

Unveiling the Black Box: A Multi-Layer Framework for Explaining Reinforcement Learning-Based Cyber Agents

Diksha Goel^{1,2}, Kristen Moore^{1,2}, Jeff Wang^{1,2}, Minjune Kim^{1,2}, and Thanh Thi Nguyen³

¹ CSIRO's Data61, Clayton, Australia

² Cyber Security Cooperative Research Centre (CSCRC), Joondalup, Australia
{diksha.goel, kristen.moore, jeff.wang, minjune.kim}@data61.csiro.au

³ Monash University, Melbourne, Australia
thanh.nguyen9@monash.edu

Abstract. Reinforcement Learning (RL) agents are increasingly used to simulate sophisticated cyberattacks, but their decision-making processes remain opaque, hindering trust, debugging, and defensive preparedness. In high-stakes cybersecurity contexts, explainability is essential for understanding how adversarial strategies are formed and evolve over time. In this paper, we propose a unified, multi-layer explainability framework for RL-based attacker agents that reveals both strategic (MDP-level) and tactical (policy-level) reasoning. At the MDP level, we model cyberattacks as a Partially Observable Markov Decision Processes (POMDPs) to expose exploration-exploitation dynamics and phase-aware behavioural shifts. At the policy level, we analyse the temporal evolution of Q-values and use Prioritised Experience Replay (PER) to surface critical learning transitions and evolving action preferences. Evaluated across CyberBattleSim environments of increasing complexity, our framework offers interpretable insights into agent behaviour at scale. Unlike previous explainable RL methods, which are often post-hoc, domain-specific, or limited in depth, our approach is both agent- and environment-agnostic, supporting use cases ranging from red-team simulation to RL policy debugging. By transforming black-box learning into actionable behavioural intelligence, our framework enables both defenders and developers to better anticipate, analyse, and respond to autonomous cyber threats.

Keywords: Explainable Artificial Intelligence · Reinforcement Learning · Autonomous Cyber Operations · Adversarial AI · Cyberattack Modelling

1 Introduction

Modern enterprise networks face persistent threats from sophisticated adversaries who exploit misconfigurations to conduct lateral movement and privilege escalation attacks. As these threats grow in scale and complexity, static and reactive defences often fall short. They lack the agility to detect and respond to dynamic,

multi-stage campaigns, highlighting the need for intelligent and adaptive security solutions [17]. Reinforcement Learning (RL)-based autonomous agents have emerged as promising solutions for modelling such adversarial behaviour, with growing use in platforms like CyberBattleSim [18], MITRE’s FARLAND [11], and CybORG [1].

Despite their increasing adoption, RL agents remain *opaque black boxes*, providing little visibility into why particular actions are taken, how strategies evolve, or which factors drive success or failure. This lack of interpretability poses barriers for both cybersecurity professionals and RL developers. For defenders, it limits the ability to anticipate threats or understanding adversarial reasoning. For developers, it hinders debugging, validation, and policy refinement in complex environments where decisions unfold over time under uncertainty.

Explainable Reinforcement Learning (XRL) aims to bridge this gap by converting RL agent behaviour into interpretable, context-aware insights that support trust, transparency, and safe deployment [22]. However, most prior explainability work in cybersecurity is limited to narrow tasks (e.g., malware detection, blockchain mining, autonomous driving, or small-scale intrusion detection [2,6,10,15,23]) and often relies on shallow, post-hoc methods. Few approaches support structured, temporally-aware explainability for RL agents operating in realistic, sequential, and partially observable adversarial settings [9,12].

To address this gap, we introduce a unified, multi-layer explainability framework tailored for RL agents in cybersecurity. While we demonstrate it with autonomous attacker agents, the framework is *agent- and environment-agnostic* and applies equally to defenders. It provides two complementary lenses. At the MDP level, we model the environment as a Partially Observable Markov Decision Process (POMDP), enabling analysis of exploration-exploitation dynamics and behavioural shifts across early and late attack phases. At the policy level, we track the temporal Q-value evolution and incorporate Prioritised Experience Replay (PER) to identify critical learning transitions and preference formation over time. Together, these layers reveal how the agent behaviour matures, identifying intent, surfacing key decision points, and clarifying strategic inflection. This empowers analysts to trace agent learning over time and supports use cases ranging from red-team planning to model debugging.

We evaluate our framework using *Microsoft’s CyberBattleSim* [18], which simulates attacker behaviour in networked environments with privilege escalation, credential reuse, and lateral movement. Across environments of increasing complexity, our framework demonstrates how strategic patterns and tactical choices emerge under uncertainty. While our analysis focuses on attackers, the methodology generalises to defenders and other sequential decision-making domains as well such as malware containment, deception planning, or autonomous incident response. *This positions our framework as a unified approach for explaining both red- and blue-team RL agents, enabling phase-aware defence design, training simulation enhancement, and policy transparency in autonomous cyber operations.* Our key contributions are as follows:

- ***Multi-layer explainability framework:*** We present a unified, multi-layer explainability framework for RL-based attacker agents. The framework inte-

grates strategic (MDP-level) and tactical (policy-level) insights, bridging the transparency gap and enabling end-to-end interpretability of autonomous attacker behaviour.

- **MDP-level explainability:** We model attacker behaviour as POMDP, enabling interpretable analysis of uncertainty handling and strategic progression through early and late phases of attack.
- **Policy-level explainability:** We use temporal Q-value tracking and prioritised experience replay to reveal how action preferences and critical decisions evolve during learning.
- **Empirical analysis:** We conduct rigorous experiments in CyberBattleSim environments of increasing scale and complexity, showcasing the framework’s scalability and insightfulness.
- **Operational and developmental utility:** Our framework supports both cyber defenders and RL developers by providing interpretable signals for anticipating behaviour, debugging policies, and designing adaptive responses.

