value function loss $\mathcal{L}_V(\theta)$
8: Compute entropy bonus $\mathcal{L}_{\text{entropy}}(\theta)$
9: Compute importance ratio $r'_t(\theta)$
10: Approximate reverse KL divergence $\mathcal{L}_{\text{KL}}(\theta)$
11: Combine loss $\mathcal{L}'(\theta)$
12: Update policy parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}'(\theta)$$

13: **end for**
14: **end for**
15: **return** $\pi_{\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}}$

Algorithm 1 outlines the training procedure for our modified PPO algorithm that incorporates knowledge retention. The process begins by initialising the policy π_θ with parameters from the policy pretrained on task $\mathcal{T}_i$ (Line 1). The training loop then proceeds for N iterations (Line 2), where each iteration consists of interacting with the new task $\mathcal{T}_j$ to collect a batch of trajectories (Line 3).

From these trajectories, the agent computes advantage estimates and value function targets (Line 4), which are required for the surrogate loss and value updates. For each mini-batch sampled from the collected data (Line 5), several loss components are computed: the clipped policy loss ensures stable updates (Line 6), the value function loss promotes accurate state-value estimates (Line 7), and the entropy bonus encourages exploration (Line 8).

To retain knowledge from task $\mathcal{T}_i$, an importance sampling ratio (Line 9) is calculated with respect to the reference policy $\pi_{\theta_{\text{ref}}}$, and a reverse KL divergence penalty is approximated (Line 10). These are combined into a final loss function (Line 11), which balances policy improvement on $\mathcal{T}_j$ with the preservation of behaviour on $\mathcal{T}_i$.

Finally, gradient descent is used to update the policy parameters with respect to this combined loss (Line 12). Once the loop is complete the final policy $\pi_{\theta_{\mathcal{T}_i \to \mathcal{T}_i \cup \mathcal{T}_j}}$ is returned (Line 15).

4 CybORG - TTCP CAGE Challenge 2

This study uses CybORG, a simulation framework tailored for research in cyber defence, particularly reinforcement learning (RL)-based defence strategies. Within CybORG, agents are organised according to their functional roles: red agents simulate attackers aiming to compromise systems, blue agents defend the network from such threats, and green agents represent benign user behaviour.

The experiments are conducted using the TTCP CAGE Challenge 2 scenario, which features a simplified network structure. In this scenario, red agents attempt to compromise an operational server and disrupt its functionality, while the blue agent's objective is to prevent this by detecting and mitigating the red agent's actions [9].

Red agent behaviours in the scenario are structured around the MITRE ATT&CK[2] framework and emulate real-world vulnerabilities and attack techniques [9]. These behaviours typically begin with reconnaissance, such as remote system discovery (T1018) via ICMP pings, followed by network service enumeration (T1046). The agent then attempts to exploit vulnerable services (T1210), escalate privileges (TA0004), and ultimately disrupt targeted services (T1489) to achieve denial of service.

This research focuses on three red agents. Two red agents are part of the TTCP CAGE Challenge 2 scenario, and the third one, generated for this paper, is a hybrid variant of the original two red agents.

- **Meander:** Operates without prior knowledge of the network topology. It sequentially attempts to compromise all nodes in one subnet before progressing to the next.
- **BLine:** Has full knowledge of the network and follows predefined sequences of actions (attack paths) that lead directly to the operational server.

[2] `https://attack.mitre.org/techniques/enterprise/`

– **Hybrid:** Combines elements from both *Meander* and *BLine*. It begins with exploratory behaviour like *Meander*, aiming to control entire subnets, before transitioning to a targeted attack path akin to *BLine*.

The implementation of the *Hybrid* red agent is openly available here.

5 Experiments

This section describes the experiments performed that justify the use of a modified version of the Proximal Policy Optimisation (PPO) reinforcement learning algorithm for retaining previously acquired knowledge. It consists of the following key parts: the set up used for our experiments, a comparison of the standard PPO and modified PPO, an evaluation of whether knowledge retention can be used to generate generic policies to handle new attacks, and finally, a study on how the initial policy may affect the generation of the generic policy.

5.1 Experimental Setup

In multi-task continual learning scenarios, retaining previously acquired knowledge while learning new tasks is critical for developing scalable agents. To evaluate knowledge retention, experiments were conducted across sequential task combinations using distinct red agents, specifically: *Meander*, *BLine*, and a combined one involving both behaviours (*Hybrid*).

