

LLM-guided RL for Cybersecurity Orchestration

Hang Shen, *Member, IEEE*, Kerui Li, Xin Liang, Tianjing Wang, *Member, IEEE*, and Guangwei Bai

Abstract—Cybersecurity orchestration often relies on static playbooks and reinforcement learning (RL) policies defined over predefined action spaces, limiting their adaptability to unfamiliar attacks and changing operational environments. To address this limitation, this paper proposes an adaptive security-orchestration framework that integrates semantic retrieval, large language model (LLM) reasoning, and RL-based decision optimization. First, relevant strategy atoms are retrieved to ground LLM reasoning, while the LLM identifies coverage gaps and generates supplementary defensive actions. The generated atoms are validated against operational constraints and executable action templates before entering the playbook construction phase. Second, a Tree-of-Thoughts (ToT) prompting mechanism evaluates candidate atoms from multiple operational perspectives to improve their relevance and practical utility. Third, a dynamic-action RL mechanism combines semantic action representations, feasibility masking, and LLM-derived action priors to support scenario-dependent candidate action sets. High-quality RL trajectories are subsequently used to align the LLM with effective orchestration decisions. Experiments in a heterogeneous 5G-enabled Internet of Things (5G-IoT) enterprise security environment with real-world vulnerabilities demonstrate that the proposed framework improves robustness under unseen threats and varying operational loads, reduces operational failure rates, and enhances reference-based playbook quality compared with representative RL- and LLM-based baselines.

Index Terms—Cybersecurity Orchestration, Large Language Model (LLM), Reinforcement Learning (RL), Tree-of-Thoughts Prompting, Strategy-Atom Augmentation.

I. INTRODUCTION

CYBERSECURITY operations are becoming increasingly challenging as enterprise infrastructures grow more distributed, heterogeneous, and dynamically interconnected. The rapid adoption of large-scale services, cloud-edge platforms, heterogeneous endpoints, and diverse communication protocols has significantly expanded the attack surface, while modern adversaries continue to employ stealthy, persistent, and multi-stage attack strategies [1]–[3]. This growing complexity places substantial pressure on Security Operations Centers (SOCs), which must continuously monitor cyber assets and coordinate timely responses across fragmented environments [4]. As a result, improving the scalability, adaptability, and oper-

ational efficiency of security response has become a critical requirement for modern cyber defense.

Security Orchestration, Automation, and Response (SOAR) platforms have emerged as an important means of improving the efficiency and consistency of security operations [5], [6]. By integrating threat intelligence, situational awareness, and automated workflows, SOAR can reduce manual effort and accelerate incident handling. However, many existing SOAR systems depend on static playbook repositories and rule-based decision logic, which limits their ability to adapt to unfamiliar attacks and rapidly changing environments. Although recent advances in artificial intelligence and machine learning have introduced data-driven capabilities such as threat intelligence analysis and malicious behavior classification [7], these techniques generally do not support sequential, context-aware, and dynamically evolving playbook generation.

Reinforcement learning (RL) provides a promising direction for adaptive security orchestration by formulating playbook generation as a sequential decision-making problem over states, actions, and rewards [8]–[10]. Through interaction with the environment, RL can progressively optimize response policies without relying entirely on manually designed rules [11]. Nevertheless, existing RL-based orchestration systems are commonly built on predefined playbook repositories and fixed atomic action spaces, which limit their ability to generalize to unseen attack patterns, heterogeneous platforms, and evolving network conditions [9], [10]. Moreover, trial-and-error exploration can be sample inefficient during pre-deployment training when useful prior knowledge is limited, while conventional fixed-output policies cannot directly accommodate candidate actions that vary across threat scenarios. Hence, relying on RL alone remains insufficient for practical security orchestration that demands both adaptability and reliability.

Pre-trained large language models (LLMs) have recently shown strong potential for cybersecurity reasoning and decision support [12]–[14]. With their knowledge transfer and in-context learning capabilities, LLMs can help expand the candidate action space beyond predefined defensive knowledge and provide context-aware guidance for unfamiliar threat scenarios. Meanwhile, RL remains effective for iterative policy refinement and long-horizon optimization in dynamic environments [15], [16]. This complementarity motivates integrating LLM reasoning with RL decision-making, enabling security orchestration systems to generate adaptive, context-aware response playbooks for emerging cyber threats [17], [18].

A. Challenges and Related Works

However, such integration remains nontrivial because LLM-generated actions must be executable by the underlying security platform, while RL policies must accommodate scenario-dependent candidate sets without violating action feasibility.

Manuscript received XX April 2026; revised XX XXX 2026; accepted XX XXX 2026. This work was supported in part by the National Natural Science Foundation of China under Grants 61502230 and 61501224, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20201357, in part by the State Administration for Market Regulation Science and Technology Program under Grant 2025MK183, and in part by the Market Supervision Administration Science and Technology Fund of Jiangsu Province under Grant KJ2026027. (Corresponding author: Hang Shen)

Hang Shen, Kerui Li, Xin Liang, Tianjing Wang, and Guangwei Bai are with the College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing, 211816, China. E-Mail: {hshen, likerui, wangtianjing, bai}@njtech.edu.cn, conscienceoo@163.com

Digital Object Identifier: xxx

1) *Reliability and operational validity of LLM-generated strategy atoms.* LLMs can expand the candidate action space beyond predefined knowledge bases, which is valuable for handling unfamiliar or evolving threats. However, as probabilistic generators, they may produce actions that are syntactically plausible but operationally unsafe, logically inconsistent, or non-compliant with security policies [19]. Such errors are particularly critical in cybersecurity orchestration, where generated actions may directly affect mitigation workflows, service availability, and access-control decisions. Retrieval augmentation can improve semantic grounding, while prior studies on playbook generation and declarative orchestration have emphasized workflow coherence and semantic validation [20], [21]. Existing hallucination-mitigation techniques, such as inconsistency detection [22] and knowledge injection [23], improve factual reliability but do not guarantee that generated actions are executable under the current operational context. Therefore, LLM-generated atoms should be validated in terms of template registration, contextual parameters, tool availability, execution dependencies, and operational scores before playbook construction.