2 Related Work

2.1 Reinforcement Learning for Cybersecurity

2.1.1 Simulation Environments. Several simulation environments support RL research in cybersecurity, each with distinct emphases. CyberBattleSim [18] models enterprise networks as graphs to train attacker agents on privilege escalation, lateral movement, and exploitation, with rewards tied to node criticality. NASim [14] offers a lightweight, Gym-compatible framework for penetration testing under partial observability. In contrast, CybORG [1] and FARLAND [11] focus on defender training, integrating emulation (CybORG) or scalable probabilistic models (FARLAND) to support adaptive blue team agents and complex attack scenarios.

2.1.2 RL-based Cyber Defence and Adversarial Simulation. Recent work has explored both defensive and offensive RL agents. Thompson et al. [19] used entity-based RL with transformer policies to enable defensive generalisation in dynamic networks. Goel et al. [7] co-trained attacker and defender policies in a Stackelberg game for Active Directory protection. On the offensive side, Sultana et al. [16] trained deep RL agents to learn multi-layered cyberattack strategies across network, service, and application layers. Other approaches use RL to simulate red team behaviour for defender training [3, 5].

Unlike these efforts, which aim to optimise RL agent performance, our approach focuses on interpreting agent behaviour, providing a principled, multi-layer framework for understanding attacker decision-making in adversarial settings.

2.2 Explainable Reinforcement Learning (XRL)

2.2.1 General XRL Techniques. Explainable reinforcement learning methods fall into two categories: intrinsic approaches [2], which embed interpretability

during training (e.g., policy distillation, influence modelling), and post-hoc techniques [8, 10], which generate explanations after training using tools like SHAP, LIME, or counterfactual reasoning. For example, Alabdulkarim et al. [2] developed an intrinsic approach by training influence predictors alongside RL policies, enabling real-time analysis of how different actions shape agent behaviour over time. In contrast, Mathes et al. [10] introduced CODEX, a post-hoc, cluster-based XRL technique that semantically groups and summarizes both factual and counterfactual behaviours in environments such as MiniGrid and StarCraft II. Gyevnar et al. [8] proposed a social XAI framework for multi-agent autonomous driving, using post-hoc counterfactual causal selection to generate contrastive explanations for sequential decisions.

2.2.2 XRL Techniques in Cybersecurity. XRL in cybersecurity has largely targeted static classification tasks (e.g., malware detection [23], intrusion analysis [15]) rather than sequential decision-making. Yu et al. [23] incorporated explainability into malware mutation agents, while Sharma et al. [15] explored adversarial XRL for classifier sensitivity. Foley et al. [6] proposed a post-hoc XRL framework for defender agents in CybORG using SHAP, feature ablations, and state visualisations to improve situational awareness. However, their attacker agents follow fixed, hierarchical rules, limiting insight into real-time red-team adaptation.

By contrast, our work places the attacker at the centre of the explanation. We introduce a general, environment-agnostic XRL framework that captures attacker behaviour across both strategic (MDP-level) and tactical (policy-level) layers. Unlike prior work, it supports both during- and post-training analysis and reveals how adversarial agents adapt over time. This empowers defenders to pre-empt adversarial tactics and developers to refine agent training, while offering actionable insight into sequential behaviours that remain opaque in existing approaches.

3 Problem Formulation

Enterprise networks are increasingly targeted by lateral movement attacks, where adversaries navigate interconnected systems to escalate privileges and compromise critical assets. In Microsoft’s *CyberBattleSim*, such networks are modelled as directed graphs $G = (V, E)$, where nodes $v \in V$ represent machines (e.g., workstations, servers, domain controllers) and edges $e \in E$ represent communication links. Each node has an associated reward $r(v)$, reflecting its operational importance. An attacker, modelled as a RL agent, seeks to maximise cumulative reward $R = \sum_{v \in V_C} r(v)$, where $V_C \subseteq V$ denotes the set of compromised nodes. The agent operates under POMDP, reflecting the uncertainty and limited visibility typical of real-world cyberattacks. While RL agents can learn effective attack policies in this setting, their decision-making processes remain opaque. Traditional explainability methods, such as static feature attribution and post-hoc visualisations, fall short of capturing the sequential, adaptive nature of adversarial behaviour. As a result, defenders lack insight into *when*, *why*, and *how* attackers

pivot laterally, escalate privileges, or shift tactics, insights that are critical for proactive defence and trust in autonomous systems.

4 CyberBattleSim Platform

CyberBattleSim is an open-source simulation platform designed to model adversarial cyber operations within enterprise network environments. Networks are represented as directed graphs, where nodes encapsulate system configurations such as operating systems, services, known vulnerabilities, firewall rules, and edges represent connectivity. The environment is inherently attacker-centric: it trains RL agents to autonomously discover and exploit vulnerabilities through local privilege escalation, remote exploitation, and credential-based lateral movement. Unlike CybORG [1], which emphasises defender modelling and blue-team coordination, CyberBattleSim provides a focused setting for evaluating autonomous red-team strategies. Defender agents are not modelled explicitly; instead, defensive behaviour is encoded through static network configurations, access control rules, and hidden vulnerabilities.

Agents begin with limited visibility and explore to uncover system properties and reachable nodes. The reward model incentivises strategic behaviour, assigning higher rewards to the compromise of mission-critical nodes (e.g., 100 for a domain controller vs. 10 for a workstation). Successfully breaching a node may also reveal previously hidden segments of the network, enabling progressive expansion of the attack agent’s surface. CyberBattleSim supports rich action spaces and complex environment dynamics, making it well-suited for studying both autonomous attacker behaviour and explainability in sequential, partially observable domains.

4.1 CyberBattleSim Environments

CyberBattle Capture-the-Flag (CTF) Environment. This environment uses a loosely connected, hub-and-spoke topology with nodes representing services like web apps, GitHub, Azure storage, SharePoint, and cloud VMs. Each node supports multiple exploit paths, such as scanning content, mining Git history, or reusing leaked credentials. Supported actions include SearchEdgeHistory, ScanPageContent, NavigateWebDirectory, AccessDataWithSASToken, and privilege escalations. Vulnerabilities are contextually embedded (e.g., exposed tokens, weak MySQL/SSH logins), requiring chained reasoning. Its non-linear structure supports analysis of both strategic variability and tactical adaptation.