The evaluation follows a two-stage training procedure. First, a base policy is trained on a source task (e.g., *Meander*) for 600,000 timesteps. Subsequently, the policy then learns any new target task (e.g., *BLine*) for a further 400,000 timesteps.

Each policy is evaluated over 1,000 episodes, with each episode consisting of 50 timesteps. To address variance in training, ten independent seeds are used per configuration, and results are reported as the mean episodic reward with standard error (SE) following established guidelines [7].

All experiments were run on an Intel Xeon Gold 5317 processor with an NVIDIA A100 Tensor Core GPU. The module versions used for Python (3.10.12) are NumPy (2.1.3) and Matplotlib (3.10.1) for data processing and visualisation. PyTorch (2.6.0+cu124) supported model development, while Stable-Baselines3 (1.8.0), Gym (0.21.0), and Gymnasium (0.29.1) were used for reinforcement learning.

5.2 Comparing standard PPO with modified PPO

To motivate the use of our proposed modified PPO approach for knowledge retention, this section presents experiments designed to evaluate the ability of each method to preserve performance across multiple tasks. Both algorithms are trained and evaluated within the same set up to ensure a fair comparison. The standard PPO is selected as the baseline due to its established stability

and strong performance in the TTCP CAGE Challenge 2 environment [22]. The modified PPO incorporates additional mechanisms specifically intended to mitigate catastrophic forgetting and enhance knowledge retention when transferring between tasks. The following experiments aim to assess the extent to which these modifications improve policy generalisation without sacrificing performance on previously learned tasks.

Table 1 shows the performance of the standard PPO when trained on either *Meander* or *BLine* and then further trained on the other red agent. The notation $\mathbf{PPO}_S$**[**M**]** shows that the policy was trained on *Meander*. The policy $\mathbf{PPO}_S$**[**M**]** is used to generate the policy $\mathbf{PPO}_S$**[**M→MB**]**, which shows a policy that was trained on *Meander* and then on *BLine*.

Table 1. Performance of Sequential Training of Standard PPO Policies ($\mathbf{PPO}_S$) Across Different Environments (without and with Green Agents)

(a) Without Green Agents

Policy	Meander (M)	BLine (B)	Total
$\mathbf{PPO}_S$**[**M**]**	-25.92 ± 0.17	-94.69 ± 1.13	-120.61 ± 1.14
$\mathbf{PPO}_S$**[**M→MB**]**	-46.28 ± 0.45	-18.16 ± 0.34	-64.44 ±0.56
$\mathbf{PPO}_S$**[**B**]**	-46.90 ± 0.56	**-18.04 ± 0.43**	-64.94 ± 0.71
$\mathbf{PPO}_S$**[**B→BM**]**	**-20.84 ± 0.10**	-37.11 ± 0.55	**-57.95 ± 0.56**

(b) With Green Agents

Policy	Meander (M)	BLine (B)	Total
$\mathbf{PPO}_S$**[**M**]**	**-26.67 ± 0.11**	-50.03 ± 0.64	-76.70 ± 0.65
$\mathbf{PPO}_S$**[**M→MB**]**	-31.03 ± 0.25	**-19.20 ± 0.35**	**-50.23 ± 0.43**
$\mathbf{PPO}_S$**[**B**]**	-49.30 ± 0.39	-24.89 ± 0.39	-74.19 ± 0.55
$\mathbf{PPO}_S$**[**B→BM**]**	-29.99 ± 0.16	-70.01 ± 0.72	-100.00 ± 0.74

Table 2 presents the performance of the modified PPO algorithm when initially trained on either *Meander* or *BLine*, and subsequently fine-tuned on the other red agent. The notation follows that of Table 1, with the addition that, alongside copying the network weights, the previous policy is also incorporated as a reference policy within the modified PPO algorithm. For these experiments, a single KL coefficient was selected and applied across all runs. However, further performance improvements can be achieved by tuning this coefficient for each specific training scenario.

When comparing Table 1 and Table 2, it is evident that the modified PPO algorithm produces policies that generalise more effectively across both environments compared to the standard PPO approach. On average, policies trained with the modified PPO achieve rewards that are 22.11% higher than those produced by the standard PPO. The most significant improvement is observed when the model is first trained on *BLine* and subsequently fine-tuned on *Meander*, where the total mean reward improves from -100.00 to -59.38, representing a 40.62% increase.

