Cyber Threat Mitigation with Knowledge-Infused Reinforcement Learning and LLM-Guided Policies

Md. Shamim Towhid*†, Shahrear Iqbal*, Euclides Carlos Pinto Neto*, Nashid Shahriar†, Scott Buffett*, Madeena Sultana‡, Adrian Taylor‡
*National Research Council, Canada, {shamim.towhid, shahrear.iqbal}@nrc-cnrc.gc.ca
* {euclidescarlos.pintoneto, scott.bufett}@nrc-cnrc.gc.ca
†Department of Computer Science, University of Regina, {mty754, nashid.shahriar}@uregina.ca
‡Defence Research and Development Canada, {madeena.sultana, adrian.taylor}@forces.gc.ca

***Abstract*—As cyber threats continue to evolve, there is a need for autonomous cyber defense (ACD) strategies capable of fast and context-aware responses. Reinforcement learning (RL) has shown promise for automating cyber defense by exploring and learning effective countermeasures, yet it often struggles with sparse reward signals and insufficient context to handle diverse attack scenarios. Furthermore, the convergence time taken by an RL agent is often high, which makes it difficult to train the RL agent in online settings. To address these challenges, we propose a large language model (LLM)-enhanced RL method that builds and queries a knowledge graph (KG) derived from agent–environment interactions. We leverage the pre-trained knowledge of an LLM on different cybersecurity frameworks and use the LLM to analyze a part of the KG to generate appropriate actions for the RL agent. We infuse the knowledge extracted from the LLM into the RL agent's training loop in two ways. First, the state vector of the RL agent is augmented with the most effective action and its corresponding reward, as determined from the KG. Second, the suggested action from the LLM is used as a reference policy. In addition, we introduce a regularization term in the loss function to make the RL policy close to the reference policy. To validate our approach, we develop a custom RL environment guided by the MITRE ATT&CK framework, enabling the agent to generate tailored mitigation strategies for detected cyber attacks. Experimental results show that our proposed approach significantly outperforms the baseline RL by over 75% in terms of taking better mitigation actions.**

***Index Terms*—Reinforcement Learning, Large Language Models, Knowledge Graphs, Cybersecurity Automation, Autonomous Cyber Defense**

I. Introduction

Cyber threats continue to become more dangerous and complex. Attackers constantly change their methods, and defenders must respond just as quickly. Traditional intrusion detection systems (IDS) can flag suspicious traffic, but they rarely adapt well to new attacks [1]. This gap in adaptability makes it difficult to choose the right mitigation action at the right time. As a result, cybersecurity research is increasingly focusing on methods that can make real-time defense decisions, considering that IDS can fail to detect cyber attacks correctly.

This paper focuses on using RL to handle the challenge of selecting proper mitigation actions, even when the detection module may misclassify network flows. We treat the defense problem as a sequence of actions that must adapt to the possibility of detection errors. Our approach draws on large language models (LLMs) that have been trained on diverse cybersecurity frameworks, such as MITRE ATT&CK [2]. By leveraging these models, we aim to provide the RL agent with a dynamic knowledge base that can be updated and queried during training to supply context to an open-source LLM, which in turn guides the RL agent's policy.

Existing RL-based methods for cyber defense often suffer from two issues: slow convergence and a lack of context-aware strategies [3]. When the environment provides only sparse or misleading reward signals, an RL agent might need a large amount of data to learn effectively. Moreover, many RL agents [4]–[7] do not incorporate external knowledge about known attack methods or best practices, which hampers their adaptability to newer attacks. LLMs, on the other hand, are pre-trained on vast amounts of Internet data, which includes references to frameworks like Cyber Kill Chain [8] and Unified Kill Chain [9]. By connecting RL with these LLMs, we can shorten the learning curve and improve decision-making under uncertainty.

Despite the potential of RL in ACD, most existing solutions assume that the detection module accurately classifies malicious traffic [10]. However, detection modules are often imperfect in reality, and misclassifications can lead to suboptimal or even harmful mitigation actions. Current RL-based approaches [11]–[16] lack mechanisms to explicitly reason under such uncertainty. Our work addresses this critical gap by proposing a novel system that augments RL agents with detection confidence information and leverages an LLM-guided KG to select better mitigation actions even when the traffic classification is uncertain. To the best of our knowledge, this is the first framework that systematically integrates real-time KG construction, querying, and LLM-driven knowledge infusion into the RL training loop for cyber defense. Unlike prior work that only uses RL for policy guidance, our approach actively builds and queries a KG during training, allowing the agent to continuously refine its decision-making based on evolving knowledge from past interactions.

The motivation for introducing the KG is twofold. First, the KG captures dynamic relationships between detected attack types, detection confidence levels, and corresponding mitigation actions along with their historical rewards. This structured representation helps the agent reason over uncertainty rather than relying solely on the current noisy observation. Second, by querying the KG, an LLM can suggest historically effective

actions, providing an external reference policy that guides the RL agent's exploration and policy updates through regularization. We explore the following research questions:

- How can RL be guided by an LLM and a dynamically updated KG to choose effective mitigation actions when the attack detection is uncertain or potentially incorrect?
- To what extent can aligning the RL agent's policy with an LLM-guided reference policy improve decision quality and accelerate learning speed?
- What specific advantages and disadvantages does the integration of RL, LLMs, and KGs offer for ACD, compared to using RL alone?

To answer these questions, we formulate a contextual bandit [17] problem, modeled as a single-step Markov Decision Process (MDP). In this setting, the agent receives a state vector containing detection confidence and contextual information, and must select an appropriate mitigation action in a single decision step. To evaluate the effectiveness of our method, we compare it against a baseline RL agent without LLM guidance, an RL agent using a KG and LLM assistance, and a supervised deep learning (DL) approach trained to directly predict mitigation actions from the state vector. Our main contributions are:

- **KG Creation:** We build a graph that links traffic signatures, predicted traffic types, and actions, weighted by rewards and confidence scores.
- **LLM-Driven Mitigation Suggestions:** We employ an open-source LLM to analyze the KG and recommend mitigation actions grounded in cybersecurity best practices.
- **Knowledge Infusion:** We embed the LLM-suggested action and reward into the RL state, and we introduce a regularization term based on the Kullback–Leibler (KL) divergence [18] to align the RL agent's policy with the LLM's suggestions.
- **Custom RL Environment:** We design and test our approach in a custom environment guided by the MITRE ATT&CK framework and publicly available intrusion detection dataset.