CyberBattleChain Environment. CyberBattleChain simulates a linear, fixed-sequence network topology alternating between Linux and Windows nodes: `start → (Linux → Windows)n → Linux[Flag]`. Progression requires exploiting platform-specific vulnerabilities and reusing credentials revealed upon node compromise (e.g., SSH credentials from Windows to the subsequent Linux node). Embedded traps lure agents into suboptimal paths, requiring strategic discernment to avoid costly dead ends. The environment ends with a high-value target ("flag" node), making it ideal for evaluating long-horizon planning, sequential decision-making, and explainability in high-stakes attacker behaviour.

5 Multi-Layer Explainability Framework for RL-Based Attackers

To address the opacity of RL agents in adversarial environments, we propose a *unified, multi-layer explainability framework* that reveals both strategic intent and tactical decision-making over time. Grounded in core explainable AI principles, such as transparency, temporal insight, and actionable reasoning, our framework operates at two complementary levels:

1. **Strategic-Level (MDP):** Modelling attacker behaviour under uncertainty, capturing exploration-exploitation dynamics and phase transitions across the attack lifecycle.
2. **Tactical-Level (Policy):** Tracing evolving action preferences and identifying high-impact decision points using temporal Q-value analysis and prioritised experience replay.

This layered design enables defenders to interpret not just what actions were taken, but why and when they were chosen, supporting trust, auditability, and more targeted defence planning.

5.1 Strategic-Level (MDP) Explainability

We formalise attacker behaviour as *Partially Observable Markov Decision Process*, defined by the tuple $(S, A, \Omega, O, T, R, \gamma)$, where S is the set of true (hidden) states, A the action space, Ω the set of possible observations, $O : S \times A \rightarrow \mathcal{P}(\Omega)$ is the observation function (a probability distribution over observations), $T : S \times A \rightarrow \mathcal{P}(S)$ the state transition function, $R : S \times A \rightarrow \mathbb{R}$ the reward function, and $\gamma \in [0, 1]$ the discount factor. This formulation captures uncertainty and limited visibility adversaries face in real networks, and allows us to analyse strategic adaptations over time.

5.1.1 Exploration-Exploitation Dynamics

We examine how different exploration strategies shape the agent’s learning and behaviour:

1. **Early Exploitation Strategy:** This strategy models adversaries who aim for fast wins. The agent begins with a low exploration rate (ϵ_{low}) up to step T_{exploit} , after which the exploration rate gradually increases, prioritising rapid exploitation before defences adapt [20]. We define the exploration rate ϵ_t at time step t as:

$$\epsilon_t = \begin{cases} \epsilon_{\text{low}}, & t \leq T_{\text{exploit}} \\ \epsilon_{\text{low}} + \frac{(\epsilon_{\text{high}} - \epsilon_{\text{low}})(t - T_{\text{exploit}})}{T - T_{\text{exploit}}}, & t > T_{\text{exploit}} \end{cases}$$

2. **Standard Exploration Strategy:** This strategy reflects more reconnaissance-heavy adversaries. The agent begins with high exploration (ϵ_{high}) until step T_{explore} , after which exploration gradually decays in favour of exploiting the

learned policy. Such agents aim for long-term advantage by systematically mapping vulnerabilities and refining their actions for maximum impact.

$$\epsilon_t = \begin{cases} \epsilon_{\text{high}}, & t \leq T_{\text{explore}} \\ \epsilon_{\text{high}} - \frac{(\epsilon_{\text{high}} - \epsilon_{\text{low}})(t - T_{\text{explore}})}{T - T_{\text{explore}}}, & t > T_{\text{explore}} \end{cases}$$

where, T denotes the total number of training steps; t is the current training step, T_{exploit} and T_{explore} specify the step thresholds that mark the end of the initial low- and high-exploration phases, respectively, for each strategy.

Comparing these schedules reveals how different attacker profiles prioritise speed, breadth, or depth in their learning trajectories and strategic development.

5.1.2 Phase Transitions and Behaviour Evolution

To further capture strategic adaptation, we segment attacker behaviour into *early* and *late* attack phases based on the network compromise ratio (C_t):

$$C_t = \frac{|\text{Compromised nodes at time } t|}{|N|}$$

where C_t represents the proportion of compromised nodes, and $|N|$ denotes the total number of nodes in the network.

1. **Early Phase:** In the early stages of an attack, strong defences and limited visibility are expected to constrain the agent’s progress, prompting a cautious, reconnaissance-driven approach. This phase is typically characterised by uncertainty, minimal network compromise, and broad probing, as the agent gathers information and identifies potential vulnerabilities. To capture this behaviour, our framework isolates transitions from early phases of the attack, where only a small fraction of the network has been compromised, enabling analysis of how strategies form under high uncertainty ($C_t < \text{Threshold}$).
2. **Late Phase:** As more nodes are compromised and visibility improves, the agent is expected to transition to a more aggressive phase of the attack. With increased access and diminishing defensive barriers, this phase is typically associated with behaviours such as privilege escalation, lateral movement, and targeting of high-value assets. To capture this shift, our framework isolates transitions where the network compromise ratio exceeds a defined threshold, enabling analysis of how attacker strategies evolve from broad exploration to more focused, high-impact exploitation ($C_t > \text{Threshold}$).

5.2 Tactical-Level (Policy) Explainability

To complement the strategic lens provided by MDP-level modelling, our framework incorporates a tactical layer focused on how the agent’s preferences evolve during learning. This layer provides insights into how specific actions become favoured over time, and which experiences most influence policy updates. We achieve this through two techniques: (1) tracking temporal Q-value evolution and (2) leveraging Prioritised Experience Replay. Together these approaches support temporal attribution and highlight learning dynamics behind critical decisions, key goals in explainable AI.

