Building Better Environments for Autonomous Cyber Defence

Chris Hicks[1], Elizabeth Bates[1], Shae McFadden[1,2,3], Isaac Symes Thompson[1], Myles Foley[1], Ed Chapman[1], Nickolas Espinosa Dice[4], Ankita Samaddar[5], Joshua Sylvester[1], Himanshu Neema[5], Nicholas Butts[6], Nate Foster[7], Ahmad Ridley[8], Zoe M[1], Paul Jones[1]

[1]The Alan Turing Institute, [2]University College London, [3]King's College London, [4]Cornell University, [5]Vanderbilt University, [6]Microsoft, [7]EPFL, [8]NSA

ABSTRACT

In November 2025, the authors ran a workshop on the topic of *what makes a good reinforcement learning (RL) environment for autonomous cyber defence (ACD)*. This paper details the knowledge shared by participants both during the workshop and shortly afterwards by contributing herein. The workshop participants come from academia, industry, and government, and have extensive hands-on experience designing and working with RL and cyber environments. While there is now a sizeable body of literature describing work in RL for ACD, there is nevertheless a great deal of tradecraft, domain knowledge, and common hazards which are not detailed comprehensively in a single resource. With a specific focus on building better environments to train and evaluate autonomous RL agents in network defence scenarios, including government and critical infrastructure networks, the contributions of this work are twofold: (1) a framework for decomposing the interface between RL cyber environments and real systems, and (2) guidelines on current best practice for RL-based ACD environment development and agent evaluation, based on the key findings from our workshop.

KEYWORDS

Autonomous Cyber Defence, Reinforcement Learning, Resilience

1 INTRODUCTION

Cyber attacks pose serious risks to individuals, businesses, and governments, whose daily operations all depend on networked systems. In recent history these risks have manifested as costly attacks on critical systems including nuclear facilities [41] and power grids [22]. The risk posed by threat actors is managed through a combination of technical controls (e.g., antivirus software, secure boot, air gaps), policy-based measures (e.g., a white list of allowed IP addresses, password policy), and human-focussed measures (e.g., phishing awareness, incident response drills). Despite these preventative measures the asymmetric incentives of adversaries [4], and emerging capabilities enabled through AI misuse [1], make cyber attacks both a significant and growing problem.

To counter such threats, agent-based Autonomous Cyber Defence (ACD) systems that can immediately monitor, adapt, and respond are key. In particular, Reinforcement Learning (RL), a subset of Machine Learning (ML) that learns through interaction with an environment has received significant interest in the literature [7, 18, 20, 23, 28, 29, 42, 69]. The unique advantage of RL, compared to agents based on pre-trained generative models, is the ability to learn how to achieve a specific goal exclusively from interacting with an environment. RL agents, therefore, do not rely on prior human knowledge or understanding. This is especially important for ACD as adversaries frequently exploit the assumptions made when systems were designed and configured.

For RL to succeed at ACD, there are at least three core challenges: (1) the environment must accurately and efficiently represent the real-world network defence problem at an appropriate level of fidelity (e.g., via a cyber environment) including, but not limited to, the network topology, attacker behaviour, and the functionality of software components; (2) proper consideration must be given to the interface between the learning agent and the environment, including the identification of a suitable numerical reward mechanism; and (3) a robust evaluation methodology, including appropriate measures of success and statistical significance, must be implemented. Towards addressing the core challenges in ACD, this work makes the following main contributions:

- We *propose a framework* for decomposing the components and modelling choices involved in mapping between ACD environments and real systems.
- We *provide guidelines* on best-practice in relation to each framework component, based on the findings from our workshop, providing a valuable reference for anyone engaged in cyber environment development.

The remainder of this paper is organised as follows: Section 2 provides a background on formulating ACD for RL; Section 3 details our framework and how each environment component contributes to the transfer of agent performance from training to deployment; Section 4 distils the insights gathered from our workshop into best-practice guidelines; Finally, Section 5 reviews related work and Section 6 concludes the paper.

Workshop Methodology

This paper is informed by a structured online workshop bringing together 25 domain experts from across academia, industry and government with experience in RL-based cyber defence. The workshop included focused discussions and interactive activities to extract limitations, design considerations, and best-practice recommendations for ACD. These were explored from three distinct perspectives: security, RL, and environment development. Insights were gathered via collaborative tools and facilitator notes, then processed through thematic analysis into the framework and best practice recommendations presented in this work.

2nd Workshop on AI for Critical National Infrastructure (AI4CNI-26) @ AAMAS 2026, , May 25 – 29, 2026, Paphos, Cyprus.

2 RL FOR CYBER DEFENCE

Both RL and cyber defence are large fields of study comprising many problem types and solution methods. In this work we focus on formulating cyber defence as an RL problem. This section firstly clarifies the definition of an RL problem and then details the scope of network defence commonly encountered in ACD.

2.1 The RL Learning Problem

RL is a field of study, a class of problems, and a class of solution methods for solving such problems [66]. It is the authors' view that RL solution methods (e.g., model-free, on-policy) and algorithms that apply them (e.g., PPO, DQN) are not the main bottlenecks in RL for ACD. Instead, the core challenge of ACD remains how best to formulate a particular network defence problem within the RL framework.

Informally, the RL learning problem is one where an agent learns how to act in an environment over time to achieve a goal. At each step, the agent selects an action, and then receives an observation and a reward. Through sequential interaction with the environment, the agent learns a policy that aims to maximise the cumulative expected reward. This interaction is modelled as a Markov Decision Process (MDP) with the assumption (a.k.a the Markov property) that the future is independent of the past given the present. In other words, the distribution of the next observation and reward must depend only on the current observation and action (and not those seen or taken previously). More formally, Sutton and Barto [66] define an RL task; one instance of the RL learning problem, as a complete specification of an environment. An RL task is typically formalised as an MDP $\mathcal{M}$ defined by the tuple:

$$\mathcal{M} = (S, A, P, R, \gamma)$$

where S is the set of states, A is the set of actions, $P(s' \mid s, a)$ is the state-transition probability function, $R \subset \mathbb{R}$ is the reward space, and $\gamma \in [0, 1]$ is the discount factor. Given the current state $s \in S$ and action $a \in A$, the transition function $P(s' \mid s, a)$ gives the probability of the next state s':

$$p(s' \mid s, a) = \Pr\left(S_{t+1} = s' \mid S_t = s,\ A_t = a\right) = \sum_{r \in R} p(s', r \mid s, a)$$

Given an MDP $\mathcal{M}$, the objective is to learn a policy $\pi(a \mid s)$ that maximises the discounted sum of future rewards G_t after t time steps:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

i.e., the optimal policy satisfies $\pi^\star = \arg\max_\pi \ \mathbb{E}_\pi\left[G_t\right]$. The horizon for future rewards is most often bounded by an *episode*, which starts in an initial state and ends upon arriving at a terminal state (or condition). The horizons induced by episodes can be *fixed*, if termination occurs after a fixed number of steps or *variable*, if termination requires certain environmental conditions to occur (e.g., win or loss conditions). Finally, when the state space of the environment is not able to be fully observed by the agent, the task is instead modelled as a partially observable MDP (POMDP) defined by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, P, R, \Omega, O, \gamma)$ where $\mathcal{S}, \mathcal{A}, P, R, \gamma$ are as in an MDP and additionally Ω is the set of observations and $O(o \mid s, a)$ is the observation model.