The remainder of the paper is structured as follows: We discuss the related works from the literature in Section II. Our proposed approach, in addition to our custom RL environment, is discussed in Section III. Section IV outlines the conducted experiments and results. The next section discusses the impact of using LLM-enhanced RL for the mitigation action selection task along with some limitations of the current work. This section also mentions our future research directions. Finally, we conclude the paper in Section VI.

II. Related Works

In this section, we review existing work on using RL for cyber defense and highlight hybrid approaches that merge RL with other AI paradigms. We then discuss recent research on LLM-enhanced RL methods, focusing on how LLMs can serve as knowledge repositories, planners, or policy guides. Finally, we compare these studies with our proposed approach, showing where our approach departs from conventional RL frameworks by incorporating an explicit knowledge graph, Bayesian inference, and an LLM-guided reference policy.

A. RL-based Cyber Defense

One of the earliest trends in ACD was to apply deep reinforcement learning (DRL) to automate detection and mitigation workflows. For example, Liu et al. [11] developed a DRL model on an extended version of the MITRE ATT&CK framework, demonstrating the feasibility of mapping attack tactics to prioritized nodes in a network. By selectively deploying mitigations on these critical points, their model improved defensive efficacy against sophisticated cyberattacks. In contrast, Huang and Zhu [12] utilized a Bayesian game framework to model the defender–attacker interplay, particularly for advanced persistent threats (APTs). While their model captured attacker stealth, it depended on accurately identifying attacker behavior probabilities—an assumption that often falters in real-world conditions.

Similarly, Gao and Wang [14] investigated DRL-based moving target defense (MTD) to thwart DDoS attacks, showcasing the promise of dynamic reconfiguration strategies. Becker et al. [13] evaluated asynchronous actor-critic (A3C) and Q-learning for penetration testing, revealing the potential of DRL to generalize across diverse network setups. However, these RL-centric methods typically assume reliable detection labels or well-shaped rewards—conditions not guaranteed in practice, especially when the IDS or signature database may misclassify incoming traffic. This issue motivates hybrid approaches that embed supplementary knowledge into the RL pipeline.

B. Hybrid Methods for Cybersecurity

To reduce the dependency on purely data-driven models, researchers have explored combining RL with symbolic reasoning, causal inference, and multi-agent or hierarchical frameworks. Peng et al. [15] designed a causality-driven hierarchical RL framework to expedite exploration in adversarial environments. Zhu et al. [19] introduced a deconfounding strategy for offline DRL, re-weighting observational data to minimize bias introduced by unseen confounders. Another angle focuses on partially modeling the environment's causal relationships in RL. Rezende et al. [16] argued that flawed causal assumptions could lead to suboptimal actions, especially in high-dimensional or noisy settings.

Hybrid AI designs that fuse data-driven learning with human-readable rules or knowledge graphs have also gained traction. Piplai et al. [20], for example, examined neuro-symbolic methods that integrate neural networks for pattern recognition with symbolic components for interpretability. While these efforts improve the transparency and adaptability of ACD systems, they typically do not leverage large-scale pre-trained knowledge from language models. Consequently, these methods may still require extensive domain engineering or rely on limited training data to capture the breadth of evolving cybersecurity threats.

C. LLM-Enhanced RL: From General Paradigms to Cybersecurity

A recent survey by Cao et al. [21] offers a unified view of LLM-Enhanced RL, categorizing five fundamental integration strategies: (i) treating the LLM as an information processor, (ii) using the LLM to design or shape rewards, (iii) aligning the RL policy with an LLM decision-maker, (iv) employing the LLM as a generator for environment simulation or policy explanation, and (v) hybrid approaches combining these roles. In general, LLM-powered methods have produced encouraging results in robotics, text-based adventures, and planning-based tasks, where the model's knowledge and reasoning capabilities can mitigate sparse rewards or data limitations.

However, direct application of LLMs on cybersecurity is comparatively sparse. Existing RL frameworks for cyber defense rarely incorporate large-scale pre-trained language models to manage knowledge, especially when detection modules mislabel traffic or produce uncertain classification scores. The few [20], [22] that integrate textual knowledge (e.g., intrusion logs or threat intelligence) generally use them as offline references rather than dynamic advisors. As a result, many RL-based defense systems continue to rely on handcrafted reward functions or one-off heuristics, yielding slower convergence under partial information.

Our proposed approach addresses these gaps by combining an explicit KG derived from RL agent–environment interactions with an open-source LLM to generate mitigation actions. This design simultaneously tackles detection uncertainty and sparse reward issues. Unlike prior methods, such as (Gao and Wang, [14] or Liu et al., [11])—that assume a static or reliable detection stage, we store and periodically update Bayesian confidence scores for each signature in our KG, allowing the RL agent to ignore signatures with low confidence and focus on those with higher accuracy. Furthermore, the reward function in our environment incorporates an LLM-curated action list for each attack type, incentivizing the RL agent to match the top-ranked mitigating measures. Finally, by embedding the LLM-suggested action and its historical reward into the agent's state vector and adding a KL divergence regularization term in the loss, we ensure the RL policy remains aligned with the LLM's knowledge.

III. LLM-Enhanced RL Agent

In this section, we discuss our proposed approach in detail. First, we discuss the RL model used in our approach, including its states, actions, and rewards. Second, the KG creation process is discussed. Finally, how we create the prompt for the LLM model using the KG and how the generated knowledge from LLM is infused with RL training are discussed.

A. RL Environment

The RL environment is crucial to train a successful RL agent because the agent learns by interacting with the environment. In this paper, we leverage the CICIDS2017 [23] dataset available publicly to prepare our custom RL environment. The CICIDS2017 is a supervised dataset for intrusion detection. The data is available in terms of network flow. A network flow is defined as a collection of network packets with the same source and destination IP, ports, and protocol. There are more than 80 network flow features available in this dataset. The CICFlowMeter [24] is used to extract these features from the dataset. There are fourteen types of attack traffic in this dataset. We combine similar attack classes into one class to make the dataset less skewed and have better detection accuracy. For example, three types of web attacks (SQL Injection, Brute Force, and Cross Site Scripting) are combined into one class named "Web". Table I shows the number of flows per class after combining similar classes into one. This dataset represents the real scenario, as we can see from Table I that the number of benign flows is much larger than attack flows. As the dataset is for intrusion detection, there is no information regarding mitigation actions. However, the network configuration and settings are mentioned in [23]. We use this information to generate mitigation actions with the help of three proprietary LLMs as discussed later in this section.