5.2.1 Temporal Q-Value Evolution and Action Preference

We track the evolution of Q-values over time to analyse how the agent’s assessment of state-action pairs changes with experience. In RL settings, Q-values begin largely undifferentiated due to sparse early feedback, and progressively evolve to reflect long-term reward expectations. Our framework leverages this temporal progression to surface how tactical preferences may emerge over time, offering insight into the agent’s evolving prioritisation of specific actions. The Q-values are updated via the Q-learning rule [21]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

where $\alpha \in (0, 1]$ is the learning rate controlling the update magnitude, γ is the discount factor, r is the immediate reward, and s' is the next state.

Over time, the Q-values converge toward the expected return defined by the Bellman expectation equation under the learned policy [4]:

$$Q(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

where $Q(s, a)$ denotes the expected cumulative reward for taking action a in state s , γ is the discount factor determining the importance of future rewards, and r_t is the reward received at time step t .

Tracking Q-value progression allows us to infer *when* an agent’s behaviour transitions from broad exploration (e.g., probing for vulnerabilities) to targeted exploitation (e.g., privilege escalation or lateral movement toward critical assets). This *temporal Q-value analysis* supports temporally grounded explanations of evolving intent and highlights emerging tactical focus within agent’s policy.

5.2.2 Policy Interpretation via Prioritised Experience Replay

Prioritised Experience Replay complements Q-value evolution by revealing what drives learning. It amplifies the influence of high-impact transitions, those with large Temporal-Difference (TD) errors, focusing the agent’s attention on experiences that most significantly shape policy updates.

Prioritised Experience Replay. Each experience tuple $\tau_i = (s_i, a_i, r_i, s'_i, d_i)$ is stored in a replay buffer \mathcal{D} , where $d_i \in \{0, 1\}$ is a binary done flag indicating whether the transition leads to episode termination. Rather than sampling these transitions uniformly, PER prioritises transitions with higher learning potential and assigns a priority p_i based on the transition’s TD error δ_i [13] as:

$$\delta_i = r_i + \gamma \max_{a'} Q(s'_i, a') - Q(s_i, a_i)$$

Transitions with large δ_i (e.g., unexpected outcomes, high-value exploits) receive higher priority:

$$p_i = |\delta_i| + \epsilon_0, \quad P(\tau_i) = \frac{p_i^\alpha}{\sum_{j \in \mathcal{D}} p_j^\alpha},$$

where p_i is the priority; ϵ_0 avoids zero priority; α controls prioritisation; $P(\tau_i)$ is the sampling probability; \mathcal{D} is the replay buffer. Importance-sampling weights are applied to ensure unbiased learning updates.

By surfacing the transitions that most influence learning, such as early successes in credential acquisition or late-stage access to high-value nodes, PER reveals the moments of highest explanatory value in the agent’s trajectory. These inflection points help clarify how the agent’s tactics evolve in response to new information and highlight where strategic shifts are most likely to occur.

By integrating strategic-level progression with tactical decision analysis, our framework provides a comprehensive and temporally grounded explanation of RL-based attacker behaviour. It elucidates not only the actions taken by the agent, but also the underlying rationale, timing, and policy evolution driving those decisions. This layered interpretability enhances transparency, supports informed defensive planning, and enables the development of phase-aware countermeasures aligned with the agent’s behavioural dynamics.

6 Experimental Evaluation

We evaluate our framework across five experimental setups designed to highlight key explainability dimensions over raw performance: (1) baseline policy selection, (2) exploration-exploitation behaviour, (3) phase-aware behavioural shifts, (4) temporal Q-value evolution, and (5) strategic learning signals via PER. These experiments serve as illustrative case studies demonstrating how our framework can analyse and interpret RL agents in adversarial cybersecurity scenarios.

Environment Description. We evaluated our framework using *CyberBattleSim* [18] across two core environments: (1) *CTF*, a 12-node targeted attack scenario, and (2) *CyberBattleChain*, with three scalable variants, CC22, CC100, and CC500, simulating networks of 22, 100, and 500 nodes, respectively. Table 1 summarises these configurations, where “Size” denotes exploitable nodes and “Max Nodes” includes the entire infrastructure, such as firewalls and auxiliary components.

Training Parameters. We scaled the architecture of the Deep Q-Network to the environment complexity: 3 hidden layers were used for smaller scenarios (CTF, CC22), and 5 layers for larger ones (CC100, CC500). Agents used a replay buffer of 20000 transitions, a target network update frequency of 10 episodes, a learning rate of 0.005 with the Adam optimiser, and batch size of 128. The

Table 1: Description of CyberBattleSim Environment.

Environment	Size	Max Nodes
CTF	9	12
CC22	12	22
CC100	70	100
CC500	350	500

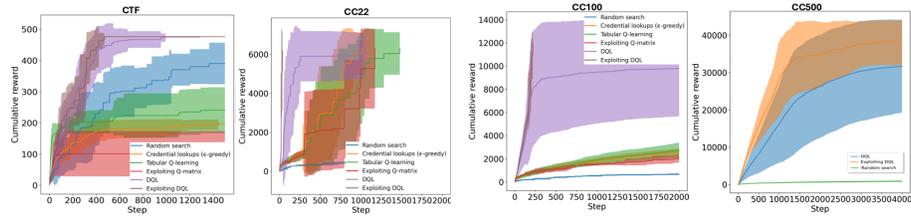


Fig. 1: Cumulative Rewards Comparison of Attacker Policies Across Environments (The shaded region in the figures represents the standard deviation of cumulative rewards across training steps for each agent).

discount factor was set to $\gamma = 0.95$. Unless stated otherwise, exploration followed a standard ϵ -greedy policy, decaying from $\epsilon = 0.90$ to $\epsilon = 0.10$ over 5000 steps. All experiments are implemented in PyTorch and run on a high-performance cluster with 1 CPU and 256 GB RAM.