2.2 Cyber Defence as an RL Task

Some famous RL tasks (e.g., chess) fit the RL paradigm without significant alteration. However, cyber defence is a complex set of real world problems that does not naturally present a single RL task [53]. Beyond comprising multiple tasks and objectives that may change across time, cyber defence is a challenging application for RL because the state of the environment is both extremely large and almost never completely known; adversaries and users introduce non-stationary dynamics; representing cyber security goals as scalar rewards is difficult; operational constraints may be difficult to represent in the action space; and accurate simulators are necessary (e.g., for safety and efficiency) but challenging to build and interface with correctly. Nevertheless, RL offers a principled framework for sequential decision making under uncertainty, enabling agents to learn potentially novel strategies for defending networks in excess of current human-designed approaches.

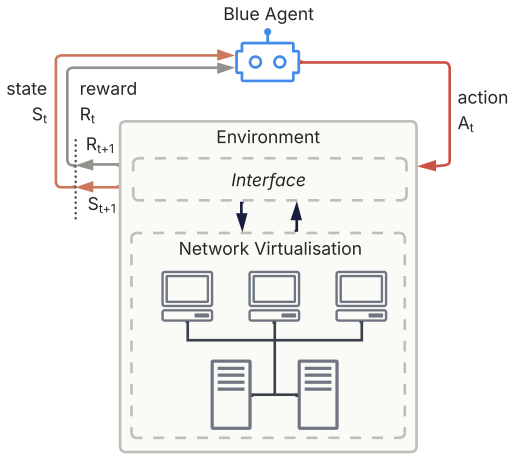

Figure 1: Network defence as an RL task.

Although cyber defence is not a single problem, cyber environments have thus far converged on a relatively constrained interpretation: a defensive blue agent is tasked with defending a network of hosts against one or more fixed-strategy adversaries.

The action space is usually an enumeration of high-level command and control (C2) techniques (e.g., scan, deny, restore) and the observation space (i.e., the real network state is not completely known) is a vector of selected attributes from the underlying network (e.g., one bit per host to indicate whether a compromise has been detected). The reward function is typically engineered by domain experts and includes penalties and/or incentives for various network states and actions (e.g., -2 when a host is compromised, -1 for restoring a host, +0.1 for making no intervention). In other words, defending the network becomes the task of minimising the number of compromised hosts whilst making conservative use of costly actions.

Thus, this network defence task of ACD is commonly framed as a finite horizon sequential decision-making problem in which a blue agent receives partial observations from an environment comprising a network of hosts, an adversarial red agent, and possibly benign green agents representing regular user activity. At each

time step, the blue agent chooses an action and receives both a new observation, and a reward, based on the state of the environment and which action was chosen.

Crucially in this framing, the environment comprises everything that is not the blue agent: the network of hosts, regular users, and the adversary - see Figure 1. Since the real network is likely of operational significance, and the RL learning framework includes learning from "mistakes" to choose better actions over time, a virtual cyber environment is the standard way to train a network defence agent. Whether blue agents are useful for the real network defence problem therefore depends significantly on how well the cyber environment models the real environment.

3 A FRAMEWORK FOR ACD ENVIRONMENTS

The divergence between environments and real-world systems, known as the *sim-to-real* gap, hinders the transfer of agent performance from training to deployment. Designing high fidelity network environments can help to close the gap but is not entirely sufficient. Equally, the formulation of the RL task matters as it fundamentally constrains both what and how (i.e., learning dynamics) the agent learns. Thus, as shown in Figure 2, the sim-to-real gap comprises two key components: virtualisation of the environment and modelling of the task. Insufficiency in either simulation or modelling limit the real world performance ceiling of RL agents by creating poorly aligned and artificially challenging RL tasks.

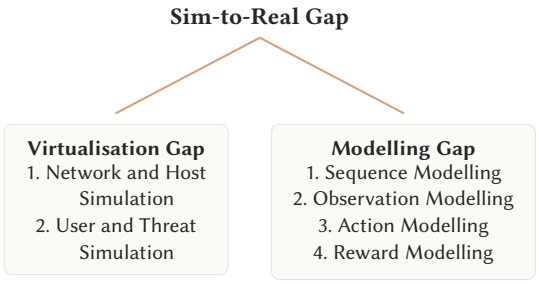

Figure 2: Breakdown of sim-to-real gap components.

3.1 The Virtualisation Gap

The virtualisation gap encompasses the difference between the simulated and real-world network defence problems, seen in Figure 3. If RL agents could be trained on a real production network, with real users and attackers, then there would be no virtualisation gap. The virtualisation gap can be split into two components: (1) the network and host simulation including its topology and operational characteristics, and (2) user and threat simulation.

Network and Host Simulation High-fidelity simulation of the network and hosts is both feasible, owing to widely available virtualisation software, and preferable for minimising the simulation gap. Low fidelity cyber environments offer high efficiency, making it cheaper to train RL agents, but introduce the significant risk of simulator artifacts. Agents learning from artifacts may perform very well in the simulator yet behave poorly, and unpredictably, in the real world.

User and Threat Simulation Both regular network users and adversarial threats are an essential part of modelling a network defence problem. Within a cyber environment these are represented by green and red agents, respectively. In proportion to the realism of their behaviour both green and red agents have a large impact on the simulation gap. Since red agents model adversarial behaviour, they determine the distribution and severity of threats encountered by the blue agent in the environment. Therefore, if the strategies they simulate are not sufficiently realistic (e.g., simplistic, predictable, or overt) then the defensive policies trained will be brittle with marginal success transferring to real-world deployments. Green agents are of equal importance, as without benign user simulation the blue agent may learn policies that succeed at defence but significantly disrupt normal traffic.

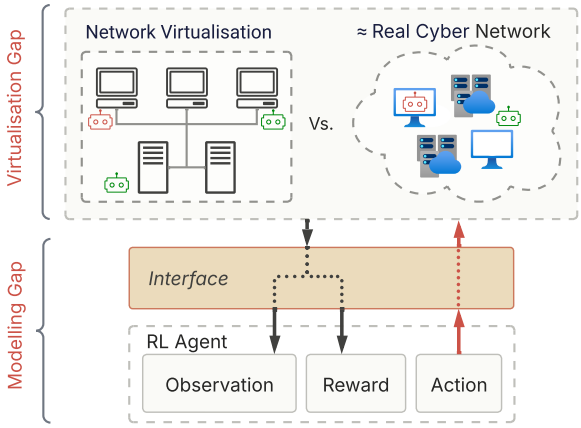

Figure 3: Virtualisation vs. modelling gap.