Class	Number of Flows
Benign	2271320
DDoS	128025
DoS	251712
FTP-Patator	7935
Infiltration	2003
PortScan	158804
SSH-Patator	5897
Web	2180

TABLE I: Number of flows per class in modified CICIDS2017

As a first step to build our custom RL environment, we decouple the intrusion detection functionality (i.e., detecting anomalous traffic) from the mitigation task. For the detection task, we train a decision tree (DT) classifier on the CICIDS2017 dataset (which is split into a 70% training set and a 30% test set). From this trained classifier, we extract all possible decision paths, resulting in 1,451 signatures. These signatures, along with their corresponding predicted class labels, form a signature database for the RL environment shown in Fig. 1.

Each signature sig_i in the database stores: (i) the predicted traffic class label (i.e., attack type or benign), (ii) the total number of flows processed by that signature (N_i), (iii) the number of correctly classified flows by that signature (C_i), and (iv) a Bayesian posterior confidence score B_i. As training progresses, whenever a flow arrives and matches a particular signature, N_i and C_i are updated to reflect the newly observed data, and B_i is recalculated. This enables dynamic adaptation of the signature's reliability based on real-time evidence.

We define the Bayesian posterior confidence score B_i for each signature sig_i using a Beta distribution with conjugate prior. Let α and β be hyperparameters for the Beta prior, typically initialized to reflect a uniform prior (e.g., α = 1 and β = 1). After observing N_i flows, out of which C_i were correctly classified, the posterior confidence for signature sig_i at time step t is given by Equation 1.

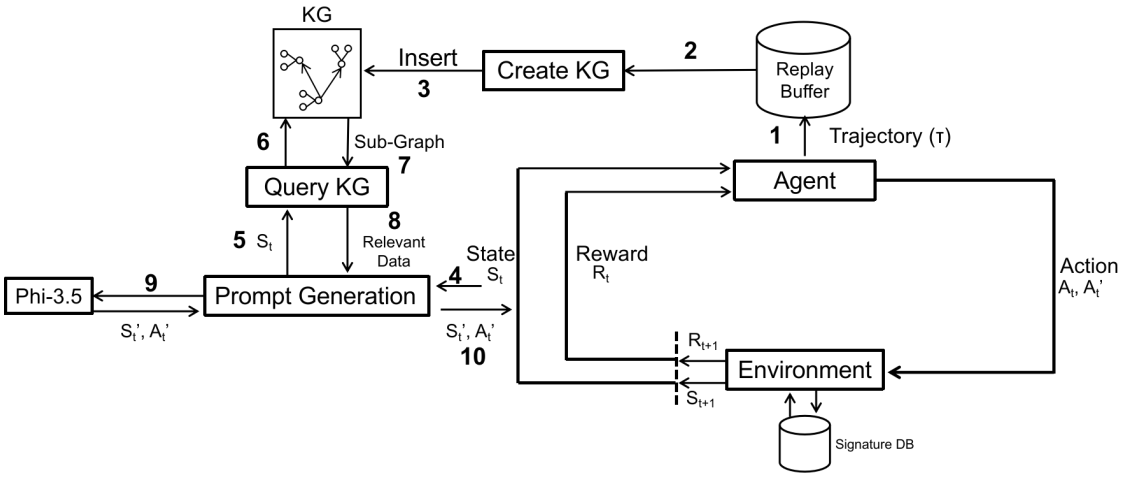


Fig. 1: Overview of our proposed approach

$$B_i^{(t)} = \frac{\alpha + C_i}{\alpha + \beta + N_i} \tag{1}$$

By incorporating $B_i^{(t)}$ in the RL observation, the agent is able to assess the reliability of each signature when deciding on the appropriate mitigation action.

State, Action, and Reward Function: At each episode, a single network flow is sampled from the training set. The flow is classified by one of the signatures in the database, say sig_i, and assigned a predicted traffic type y_i (e.g., benign, DoS, brute force, etc.). Hence, at time step t, the RL environment provides an observation vector as follows.

$$S_t = \left[y_i,\, \text{ID}(\text{sig}_i),\, N_i,\, C_i,\, B_i^{(t)}\right] \tag{2}$$

Here y_i is the predicted class label of the flow *i*, $\text{ID}(\text{sig}_i)$ is the identifier of the signature that matched the flow, N_i and C_i respectively denote the total and correctly classified flow counts for sig_i, $B_i^{(t)}$ is the current Bayesian posterior confidence for sig_i. The RL agent then chooses a mitigation action A_t from a fixed set of 14 possible mitigation actions derived from the MITRE ATT&CK framework (e.g., 'Filter Network Traffic", "Re-authenticate user" etc.), along with a special "No Action" choice for benign traffic. These 14 actions are curated and ranked using three proprietary LLMs, OpenAI's O1, GPT 4O, and O3 mini—which act as domain experts. The commonly recommended actions from these LLMs are compiled into an ordered list of five prioritized mitigations for each attack type. The selected mitigation actions for each of the network traffic are shown in Table II.

To guide the RL agent toward selecting effective mitigations, we design a reward function that leverages the LLM-recommended action list. Let $\mathcal{A}_{\text{LLM}} = \{a_1, a_2, a_3, a_4, a_5\}$ be the ordered list of five recommended actions from the LLMs for a given attack type. The reward R_t at time step t is defined as follows:

- If the agent selects the top-ranked action a_1, it receives +5.
- If it selects the k-th recommended action a_k (where $1 < k \leq 5$), it receives $+(6 - k)$. Hence, +4 for the second recommended action, down to +1 for the fifth recommended action.

TABLE II: Selected mitigation actions for each traffic type

Traffic Type	Mitigation actions	Mitigation ID from MITRE ATT&CK
PortScan	Filter Network Traffic	M1037
	Network Intrusion Prevention	M1031
	Network Segmentation	M1030
	Limit Access to Resource Over Network	M1035
	Disable or Remove Feature or Program	M1042
DoS	Filter Network Traffic	M1037
	Network Intrusion Prevention	M1031
	Network Segmentation	M1030
	Limit Access to Resource Over Network	M1035
	Audit	M1047
Benign	No Action	-
DDoS	Filter Network Traffic	M1037
	Network Intrusion Prevention	M1031
	Network Segmentation	M1030
	Limit Access to Resource Over Network	M1035
	Audit	M1047
SSH-Patator	Password Policies	M1027
	Multi-factor Authentication	M1032
	Account Use Policies	M1036
	Filter Network Traffic	M1037
	Behavior-based Traffic Analysis	M1040
FTP-Patator	Password Policies	M1027
	Multi-factor Authentication	M1032
	Account Use Policies	M1036
	Filter Network Traffic	M1037
	Behavior-based Traffic Analysis	M1040
Web	Input Validation	M0818
	Multi-factor Authentication	M1032
	Execution Prevention	M1038
	Privileged Account Management	M1026
	Behavior-based Traffic Analysis	M1040
Infiltration	Filter Network Traffic	M1037
	Update Software	M1051
	Multi-factor Authentication	M1032
	Privileged Account Management	M1026
	Password Policies	M1027

- If the agent selects an action not in $\mathcal{A}_{\text{LLM}}$, it receives -5.
- For benign traffic, the correct action is "No Action". Selecting any other action results in a -5 reward.