Table 2. Performance of Sequential Training of Modified PPO Policies ($\mathbf{PPO}_M$) Across Different Environments (without and with Green Agents)

(a) Without Green Agents

Policy	KL Coefficient	Meander (M)	BLine (B)	Total
$\mathbf{PPO}_M$[M]	N/A	-27.94 ± 0.15	-93.60 ± 0.98	-121.54 ± 0.99
$\mathbf{PPO}_M$[M→MB]	0.002	**-23.31 ± 0.11**	-21.45 ± 0.34	**-44.76 ± 0.36**
$\mathbf{PPO}_M$[B]	N/A	-46.78 ± 0.52	**-12.57 ± 0.19**	-59.35 ± 0.55
$\mathbf{PPO}_M$[B→BM]	0.002	-27.07 ± 0.24	-21.73 ± 0.51	-48.80 ± 0.56

(b) With Green Agents

Policy	KL Coefficient	Meander (M)	BLine (B)	Total
$\mathbf{PPO}_M$[M]	N/A	**-25.01 ± 0.12**	-61.00 ± 0.73	-86.01 ± 0.74
$\mathbf{PPO}_M$[M→MB]	0.002	-25.70 ± 0.11	**-23.79 ± 0.32**	**-49.49 ± 0.34**
$\mathbf{PPO}_M$[B]	N/A	-39.72 ± 0.30	-24.56 ± 0.30	-64.28 ± 2.34
$\mathbf{PPO}_M$[B→BM]	0.002	-28.15 ± 0.12	-31.23 ± 0.53	-59.38 ± 0.54

Figures 1 and 2 show the impact of varying the KL coefficient on the performance of the modified PPO algorithm. In both cases, the mean reward is plotted on the y-axis while the KL coefficient is plotted on the x-axis. As expected, when the KL coefficient is set too high, the agent is overly constrained by the reference policy, resulting in poor learning of the new agent's behaviour. On the other hand, when the KL coefficient is too low, the algorithm allows for greater deviation from the reference policy, which enables adaptation to the new agent but leads to partial forgetting of the original behaviour. These results highlight the impact of the KL coefficient and the importance of fine-tuning it to balance knowledge retention with the ability to learn new behaviours.

Fig. 1. Results Showing the Effect of the KL Coefficient on Mean Rewards (without Green Agents)

Figures 3 and 4 illustrate the KL divergence between the final policy and the reference policy based on the KL coefficient. In both cases, the KL coefficient is plotted on the x-axis and the resulting KL divergence on the y-axis. As expected,

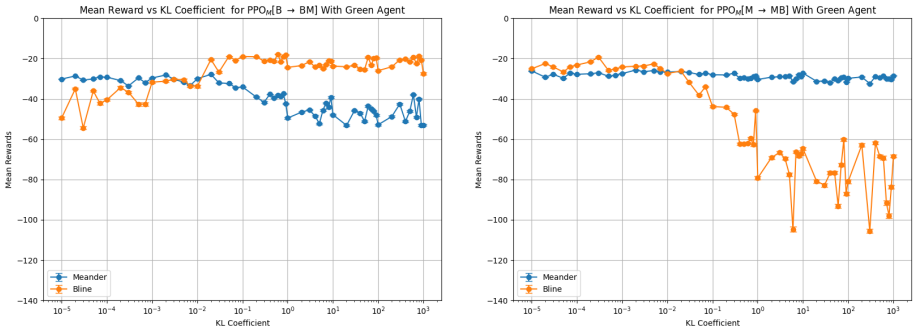

Fig. 2. Results Showing the Effect of the KL Coefficient on Mean Rewards (with Green Agents)

lower KL coefficients permit greater deviation from the reference policy, resulting in higher KL divergence, while higher coefficients enforce stronger regularisation, keeping the policy closer to the reference. Notably, there is a clear difference between the two initialisation conditions: when starting from the *Meander* policy, the final KL divergence remains relatively lower across the coefficient range, suggesting that the policy requires smaller adjustments to accommodate the new task. In contrast, when starting from the *BLine* policy, substantially higher KL divergences are observed at lower coefficients, indicating that greater policy change is necessary to integrate the new behaviour.