3.2 The Modelling Gap

The modelling gap comprises the interface between the network defence problem (e.g., simulation model) and the RL task (i.e., a POMDP), seen in Figure 3. The modelling gap includes observation modelling (e.g.,a vector derived asynchronously from network state information), action modelling (e.g., an enumeration of actions and how they are executed on network hosts), sequence modelling (i.e., how POMDP time steps relate to the occurrence of network events) and reward modelling (i.e., how to provide scalar rewards in alignment with real-world goals). Modelling choices significantly impact how effective learned agent policies are, irrespective of the simulation gap, but are also constrained by simulation fidelity.

Sequence Modelling. Hundreds of processes run concurrently on modern operating systems and network packets are sent, reordered, dropped and received between devices without coordinated clocks. Sequence modelling captures the temporal relationship between continuous, asynchronous network activity and the discrete-time sequence of agent actions, observations and rewards required for RL, seen in Figure 4. Sequence modelling is crucial because the action, observation and reward modelling components must preserve causal ordering, allowing meaningful state transitions and reward attribution, which all depend on understanding how the network,

hosts and other agents may advance between each time step. The MDP discount factor γ, which determines an effective prior over the relevance of future rewards, and the RL task episode length when fixed, both typically assume fixed per-step wall time and therefore depend greatly on sequence modelling.

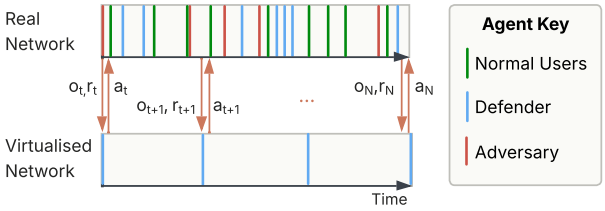

Figure 4: Sequence Modelling: the relationship between continuous, asynchronous real-world network activity (top) and the discrete-time RL sequence trajectory (bottom).

Observation Modelling. The true underlying state of the network is far too large to input to any practical algorithm (e.g., a single workstation hard disk may have 1TB of state information) and is distributed over many devices. Thus, the network state must be aggregated and parsed to provide vector observations for the RL agent. The crucial modelling constraint of the underlying state is the Markov property: the distribution of the next state and reward can only depend on the current state and action. Thus, the observation should approximate the Markov property as closely as possible, using any data reasonably available to a defender, whilst respecting realistic monitoring limitations. At the same time, irrelevant features might increase the attack surface and, when featuring high-variance, increase the learning difficulty.

Action Modelling. In network defence, operations are usually carried out using highly expressive graphical and command-line user interfaces. Action modelling involves constructing a set (discrete, continuous, or both) of actions that map to a useful set of choices relevant to the network defence task.

Reward Modelling. The goal of network defence is to minimise adversarial disruption while maintaining normal operations for non-malicious users. Reward modelling involves identifying scalar incentives and penalties which correspond to achieving one or more network defence goals when their combination is maximised over the discounted horizon.

4 TOWARDS BETTER CYBER DEFENCE ENVIRONMENTS

This section distils best-practice insights from domain experts into actionable guidelines, designed to maximise RL task effectiveness and real-world transfer, structured around the framework introduced in Section 3. As the main focus of academic work to date [71], this best-practice focusses on single-agent ACD environments for network defence.

4.1 Characterising the Real-World Problem

Before any simulation or RL modelling decisions are made, it is essential to clearly define the target network defence problem. The problem scope will determine which real-world dynamics must be captured, which abstractions may be acceptable, and how problem complexity can be introduced over time.

Specify the Problem, Constraints & Success Criteria. The first step is to fully specify the network defence problem to be investigated. The specification should characterise the network environment (e.g., topology, scale, host details and roles), operational and business requirements (e.g., user data must have high availability), assets to be protected (e.g., critical hosts, sensitive data), adversarial assumptions (e.g., goals, tactics, techniques, and procedures), and the capabilities and constraints of defensive actors.

Identify Areas of Uncertainty. A key consideration is characterising uncertainty about the real-world network defence problem. Uncertainty may arise from: the network environment (e.g., variations in network topology and host behaviour), the distribution of expected user and attacker profiles, and defender errors (e.g., malware misclassification). Characterising uncertainty helps to identify generalisation requirements and may improve alignment between the training, evaluation and deployment performance by allowing agents to be robust to unseen distributions of trajectories.

Detail a Minimum Viable Problem. The combination of a complex full-scale network defence problem, and well-known RL learning reliability challenges [2], is ill-suited to efficient debugging and early experimental iteration. Thus, it is important to construct a well-scoped minimum viable problem that can eventually scale to the required complexity. This abstraction should preserve the core decision-making structure and causal factors of the task while simplifying confounding factors including network scale, state dimensionality, action branching, and non-stationary user and threat behaviour. Once the core problem structure is understood, realism can be systematically reintroduced to recover the full-problem.

4.2 Virtualisation

Having characterised the minimum viable problem scope and success criteria, the goal is to build a virtual environment that captures the necessary problem structure, users, and adversaries with scope to eventually scale to the full problem. The virtualisation gap constrains the behaviours, uncertainties, and causal relationships that an RL agent can learn from; setting an upper bound on the achievable real-world performance, irrespective of how sophisticated the subsequent RL modelling may be.

Scope Existing Environments. While a pragmatic first step is to review existing cyber environments [71], any suitably well-characterised (i.e., specific) network defence problem is likely to require significant customisation. If a new environment is judged necessary, then developers must decide between a simulation, emulation, or hybrid approach, balancing efficiency against the risk of simulator artefacts. This decision should be informed by the minimum viable problem, minimising any unnecessary complexity that does not contribute to the learning signal.

Validate Connection to High-Fidelity. While this may involve low-fidelity simulation initially, it is possible that the correct problem structure and causal factors are not fully understood. Thus, it

is important to validate that the resulting observations, action feasibility, state transitions, and rewards correspond to those produced by realistic interactions between emulated or real-world software components, hosts, and network infrastructure. Implementing a bespoke mid-fidelity simulation should be approached with caution as there is a risk of both including unnecessary complexity and failing to match the real-world problem dynamics.

Inform Design Choices through Problem Scale. The problem scale should inform the choice of virtualisation technologies, and non-negotiable configurability (e.g., topology, host details, user and threat profiles), ensuring the feasibility of systematic scaling once the minimum viable problem has been satisfied. The network virtualisation, capabilities and constraints of defensive actors, alongside the measurable criteria for success or failure should inform what monitoring data is available to the defender. Fidelity, noise, and delay characteristics of monitoring information should reflect realistic operational constraints.

4.2.1 Network and Host Virtualisation

At a minimum, the virtualised network should reflect the topology, traffic flows, service dependencies, host roles, and configuration necessary for both defensive and adversarial actions to produce plausible effects. Ideally, high-fidelity virtual machines, containerised instances, and virtual networks running real-world software stacks (e.g., applications, databases and network services) representative of the full-scale problem should be used to provide network and host virtualisation. The network and host state information available to defenders should plausibly represent data produced by the suite of monitoring tools found in the real-world problem.