6.1 Setup 1: Baseline Policy Selection

To ensure that the interpretability results are grounded in a stable, scalable attacker policy, we benchmarked six RL-based agents: Random Search, Credential Lookup (ϵ -greedy), Tabular Q-Learning, Exploiting Q-Matrix, Deep Q-Learning (DQL), and Exploiting DQL. These agents were evaluated across CTF, CC22, CC100, and CC500 using cumulative reward as the primary metric.

Results. Figure 1 presents a comparative summary of agent performance across all environments and the results show that DQL significantly outperformed the baseline agents in terms of stability, scalability, and cumulative reward. Classical methods such as Tabular Q-Learning and Credential Lookup struggled in larger environments (CC100+, with no viable results in CC500). Random search underperformed in all structured settings. While the Exploiting DQL variant achieved slightly higher cumulative rewards, it is not a standalone method; it relies on a fully trained DQL agent and simply exploits the final policy without any further training or adaptation. As such, it does not reflect the ongoing decision-making process that our explainability framework aims to explain. We therefore, selected DQL as the attacker policy for all subsequent experiments. We emphasise that the results in Figure 1 serve to establish a realistic setting to demonstrate the utility of our multi-layer explainability framework.

6.2 Setup 2: Exploration-Exploitation Dynamics

To analyse how exploration-exploitation strategies shape agent behaviour, we compare two strategies: Early and Standard Exploration and track their impact on cumulative reward and node discovery rate across environments. This reveals whether agent behave opportunistically, methodically, or inefficiently over time.

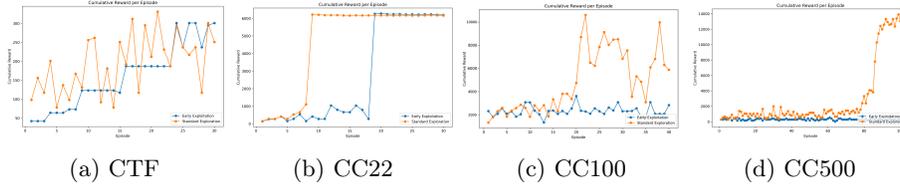


Fig. 2: Impact of Exploration Strategies on Cumulative Rewards Across Environments.

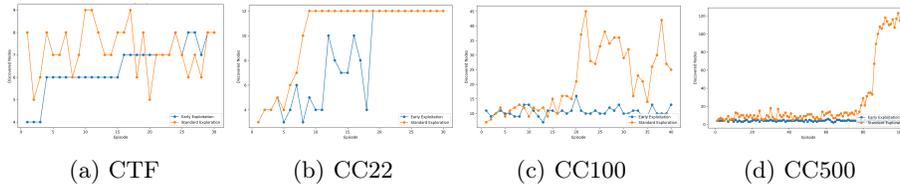


Fig. 3: Impact of Exploration Strategies on Node Discovery Rate Across Environments.

Results. For Early Exploitation, we initialised ϵ at 0.001 and gradually increased it to 0.9 at a growth rate of 0.005. In contrast, Standard Exploration began with $\epsilon = 0.9$ and decayed to 0.01 at a rate of 0.95. Training was conducted for 30 episodes (200 iterations) in CTF and CC22, 40 episodes (300 iterations) in CC100 and 100 episodes (300 iterations) in CC500. Figure 2 shows that in smaller topologies (CTF and CC22), *Standard Exploration* quickly outperformed *Early Exploitation*, benefitting from more aggressive probing. In larger environments (CC100 and CC500), this advantage widened considerably, with Early Exploitation often stagnating at lower cumulative rewards, especially in CC500, where Standard Exploration achieved a notable reward surge after sustained, wide-ranging probing. These patterns are echoed in the node discovery rates in Figure 3 where Standard Exploration uncovered more nodes earlier, while Early Exploitation remained constrained.

Our findings highlight how exploration strategy shapes agent performance and behavioural style. Standard Exploration encourages deliberate, reconnaissance-driven tactics, while Early Exploitation creates opportunistic agents well-suited to complex environments. Our framework surfaces these differences, supporting interpretable assessments of attacker risk posture and planning style.

6.3 Setup 3: Phase-Aware Behaviour Evolution

This setup supports phase-aware explainability by segmenting attacker behaviour into early and late stages, using the compromise ratio C_t (the proportion of compromised nodes) to highlight when strategy shifts occur.

Results. Agents were trained for 20 episodes (200 iterations) in CTF and CC22 environments, 100 episodes (400 iterations) in CC100, and 200 episodes (400 iterations) in CC500. *Early phase* is defined as network compromise ratio $C_t < 0.5$,

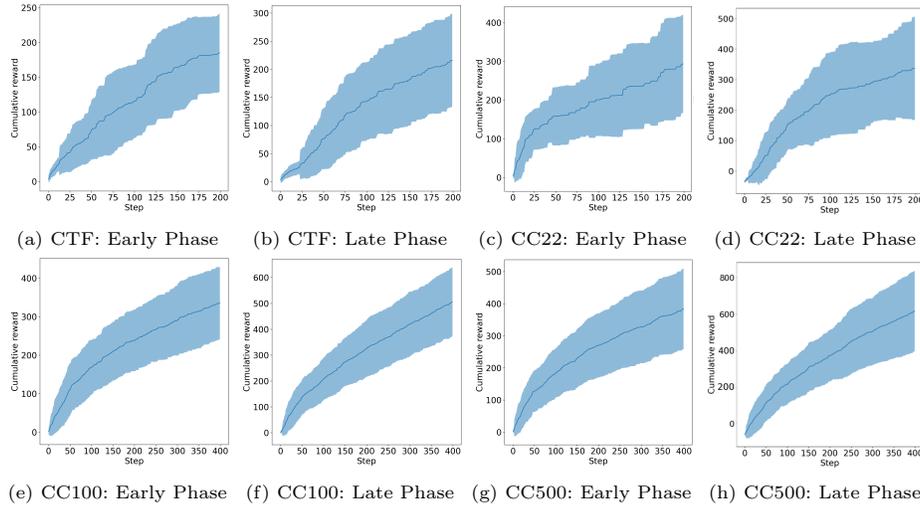


Fig. 4: Cumulative Rewards Comparison in Early vs. Late Phase Attack Across Environments.

and *late phase* as $C_t \geq 0.5$. Figure 4 shows that early-phase rewards remain modest due to limited visibility and cautious probing. Late-phase curves begin with higher rewards and steeper slopes, reflecting the agent’s growing confidence and accumulated access.