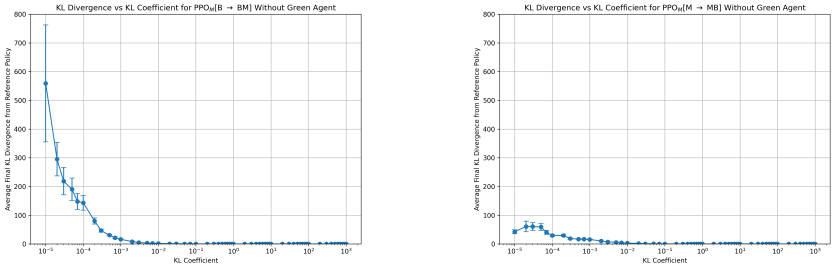

Fig. 3. Results Showing the Effect of the KL Coefficient on KL Divergence (without Green Agents)

5.3 Transfer Learning Multiple Times

In this experiment, we evaluate the ability of the modified PPO algorithm to perform sequential task learning across multiple red agent behaviours while retaining previously acquired knowledge. The setup involves taking the policies from the previous experiment and then continuing training on the other red

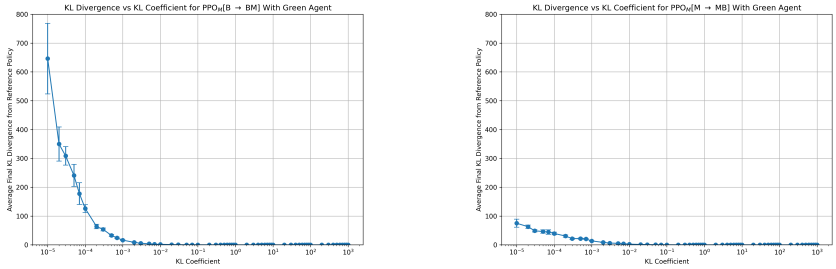

Fig. 4. Results Showing the Effect of the KL Coefficient on KL Divergence (with Green Agents)

agent (*Hybrid*) using the modified PPO approach, which incorporates the previous policy as a reference policy. The objective is to investigate how the initial policy may affect the generation of the generic policy.

As shown in Table 3, the modified PPO policies maintain stable performance even when the process is repeated. The results indicate that successive fine tuning steps do not lead to significant degradation in performance across the evaluated environments, suggesting that the method is robust to multiple rounds of this process.

Table 3. Performance of Modified PPO Policies when additional Red Agents are considered

Policy	KL Coefficient	Meander (M)	BLine (B)	Hybrid (H)	Total
$\mathbf{PPO}_M$[M→MB]	0.002	-25.70 ± 0.11	-23.79 ± 0.32	-19.10 ± 0.15	-68.59 ± 0.37
$\mathbf{PPO}_M$[MB→MBH]	0.002	**-25.44 ± 0.12**	-21.33 ± 0.32	-19.55 ± 0.16	**-66.32 ± 0.38**
$\mathbf{PPO}_M$[B→BM]	0.002	-28.15 ± 0.12	-31.23 ± 0.53	-22.48 ± 0.27	-81.86 ± 0.61
$\mathbf{PPO}_M$[BM→BMH]	0.002	-27.24 ± 0.13	**-21.28 ± 0.34**	**-19.44 ± 0.16**	-67.96 ± 0.40

Figure 5 shows the effect of increasing the KL coefficient on the overall policy performance after repeating the process. As the coefficient increases, performance degradation that occurred during earlier transfer stages is carried forward, resulting in reduced performance not only on the previously learned tasks but also on the most recently introduced red agent. This suggests that overly constraining the policy to remain close to the reference can limit its capacity to recover from prior sub-optimal adaptations, ultimately harming performance across both old and new tasks. However, when the KL coefficient is lower, the mean rewards are stable, when compared with Figure 2, lower values of the KL coefficient are more stable, suggesting that the more the procedure is repeated, the more stable it would become as it finds the optimal generic policy.

Figure 6 presents the KL divergence of the final policy with respect to the reference policy after repeating the process, plotted against the KL coefficient. Interestingly, compared to Figure 4, the trend has now reversed: the policy that

Fig. 5. Results Showing the Effect of the KL Coefficient on Mean Rewards when Repeating the Process for a Third Red Agent (with Green Agents)