Ensure Problem Structure is Preserved. When a highly abstract network representation is justified by a simple problem structure, the resulting environment dynamics should be validated against a high-fidelity network environment, or using data from the real-world problem, to ensure that problem structure has been preserved as expected. Towards scaling from minimum viable problem to full-scale realism, network topologies, host profiles, services, and monitoring capabilities should be easily configurable.

Examples of Best-Practice. Current best-practice examples include Cyberwheel [55], which ensures network and host state information is designed to mimic real-world monitoring and detection pipelines, and NASimEmu [34] which validates abstract red team simulation dynamics against high-fidelity emulation using a shared simulator-emulator interface.

4.2.2 User and Threat Virtualisation

User and threat virtualisation determine the distribution of benign and adversarial network activity encountered during training and evaluation. Both must be modelled with sufficient realism to ensure agents remain effective in the real-world, avoiding policies that depend on misrepresentative user activity or attacker behaviour.

Move Beyond Fixed Attacker Strategies. Fixed-strategy, deterministic red agents can be useful when implementing a minimum viable problem, as they remove a source of non-stationarity and thereby ease debugging and early experimental iteration. However, RL policies are known to be brittle under modest shifts in environment dynamics [56, 61]. As a result, environments that rely solely on stationary adversaries are likely to induce policies which fail to generalise or transfer satisfactorily to real-world settings. Environments should therefore support diverse and configurable red-agent behaviour, including stochasticity, strategy switching, and adaptive decision-making. Where feasible, environments should support competitive multi-agent training; enabling red and blue agents to co-evolve, and exposing defender policies to diverse offensive strategies. In principle, co-evolution may even lead to state-of-the-art emergent strategies providing pre-emptive mitigation techniques for real-world networks.

Aim for Representative User Activity. Legitimate user activity should be simulated with sufficient realism to reflect the diversity and variability of real-world network traffic. Green agent behaviour should generate representative background traffic, service interactions, and benign anomalies that meaningfully interfere with detection and response. Where possible, real-world logs or statistically grounded models of network activity should be used to inform and validate user behaviour, rather than relying on scripted profiles.

Consider Generative Green and Red Agents. Virtualisation environments should support multiple approaches to modelling agents. Alongside fixed or learning-based agents, emerging LLM-based, agentic adversaries may allow generating diverse, adaptive, goal-directed agent behaviour without requiring explicit modelling.

4.3 RL Modelling

Given a simulated network defence problem, the next step is to develop a suitable RL task formed from observations, actions, temporal structure, and rewards. These modelling choices determine what information the RL agent can learn from, how it can interact, how causality unfolds over time, and how success is incentivised.

4.3.1 Sequence Modelling

When building an RL task for a network defence problem, sequence modelling ensures that asynchronous real-world network behaviour is properly mapped onto the discrete-time sequence of actions, observations, and rewards that structure RL learning.

Addressing sequence modelling remains an open challenge for RL in cyber defence, particularly given long-horizon dependencies, asynchronous dynamics, and persistent adversaries. However, explicitly reasoning about temporal structure, action duration, and concurrency is essential for closing the modelling gap and producing agents whose behaviour transfers to the real-world.

Justify Episodic Structure and Horizon Length. Many ACD environments divide agent-environment interactions into discrete episodes, either with a fixed number of time steps (i.e., fixed-horizon) or explicit terminal states (i.e., episodic). An episodic task formulation simplifies training and evaluation by ensuring that returns are finite and well-scaled, and also by providing a reset mechanism which can increase exposure to important states [66]. However, episodic formulations impose strong assumptions about temporal structure that often do not hold in operational networks. Real-world networks are typically persistent and non-stationary, challenging the possibility of a "reset" and discrete episodes. Another difficulty

is that episodic formulations impose a bounded decision horizon, eliminating the possibility of learning causal structure beyond the episode bounds and biasing RL agents towards short-term wins.

Best practice therefore is to explicitly justify the temporal task modelling, based on the target problem, rather than adopting an episodic, fixed-horizon by default. If the problem is inherently continuous or long horizon, exemplified by Advanced Persistent Threat (APT) actors, who may gain access to a network months or years before executing their attack, then open-ended or infinite-horizon formulations should be considered. For example, Hammar and Stadler [27] formulate network defence as an optimal stopping problem, where the defensive agent has two possible actions: "stop", corresponding to a defensive intervention, and "continue".

Account for Temporal Variability and Action Duration. In real networks, defensive actions are neither instantaneous nor uniform in duration. Scanning, isolating hosts, restoring services, and reconfiguring access controls may take seconds, minutes, or longer, and can overlap with other defensive or adversarial activities. Similarly, attacks unfold asynchronously, driven by independent processes and external timing constraints. In contrast, the standard RL assumption is that each time step corresponds to a fixed quantity of wall-time (i.e., each time step is weighted equally when calculating reward attributions). Thus, to support real-world transfer, it is best practice to represent temporal variability and action duration explicitly within the environment. Where possible, environments should support variable action durations, varying delays between action initiation and effect observation, overlapping and concurrent execution, and events occurring independently of agent actions. R3ACE [11] implements continuous-time, event-driven RL modelling which is aligned with best practice, although only demonstrated on a very small problem space.

Reconsider Turn-Based Modelling. Many cyber environments model red, green, and blue agent interactions in a turn-based manner, implicitly enforcing an ordering of each agent's action. This abstraction is not representative of real networks, where attackers, users, and defenders act concurrently, opportunistically, and without coordination. At baseline, it is important to be aware that turn-based agent interaction causes non-trivial impacts on RL learning outcomes in ACD [8, 9]. Ideally, turn-based modelling should be avoided except where the problem structure imparts an ordering distribution that can be well-determined. If turn-based structure is retained for tractability, then the sensitivity of trained policies should be evaluated.

4.3.2 Observation Modelling

Observations in ACD environments are often fixed-size, discrete vectors representing evidence about network status. However, workshop participants repeatedly highlighted that unrealistic "magic" observations and overly lossy encodings (e.g., minimal bit-vectors) undermine both realism and transfer, while also obscuring what a defender is assumed to know at decision time.

Define a Realistic Observation Pipeline. The information contained in observations should reflect what is genuinely available in practice from realistic sources such as monitoring tools, logs, and alerts. When observations are drawn directly from the raw simulator or emulator state, they risk incorporating features that could not be reliably produced in real deployments (e.g., listing if a host is compromised in the state). This creates a gap between the environment observed by the agent and real-world conditions, as these "magic" observations embed convenient but impossible information that undermine practical transferability. Equally, observations modelled directly on virtualised network state also risk containing less information than a cyber operator would typically have access to, under-representing true monitoring capability. Since magic observations cannot be reproduced outside of the virtualised environment, observation modelling must be grounded in practically available information. For example, the Cyberwheel [55] environment demonstrates this by constructing observations directly from existing cyber detection tools.