In CyberBattleChain environments, the attack path is fixed: each node grants access to the next. Behavioural variation therefore reflects how quickly and efficiently the agent escalates privileges, selects exploits, and avoids traps, rather than differences in path selection. The observed late-phase deceleration likely reflects structural limitations rather than a shift in strategy.

In contrast, the CTF environment offers a more open topology. Here, phase-aware analysis can uncover how strategy evolves across multiple paths, revealing, for example, which nodes are prioritised once situational awareness improves. These insights can guide defenders in identifying high-risk services and timing of potential escalation.

More generally, in flexible or real-world networks, this type of analysis could help detect when an agent shifts from reconnaissance to exploitation, supporting adaptive defences or deception deployment. For developers, reward trajectories across phases provide feedback on convergence quality and policy generalisation. This setup thus enables interpretable comparisons across environments with varying structural complexity.

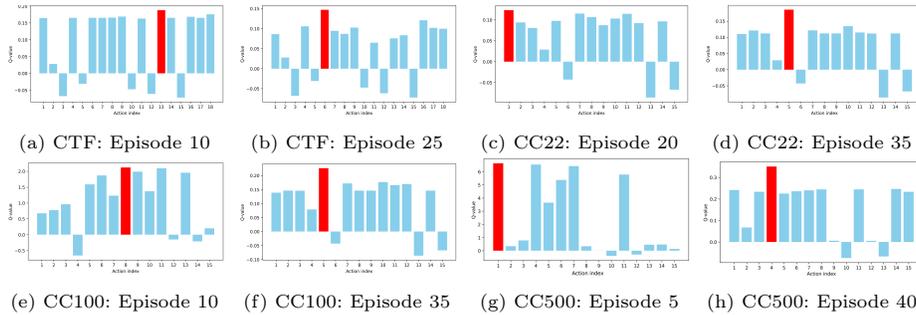


Fig. 5: Emergence of Action Preferences via State-Aggregated Q-Values Across Episodes.

6.4 Setup 4: Temporal Q-Value Analysis

In this setup, we track how action preferences evolve over training, revealing shifts in tactical focus. We compute state-aggregated Q-values per action across episodes to visualise which actions the agent increasingly favours as it learns.

Results. We trained the attacker for 40 episodes (400 iterations), logging Q-values for all visited state-action pairs. Figure 5¹ shows Q-value distributions per action at different training stages across environments, with Table ?? (refer appendix) mapping action indices to semantic descriptions. In early episodes, Q-values are relatively low and undifferentiated, indicating uncertainty and limited experience. As training progresses, a preference hierarchy emerged, with one single action (highlighted in red) attaining higher mean Q-values across the states, indicating stronger expected long-term value.

The sharpness and timing of this differentiation varies by environment. For **CC22**, dominant early actions include `local(CrackKeepPassX)`, `connect(RDP)`, and `connect(GIT)` (actions 1, 7, and 11) by episode 20 (Figure 5c), reflecting early experimentation with lateral movement and credential reuse. By episode 35 (Figure 5d), it converges on `local(CrackKeepPass)` (action 5), suggesting a shift toward more reliable or higher-reward local exploitation once basic network structure is learned. For **CC100**, a similar pattern is observed, the agent begins with broad action exploration, including `connect(SSH-key)`, `connect(MySQL)`, `connect(GIT)`, and `local(ScanExplorerRecentFiles)` (actions 8, 9, 11, 13) at episode 10 (Figure 5e), before narrowing its focus to `local(CrackKeepPass)` (action 5) by episode 35 (Fig. 5f). This likely reflects convergence on a consistently valuable tactic once credential chains are established. For **CC500**, in contrast to the smaller environments, the agent ultimately prefers `remote(ProbeWindows)` (action 4) by episode 40 (Fig. 5h), after initial attempts using actions like `local(CrackKeepPassX)` and `connect(RDP)`. This suggests that in more com-

¹

For the CTF environment, actions 1–18 are: `local(ScanBashHistory)`, `local(ScanExplorerRecentFiles)`, `local(SudoAttempt)`, `local(ExfiltrateFlag)`, `local(CrackKeepPassX)`, `remote(ProbeLinux)`, `remote(ProbeWindows)`, `remote(ProbeSQLServer)`, `connect(HTTPS)`, `connect(GIT)`, `connect(SSH)`, `connect(RDP)`, `connect(MySQL)`, `connect(SSH-key)`, `connect(PING)`, `connect(su)`, `remote(ProbeFlagServer)`, and `local(CrackKeepPass)`. For the Chain environment, actions follow this sequence: `local(CrackKeepPassX)`, `remote(ProbeLinux)`, `local(ScanBashHistory)`, `remote(ProbeWindows)`, `local(CrackKeepPass)`, `connect(SSH)`, `connect(RDP)`, `connect(SSH-key)`, `connect(MySQL)`, `connect(HTTPS)`, `connect(GIT)`, `connect(PING)`, `local(ScanExplorerRecentFiles)`, `local(SudoAttempt)`, `connect(su)`.