Formally, we can write as follows:

$$R_t(A_t) = \begin{cases} 6 - k, & \text{if } A_t = a_k \in \mathcal{A}_{\text{LLM}}, \\ -5, & \text{otherwise.} \end{cases} \tag{3}$$

The above reward scheme aligns the RL agent's actions with expert-recommended mitigations from the LLMs, assigning higher rewards to more effective responses and penalizing unsafe or irrelevant actions. Therefore, it guides the agent toward safer and context-aware decisions. By emphasizing alignment with a prioritized list of suggested mitigations, the RL agent is encouraged to converge toward a policy that is both effective and contextually consistent with established cybersecurity practices. Each episode terminates once the reward is assigned, after which a new flow is sampled from the dataset in the next episode, allowing the RL agent to iteratively refine its mitigation policy.

Although cyber attack mitigation often involves long action sequences, our current setting reduces to a one-step episodic Markovian Decision Process (MDP), which is known as the contextual bandit [17]. The RL environment has no dynamics in a bandit problem, which means the reward depends on the current action and the observation only. Since the RL agent only observes predicted traffic type with its confidence and statistics, the true state is never revealed to the agent. The agent's objective is to learn a policy that maximizes the expected reward under the unknown joint distribution of true state and the observed state. This is the objective solved by RL algorithms in a contextual bandit form. RL is therefore both applicable and necessary, as it can discover the optimal action selection strategy despite observation noise, sparse rewards, and a non-stationary traffic distribution.

B. Knowledge Graph Creation

During each training cycle, the RL agent interacts with our custom environment over a fixed number of episodes. Each interaction (i.e., observed state, chosen action, and corresponding reward) is stored in a replay buffer as a trajectory, τ. After completing these episodes, we perform two updates. First, we train the policy network and update the value function using the saved trajectories. Second, we create or update the KG based on the information contained in the replay buffer. The goal of this KG is to capture and aggregate knowledge about how different signatures, traffic types, and mitigation actions relate to one another, thereby providing a richer context to guide subsequent LLM policy improvements.

The KG is a weighted graph featuring three distinct types of nodes: traffic types, mitigation actions, and signature IDs, and three types of weighted edges. The first edge type links the detected traffic type to the selected mitigation action, capturing the cumulative reward obtained for that particular pairing. The second edge type connects a signature ID to its predicted traffic type, weighted by the Bayesian posterior confidence B_i. The third edge connects two traffic-type nodes representing misclassification of one traffic type to another. This edge is weighted by three attributes: percentage of misclassification, signature used for the prediction and confidence score when the traffic is misclassified by the detector. A simplified version of the KG is shown in Fig. 2, where three types of nodes and edges are visible. The edge between Benign and DDoS traffic means, when signature 3776 detects "Benign" traffic, 15% of the time it is actually "DDoS" traffic, and the confidence score for this misclassification is less than 0.92. As the agent processes more network flows, the Bayesian confidence values stored on these edges are recalculated to reflect the updated statistics of correctly classified versus total flows for the relevant signature.

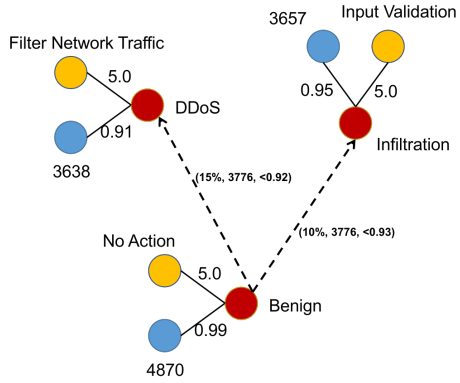


Fig. 2: A simplified KG

A key challenge in our setup arises from the possibility that the same action can produce both positive and negative rewards for the same traffic type. For instance, if a flow is initially misclassified by a low-confidence signature, the "correct" action for that predicted traffic category may lead to a negative reward. Over time, this duality is also reflected in the KG: edges between traffic types and actions accumulate weighted feedback for both correct and incorrect classification scenarios. Through repeated updates, the RL agent can learn to rely less on signatures with low confidence scores and develop robust mitigation strategies even under partial or inaccurate information. After the training phase, the constructed KG contained 1,306 nodes and 1,382 edges, reflecting the diversity of observed traffic patterns and mitigation outcomes.

C. Knowledge Infusion

In our proposed framework, once the RL agent receives a new state S_t from the environment, it propagates this information to a prompt generation module (see step 4 in Fig. 1). The module first queries the existing KG with the predicted attack type and the identifier of the signature that triggered this detection. The KG responds with all connected nodes (mitigation actions, misclassified traffic types) and their associated edge weights. These edges are then serialized into text using a predefined template.

Following this serialization, any actions associated with negative rewards and not connected to misclassified traffic

nodes are filtered out, leaving only actions that historically have led to positive outcomes in similar states. Table III shows the core template of our prompts, where three placeholders: $actions$, $records$, and $attack$ are replaced by:

- $attack$: the predicted attack type derived from the KG query, the signature used for this prediction,
- $records$: the serialized edges for all actions that have positive rewards for the identified attack type, the misclassification rate, and confidence of the signature from KG, actual traffic type in the case of misclassification, the correct mitigation action for the misclassified node,
- $actions$: the subset of the total 14 unique mitigation actions remaining after filtering out those with consistently negative rewards and not connected to misclassified traffic.

The completed prompt is then passed to Microsoft's Phi-3.5 mini instruct [25] language model for local inference. Phi-3.5, with its 3.82B parameters, requires approximately 15 GB of GPU memory in float32 precision and provides an average inference time of ~ 0.58 seconds given our prompt structure. We select Phi-3.5 mini instruct model because it can fit into one single GPU and it is an instruction fine-tuned model, which means it can follow instructions. Moreover, Phi-3.5 model performs decently on the Cybermetric benchmark dataset as reported in [26]. Cybermetric evaluates the ability of LLM models to answer cybersecurity questions. The LLM module returns two key attributes: (i) the most effective mitigation action for the attack observed in the past, and (ii) the corresponding reward that action yielded from KG. A post-processing module ensures the correctness and format of these raw outputs of the LLM.