Consider Observation Representation. For ACD with a single blue agent, the most common observation design is to aggregate information from all network nodes into a single observation vector, mimicking the design of most traditional RL environments. However, this approach can lead to limitations in expressive power and generalisability when using standard neural network architectures [50]. This is especially problematic in environments with varying network topologies, which can correspond to changing the dimensions of the observation vector. An alternative is to factorise observations into individual network node features, and apply GNN [16, 34, 40, 54] or attention-based [49, 67] architectures, leveraging their permutation equivariance to produce policies that are more robust across varying network configurations.

4.3.3 Action Modelling

To enable learning effective strategies, agents need useful actions that correspond to real-world capabilities. Action modelling requires carefully mapping relevant capabilities into a set of actions that can be chosen from by the RL agent.

Provide a Sufficient and Representative Action Space. The action space fundamentally defines an agent's capabilities and must therefore provide sufficient scope to enable learning diverse and effective strategies. If the action space is very narrow or abstract the agent will likely be constrained to a small set of behaviours, limiting its capabilities below what may be effective in deployment. A representative action space should map to realistic operations that cyber defenders can perform on the network, preserving the temporal and causal relationships between system state, actions, and their outcomes in the environment.

Consider the Granularity-Capability Trade-off. Fine-grained actions can lead to large action spaces, increasing the difficulty of learning an effective policy. Whereas coarse abstractions (e.g. Mitre D3FEND [35] or Open C2 [64]) risk removing decision-making capability and breaking causal realism. The granularity of actions should reflect the intended operator role and specified network defence problem. Where possible, the granularity should be validated against real-world operational practice and be mapped to actual behaviour in high-fidelity emulation. Iteratively increasing the action granularity is consistent with the best practice of starting with an MVP and systematically scaling up to the full network defence

problem. As realism scales up, and the action space becomes more fine grained, additional capability and flexibility granted to the blue agent may come at the expense of learning efficiency. This trade-off could be managed through iterations of task modelling. To deal with large combinatorial action spaces, first consider approaches from the RL action space decomposition literature [51].

Mask Invalid Actions. Not every action is necessarily meaningful or feasible in a given state, as determined by the specific network conditions and privileges given to the agent. Allowing agents to select invalid actions can introduce undefined behaviour, and wastes training time on zero-value choices, rather than learning effective strategies. Masking invalid actions ensures the action space appropriately reflects operational reality and supports more efficient learning. Molina-Markham *et al.* [53] suggest using the Planning Domain Definition Language (PDDL) for modelling network defence tasks, including using action preconditions to determine masks.

Ensure Effects are Reflected in Observations. Valid actions that do not have observable consequences do not provide a useful learning signal and introduce noise into training that can impede policy learning. Furthermore, the success of a chosen action is not guaranteed in realistic deployments. Therefore, if available in practice, actions should have a discernible effect that can be observed by the agent, either directly or indirectly over subsequent states. This dependency between action and state modelling should be validated to ensure environments facilitate effective agent training.

Make Configuration Easy. The interface should facilitate the addition, removal, or changing of action constraints to allow for the modelling of specific network defence subtasks. This flexibility enables experimentation with capability assumptions, promoting the reuse of principled environments and improving reproducibility.

4.3.4 Reward Modelling

The reward function fundamentally defines the learning objective of an RL agent, and thus the behaviour of the policy. This may be especially consequential for ACD, as an incorrectly specified reward function can lead to policies that appear successful in maximising long-term rewards, but fail to genuinely achieve defensive objectives.

Motivate the Reward Function. Reward functions should be motivated and justified with respect to the underlying defensive objective. The general goal in network defence is typically the sustained operation of a system that continues to meet user demands while hosts remain uncompromised. The decisions relating to reward function modelling must be justified and documented, ensuring that the chosen reward function reflects operational requirements rather than arbitrary design choices.

Align with the ACD Problem Goal. It is important to consider the alignment of rewards to the overarching goal. Reward functions fundamentally assign relative value to states and actions, which can unintentionally introduce equivalencies between meaningfully different network states (e.g., restoring a user host versus an operational server). Reward misalignment can lead the agent to "game" the task by maximising rewards without producing a valid defensive policy, referred to as reward hacking. Therefore, principled reward design is necessary to avoid reward hacking and produce operationally valuable behaviour.

Simplify Instead of Over Engineering. Where possible, reward functions should minimise unnecessary shaping and avoid encoding detailed domain heuristics directly into scalar rewards. Highly engineered dense rewards can introduce unintended biases and impose arbitrary trade-offs between defensive actions and network states. As a result, reward functions should ideally be sparse such that only a few, but feasibly reachable, goal-aligned state-action pairs provide a reward signal [8, 9]. This is achievable in ACD RL tasks where episodes begin in a goal state (i.e., an uncompromised network). Sparse rewards place fewer constraints on agent behaviour, support long-horizon planning, and reduce the likelihood of learning policies that exploit artefacts of reward design rather than achieving goal objectives. However, sparsity must be balanced against exploration and learning stability [10]. Scaling from smaller, less complex networks to large networks of interest throughout training, as a form of curriculum learning, may help to mitigate some of these problems.

4.4 Evaluation

Comprehensive evaluation methods are critical for building a good RL cyber environment. Thorough evaluations in complex RL environments illuminate any unknowns about true agent performance, revealing issues in both modelling and network virtualisation that might otherwise be missed. The process of realistically simulating or emulating networks is inherently noisy. Therefore, robust evaluation measures are needed to give assurances about the reproducibility, statistical significance, and risk profile of agent performance relative to the specified ACD problem.

Understanding the Learnt Policy. Typical RL evaluation practices for ACD problems include reporting average episodic rewards [15, 39, 68], and variance metrics [27, 34]. Although, this alone is not sufficient to get a complete picture of an agent's learnt policy, nor is it an independent measure of how correct an MDP is.

Ensure Statistical Validity. Sufficient analysis of variance in policy performance is frequently missing in research applying RL to cybersecurity tasks [48]. Statistical validity is only achievable using multiple independent training runs with different random seeds to establish confidence intervals and ensure reproducibility [58].

Evaluate Behaviour at the System-Level. An RL agent will always seek to maximise the reward signal it receives and simply evaluating using average episodic rewards only shows how well the agent has optimised its behaviour with respect to *that* signal.

Hence, evaluating using average episodic rewards is not sufficient to get a full picture of the trained agent's policy, nor is it an independent measure of how correct your task modelling is, reflecting Goodhart's law: when a metric becomes the optimisation target, it ceases to be a reliable measure [6, 36].

Agent evaluation must move beyond only episodic rewards and extend its metrics to measure system-level activity relevant to the goal: how often are hosts attacked? Which hosts? What kind of attack? Analysis of the blue agent's learnt policy can shed light on where in the network the blue agent is acting most and what the

average action distribution looks like. This builds a better picture of the defensive strategy the agent has learnt and provides granular policy information to critically assess how well the observations, actions and rewards are being modelled. The information to track these metrics and extract agent trajectories is found at the simulation level of the cyber environment, where the state of the network can be assessed in relation to the foundational network defence goals independently of the RL task.