2) *RL decision making over varying and constrained action spaces.* LLM-based augmentation makes the candidate action set scenario-dependent, whereas conventional discrete RL policies usually assume a predefined action space. Dulac-Arnold *et al.* [24] addressed large discrete action spaces by embedding actions into a continuous representation and retrieving discrete actions through nearest-neighbor search. In cybersecurity, APRIL [25] further exploits semantic correlations among penetration-testing actions to construct a continuous action-embedding space for scalable policy learning and cross-scenario transfer. Terranova *et al.* [26] introduced continuous invariant input and output spaces for attack-path discovery, improving policy generalization across varying network structures and vulnerability sets. Invalid-action masking provides a complementary mechanism for removing infeasible actions from the policy support during optimization [27]. However, these studies primarily focus on penetration testing or attack-path discovery, leaving LLM-generated, scenario-dependent actions in defensive orchestration insufficiently explored.

3) *Reliable alignment between LLM guidance and RL policies.* RL has shown promise for adaptive cyber defense under uncertain and evolving attack conditions [28], [29], while LLMs can provide semantic priors that reduce uninformed exploration. However, combining their action distributions may introduce inconsistent preferences when the LLM assigns probabilities to infeasible actions. Moreover, feeding RL trajectories back to the LLM raises the question of whether trajectory-based alignment improves transferable orchestration knowledge or merely reinforces exploitation patterns already favored by the learned policy. Existing cybersecurity orchestration studies have highlighted the importance of policy-driven automation and operational control [30], while knowledge-enhanced LLM reasoning [31], symbolic and hierarchical RL [32], and LLM-assisted explanation mechanisms [33] provide complementary means of improving decision transparency and interpretability. Hybrid LLM-RL [34] demonstrates the potential of combining reasoning with policy op-

timization. However, aligning LLM and RL decisions over the same feasible action set, and using RL trajectories to improve LLM guidance without compromising robustness and traceability, remain insufficiently explored.

B. Contributions and Organization

Motivated by the above challenges, this paper develops an LLM-guided RL framework for adaptive security orchestration. The proposed framework couples LLM-based knowledge reasoning with RL-based policy optimization to improve the adaptability of strategy generation and the quality of security playbooks. The main contributions are threefold:

- We develop a strategy-atom-based representation for security orchestration and an LLM-assisted augmentation mechanism to address the coverage and quality limitations of semantic retrieval. A Tree-of-Thoughts (ToT) prompting scheme evaluates candidate atoms across multiple operational dimensions, while generated atoms are further validated for operational feasibility before entering playbook construction.
- We develop an RL decision mechanism for scenario-dependent candidate action spaces using semantic action representations and dynamic invalid-action masking. LLM-derived preferences are incorporated as priors over the same feasible action set, enabling adaptive action selection while excluding infeasible actions.
- We align the LLM with learned orchestration policies using high-quality RL trajectories. The state, feasible candidate set, and RL-selected action are serialized into structured supervision for lightweight LLM adaptation, enabling effective action preferences learned by the RL policy to be transferred back to the LLM.

To validate the proposed framework, we construct a heterogeneous 5G-enabled Internet of Things (5G-IoT) enterprise security environment with real-world vulnerabilities and evaluate it under diverse threat and operational conditions. The experiments examine strategy-atom quality and validity, RL optimization, varying action spaces, cross-scenario robustness, and RL-trajectory alignment. The evaluation addresses the following four research questions:

- **RQ1:** How effectively do knowledge enhancement and validation improve strategy-atom quality and validity?
- **RQ2:** Can the dynamic-action mechanism adapt to varying and constrained action spaces efficiently?
- **RQ3:** How well does the orchestration policy generalize to unseen threats and varying operational loads?
- **RQ4:** Can RL-trajectory alignment improve the LLM action prior without reinforcing biased policy preferences?

The rest of this paper is organized as follows. Section II presents the system model and problem formulation. Section III describes the proposed LLM-guided RL framework. Section IV introduces the experimental setup and dataset construction. Section V reports and analyzes the experimental results. Finally, the paper is concluded in the last section. The main variables and symbols are summarized in Table I.

TABLE I: Main Notations and Variables

Symbol	Description
a_t	Action selected at decision step t
$\mathcal{C}$	Current threat context
d_n	Description of strategy atom n
$\mathcal{E}$	Dependency-edge set of candidate actions
$\mathcal{H}$	Validated LLM-generated strategy-atom set
K	Number of retrieved strategy atoms
$\mathcal{N}$	Strategy-atom set in the knowledge base
$\mathcal{N}'$	Semantically retrieved strategy-atom set
$\mathcal{N}^*$	Final candidate strategy-atom set
r_t	Immediate reward at step t
s_t	Orchestration state at step t
v_n	Graph node corresponding to strategy atom n
x_n	Selection indicator of node v_n
y_n	Attack category of strategy atom n
$y_{m,n}$	Selection indicator of edge (v_m, v_n)
$\mathcal{Y}_C$	Attack-category set of threat context $\mathcal{C}$
$\mathcal{Y}'$	Attack-category set covered by $\mathcal{N}'$
$\alpha_{n,k}$	k -th operational attribute of strategy atom n
α_n^*	Refinement indicator of retrieved atom n
$\bar{\alpha}^{(4)}$	Average defensive-effectiveness score in $\mathcal{N}'$
β	Weight of the LLM action prior
ε	Minimum defensive requirement from threat severity
λ_l	Weight of the l -th operational cost
τ	Security-orchestration trajectory
θ	RL policy parameters

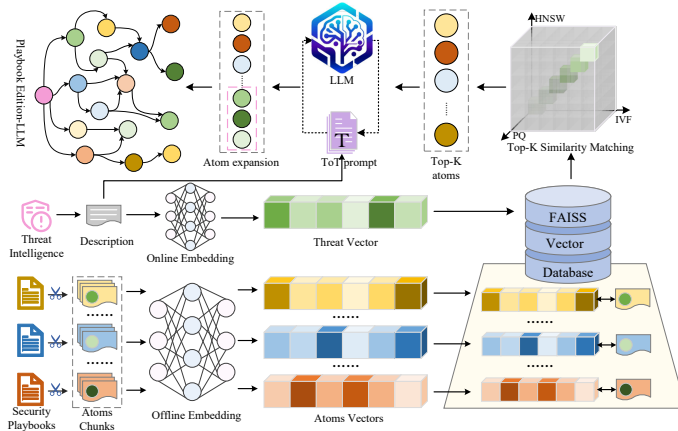


Fig. 1: Strategy-atom matching and augmentation process.