Once the LLM-suggested action and historical reward are obtained, they are fused into the RL loop. Formally, we extend S_t as follows:

$$S'_t = \big[S_t,\, A_{\text{LLM}},\, R_{\text{LLM}}\big] \tag{4}$$

Here A_{LLM} is the action recommended by the language model and R_{LLM} is the reward historically observed for $\big[y_i, A_{\text{LLM}}\big]$. This augmented state enables the agent to leverage both the Bayesian confidence from the KG and the expert-like insights provided by the LLM.

In addition to augmenting the RL state, we treat the LLM's behavior as a reference policy, denoted $\pi_{\text{LLM}}(A \mid S_t)$. The RL agent's policy is $\pi_\theta(A \mid S_t)$, parameterized by θ. To align the agent's policy with the LLM policy, we introduce the KL divergence [18] term as a regularization component in the loss function. Specifically, we compute the KL divergence:

$$D_{\text{KL}}\big(\pi_\theta(\cdot \mid S_t) \,\|\, \pi_{\text{LLM}}(\cdot \mid S_t)\big) \;=\; \sum_A \pi_\theta(A \mid S_t)\, \ln\Big[\tfrac{\pi_\theta(A|S_t)}{\pi_{\text{LLM}}(A|S_t)}\Big] \tag{5}$$

Let $\mathcal{L}_{\text{RL}}(\theta)$ denote the original RL loss (e.g., a policy gradient objective). We augment it with the KL regularization term weighted by λ:

$$\mathcal{L}_{\text{total}}(\theta) \;=\; \mathcal{L}_{\text{RL}}(\theta) \;+\; \lambda\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid S_t) \,\|\, \pi_{\text{LLM}}(\cdot \mid S_t)\big) \tag{6}$$

By penalizing large deviations from the LLM's policy distribution, the RL agent is gently guided toward the expert-like behavior proposed by Phi-3.5. Over multiple training epochs, this knowledge infusion helps the agent converge more efficiently, especially in scenarios where the state and reward signals are partially misleading. The RL agent balances between exploration and exploitation by trying random actions beyond LLM suggestions and favoring expert-informed actions through LLM-guided policy alignment.

TABLE III: Prompt template for the local LLM

—System—
You are a Cybersecurity expert. Given a predicted attack on networks, you must select one mitigation action from the following list separated by comma to mitigate the predicted attack.

Some previous records of predicted traffic and selected actions with its rewards are also given below. You must select the action with reward 5.00. If the records do not have any action with reward 5.00, select based on your knowledge of the MITRE ATT&CK framework.

In addition to previous records, which signature used for prediction and confidence of prediction are given. Sometimes the prediction can be wrong. According to previous records, how many times the particular prediction was wrong and what was actual traffic are given below.

Reason thoroughly before answering and consider the fact that predictions can be wrong. When prediction is wrong, some possible true attack list is also given below.

—User—
Now, if **{attack}** traffic is predicted. Select one mitigation action from the given list. Output only the mitigation action name, nothing else.
Mitigation actions: **{actions}**
Previous records: **{records}**

—Assistant—

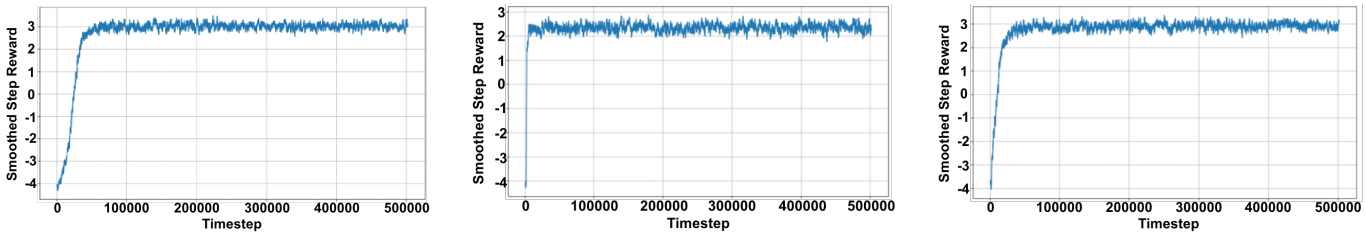
IV. Evaluation Results

In this section, we describe our evaluations and results. At first, we vary the value of the KL coefficient λ to control the KL penalty and observe the effect of it. Observing the effect of the KL coefficient, we decide to anneal the value of this coefficient in our next experiment. Finally, our third experiment shows the improvement compared to the baseline where we introduce supervised pre-training.

A. Experimental Settings

We train the RL policy by using the Proximal Policy Optimization (PPO) algorithm [27]. Due to its training stability and popularity in the field of Cybersecurity, we select the PPO algorithm. All the experiments are performed on an 11^{th} Gen i7 machine that has one GeForce RTX 3090 GPU with 24GB memory and 32GB main memory.

As mentioned in Section III-A, we divide the CICIDS2017 dataset into two sets. The RL training happens by using the network flow in the training set of CICIDS2017. Later, we evaluate the RL policy on the network flows of the test set



(a) PPO (b) PPO-KG $\lambda = 1.0$ (c) PPO-KG $\lambda = 0.1$

Fig. 3: The effect of varying the value KL coefficient (λ)

of CICIDS2017. During the training of the RL agent, we traverse through the network flows of the training set five times in total. We test the learned policy on all the network flows of the test set. We report both training time reward and testing time reward to evaluate our proposed approach in comparison with the baseline. We select the performance of the PPO algorithm as baseline, referred to as PPO, in our experiment. Our goal is to improve the performance of the RL agent by using LLM and KG. Hence, we combine LLM and KG with the RL agent (referred to as PPO-KG) in one of our experiments. In another experiment, we initialize the RL agent's policy network with pre-trained weights rather than random initialization, and then integrate both the LLM and KG into the RL training (referred to as PPO-KG-PT). We evaluate the performance of the LLM that leverages the KG without any RL agent (LLM-KG). A supervised DL approach that directly maps the state vector to mitigation actions is also compared with our proposed approach.

B. Effect of varying the value of KL coeffcient

In our first experiment, we showcase the effect of varying the value of KL coefficient (λ). We control the amount of penalty given to the RL policy with this parameter. The value of λ ranges between 0 and 1. A high value of this parameter $(\lambda = 1)$ means we give more penalty if the RL policy is not close to the LLM policy. On the other hand, a low value of this coefficient means we give less penalty. We experiment with two values of λ. For the high value, we use $\lambda = 1$ and for the low value, we use $\lambda = 0.1$. When $\lambda = 1$, we take the whole KL divergence and apply that as regularization. $\lambda = 0.1$ means we take only 10% of the penalty for regularization. Fig. 3 shows the average training reward obtained by each of the variations of the PPO algorithm.