5 RELATED WORK

Motivations and Limitations of existing ACD Environments. An early effort at designing an RL environment for ACD was FARLAND [52], "*a framework for advanced Reinforcement Learning for autonomous network defense*". This work emphasised the importance of configurability in the underlying environment and the ability to perform curriculum learning by training across different scenarios of varying complexity. FARLAND also incorporated adversarial red behaviour that manipulates the observations of the blue agent, as opposed to simply attempting to infiltrate the network. CybORG is a simulator developed for the "Cyber Autonomy Gym for Experimentation" (CAGE) challenges [26, 37–39, 65], which has become a standard benchmark environment for the development of RL agents for ACD. The CybORG version most commonly used in the literature is CAGE 2, but a number of authors identify fundamental issues with this environment. CybORG++ [17] is an iteration on the CAGE 2 version of CybORG that identifies and fixes problems with the original version and offers a lightweight alternative "MiniCAGE". A more recent environment is Cyberwheel [55], which provides both a simulator and emulator. As part of the motivation for developing Cyberwheel, the authors discuss their attempt to extend the CAGE 2 version of CybORG and identify six core issues that led them to develop a new environment: Lack of network topology configurability, inadequate red agent behaviour, no emulation environment, no visualisation tools, poorly scalable observation design (see section 4.3.2), and dead code.

As part of Dstl's ARCD (Autonomous Resilient Cyber Defence) programme, four simulators were developed [51, 63]. Yawning Titan [5] is a low-fidelity, abstract simulated environment, intended to encourage fast experimental iteration of defensive agent architectures. Within the constraints of its abstract setting, it allows for a great degree of configuration of network topology, node types, reward functions and action types. PrimAITE [15] is a higher fidelity simulated environment allowing for the configuration of Information Exchange Requirements (IERs) and red and green patterns-of-life at a node level (as opposed to simulated red and green agents). In principle, this could allow for training a defensive agent on a large distribution of possible scenarios, provided it is feasible to generate sufficient numbers of realistic pattern-of-life configuration files. Imaginary Yak is a closed-source emulated environment that can operate on containers and virtual machines, and PalisAIDE the highest fidelity environment from ARCD, operating on virtual machines and hardware. One aim of this suite of environments was to transfer an agent trained in a simulated environment (PrimAITE) to a real system (PalisAIDE). Short [63] reports there was some success here, but involved a significant engineering challenge, addressing various sim-to-real pitfalls.

ACD Survey Papers. Vyas *et al.* [71] provide a survey of environments for, and approaches to autonomous cyber network defence. Whilst similar in motivation, our paper differs in that we take steps towards formulating a comprehensive framework for building a realistic cyber defence simulator, based on feedback from practitioners during the workshop. Palmer *et al.* [57] provide another review of deep RL for ACD, with a particular emphasis on scalability challenges and an overview of existing ACD environments. In addition to reviewing existing DRL approaches, they outline desirable properties for benchmarking environments that range from appropriate fidelity to the need for emulators.

Multi-agent Reinforcement Learning for ACD. While this paper focuses on single-agent formulations, MARL offers a complementary perspective for ACD, where red and blue agents can co-evolve through self-play [73] and multiple defensive agents can coordinate across network segments. However, multi-agent training introduces additional challenges, including non-stationarity [25] and exponentially growing joint state-action spaces [33, 60]. Many of the principles discussed in this paper remain relevant in multi-agent settings.

RL for Network Offence. Whilst this paper focuses on RL for network defence, there is also a large body of research [12, 21, 32, 43, 45, 72] on RL environments for penetration testing. For example, CyberBattleSim [68] and NASim [62]. NASimEmu [34] extends NASim to have an emulation component, enabling the transfer of agents trained in simulation to an emulated network.

RL for Cybersecurity. Outside of network security, RL has been applied to a range of different cybersecurity applications. McFadden *et al.* [48] surveyed and systematised common pitfalls of the domain at large. Common applications include: *evading detection* [13, 30, 70], *detecting malicious activity* [46, 59], *vulnerability discovery* [3, 19, 47], blockchain security [14, 31], and hardware security [24, 44].

6 CONCLUSION

Building effective RL environments for ACD requires principled decisions at every stage of design, from characterising the real-world problem to virtualisation of the network and modelling the agent interface. This paper provides a conceptual framework that maps the core components of ACD environments to two parts of the *sim-to-real* gap, virtualisation and modelling. Using this framework, the paper provides a systematisation of best practice guidelines. The central theme of these guidelines is that utility of an ACD environment is determined by how faithfully its design reflects the structure and constraints of the real-world problem; without this grounding, agents risk being optimised for performance in the training environment that does not transfer to capabilities in the operational task. We hope that the framework and best practice guidelines provided in this paper enable the development of more effective ACD environments.

ACKNOWLEDGMENTS

We would like to additionally thank Dr. Andres Molina-Markham for their contributions to this work both in the initial workshop and during the write-up.

REFERENCES

[1] Disrupting the first reported AI-orchestrated cyber espionage campaign. Tech. rep., Anthropic (2025), (Online, Accessed 21st January 2026) https://assets.anthropic.com/m/ec212e6566a0d47/original/Disrupting-the-first-reported-AI-orchestrated-cyber-espionage-campaign.pdf

[2] Agarwal, R., Schwarzer, M., Castro, P.S., Courville, A., Bellemare, M.G.: Deep reinforcement learning at the edge of the statistical precipice. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21 (2021)

[3] Al Wahaibi, S., Foley, M., Maffeis, S.: SQIRL: Grey-box detection of sql injection vulnerabilities using reinforcement learning. In: 32nd USENIX Security Symposium (USENIX Security 23) (2023)

[4] Anderson, R., Moore, T.: The economics of information security. science **314**(5799), 610–613 (2006)

[5] Andrew, A., Spillard, S., Collyer, J., Dhir, N.: Developing optimal causal cyber-defence agents via cyber security simulation. In: Workshop on Machine Learning for Cybersecurity (ML4Cyber) (07 2022)

[6] Ashton, H.: Causal campbell-goodhart's law and reinforcement learning (2021), https://arxiv.org/abs/2011.01010

[7] Bates, E., Mavroudis, V., Hicks, C.: Reward shaping for happier autonomous cyber security agents. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (2023)

[8] Bates, E., Hicks, C., Mavroudis, V.: Less is more? rewards in rl for cyber defence. arXiv preprint arXiv:2503.03245 (2025)

[9] Bates, E., Hicks, C., Mavroudis, V.: Beyond rewards in reinforcement learning for cyber defence (2026), https://arxiv.org/abs/2602.04809

[10] Bates, E., Mavroudis, V., Hicks, C.: Reward shaping for happier autonomous cyber security agents (2023), https://arxiv.org/abs/2310.13565