II. SYSTEM MODEL AND PROBLEM FORMULATION

This section presents the system model and problem formulation for cybersecurity playbook generation. It first describes how relevant defensive strategy atoms are retrieved and augmented in the current threat context, and then introduces the structured playbook representation that captures action dependencies and execution constraints for subsequent orchestration.

A. Strategy Atom Matching and Augmentation

Fig. 1 illustrates the proposed strategy-atom matching and augmentation process. Given an incoming threat context, the system first performs semantic retrieval over a knowledge base of defensive strategy atoms to identify the most relevant candidates. These retrieved atoms are then provided to the LLM as contextual references, enabling it to detect coverage

deficiencies and generate supplementary strategy atoms. In this way, the candidate action space can be expanded beyond purely retrieval-based matching, thereby improving threat coverage and orchestration flexibility.

A security playbook consists of multiple executable defensive actions, such as isolating a compromised host, blocking malicious traffic, or updating access-control policies. To facilitate modeling and optimization, each playbook is decomposed into minimally executable units, referred to as *strategy atoms*. Let $\mathcal{N}$ denote the complete set of strategy atoms in the knowledge base. For atom $n \in \mathcal{N}$, its textual description is denoted by d_n , and its associated attack category is denoted by y_n . In addition, each atom is annotated with four quantitative attributes, $\{\alpha_{n,1}, \alpha_{n,2}, \alpha_{n,3}, \alpha_{n,4}\}$, which represent response time, resource consumption, human involvement, and defensive effectiveness, respectively. For retrieved atoms, these attributes are initialized using tool metadata, historical execution records, and rule-based annotations. For newly generated atoms, the LLM provides initial estimates according to a predefined scoring rubric. The estimates are subsequently calibrated or checked using tool profiles, sandbox measurements, predefined rules, and expert annotations before being used for candidate filtering and downstream optimization.

Given the current threat context $\mathcal{C}$ with textual description d_C , the relevance between strategy atom n and the threat context is measured by a similarity function $\Phi(d_n, d_C)$. Based on this relevance score, the system selects the top- K candidate atoms from the knowledge base. Let $\mathcal{N}' \subseteq \mathcal{N}$ be the selected subset. The retrieval step is formulated as

$$\mathcal{N}' = \arg \max_{\substack{\mathcal{S} \subseteq \mathcal{N} \\ |\mathcal{S}|=K}} \sum_{n \in \mathcal{S}} \Phi(d_n, d_C). \quad (1)$$

Although semantic retrieval can identify atoms that are relevant to the current threat context, the retrieved subset may still suffer from insufficient threat coverage or limited defensive quality. To quantify the coverage of the retrieved atoms, let $\mathcal{Y}_C$ denote the set of attack categories associated with the current threat context, and let $\mathcal{Y}' = \{y_n \mid n \in \mathcal{N}'\}$ denote the attack categories covered by the retrieved atoms. The attack-category coverage ratio is denoted by

$$\rho_C = \frac{|\mathcal{Y}' \cap \mathcal{Y}_C|}{|\mathcal{Y}_C|}. \quad (2)$$

If $\rho_C < \Omega$, where Ω is a predefined coverage threshold, the retrieved set is considered insufficient in terms of threat-category coverage, with uncovered categories being $\mathcal{Y}_C \setminus \mathcal{Y}'$.

In addition to coverage, the quality of the retrieved atoms should be examined. Since defensive effectiveness serves as a critical benefit-oriented criterion in the subsequent orchestration process, it is used to identify atoms requiring refinement or replacement. Let $\bar{\alpha}^{(4)}$ denote the average defensive-effectiveness score of atoms in $\mathcal{N}'$, i.e.,

$$\bar{\alpha}^{(4)} = \frac{1}{|\mathcal{N}'|} \sum_{n \in \mathcal{N}'} \alpha_{n,4}. \quad (3)$$

For retrieved atom $n \in \mathcal{N}'$, a binary indicator is introduced

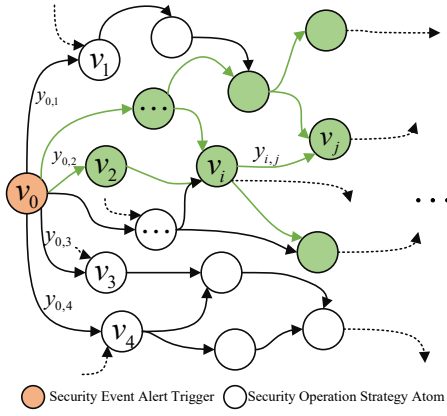


Fig. 2: Graph model for cybersecurity playbook.

as

$$\alpha_n^* = \begin{cases} 1, & \text{if } \alpha_{n,4} < \bar{\alpha}^{(4)}, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Atoms whose defensive-effectiveness scores fall below the average level are marked as candidates for refinement.

To address both uncovered attack categories and low-quality retrieved atoms, the LLM is used to generate a supplementary atom set. Conditioning on the retrieved subset $\mathcal{N}'$, the threat context $\mathcal{C}$, the uncovered category set $\mathcal{Y}_C \setminus \mathcal{Y}'$, and the refinement indicators $\{\alpha_n^*\}_{n \in \mathcal{N}'}$, the LLM initially produces a group of augmented strategy atoms, denoted by

$$\mathcal{H} = \mathcal{G}_{\text{LLM}}(\mathcal{N}', \mathcal{C}, \mathcal{Y}_C \setminus \mathcal{Y}', \{\alpha_n^*\}_{n \in \mathcal{N}'}). \quad (5)$$

The generated atoms are validated before being retained in $\mathcal{H}$. Specifically, atoms with unregistered strategy templates, ungrounded parameters, policy violations, unavailable tools, unmet reversibility requirements, or unsatisfied execution dependencies are removed. For notational simplicity, $\mathcal{H}$ continues to denote the generated set after these validity checks. Each retained atom must further either cover an uncovered attack category or provide an alternative to a retrieved atom marked for refinement, while satisfying the defensive-effectiveness requirement. Accordingly, $\mathcal{H}$ is updated as

$$\mathcal{H} = \left\{ h \in \mathcal{H} \mid \begin{array}{l} \mathcal{Y}_h \cap (\mathcal{Y}_C \setminus \mathcal{Y}') \neq \emptyset \\ \vee (\exists n \in \mathcal{N}' : \alpha_n^* = 1, \mathcal{Y}_h \cap \mathcal{Y}_n \neq \emptyset) \\ \wedge \alpha_{h,4} \geq \bar{\alpha}^{(4)} \end{array} \right\}, \quad (6)$$

where $\alpha_{h,4}$ is the defensive-effectiveness score after validation.