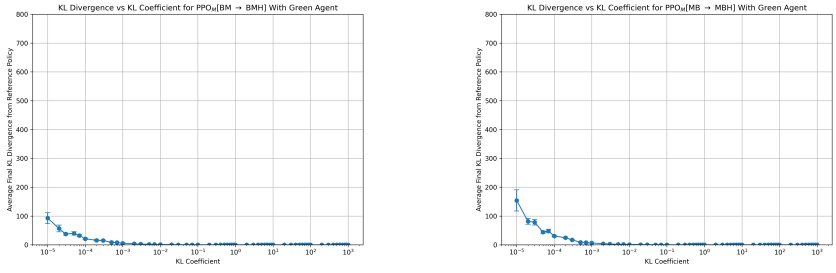

Fig. 6. Results Showing the Effect of the KL Coefficient on KL Divergence when Repeating the Process for a Third Red Agent (with Green Agents)

was initially trained on *BLine* exhibits consistently lower KL divergence, while those that started from *Meander* display higher divergence across the coefficient range. This suggests that after multiple transfer stages, the policy initialised from *BLine* remains closer to its reference, whereas the policy initialised from *Meander* requires larger adjustments to accommodate the accumulated changes introduced by subsequent tasks. This observation highlights how the initial training path can influence the long-term policy stability under knowledge retention.

Table 4 demonstrates that when this process is initialised from the *Hybrid* red agent, the resulting policies maintain comparable performance. The exception is $\mathbf{PPO}_M$**[**HB→HBM**]**, where performance against the *BLine* agent deteriorates significantly. This decline is caused by an insufficient KL coefficient, as illustrated in Fig. 7, which shows that increasing the KL coefficient yields better performance.

6 Conclusion

To enable scalable and generic reinforcement learning policies for network defence, we present a modified PPO algorithm designed for sequential, continual

Table 4. Performance of Modified PPO Policies when initialised from the *Hybrid* Red Agent

Policy	KL Coefficient	Meander (M)	BLine (B)	Hybrid (H)	Total
$\mathbf{PPO}_M$ [H]	N/A	-36.49 ± 0.24	-30.15 ± 0.45	-21.93 ± 0.22	-88.57 ± 0.56
$\mathbf{PPO}_M$ [H→HM]	0.002	-28.78 ± 0.11	-29.21 ± 0.25	-22.32 ± 0.15	-80.31 ± 0.31
$\mathbf{PPO}_M$ [HM→HMB]	0.002	**-27.37 ± 0.15**	-22.02 ± 0.31	-19.43 ± 0.17	**-68.82 ± 0.38**
$\mathbf{PPO}_M$ [H→HB]	0.002	-39.88 ± 0.34	**-20.94 ± 0.36**	-21.63 ± 0.26	-82.45 ± 0.56
$\mathbf{PPO}_M$ [HB→HBM]	0.002	-29.35 ± 0.16	-37.39 ± 0.68	-21.97 ± 0.29	-88.71 ± 0.76

Fig. 7. Results Showing the Effect of the KL Coefficient on Mean Rewards when Initialised from the *Hybrid* Red Agent (with Green Agents)

multi-task learning. Our approach allows agents to efficiently incorporate new attack behaviours while retaining previously acquired knowledge. Unlike existing methods, it enables controlled knowledge retention during transfer, balancing stability and plasticity. By building on existing policies rather than retraining from scratch, this method reduces computational cost and data requirements, while supporting continuous learning as new threats emerge.

We have demonstrated that the modified PPO algorithm consistently outperforms standard PPO in scenarios involving learning sequential multiple attack behaviours. The inclusion of the reference policy and KL coefficient enables the agent to generalise across distinct attacker behaviours, achieving higher average performance while mitigating catastrophic forgetting. However, results suggest that since the initial policy becomes a reference for controlling knowledge retention, depending on the initial policy, we may achieve a different final policy.

We also observed that, depending on the policy and the new attack behaviour, there is a need for a different KL coefficient, which would require performing hyperparameter tuning for any new combination between existing policy and attack behaviour. An alternative solution, to be investigated as future work, could be to include the KL coefficient as a learnable parameter.

Bibliography

[1] Acuto, A., Maskell, S., Jack, D.: Defending the unknown: Exploring reinforcement learning agents' deployment in realistic, unseen networks. In: CAMLIS, pp. 22–35 (2023)

[2] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV), pp. 139–154 (2018)

[3] Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. In: International conference on machine learning, pp. 166–175, PMLR (2017)

[4] Dehghantanha, A., Yazdinejad, A., Parizi, R.M.: Autonomous cybersecurity: Evolving challenges, emerging opportunities, and future research trajectories. In: Proceedings of the Workshop on Autonomous Cybersecurity, p. 1–10, AutonomousCyber '24, Association for Computing Machinery, New York, NY, USA (2024), ISBN 9798400712296, https://doi.org/10.1145/3689933.3690832, URL https://doi.org/10.1145/3689933.3690832