[11] Chapman, E., Hicks, C., Mavroudis, V.: r3ace. https://doi.org/10.5281/zenodo.15147271, https://github.com/alan-turing-institute/r3ace

[12] Chen, J., Hu, S., Zheng, H., Xing, C., Zhang, G.: GAIL-PT: An intelligent penetration testing framework with generative adversarial imitation learning. Computers & Security (2023)

[13] Chen, X., Nie, Y., Guo, W., Zhang, X.: When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. Advances in Neural Information Processing Systems (2024)

[14] De Silva, R., Guo, W., Ruaro, N., Grishchenko, I., Kruegel, C., Vigna, G.: {GuideEnricher}: Protecting the anonymity of ethereum mixing service users with deep reinforcement learning. In: 33rd USENIX Security Symposium (USENIX Security 24) (2024)

[15] Defence Science and Technology Laboratory UK: Primaite (primary-level ai training environment), https://github.com/Autonomous-Resilient-Cyber-Defence/PrimAITE, gitHub repository (tag: v4.0.0). Accessed 2026-02-19

[16] Dudman, T., Bull, M.: Towards a generalisable cyber defence agent for real-world computer networks (2025), https://arxiv.org/abs/2511.09114

[17] Emerson, H., Bates, L., Hicks, C., Mavroudis, V.: Cyborg++: An enhanced gym for the development of autonomous cyber agents (2024), https://arxiv.org/abs/2410.16324

[18] Foley, M., Hicks, C., Highnam, K., Mavroudis, V.: Autonomous Network Defence Using Reinforcement Learning. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. ASIA CCS '22 (2022), https://doi.org/10.1145/3488932.3527286

[19] Foley, M., Maffeis, S.: Apirl: Deep reinforcement learning for rest api fuzzing. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)

[20] Foley, M., Wang, M., M, Z., Hicks, C., Mavroudis, V.: Inroads into Autonomous Network Defence using Explained Reinforcement Learning. In: Conference on Applied Machine Learning in Information Security (CAMLIS) (2022)

[21] Gangupantulu, R., Cody, T., Park, P., Rahman, A., Eisenbeiser, L., Radke, D., Clark, R., Redino, C.: Using cyber terrain in reinforcement learning for penetration testing. In: 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS). IEEE (2022)

[22] Geiger, M., Bauer, J., Masuch, M., Franke, J.: An analysis of black energy 3, crashoverride, and trisis, three malware approaches targeting operational technology systems. In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). vol. 1, pp. 1537–1543. IEEE (2020)

[23] 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. Springer (2024)

[24] Gohil, V., Guo, H., Patnaik, S., Rajendran, J.: Attrition: Attacking static hardware trojan detection techniques using reinforcement learning. In: Proceedings of the 2022 ACM SIGSAC conference on computer and communications security (2022)

[25] Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review **55**(2), 895–943 (2022)

[26] Group, T.C.W.: Ttcp cage challenge 3. https://github.com/cage-challenge/cage-challenge-3 (2022)

[27] Hammar, K., Stadler, R.: Intrusion prevention through optimal stopping. IEEE Transactions on Network and Service Management **19**(3), 2333–2348 (2022). https://doi.org/10.1109/TNSM.2022.3176781

[28] Han, Y., Rubinstein, B.I., Abraham, T., Alpcan, T., De Vel, O., Erfani, S., Hubczenko, D., Leckie, C., Montague, P.: Reinforcement learning for autonomous defence in software-defined networking. In: International conference on decision and game theory for security. Springer (2018)

[29] Hicks, C., Mavroudis, V., Foley, M., Davies, T., Highnam, K., Watson, T.: Canaries and Whistles: Resilient Drone Communication Networks with (or without) Deep Reinforcement Learning. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security. AISec '23 (2023)

[30] Hore, S., Ghadermazi, J., Paudel, D., Shah, A., Das, T., Bastian, N.: Deep packgen: A deep reinforcement learning framework for adversarial network packet generation. ACM Transactions on Privacy and Security (2025)

[31] Hou, C., Zhou, M., Ji, Y., Daian, P., Tramer, F., Fanti, G., Juels, A.: Squirrl: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. arXiv preprint arXiv:1912.01798 (2019)

[32] Hu, Z., Beuran, R., Tan, Y.: Automated penetration testing using deep reinforcement learning. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE (2020)

[33] Huh, D., Mohapatra, P.: Multi-agent reinforcement learning: A comprehensive survey. arXiv preprint arXiv:2312.10256 (2023)

[34] Janisch, J., Pevnỳ, T., Lisỳ, V.: Nasimemu: Network attack simulator & emulator for training agents generalizing to novel scenarios. In: European Symposium on Research in Computer Security. pp. 589–608. Springer (2023)

[35] Kaloroumakis, P., Smith, M.: Toward a knowledge graph of cybersecurity countermeasures (2020)

[36] Karwowski, J., Hayman, O., Bai, X., Kiendlhofer, K., Griffin, C., Skalse, J.: Goodhart's law in reinforcement learning (2023), https://arxiv.org/abs/2310.09144

[37] Kiely, M., Ahiskali, M., Borde, E., Bowman, B., Bowman, D., van Bruggen, D., Cowan, K., Dasgupta, P., Devendorf, E., Edwards, B., et al.: Exploring the efficacy of multi-agent reinforcement learning for autonomous cyber defence: A cage challenge 4 perspective. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 28907–28913 (2025)

[38] Kiely, M., Ahiskali, M., Borde, E., Bowman, B., Bowman, D., Van Bruggen, D., Cowan, K., Dasgupta, P., Devendorf, E., Edwards, B., et al.: Cage challenge 4: A scalable multi-agent reinforcement learning gym for autonomous cyber defence. AI Magazine **46**(3), e70021 (2025)

[39] Kiely, M., Bowman, D., Standen, M., Moir, C.: On autonomous agents in a cyber defence environment (2023), https://arxiv.org/abs/2309.07388

[40] King, I.J., Bowman, B., Huang, H.H.: Automated cyber defense with generalizable graph-based reinforcement learning agents (2025), https://arxiv.org/abs/2509.16151

[41] Kushner, D.: The real story of stuxnet. ieee Spectrum **50**(3), 48–53 (2013)

[42] Kvasov, A., Sahin, M., Hebert, C., De Oliveira, A.S.: Simulating deception for web applications using reinforcement learning. In: European Symposium on Research in Computer Security. Springer (2023)

[43] Li, Q., Hu, M., Hao, H., Zhang, M., Li, Y.: INNES: An intelligent network penetration testing model based on deep reinforcement learning. Applied Intelligence **53**, 27110–27127 (2023)

[44] Luo, M., Xiong, W., Lee, G., Li, Y., Yang, X., Zhang, A., Tian, Y., Lee, H.H.S., Suh, G.E.: Autocat: Reinforcement learning for automated exploration of cache-timing attacks. In: 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE (2023)