The final candidate atom set used for subsequent playbook construction is

$$\mathcal{N}^* = \{n \in \mathcal{N}' \mid \alpha_n^* = 0\} \cup \mathcal{H}.$$

Thus, only satisfactory retrieved atoms and validated supplementary atoms are retained in the final candidate set.

B. Security Playbook Graph Model

Given the final candidate strategy-atom set $\mathcal{N}^*$, the next step is to organize these atoms into a structured representation for downstream orchestration. Since a security playbook involves

sequential execution and interdependent defensive actions, a graph model is adopted to characterize the execution dependencies among candidate actions.

As illustrated in Fig. 2, we construct a rooted directed acyclic graph (DAG) $G = (\mathcal{V}, \mathcal{E})$, where each candidate strategy atom $n \in \mathcal{N}^*$ is mapped to a node $v_n \in \mathcal{V}$. In addition, a special root node v_0 is introduced to represent the triggering security alert or threat event that initiates the orchestration process. A directed edge $(v_m, v_n) \in \mathcal{E}$ indicates that strategy atom m can be executed before atom n , thereby encoding a feasible precedence or dependency relationship between the two defensive actions. The edges are constructed according to predefined execution stages and dependency relations, with v_0 as the unique source node, so that G is acyclic by construction.

This graph-based representation provides greater flexibility than a fixed linear playbook. Because the candidate atom set contains both retrieved and LLM-generated actions, multiple alternative responses may exist for the same threat context, and each action may connect to different possible predecessors or successors. As a result, the graph captures diverse orchestration paths and enables the subsequent decision process to select an appropriate playbook structure according to operational objectives and environmental constraints.

In the graph model, a security playbook is represented as a rooted subgraph starting at the alert node and including a valid set of executable defensive actions.

Definition 1 (Security Orchestration Strategy). A security playbook is modeled as a rooted directed acyclic graph $Z = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. The root node v_0 must be included in the playbook. Every selected non-root node must have at least one selected predecessor and be reachable from v_0 through a valid execution path. A selected node may have multiple predecessors when multiple prerequisite actions are required. An edge can be selected only if both of its incident nodes are also selected.

The first condition ensures that every playbook starts from the triggering alert. The second requires each selected non-root action to have at least one selected predecessor, thereby preserving executable ordering. The third enforces consistency between selected actions and selected transitions. Since $\mathcal{E}' \subseteq \mathcal{E}$ and G is acyclic by construction, every resulting playbook Z is also acyclic.

The graph model serves as the structural foundation for optimizing cybersecurity orchestration, which helps select an appropriate rooted execution graph that balances defensive effectiveness and operational costs under multiple constraints.

C. Multi-Constraint Security Orchestration Formulation

Based on the security playbook graph, the orchestration problem is to select an appropriate subset of strategy atoms and execution transitions to construct a valid playbook while balancing defensive effectiveness and operational cost. Let $x_n \in \{0, 1\}$ denote whether node v_n is selected, and let $y_{m,n} \in \{0, 1\}$ denote whether the directed edge (v_m, v_n) is selected in the playbook. For each candidate action v_n , let $\mathcal{D}_n$ denote the set of its mandatory predecessor nodes, where $\mathcal{D}_n \subseteq \{v_m \in \mathcal{V} \mid (v_m, v_n) \in \mathcal{E}\}$.

For candidate action v_n , we consider four context-aware state factors: network situation, system configuration, workload condition, and operational history, represented by $g_1(v_n)$, $g_2(v_n)$, $g_3(v_n)$, and $g_4(v_n)$, respectively. The four operational attributes of strategy atom n , namely response time, resource consumption, human involvement, and defensive effectiveness, are represented by $f_1(v_n)$, $f_2(v_n)$, $f_3(v_n)$, and $f_4(v_n)$, respectively. For v_n corresponding to strategy atom n , these attributes satisfy $f_k(v_n) = \alpha_{n,k}$, $k \in \{1, 2, 3, 4\}$. The attribute scores are normalized over the candidate-action nodes $\mathcal{V} \setminus \{v_0\}$. To avoid division by zero when all candidates have the same value for an attribute, a small constant $\delta > 0$ is introduced. The normalized value of $f_k(v_n)$ is expressed as

$$\tilde{f}_k(v_n) = \frac{f_k(v_n) - \min_{v_b \in \mathcal{V} \setminus \{v_0\}} f_k(v_b)}{\max \left\{ \max_{v_c \in \mathcal{V} \setminus \{v_0\}} f_k(v_c) - \min_{v_b \in \mathcal{V} \setminus \{v_0\}} f_k(v_b), \delta \right\}}. \quad (7)$$

Given the current threat context, the LLM produces a threat-severity score ε , which reflects the overall security degradation caused by the incoming threat. ε is normalized to the same scale as the defensive-effectiveness attribute. To ensure that the selected playbook provides sufficient defensive capability, the aggregated normalized defensive-effectiveness score of the selected actions should be no smaller than ε , i.e.,

$$\sum_{v_n \in \mathcal{V} \setminus \{v_0\}} x_n \tilde{f}_4(v_n) \geq \varepsilon. \quad (8)$$

The objective of security orchestration is to maximize net operational benefit by favoring highly effective actions while penalizing excessive response time, resource consumption, and human effort. Let λ_1 , λ_2 , and λ_3 denote the weighting coefficients associated with these three cost factors. The resulting optimization problem is formulated as $\mathcal{P}1$.