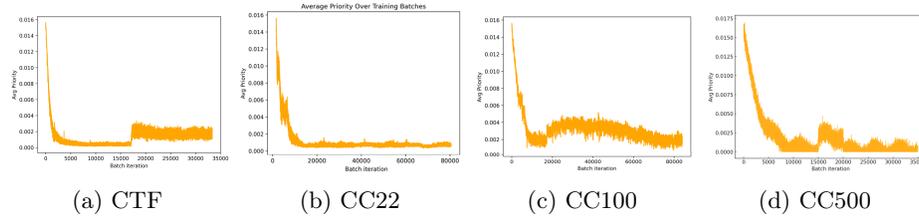


Fig. 6: Learning Impact of High-Priority Transitions (Average TD Error) Across Environments.

plex, large-scale networks, reconnaissance actions offer higher long-term strategic value, providing the visibility needed to coordinate deeper exploitation.

These results reflect the agent’s shift from broad exploration to targeted exploitation of high-reward paths. In smaller environments (CC22, CC100), the agent converges on local exploits like `local(CrackKeepPass)` (action 5), while in larger networks like CC500, it favours reconnaissance actions such as `remote(ProbeWindows)` (action 4), highlighting a preference for scalable reconnaissance over direct exploitation. By tracking how action valuations evolve over time, our framework helps developers debug brittle convergence, uncover underused actions, and evaluate policy stability. For cybersecurity teams involved in red/blue-team exercises or AI-driven threat simulation, these insights offer a window into how autonomous agents learn and which tactics they prioritise, informing training scenarios, service hardening, and timing of decoy deployment during simulations.

6.5 Setup 5: Strategic Decision Attribution via PER

This experiment demonstrates how Prioritised Experience Replay surfaces the most impactful (high-TD-error) transitions, acting as an attention mechanism that steers the agent toward strategically important experiences.

Results. PER was configured with $\alpha = 0.6$, $\beta = 0.7$, and $\epsilon_0 = 0.01$ (to avoid zero probabilities). We filtered transitions with rewards above 2.0 to isolate high-impact experiences such as privilege escalation. Figure 6 shows average transition priority over time. All environments exhibit early TD-error spikes from policy instability and exploration. Smaller networks (CTF, CC22) stabilise quickly, while larger ones (CC100, CC500) show sustained fluctuation, indicating extended learning. Notably, CC500 displays distinct spikes, suggesting newly discovered pivot points or strategic escalations. Figure 7 tracks the number of distinct high-priority states. CTF and CC22 converge to compact state sets, implying repeated exploitation of a compact, optimal state set. Larger networks (CC100, CC500) maintain broader distributions, indicating ongoing refinement and diverse learning trajectories.

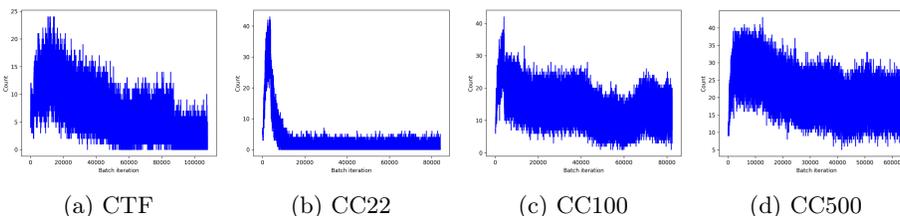


Fig. 7: Number of Key High-Priority States Across Environments.

PER reveals *where* and *when* attacker agents learn most rapidly during training. These TD-error spikes, especially in large, dynamic environments, mark key policy shifts that developers can investigate and defenders can simulate against. For developers, CC100, the smooth arc between 40k-60k batch iterations reflect gradual learning concentration, useful for assessing when and how the agent starts committing to high-impact transitions. For defenders, while PER is not observable in deployed agents, it supports red-team planning by revealing when attackers are most likely to pivot. In CC500, twin spikes around 15k and 25k iterations may reflect credential pivoting or lateral movement, ideal timing cues for testing decoy deployment or access hardening in simulation. Future agentic threats using online learning could exhibit similar dynamics.

7 Discussion

Takeaways from Analysis. Our multi-layer XRL framework provides structured visibility into attacker behaviour at both strategic and tactical levels. At the MDP level, it surfaces transitions from reconnaissance to exploitation, contextualised by uncertainty and environment progression. At the policy level, it highlights how action preferences emerge and evolve, and which transitions drive the agent’s most significant updates. Together, these layers go beyond performance metrics to explain *how* and *when* attacker strategies are formed, enabling proactive interpretation rather than retrospective outcome assessment.

Practical Use for Cybersecurity Experts. Although designed for RL explainability, our framework offers operational benefits for security teams planning for future agentic threats. As autonomous attackers become more sophisticated, defenders must understand not only *what* an agent does but also *how* it learns and adapts. Our framework enables defenders to anticipate attacker intent by highlighting when and where adversarial strategies shift, especially during red-team development or simulation-based training.

Example: In a simulated environment like CC500, defenders might observe that a DQL agent consistently spikes in transition priority around 15k and 25k iterations, signalling key learning updates such as credential pivoting. This insight helps teams identify likely escalation paths and optimal timing for deploying decoys or adjusting access controls. While such data are not available in live

attacks, it supports red-team preparation and AI adversary modelling by revealing which phases of attack progression are most adaptive.

Practical Use for ML Model Developers. For developers of RL-based cyber agents, this framework serves as a diagnostic tool for understanding training behaviour. It enables:

- Visualisation of Q-value trajectories to trace emerging action preferences,
- Identification of key transitions via TD-error spikes and PER prioritisation,
- Detection of brittle convergence or excessive determinism in policies,
- Evaluating exploration strategies by behavioural richness over performance.

These insights are especially valuable in partially observable settings like CyberBattleSim, where agent performance may hinge on sparse rewards and incomplete visibility. Our framework supports more interpretable, resilient policy development by shedding light on not just *what* the agent learns, but *how* and *when*.