Fig. 3 shows that a high value of λ helps faster convergence but the average reward is slightly lower compared to PPO (Fig. 3a). On the other hand, $\lambda = 0.1$ achieves the same average reward by compromising a little bit of convergence time. We run all these three experiments for 500K episodes. This experiment suggests that a higher value of λ achieves better convergence but lacks accuracy. By observing this result, we decide to use KL coefficient annealing. At the beginning of the training, the KL value should be high as it helps in the convergence. As training progresses, we decrease the value of λ by following a linear decay schedule so that the RL agent achieves a better average reward.

Fig. 4 shows the performance of KL coefficient annealing in comparison with the baseline PPO algorithm. In this experiment, the agent is trained for five iterations of the whole training set. As the value of λ is high at the beginning of the training, we see the PPO-KG algorithm converges to a lower average reward compared to the baseline PPO algorithm. After certain episodes, when the value of λ reduces to a certain value, the PPO-KG starts to outperform the baseline algorithm. Since the KG is updated during training, our proposed approach achieves a higher but less stable reward curve, as shown in Fig. 4.

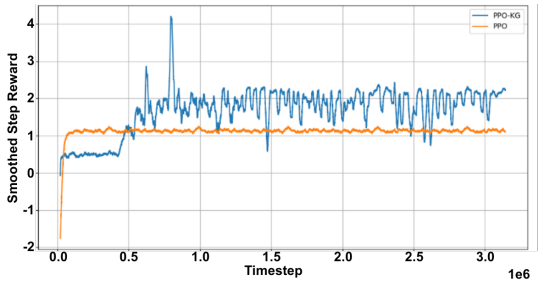


Fig. 4: PPO vs PPO-KG with KL coefficient annealing

C. Supervised Pre-training of the RL Policy Network

As our proposed approach outperforms the baseline in the training set, the next step is to evaluate the learned policy on the test set. Our proposed approach achieves a 2.88 average reward on the test set whereas the baseline PPO algorithm achieves only 1.13. Therefore, our proposed approach outperforms the baseline on average reward in the test set. However, the class-wise average reward on the test set reported in Table IV shows that the main benefit comes from the "Benign" class. As the number of network flows on "Benign" class is the highest and our proposed approach takes correct action for the large number of "Benign" traffic, the average reward for "Benign" class is higher (-0.11) than the baseline PPO (-5.0).

Table IV also shows that for certain classes, such as "SSH-Patator", "FTP-Patator", and "Infiltration" the baseline PPO algorithm achieves a better average reward compared to our proposed approach. Our investigation reveals that the PPO algorithm has gradient clipping functionality that prevents drastic changes between two consecutive policies during the training. Hence, during the random initialization of the RL policy network, if the distance between the RL policy and

LLM policy in the feature space is too high, the PPO-KG algorithm struggles to converge to the LLM policy.

To make the RL policy close to the LLM policy from the beginning of the training, we implement a behavior cloning strategy in our next experiment. Since we already have the KG and the LLM policy ready to infer, we create a supervised dataset that maps states to actions according to the LLM policy. Then, we pre-train the RL policy network for ten epochs on this supervised dataset. Instead of randomly initializing the RL policy network, we initialize it with these trained parameters. Now, the RL policy should be closer to the LLM policy from the beginning of the RL training. The fourth column of Table IV shows this result.

Class	PPO	PPO-KG	PPO-KG-PT
PortScan	5.0	5.0	5.0
DoS	5.0	4.98	4.99
Benign	-5.0	-0.11	4.99
DDoS	5.0	4.99	4.99
SSH-Patator	1.99	-3.87	1.97
FTP-Patator	1.99	-4.99	1.98
Web	-5.0	-4.88	-4.94
Infiltration	5.0	2.20	2.35
Average	1.13	**2.88**	**4.91**

TABLE IV: Class-wise performance of the proposed approach in comparison with the baseline on test dataset

Table IV shows significant improvement in our proposed approach after supervised pre-training. The average reward on the test set after supervised pre-training is 4.91 which is significantly higher than what we obtained previously. Interestingly, Table IV shows that our proposed approach "PPO-KG-PT" does not perform well on the "Web" and "Infiltration" classes. The vanilla PPO algorithm performs well on the "Infiltration" class; however, it performs the worst on the "Web" class. We argue that the baseline PPO algorithm is biased towards the most popular mitigation action in the dataset, which is "Filter Network Traffic". This is the best mitigation action for four types of attacks among the eight types of traffic in our dataset (see Table II). Therefore, the PPO algorithm is biased towards this action. The poor performance of the PPO algorithm on the "Web" class is evidence of this behavior because the best action for the "Web" attack is not filtering but "Input Validation". Hence, the PPO algorithm does not perform well on the "Web" attack class. On the other hand, the LLM-enhanced RL approach performs better than vanilla PPO on the "Web" class.

D. Performance of LLM-enhanced RL Under Uncertainty

It is crucial to evaluate the proposed approach when the detection module is incorrect. Since we propose utilizing a KG that captures the misclassification of network traffic associated with a particular signature, the LLM model should reason about this misclassification by observing the relations captured in the KG. In this subsection, we compare the performance of our proposed approach when the detection module is correct versus when it is incorrect.

First, we need to evaluate the performance of the detection module to understand its misclassification rate and assess the impact of our proposed RL-based approaches (PPO-KG and PPO-KG-PT). Table V shows the class-wise performance of the detection module. The "Support" column in Table V shows the number of data samples for a particular class in the test set of CICIDS2017. The signature-based detection module performs well in most of the classes except for "Web" and "Infiltration" classes according to Table V. In the following subsection, we investigate how the performance of different approaches varies with the performance of the detection module.