$$\mathcal{P}1: \quad \max \sum_{v_n \in \mathcal{V} \setminus \{v_0\}} x_n \left[\tilde{f}_4(v_n) - \sum_{l=1}^3 \lambda_l \tilde{f}_l(v_n) \right] \quad (9a)$$

$$\text{s.t.} \quad \begin{cases} x_0 = 1 \\ \sum_{\substack{v_m \in \mathcal{V} \\ (v_m, v_n) \in \mathcal{E}}} y_{m,n} \geq x_n, \quad \forall v_n \in \mathcal{V} \setminus \{v_0\} \\ y_{m,n} \leq x_m \quad \text{and} \quad y_{m,n} \leq x_n, \quad \forall (v_m, v_n) \in \mathcal{E} \\ y_{m,n} \geq x_n, \quad \forall v_n \in \mathcal{V} \setminus \{v_0\}, \quad \forall v_m \in \mathcal{D}_n \end{cases} \quad (9b) \quad (9c) \quad (9d)$$

Constraint (9a) ensures that alert node v_0 is always selected as the root of the playbook. Constraint (9b) requires each selected non-root action to have at least one selected predecessor. Constraint (9c) enforces consistency between selected nodes and transitions. Constraint (9d) further requires all mandatory dependency edges of a selected action to be included in the playbook. Together with (9c), this ensures that the prerequisite actions are also selected. Since the candidate graph G is constructed as a DAG according to execution ordering and dependency relations, any selected subgraph remains acyclic. Together with (8), the formulation ensures that the generated playbook satisfies both defensive-effectiveness and execution-feasibility requirements.

III. LLM-GUIDED RL FRAMEWORK DESIGN

As illustrated in Fig. 3, the proposed framework integrates LLM-based security reasoning with RL-based sequential decision making for adaptive playbook generation. It consists of four components:

- 1) *Security Situation Analysis*. Threat intelligence and the current playbook state are analyzed to obtain context-aware information for subsequent decision making.
- 2) *State and Action Space Enhancement*. Strategy-atom augmentation expands the candidate action space, while semantic action representations and feasibility masking allow the RL policy to process scenario-dependent candidate sets without changing the network architecture.
- 3) *LLM-Guided RL Decision Making*. The LLM provides action-preference priors to guide policy learning. The LLM prior and RL policy are restricted to the same feasible action set before action selection.
- 4) *Trajectory Feedback and Policy Alignment*. High-quality RL trajectories are fed back to the LLM for lightweight alignment, improving the consistency between LLM guidance and learned orchestration policies.

A. Security Situation Analysis

Security situation analysis provides context-aware information for RL-based orchestration. Threat intelligence collected from the network is provided to the LLM to identify current security risks and assess their severity. Based on the threat report, the LLM generates a threat score ε that quantifies the overall security impact.

During playbook construction, the LLM analyzes the evolving security situation from multiple contextual aspects, including network conditions, system configuration, workload status, and operational history. These context-aware assessments are subsequently incorporated into the RL state representation.

The LLM receives a structured prompt containing the threat description, playbook history, candidate strategy atoms, operational attributes, tool availability, and dependency information. Generated outputs are parsed and validated before entering the RL module. When an augmented atom fails validation, the validation feedback is returned to the LLM for one corrective generation attempt; atoms that remain invalid are discarded. The LLM does not directly determine environmental transitions; these are governed by the registered action templates.

Each validated strategy atom is mapped to a registered SOAR action template that specifies its executable tool or API, required parameters, preconditions, and state-transition logic. Thus, an LLM-generated atom enters the RL environment only when such a valid mapping exists. Its environment transition is executed according to the registered template rather than predicted by the LLM.

B. State and Action Space Enhancement

The playbook generation is formulated as a discrete-time Markov Decision Process (MDP), denoted by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$.

State space $\mathcal{S}$: At decision step t , the state consists of the current playbook graph G_t and four context-aware indicators derived from the security situation analysis. Specifically,

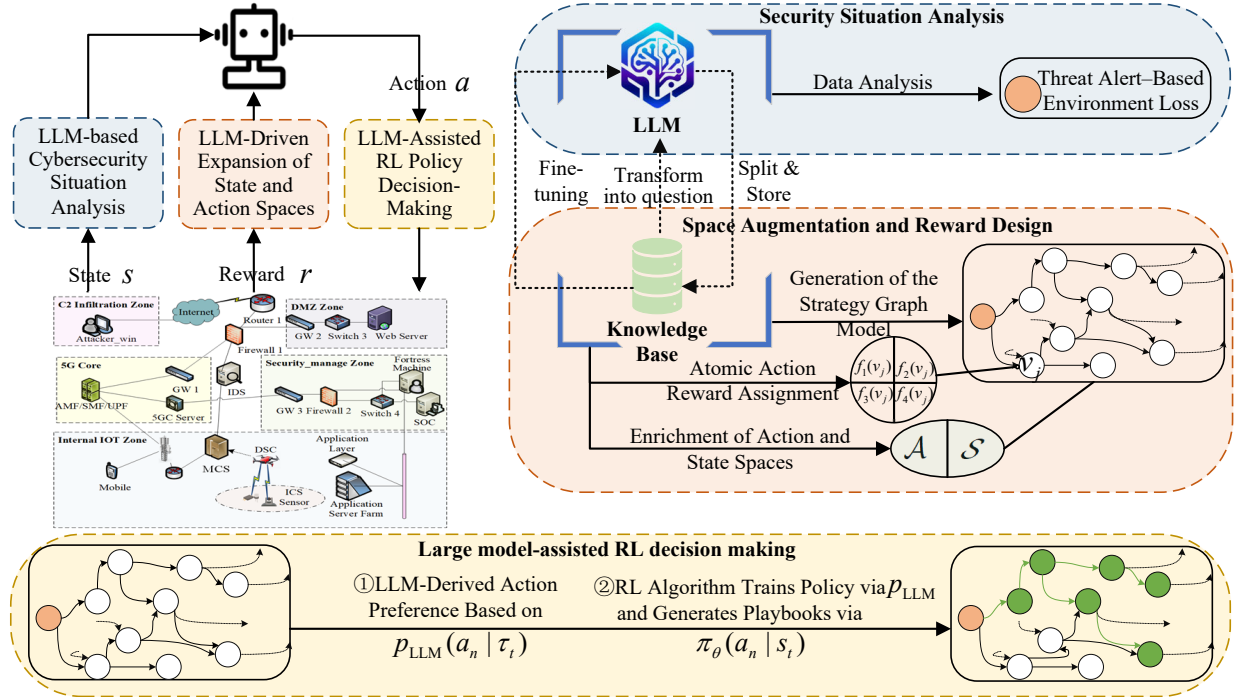


Fig. 3: Overall framework of the proposed LLM-guided RL scheme.