Limitations and Future Work. While CyberBattleSim offers a controlled and extensible environment, it simplifies many real-world complexities, such as concurrent adversaries, deception-aware agents, and sophisticated evasion tactics. Moreover, CyberBattleSim’s network architecture, while valuable for controlled experimentation, is relatively simple and lacks the heterogeneity and complexity of real-world enterprise networks, such as layered access controls, segmented subnets, and diverse user behaviours. Additionally, our analysis currently focuses on a single-agent attacker. Future work will extend this framework to multi-agent scenarios, enabling analysis of competitive or cooperative dynamics between attackers and defenders. We also plan to conduct user studies with security analysts and red-teamers to refine how explanations are delivered, ensuring insights are both technically sound and practically usable.

8 Conclusion

In this paper, we introduced a unified, multi-layer explainability framework for demystifying the decision-making of RL-based attacker agents in cyber environments. By combining MDP- and policy-level analysis, the framework offers fine-grained, temporally grounded insights into how adversaries explore networks, adapt, and exploit network vulnerabilities. Through experiments in CyberBattleSim, we demonstrated how the framework reveals strategy shifts, learning patterns, and decision inflection points often hidden in standard RL evaluations. Unlike prior post-hoc or domain-specific work, our agent-agnostic approach generalises to defender agents, enabling unified interpretation to both offensive and defensive strategies. It equips red teams with visibility into how RL agents develop adversarial tactics and offers blue teams a powerful tool to interpret and counter evolving threats. This work lays the foundation for transparent, trustworthy, and operationally useful RL in cybersecurity. Future directions include real-time integration, multi-agent extensions, and application in high-stakes AI safety contexts.

References

1. Cyber operations research gym. <https://github.com/cage-challenge/CybORG> (2022), created by Maxwell Standen, David Bowman, Son Hoang, Toby Richer, Martin Lucas, Richard Van Tassel, Phillip Vu, Mitchell Kiely, KC C., Natalie Konschnik, Joshua Collyer
2. Alabdulkarim, A., Singh, M., Mansi, G., Hall, K., Riedl, M.O.: Experiential explanations for reinforcement learning. arXiv preprint arXiv:2210.04723 (2022)
3. Andrew, A., Spillard, S., Collyer, J., Dhir, N.: Developing optimal causal cyber-defence agents via cyber security simulation. arXiv preprint arXiv:2207.12355 (2022)
4. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton, NJ (1957)
5. Bierbrauer, D.A., Schabinger, R.M., Carlin, C., Mullin, J., Pavlik, J.A., Bastian, N.D.: Autonomous cyber warfare agents: dynamic reinforcement learning for defensive cyber operations. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V. vol. 12538, pp. 42–56. SPIE (2023)
6. Foley, M., Wang, M., Hicks, C., Mavroudis, V., et al.: Inroads into autonomous network defence using explained reinforcement learning. arXiv preprint arXiv:2306.09318 (2023)
7. Goel, D., Moore, K., Guo, M., Wang, D., Kim, M., Camtepe, S.: Optimizing cyber defense in dynamic active directories through reinforcement learning. In: European Symposium on Research in Computer Security. pp. 332–352. Springer (2024)
8. Gyevnar, B., Wang, C., Lucas, C.G., Cohen, S.B., Albrecht, S.V.: Causal explanations for sequential decision-making in multi-agent systems. arXiv preprint arXiv:2302.10809 (2023)
9. Jesus, S., Belém, C., Balayan, V., Bento, J., Saleiro, P., Bizarro, P., Gama, J.: How can i choose an explainer? an application-grounded evaluation of post-hoc explanations. In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. pp. 805–815 (2021)
10. Mathes, T.K., Inman, J., Colón, A., Khan, S.: Codex: A cluster-based method for explainable reinforcement learning. arXiv preprint arXiv:2312.04216 (2023)
11. Molina-Markham, A., Winder, R.K., Ridley, A.: Network defense is not a game. arXiv preprint arXiv:2104.10262 (2021)
12. Retzlaff, C.O., Angerschmid, A., Saranti, A., Schneeberger, D., Roettger, R., Mueller, H., Holzinger, A.: Post-hoc vs ante-hoc explanations: xai design guidelines for data scientists. *Cognitive Systems Research* **86**, 101243 (2024)
13. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: Proceedings of the 4th International Conference on Learning Representations (ICLR) (2016), <https://arxiv.org/abs/1511.05952>, arXiv:1511.05952
14. Schwartz, J., Kurniawatti, H.: Nasim: Network attack simulator. <https://networkattacksimulator.readthedocs.io/> (2019)
15. Sharma, D.K., Mishra, J., Singh, A., Govil, R., Srivastava, G., Lin, J.C.W.: Explainable artificial intelligence for cybersecurity. *Computers and Electrical Engineering* **103**, 108356 (2022)
16. Sultana, M., Taylor, A., Li, L.: Autonomous network cyber offence strategy through deep reinforcement learning. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III. vol. 11746, pp. 490–502. SPIE (2021)
17. Sun, N., Ding, M., Jiang, J., Xu, W., Mo, X., Tai, Y., Zhang, J.: Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives. *IEEE Communications Surveys & Tutorials* **25**(3), 1748–1774 (2023)

18. Team, M.D.R.: Cyberbattlesim. <https://github.com/microsoft/cyberbattlesim> (2021), 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
19. Thompson, I.S., 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 (2024)
20. Tokic, M.: Adaptive ϵ -greedy exploration in reinforcement learning. In: Proceedings of the 22nd International Conference on Tools with Artificial Intelligence (ICTAI). pp. 243–250. IEEE (2010). <https://doi.org/10.1109/ICTAI.2010.41>
21. Watkins, C.J.C.H., Dayan, P.: Q-learning. *Machine Learning* **8**(3-4), 279–292 (1992). <https://doi.org/10.1007/BF00992698>
22. Wei, C.Y., Luo, H.: Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach. In: Conference on learning theory. pp. 4300–4354. PMLR (2021)
23. Yu, J., Guo, W., Qin, Q., Wang, G., Wang, T., Xing, X.: {AIRS}: Explanation for deep reinforcement learning based security applications. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 7375–7392 (2023)