[5] Ernst, D., Stan, G.B., Goncalves, J., Wehenkel, L.: Clinical data based optimal sti strategies for hiv: a reinforcement learning approach. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 667–672, IEEE (2006)

[6] Farooq, M.O., Kunz, T.: A Generic Blue Agent Training Framework for Autonomous Cyber Operations. In: 2024 IFIP Networking Conference (IFIP Networking), pp. 515–521 (Jun 2024), https://doi.org/10.23919/IFIPNetworking62109.2024.10619771, iSSN: 1861-2288

[7] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. Proceedings of the AAAI Conference on Artificial Intelligence **32** (09 2017), https://doi.org/10.1609/aaai.v32i1.11694

[8] Isele, D., Cosgun, A.: Selective experience replay for lifelong learning. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (Apr 2018), https://doi.org/10.1609/aaai.v32i1.11595, URL https://ojs.aaai.org/index.php/AAAI/article/view/11595

[9] Kiely, M., Bowman, D., Standen, M., Moir, C.: On Autonomous Agents in a Cyber Defence Environment (Sep 2023), https://doi.org/10.48550/arXiv.2309.07388, URL http://arxiv.org/abs/2309.07388, arXiv:2309.07388 [cs]

[10] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017), https://doi.org/10.1073/pnas.1611835114, URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[11] Kott, A.: Autonomous Intelligent Cyber-defense Agent: Introduction and Overview, pp. 1–15 (06 2023), ISBN 978-3-031-29268-2, https://doi.org/10.1007/978-3-031-29269-9_1

[12] Loevenich, J., Adler, E., Hürten, T., Lopes, R.R.F.: Design and evaluation of an autonomous cyber defence agent using drl and an augmented llm. Computer Networks **262**, 111162 (2025), ISSN 1389-1286, https://doi.org/https://doi.org/10.1016/j.comnet.2025.111162, URL https://www.sciencedirect.com/science/article/pii/S1389128625001306

[13] Nevmyvaka, Y., Feng, Y., Kearns, M.: Reinforcement learning for optimized trade execution. In: Proceedings of the 23rd international conference on Machine learning, pp. 673–680 (2006)

[14] Palmer, G., Parry, C., Harrold, D.J.B., Willis, C.: Deep Reinforcement Learning for Autonomous Cyber Defence: A Survey (Sep 2024), https://doi.org/10.48550/arXiv.2310.07745, URL http://arxiv.org/abs/2310.07745, arXiv:2310.07745 [cs]

[15] Parisotto, E., Ba, J.L., Salakhutdinov, R.: Actor-mimic: Deep multitask and transfer reinforcement learning (2016), URL https://arxiv.org/abs/1511.06342

[16] Peng, X.B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. In: 2018 IEEE international conference on robotics and automation (ICRA), pp. 3803–3810, IEEE (2018)

[17] Rusu, A.A., Colmenarejo, S.G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., Hadsell, R.: Policy distillation (2016), URL https://arxiv.org/abs/1511.06295

[18] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017), URL https://arxiv.org/abs/1707.06347

[19] Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International conference on machine learning, pp. 4528–4537, PMLR (2018)

[20] Symes Thompson, I., Caron, A., Hicks, C., Mavroudis, V.: Entity-based Reinforcement Learning for Autonomous Cyber Defence. In: Proceedings of the Workshop on Autonomous Cybersecurity, pp. 56–67, AutonomousCyber '24, Association for Computing Machinery, New York, NY, USA (Nov 2024), ISBN 979-8-4007-1229-6, https://doi.org/10.1145/3689933.3690835, URL https://dl.acm.org/doi/10.1145/3689933.3690835

[21] van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning (2019), URL https://arxiv.org/abs/1904.07734

[22] Wolk, M., Applebaum, A., Dennler, C., Dwyer, P., Moskowitz, M., Nguyen, H., Nichols, N., Park, N., Rachwalski, P., Rau, F., Webster, A.: Beyond CAGE: Investigating Generalization of Learned Autonomous Network Defense Policies (Nov 2022), https://doi.org/10.48550/arXiv.2211.15557, arXiv:2211.15557 [cs]

[23] Xu, Z., Wu, K., Che, Z., Tang, J., Ye, J.: Knowledge transfer in multi-task deep reinforcement learning for continuous control. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020), ISBN 9781713829546