[45] Maeda, R., Mimura, M.: Automating post-exploitation with deep reinforcement learning. Computers & Security (2021)

[46] McFadden, S., Foley, M., D'Onghia, M., Hicks, C., Mavroudis, V., Paoletti, N., Pierazzi, F.: Drmd: Deep reinforcement learning for malware detection under concept drift. In: Proc. of the AAAI Conference on Artificial Intelligence (2026)

[47] McFadden, S., Maugeri, M., Hicks, C., Mavroudis, V., Pierazzi, F.: Wendigo: Deep reinforcement learning for denial-of-service query discovery in graphql. In: IEEE Workshop on Deep Learning Security and Privacy (DLSP) (2024)

[48] McFadden, S., Foley, M., Bates, E., Tsingenopoulos, I., Vyas, S., Mavroudis, V., Hicks, C., Pierazzi, F.: Sok: The pitfalls of deep reinforcement learning for cybersecurity. arXiv preprint arXiv:2602.08690 (2026)

[49] Mern, J., Hatch, K., Silva, R., Hickert, C., Sookoor, T., Kochenderfer, M.J.: Autonomous attack mitigation for industrial control systems (2021), https://arxiv.org/abs/2111.02445

[50] Mern, J., Sadigh, D., Kochenderfer, M.J.: Exchangeable input representations for reinforcement learning (2020), https://arxiv.org/abs/2003.09022

[51] Miles, I., Farmer, S., Foster, D., Harrold, D., Palmer, G., Parry, C., Willis, C., Casassa Mont, M., Gralewski, L., Menzies, R., Morarji, N., Turkbeyler, E., Wilson, A., Beard, A., Marques, P., Francis Roscoe, J., Bailey, S., Cheah, M., Dorn, M., Haubrick, P., Lacey, M., Rimmer, D., Stone, J., Till, D., Heartfield, R., Harrison, A., Short, J., Wilson, T., H, J.: Reinforcement learning for autonomous resilient cyber defence (Aug 2024), https://www.fnc.co.uk/media/mwcnckij/us-24-milesfarmer-reinforcementlearningforautonomousresilientcyberdefence-wp.pdf, presented at Black Hat USA, August 2024

[52] Molina-Markham, A., Miniter, C., Powell, B., Ridley, A.: Network environment design for autonomous cyberdefense (2021), https://arxiv.org/abs/2103.07583

[53] Molina-Markham, A., Robaina, L., Steinle, S., Trivedi, A., Tsui, D., Potteiger, N., Brandt, L., Winder, R., Ridley, A.: Training rl agents for multi-objective network defense tasks. arXiv preprint arXiv:2505.22531 (2025)

[54] Nyberg, J., Johnson, P.: Structural generalization in autonomous cyber incident response with message-passing neural networks and reinforcement learning (2024), https://arxiv.org/abs/2407.05775

[55] Oesch, S., Chaulagain, A., Weber, B., Dixson, M., Sadovnik, A., Roberson, B., Watson, C., Austria, P.: Towards a high fidelity training environment for autonomous cyber defense agents. In: Proceedings of the 17th Cyber Security Experimentation and Test Workshop. p. 91–99. CSET '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3675741.3675752, https://doi.org/10.1145/3675741.3675752

[56] Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., Song, D.: Assessing generalization in deep reinforcement learning (2018)

[57] Palmer, G., Parry, C., Harrold, D.J.B., Willis, C.: Deep reinforcement learning for autonomous cyber defence: A survey (2024), https://arxiv.org/abs/2310.07745

[58] Patterson, A., Neumann, S., White, M., White, A.: Empirical design in reinforcement learning. Journal of Machine Learning Research (2024)

[59] Praveena, V., V., A., Chinnasamy, P., Ali, I., Alroobaea, R., Alyahyan, S.Y., Raza, M.A.: Optimal deep reinforcement learning for intrusion detection in uavs. Computers, Materials & Continua (2022)

[60] Qu, G., Wierman, A., Li, N.: Scalable reinforcement learning for multiagent networked systems. Operations Research **70**(6), 3601–3628 (2022)

[61] Samaddar, A., Potteiger, N., Koutsoukos, X.: Out-of-distribution detection for neurosymbolic autonomous cyber agents. In: 2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC). pp. 1–9 (2025). https://doi.org/10.1109/ICAIC63015.2025.10849024

[62] Schwartz, J., Kurniawatti, H.: Nasim: Network attack simulator. https://networkattacksimulator.readthedocs.io/ (2019)

[63] Short, J.: The essential role of modelling and simulation in helping ai fight cyber-attacks. In: Force Readiness for Multi-Domain Operations through Modelling and Simulation: NATO Modelling and Simulation Group (MSG) Symposium (MSG-229). No. STO-MP-MSG-229 in STO Meeting Proceedings, NATO Science and Technology Organization (STO) (2025), https://publications.sto.nato.int/publications/STO%20Meeting%20Proceedings/STO-MP-MSG-229/MP-MSG-229-02.pdf, paper MP-MSG-229-02 (Open Access)

[64] Sparrell, D.: Open command and control (openc2) architecture specification version 1.0. (2022), https://docs.oasis-open.org/openc2/oc2arch/v1.0/oc2arch-v1.0.html.

[65] Standen, M., Lucas, M., Bowman, D., Richer, T., Kim, J., Marriott, D.: CybORG: A gym for the development of autonomous cyber agents. In: IJCAI-21 1st International Workshop on Adaptive Cyber Defense (2021)

[66] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, second edn. (2018), http://incompleteideas.net/book/the-book-2nd.html

[67] Symes Thompson, I., Caron, A., Hicks, C., Mavroudis, V.: Entity-based reinforcement learning for autonomous cyber defence (2025), https://arxiv.org/abs/2410.17647

[68] 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.

[69] Terranova, F., Lahmadi, A., Chrisment, I.: Leveraging deep reinforcement learning for cyber-attack paths prediction: Formulation, generalization, and evaluation. In: Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (2024)

[70] Tsingenopoulos, I., Cortellazzi, J., Bosanskỳ, B., Aonzo, S., Preuveneers, D., Joosen, W., Pierazzi, F., Cavallaro, L.: How to train your antivirus: Rl-based hardening through the problem space. In: Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (2024)

[71] Vyas, S., Mavroudis, V., Burnap, P.: Towards the Deployment of Realistic Autonomous Cyber Network Defence: A Systematic Review. ACM Comput. Surv. **58**(1) (Aug 2025). https://doi.org/10.1145/3729213

[72] Yang, Y., Chen, L., Liu, S., Wang, L., Fu, H., Liu, X., Chen, Z.: Behaviour-diverse automatic penetration testing: a coverage-based deep reinforcement learning approach. Frontiers of Computer Science (2025)

[73] Zhang, R., Xu, Z., Ma, C., Yu, C., Tu, W.W., Tang, W., Huang, S., Ye, D., Ding, W., Yang, Y., et al.: A survey on self-play methods in reinforcement learning. arXiv preprint arXiv:2408.01072 (2024)