$g_1(v_n)$, $g_2(v_n)$, $g_3(v_n)$, and $g_4(v_n)$ represent the network situation, system configuration, workload condition, and operational history associated with the latest selected atom v_n , respectively. Let $\mathbf{g}_t$ denote the fixed-dimensional representation of G_t . The resulting state is represented as

$$s_t = [\mathbf{g}_t, g_1(v_n), g_2(v_n), g_3(v_n), g_4(v_n)]. \quad (10)$$

Action space $\mathcal{A}$: At each decision step, an action corresponds to selecting a candidate strategy atom from $\mathcal{N}^*$. Let $\mathbf{d}_n$ denote the fixed-dimensional semantic representation of the textual description d_n . The action a_n is represented as

$$\mathbf{e}_n = [\mathbf{d}_n, \tilde{f}_1(v_n), \tilde{f}_2(v_n), \tilde{f}_3(v_n), \tilde{f}_4(v_n)]. \quad (11)$$

The same action representation and policy-scoring function are shared by all candidate atoms. Therefore, the policy does not require a fixed output slot for each atom and can accommodate candidate sets of different sizes across threat scenarios.

At decision step t , a binary mask $m_t(n) \in \{0, 1\}$ is applied to each candidate atom. An atom is retained only when it has not already been selected, its required tools are available, its mandatory dependencies are satisfied, and its execution complies with the policy and ordering constraints. The resulting feasible action set is

$$\mathcal{A}_t = \{a_n \mid n \in \mathcal{N}^*, m_t(n) = 1\}. \quad (12)$$

State transition $\mathcal{P}$: The transition model characterizes how the orchestration state evolves after action a_t is executed. Each retrieved or generated atom admitted into $\mathcal{N}^*$ is associated with a registered SOAR action template containing its executable operation, preconditions, and transition logic. After a_t is selected, the template is executed by the simulator or

security tool, and the resulting environment state is used to construct s_{t+1} . Hence, an LLM-generated atom without a valid transition mapping cannot enter the RL action space.

Reward $\mathcal{R}$: The reward measures the contribution of the selected action to the orchestration objective.

C. LLM-Guided RL Decision Making

Given the scenario-dependent candidate set $\mathcal{N}^*$, the RL policy evaluates the current action set $\mathcal{A}_t$ through a shared state-action scoring function. For each $a_n \in \mathcal{A}_t$, the corresponding policy score is defined as

$$u_\theta(s_t, a_n) = F_\theta([s_t, \mathbf{e}_n]), \quad (13)$$

where $F_\theta(\cdot)$ denotes the shared policy-scoring network. The RL action distribution is given by

$$\pi_\theta(a_n \mid s_t) = \frac{\exp(u_\theta(s_t, a_n))}{\sum_{a_j \in \mathcal{A}_t} \exp(u_\theta(s_t, a_j))}. \quad (14)$$

Let $\tau_t = \{(s_0, a_0, r_0), \dots, (s_{t-1}, a_{t-1}, r_{t-1})\}$ denote the interaction history before step t . Based on the current state and interaction history, the LLM assigns a preference score $u_{\text{LLM},t}(a_n)$ to each action in $\mathcal{A}_t$. The preference distribution is

$$p_{\text{LLM}}(a_n \mid \tau_t) = \frac{\exp(u_{\text{LLM},t}(a_n))}{\sum_{a_j \in \mathcal{A}_t} \exp(u_{\text{LLM},t}(a_j))}. \quad (15)$$

In the complete configuration, $\mathcal{A}_t$ is obtained by applying the feasibility mask defined in (12). Consequently, π_θ and

p_{LLM} are normalized over the same feasible action support. They are combined to form the guided action distribution

$$q_{\theta}(a_n | s_t, \tau_t) = (1 - \beta)\pi_{\theta}(a_n | s_t) + \beta p_{\text{LLM}}(a_n | \tau_t), \quad (16)$$

where $\beta \in [0, 1]$ controls the contribution of LLM guidance. Thus, infeasible actions receive no probability mass when masking is enabled. The framework also supports ablation configurations by disabling feasibility masking or setting $\beta = 0$ to remove the LLM prior.

The immediate reward follows the operational objective formulated in $\mathcal{P}1$:

$$r_t = \tilde{f}_4(a_t) - \sum_{l=1}^3 \lambda_l \tilde{f}_l(a_t). \quad (17)$$

For trajectory τ , the cumulative discounted return is

$$\mathcal{R}(\tau) = \sum_{t=0}^T \gamma^t r_t, \quad (18)$$

where $\gamma \in [0, 1]$ is the discount factor. The general RL objective is expressed as

$$J(\theta) = \mathbb{E}_{\tau \sim q_{\theta}} [\mathcal{R}(\tau)]. \quad (19)$$

The state-action representation and LLM-guided decision mechanism are independent of a specific RL optimizer, while feasibility masking and LLM priors can be enabled or disabled to form different policy configurations. Specific RL implementations and corresponding ablation settings are introduced in the experimental section.

Algorithm 1 summarizes the overall training procedure. For each threat scenario, the validated set $\mathcal{N}^*$ provides the candidate actions. At each decision step, $\mathcal{A}_t$ is updated according to the adopted action-space configuration, after which the RL policy and, when enabled, the LLM prior are evaluated to obtain q_{θ} . The selected action is executed through its registered template, and the resulting transition and reward are used for subsequent policy optimization.

D. LLM-RL Policy Alignment via Fine-Tuning

After RL policy training, high-quality orchestration trajectories are used to align the LLM with the learned decision policy. The interaction trajectories are transformed into structured supervision containing the current state, candidate actions, and the action selected by the learned policy.

1) *Trajectory-based LoRA alignment.* To avoid full-parameter fine-tuning, Low-Rank Adaptation (LoRA) is adopted. Let $W \in \mathbb{R}^{d \times d}$ denote a pretrained weight matrix. Its adapted form is

$$W' = W + \Delta W, \quad \Delta W = BA, \quad (20)$$

where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ are trainable low-rank matrices with $r \ll d$.