Class Name	Precision	Recall	F1-score	Support
PortScan	99.33	99.96	99.64	47641
DoS	99.63	99.77	99.70	75514
Benign	99.94	99.88	99.91	681396
DDoS	99.96	99.90	99.93	38408
SSH-Patator	99.27	99.66	99.46	1769
FTP-Patator	99.92	99.79	99.85	2380
Web	95.14	95.72	95.43	654
Infiltration	71.45	72.05	71.75	601

TABLE V: Classification report of the signature-based detection module

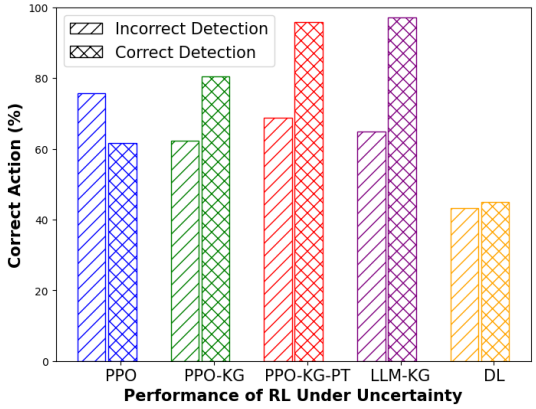


Fig. 5: Performance of RL Under Uncertainty

Figure 5 shows the percentage of correct actions taken by different agents in the case of correct versus incorrect detection. The LLM-KG approach shows what the LLM model can achieve without any fine-tuning or an RL agent. Here, the prompt for LLM is created using the KG and state vector (S_t). By taking the prompt as input, the LLM model outputs the mitigation action. Since our problem formulation is a one-step MDP, and we have the mitigation action as ground truth in the dataset, we can train a DL model in a supervised manner that takes the state vector as input and outputs a mitigation action like a classification problem. The performance of this approach is shown in Figure 5 as DL. For the model architecture in the DL approach, we select the same model architecture used as a policy network in the RL-based approaches for a fair comparison. One difference is that we iterate through the training dataset 5 times in RL-based approaches during

training, whereas we iterate through the entire training set 100 times in the DL approach.

Figure 5 shows that our proposed approach "PPO-KG-PT" is the best overall among the compared approaches. On the other hand, the supervised DL approach is the worst, according to Figure 5. Although the DL model is trained for more epochs compared to RL-based approaches, it performs the worst in terms of both correct and incorrect detection. The reason for the poor performance of the DL approach is the imbalanced dataset. As shown in Table I and Table V, both the training and test sets are highly imbalanced across different classes. Therefore, the DL approach overfits the majority class in the dataset. Three variations of our proposed approach—"PPO-KG", PPO-KG-PT", and LLM-KG"—outperform the baseline PPO algorithm and DL when the detection is correct. However, in the case of incorrect detection, the baseline PPO algorithm performs better than our proposed approach. The reason for this behavior of the baseline PPO algorithm lies in the distribution of mitigation actions in our dataset. For the majority of attack types, "Filter Network Traffic" is the mitigation action that obtains the maximum reward. Therefore, the baseline PPO selects this mitigation action most of the time. Hence, in Table IV, we observe that the baseline PPO algorithm does not perform well on the classes where "Filter Network Traffic" is not the best mitigation action.

Our proposed approaches in Figure 5 learn a policy that is close to the LLM policy. Therefore, the performance of our proposed approaches (PPO-KG and PPO-KG-PT) is close to the "LLM-KG" approach. In the case of correct detection, the LLM selects the correct mitigation action more than 98% of the time. In the case of incorrect detection, if the relation between the predicted and actual traffic is not captured by the KG, then the LLM does not receive any clue about the misclassification by the detector. Furthermore, even if there is a relation between two traffic type nodes in the KG, it is not guaranteed that the predicted traffic is misclassified. The LLM model needs to pay attention to the confidence score of the prediction and the percentage of previous misclassifications to reason about possible mitigation actions. Hence, the performance of LLM-enhanced RL is lower than that of the baseline PPO when detection is incorrect.

V. Discussion and Future Directions

The objective of this paper was to answer three research questions discussed in Section I. First, we found that RL agents can be guided by LLM to select effective mitigation actions during a cyber attack by utilizing a KG created from the agent-environment interactions. The KG must capture the relationship between predicted and actual traffic by signature in cases of misclassification. Furthermore, the KG must be incrementally updated to stay current. Data should be collected from the networks periodically to update the KG and RL policy. Our experiments show that Open-source LLMs like Phi-3.5 are capable of selecting proper mitigation actions like a human expert if given the proper context and instructions. The performance of LLMs varies depending on where instructions are placed within a prompt, for example, instructions at the beginning versus the end of the prompt may influence the performance. Every instruction-fine-tuned LLM, like phi-3.5, has a specific prompt format, and we must follow that prompt style to maximize the benefits from the LLM.

Second, aligning the RL agent's policy with an LLM-based reference policy improved its performance by approximately 75%. We achieved this by embedding the LLM-suggested action and its historical reward into the state vector and introducing a KL regularization term to penalize policy deviation. Third, combining LLM knowledge and reference policy increased the RL agent's average test-set reward from 1.13 to 4.91. Annealing the KL coefficient accelerated convergence, requiring fewer episodes. By factoring in Bayesian confidence and past rewards, the LLM effectively mitigates even when facing uncertain traffic classification, enabling faster and more reliable cyber responses than RL alone.

Our current contextual-bandit setting evaluates only a single mitigation decision per flow. It does not yet capture multi-step reasoning (e.g., triage $\rightarrow$ forensics $\rightarrow$ remediation) that real incident-response workflows require. As a next step, we will extend our work to automate cyber dynamic tasks [28]. We will use simulation platforms like CybORG [29] to create dynamic environments and evaluate our proposed approach in solving dynamic tasks. We plan to apply post-processing and human validation to mitigate potential biases in LLM recommendations and prevent unsafe actions. The proposed LLM-enhanced RL method requires approximately 0.88 seconds per inference, primarily due to LLM inference overhead, whereas the baseline PPO agent requires only 1.7 milliseconds per inference. This gap may pose challenges in high-throughput, real-time deployment scenarios. As future work, we will explore quantized, distilled, or more lightweight LLMs to reduce this computational burden and improve response latency. Additionally, we have tested only one RL algorithm (PPO) in our experiments. As future work, we aim to evaluate the proposed approach with additional RL algorithms. Specifically, we will investigate whether the requirement for supervised pre-training in our proposed approach can be eliminated by incorporating other RL algorithms.

VI. Conclusion

This paper addresses the challenge of selecting appropriate cyber threat mitigation actions under uncertainty, where intrusion detection systems may misclassify traffic and provide limited context. Traditional RL agents often struggle in such settings due to sparse rewards and slow convergence. To overcome these limitations, we propose a novel method that combines RL with external knowledge derived from an open-source LLM and a dynamically updated KG.

We formulate the mitigation task as a one-step MDP, enabling fast decision-making in the presence of noisy observations. Our custom RL environment, built on the CICIDS2017 dataset and guided by the MITRE ATT&CK framework, allows the agent to interact with realistic traffic flows and detection outputs. Our results show that this LLM-enhanced

RL agent significantly outperforms a baseline PPO agent, raising the average test-set reward from 1.13 to 4.91, which is over 75% improvement toward optimal performance. The agent also maintains robust behavior under misclassification and learns policies that align with expert-recommended actions more quickly and reliably.