The collected trajectories are ranked according to their cumulative return $\mathcal{R}(\tau)$, and trajectories satisfying $\mathcal{R}(\tau) \geq \Omega^+$ are retained. For each retained decision step, the state s_t and current action set $\mathcal{A}_t$ are mapped into the structured prompt used for alignment. Let $\Gamma(\cdot)$ denote the structured prompt

Algorithm 1: LLM-Guided RL Policy Training

Input: Threat context $\mathcal{C}$, candidate set $\mathcal{N}^*$, maximum horizon T , mixing weight β ;
Output: Optimized RL policy π_{θ}^* ;

- 1 Initialize RL parameters θ ;
- 2 **while** the policy has not converged **do**
- 3 Initialize state s_0 and trajectory τ ;
- 4 **for** $t \leftarrow 0$ **to** T **do**
- 5 Construct current action set $\mathcal{A}_t$ according to the adopted configuration;
- 6 Compute action representations $\{\mathbf{e}_n\}$;
- 7 Compute RL policy π_{θ} over $\mathcal{A}_t$;
- 8 **if** LLM guidance is enabled **then**
- 9 Compute p_{LLM} over $\mathcal{A}_t$;
- 10 Obtain q_{θ} using Eq. (16);
- 11 Sample action $a_t \sim q_{\theta}$;
- 12 Execute the registered action template and observe s_{t+1} ;
- 13 Compute r_t and update trajectory τ ;
- 14 **if** the orchestration objective is satisfied **then**
- 15 **break**;
- 16 Update RL according to the optimizer and $\mathcal{R}(\tau)$;
- 17 **return** π_{θ}^* ;

mapping of the current state and candidate actions, including their descriptions, operational attributes, and dependency information. If $a_t = a_n$ is selected by the learned policy, the corresponding supervision pair is

$$(\mathbf{x}_t, \mathbf{y}_t) = (\Gamma(s_t, \mathcal{A}_t), d_n), \quad a_t = a_n, \quad (21)$$

where d_n provides the textual target associated with the selected strategy atom. The resulting high-return dataset is

$$\mathcal{D}^+ = \{(\mathbf{x}_t, \mathbf{y}_t) \mid \mathcal{R}(\tau) \geq \Omega^+\}. \quad (22)$$

The LoRA parameters are optimized using the autoregressive negative log-likelihood

$$\mathcal{L}_{\text{LLM}} = - \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}^+} \sum_{j=1}^{|\mathbf{y}|} \log p_{\theta'_{\text{LLM}}}(y_j \mid \mathbf{x}, y_{<j}), \quad (23)$$

where θ'_{LLM} denotes the LoRA-adapted LLM parameters.

The default alignment uses high-return trajectories only. A separate configuration incorporating corrective samples derived from low-return trajectories is evaluated experimentally to determine whether such feedback provides complementary information or introduces additional noise.

2) *Stability-aware parameter merging.* To stabilize LoRA adaptation, gradient clipping is applied as

$$g'_{\Delta W} = \frac{g_{\Delta W}}{\max \left\{ 1, \frac{\|g_{\Delta W}\|_2}{c} \right\}}, \quad (24)$$

where $g_{\Delta W}$ denotes the gradient of the LoRA update and c is the clipping threshold.

After alignment, the adapted parameters are merged with

the original LLM parameters:

$$\theta''_{LLM} = \omega \theta'_{LLM} + (1 - \omega) \theta_{LLM}, \quad (25)$$

where θ_{LLM} denotes the original parameters and ω controls the contribution of the aligned model. The fusion coefficient is determined by

$$\omega = \frac{\exp(-\mathcal{L}_{LLM})}{\exp(-\mathcal{L}_{LLM}) + 1}. \quad (26)$$

Algorithm 2 summarizes the LLM alignment procedure using high-quality RL trajectories. High-return trajectories are first converted into structured supervision pairs, after which the LoRA parameters are optimized with gradient clipping. The adapted parameters are then merged with the original model to obtain the aligned LLM.

Algorithm 2: LLM–RL Policy Alignment

Input: RL trajectories, return threshold Ω^+ , clipping threshold c , LoRA learning rate ϑ , original LLM parameters θ_{LLM} , maximum epochs ψ ;
Output: Aligned LLM parameters θ''_{LLM} ;
1 Select trajectories satisfying $\mathcal{R}(\tau) \geq \Omega^+$;
2 Construct supervision pairs $(\mathbf{x}_t, \mathbf{y}_t)$;
3 Construct high-return dataset $\mathcal{D}^+$;
4 Initialize low-rank matrices A and B ;
5 **for** $e \leftarrow 1$ **to** ψ **do**
6 Compute $\mathcal{L}_{LLM}$ using Eq. (23);
7 Compute and clip the LoRA gradients;
8 Update A and B using learning rate ϑ ;
9 Obtain adapted parameters θ'_{LLM} ;
10 Compute ω using Eq. (26);
11 Merge θ'_{LLM} and θ_{LLM} using Eq. (25);
12 **return** θ''_{LLM} ;

IV. EXPERIMENTAL PREPARATION

A. Experimental Environment Setup

To validate the proposed framework under complex and heterogeneous conditions, we construct a 5G-IoT enterprise security scenario. As shown in the middle-left part of Fig. 3, the experimental environment is segmented by perimeter and internal firewalls into multiple security zones with different trust levels and operational functions.

- **C2 Penetration Zone:** Linux and Windows attacker hosts are deployed to emulate external threat sources.
- **DMZ Zone:** Externally exposed services, including database, Web, and FTP servers, are deployed as the boundary between the Internet and the internal network.
- **5G Core Network Zone:** Core 5G functions and supporting servers handle signaling, session management, and data-plane operations.
- **Security Management Zone:** Bastion hosts and SOC platforms provide centralized monitoring, event correlation, and policy enforcement.
- **Internal Application and IoT Zone:** PCs, mobile devices, sensors, IoT endpoints, and application servers constitute the main protected target space.

To increase the diversity and statistical significance of the evaluation, the vulnerability set is expanded to include 10 real-world CVE scenarios distributed across the DMZ, 5G Core, security management, internal IoT, and application-layer environments. As summarized in Table II, these vulnerabilities span heterogeneous attack surfaces and exploitation characteristics, including remote code execution, command injection, denial-of-service, and unauthorized access to information.