Beyond performance improvements, this framework exemplifies how external knowledge can be integrated into RL without compromising online adaptability. The approach is model-agnostic, meaning alternative open-source LLMs or external threat-intelligence feeds can be incorporated with minimal effort. Thus, it bridges the research communities of RL, knowledge-graph reasoning, and cybersecurity, providing a template for future hybrid systems operating under partial observability and evolving threat landscapes.

In conclusion, our study provides empirical evidence that coupling LLM with a knowledge-aware RL agent yields near-optimal, context-aware cyber mitigations in real-time. We hope this blueprint accelerates progress toward self-adaptive defense systems and inspires broader adoption of LLM-enhanced RL across cybersecurity domains.

REFERENCES

[1] M. A. Hossain and M. S. Islam, "Ensuring network security with a robust intrusion detection system using ensemble-based machine learning," *Array*, vol. 19, p. 100306, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005623000310

[2] The MITRE Corporation (2025), "MITRE ATT&CK," Accessed: 2025-03-26. [Online]. Available: https://attack.mitre.org

[3] I. Tareq, B. M. Elbagoury, S. A. El-Regaily, and E.-S. M. El-Horbaty, "Deep reinforcement learning approach for cyberattack detection," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 20, no. 05, p. pp. 15–30, Mar. 2024. [Online]. Available: https://online-journals.org/index.php/i-joe/article/view/48229

[4] W. Lalouani and M. Younis, "Robust distributed intrusion detection system for edge of things," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[5] I. A. Khan, I. Razzak, D. Pi, N. Khan, Y. Hussain, B. Li, and T. Kousar, "Fed-inforce-fusion: A federated reinforcement-based fusion model for security and privacy protection of iomt networks against cyber-attacks," *Information Fusion*, vol. 101, p. 102002, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523003184

[6] S. Otoum, N. Guizani, and H. Mouftah, "Federated reinforcement learning-supported ids for iot-steered healthcare systems," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[7] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419306815

[8] S. Cho, I. Han, H. Jeong, J. Kim, S. Koo, H. Oh, and M. Park, "Cyber kill chain based threat taxonomy and its application on cyber common operational picture," in *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, 2018, pp. 1–8.

[9] P. Pols and F. Domínguez, "The unified kill chain," 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235741204

[10] I. Miles, S. Farmer, D. Foster, D. Harrold, G. Palmer, C. Parry, C. Willis, M. C. Mont, L. Gralewski, R. Menzies *et al.*, "Reinforcement learning for autonomous resilient cyber defenc e," *Presented at Black Hat USA*, 2024.

[11] Y. Liu, Y. Guo, R. Ranjan, and D. Chen, "Optimization of mitigation deployment using deep reinforcement learning over an enhanced att&ck," *Computing*, vol. 106, no. 12, p. 4015–4038, Sep. 2024. [Online]. Available: https://doi.org/10.1007/s00607-024-01344-4

[12] L. Huang and Q. Zhu, "Adaptive strategic cyber defense for advanced persistent threats in critical infrastructure networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 2, p. 52–56, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3305218.3305239

[13] N. Becker, D. Reti, E. V. Ntagiou, M. Wallum, and H. D. Schotten, "Evaluation of reinforcement learning for autonomous penetration testing using a3c, q-learning and dqn," *arXiv preprint arXiv:2407.15656*, 2024.

[14] C. Gao and Y. Wang, "Reinforcement learning based self-adaptive moving target defense against ddos attacks," *Journal of Physics: Conference Series*, vol. 1812, p. 012039, 02 2021.

[15] S. Peng, X. Hu, R. Zhang, K. Tang, J. Guo, Q. Yi, R. Chen, X. Zhang, Z. Du, L. Li, Q. Guo, and Y. Chen, "Causality-driven hierarchical structure discovery for reinforcement learning," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[16] D. J. Rezende, I. Danihelka, G. Papamakarios, N. R. Ke, R. Jiang, T. Weber, K. Gregor, H. Merzic, F. Viola, J. Wang *et al.*, "Causally correct partial models for reinforcement learning," *arXiv preprint arXiv:2002.02836*, 2020.

[17] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 661–670. [Online]. Available: https://doi.org/10.1145/1772690.1772758

[18] F. Raiber and O. Kurland, "Kullback-leibler divergence revisited," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ser. ICTIR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 117–124. [Online]. Available: https://doi.org/10.1145/3121050.3121062

[19] W. Zhu, C. Yu, and Q. Zhang, "Causal deep reinforcement learning using observational data," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, ser. IJCAI '23, 2023. [Online]. Available: https://doi.org/10.24963/ijcai.2023/524

[20] A. Piplai, A. Kotal, S. Mohseni, M. Gaur, S. Mittal, and A. Joshi, "Knowledge-enhanced neurosymbolic artificial intelligence for cybersecurity and privacy," *IEEE Internet Computing*, vol. 27, no. 5, pp. 43–48, 2023.

[21] Y. Cao, H. Zhao, Y. Cheng, T. Shu, Y. Chen, G. Liu, G. Liang, J. Zhao, J. Yan, and Y. Li, "Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2024.

[22] J. F. Loevenich, E. Adler, R. Mercier, A. Velazquez, and R. R. F. Lopes, "Design of an autonomous cyber defence agent using hybrid ai models," in *2024 International Conference on Military Communication and Information Systems (ICMCIS)*, 2024, pp. 1–10.

[23] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," pp. 108–116, 01 2018.

[24] A. Habibi Lashkari, G. Draper Gil, M. Mamun, and A. Ghorbani, "Characterization of tor traffic using time based features," pp. 253–262, 01 2017.

[25] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[26] N. Tihanyi, M. A. Ferrag, R. Jain, T. Bisztray, and M. Debbah, "Cybermetric: A benchmark dataset based on retrieval-augmented generation for evaluating llms in cybersecurity knowledge," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2024, pp. 296–302.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] M. Sultana, A. Taylor, L. Li, and S. Majumdar, "Towards evaluation and understanding of large language models for cyber operation automation," in *2023 IEEE Conference on Communications and Network Security (CNS)*, 2023, pp. 1–6.

[29] C. Baillie, M. Standen, J. Schwartz, M. Docking, D. Bowman, and J. Kim, "Cyborg: An autonomous cyber operations research gym," *arXiv preprint arXiv:2002.10667*, 2020.