Among the ten CVE scenarios, eight are included during policy training and treated as *Seen* scenarios, while the remaining two are held out and used exclusively as *Unseen* scenarios to evaluate generalization to previously unobserved threat characteristics. Operational-load variations are further introduced to emulate changes in the execution environment, resulting in four evaluation conditions that jointly assess robustness under threat-distribution shifts and changing operational environments:

- **Seen Normal:** seen CVE scenarios under standard operational conditions;
- **Seen High:** seen CVE scenarios under increased resource contention, execution latency, tool unavailability, and action-failure probability;
- **Unseen Normal:** held-out CVE scenarios under standard operational conditions;
- **Unseen High:** held-out CVE scenarios under high operational load.

B. Knowledge Base Construction

To facilitate strategy-atom retrieval and security playbook generation, we construct a security knowledge base using mitigation and defensive-control data from the MITRE ATT&CK framework¹. The repository covers three representative security domains: Enterprise, Mobile, and Industrial Control Systems (ICS), as summarized in Table III. Within ATT&CK, each mitigation category may be associated with multiple defensive techniques. The collected techniques are therefore organized by mitigation identifier and maintained separately across domains, providing a structured and extensible repository for strategy-atom matching and augmentation.

Each strategy atom contains priority metadata and four operational attributes corresponding to the formulation in Section II: response time, resource consumption, human involvement, and defensive effectiveness. Priority serves as auxiliary information for strategy organization, while the four attributes correspond to f_1 – f_4 and are used in candidate evaluation and RL optimization. For retrieved atoms, these attributes are initialized using registered tool metadata, historical execution records, and rule-based annotations. For newly generated atoms without historical records, the LLM first provides provisional scores according to a predefined rubric. Response-time and resource-consumption scores are subsequently calibrated using tool profiles or sandbox measurements, whereas human-involvement and defensive-effectiveness scores are checked against predefined operational rules and expert annotations. The validated values are then normalized as $\tilde{f}_1$ – $\tilde{f}_4$ for candidate filtering and reward calculation.

¹<https://attack.mitre.org/>

TABLE II: CVE-based vulnerability scenarios used in the experiments

Scenario	Zone	CVE ID	CVSS	Vulnerability Description	Published Date
S01	DMZ	CVE-2023-34362	9.8	SQL injection in MOVEit Transfer allowing unauthorized database access	02/06/2023
S02	DMZ	CVE-2021-44228	10.0	JNDI injection in Apache Log4j2 allowing remote code execution	10/12/2021
S03	5G Core	CVE-2024-38063	9.8	Windows TCP/IP vulnerability allowing unauthenticated remote code execution	13/08/2024
S04	5G Core	CVE-2023-44487	7.5	HTTP/2 Rapid Reset causing resource exhaustion and denial of service	10/10/2023
S05	Security Mgmt	CVE-2022-30190	7.8	MSDT URL-protocol exploitation allowing arbitrary code execution	01/06/2022
S06	Security Mgmt	CVE-2024-3400	10.0	Command injection in PAN-OS GlobalProtect allowing root-level code execution	12/04/2024
S07	Internal IoT	CVE-2023-1389	8.8	Command injection in TP-Link Archer AX21 allowing root-level execution	15/03/2023
S08	Internal IoT	CVE-2024-3273	7.3	Command injection in end-of-life D-Link NAS devices	03/04/2024
S09	Application Layer	CVE-2024-6387	8.1	OpenSSH signal-handler race condition potentially allowing unauthenticated RCE	01/07/2024
S10	Application Layer	CVE-2024-23897	9.8	Jenkins CLI argument expansion allowing unauthenticated arbitrary file read	24/01/2024

Note: The reported scores are CVSS v3.1 base scores displayed by the National Vulnerability Database. When an NVD assessment is unavailable, the CNA score displayed by NVD is adopted.

TABLE III: Knowledge base categories and distribution

Scenario	Mitigations	Techniques
Mobile	13	153
ICS	52	331
Enterprise	43	1296

C. System Implementation

The system is implemented through a unified pipeline comprising semantic retrieval, LLM-assisted strategy augmentation, strategy validation, and RL-based playbook construction.

- 1) **Data structuring and semantic retrieval.** Preprocessed security strategies are organized into scenario-specific atomic units and stored in structured files. For hierarchical MITRE ATT&CK entries, the complete category path is retained as metadata, while the *Description* field provides the primary semantic content. Atom descriptions are encoded using *sentence-transformers/all-mpnet-base-v2* and indexed in FAISS, along with their metadata, for similarity-based retrieval.
- 2) **LLM-assisted strategy augmentation.** Retrieved strategy atoms are incorporated into a structured prompt together with threat and operational information. Qwen2.5-7B² and 14B³ are used as the base LLM in the dynamic-action and trajectory-alignment experiments. Deterministic decoding is adopted with a temperature of 0.0, top- p of 1.0, and a maximum output length of 256 tokens. The same configuration is used across the compared LLM-based variants. The LLM identifies insufficiently covered threat categories, generates supplementary strategy atoms, and provides action-preference scores for subsequent RL decision-making.

- 3) **Strategy validation and executable mapping.** Each generated atom is validated before entering $\mathcal{N}^*$ by checking required fields, scenario consistency, tool availability, policy and approval constraints, execution dependencies, and duplicated actions. Failed atoms receive one corrective generation attempt using the validation feedback and are discarded if they remain invalid. Accepted atoms are mapped to registered SOAR action templates specifying executable operations, required tools, preconditions, and transition logic. Only atoms with valid executable mappings are admitted to the RL environment.
- 4) **RL-based playbook construction.** The validated atoms form the scenario-dependent candidate set $\mathcal{N}^*$. Their semantic representations and operational attributes are used to construct the state-action representations. At each decision step, feasibility masking determines the valid action set, over which the RL policy and, when enabled, the LLM prior select the next executable strategy atom.

D. Comparison and Ablation Settings

We compare the proposed framework with two representative baselines for security orchestration:

- **Baseline-1 (CCPO):** A constrained-policy optimization method that regulates playbook generation within a limited behavioral policy space [35].
- **Baseline-2 (SecCode):** A three-stage orchestration workflow that incrementally constructs tree-structured playbooks through initial generation, vulnerability-aware refinement, and logic cross-validation [36].

We construct a series of variants to evaluate the effects of knowledge enhancement, ToT reasoning, LLM scale, RL optimization, LLM-RL alignment, and dynamic-action handling, as summarized in Table IV. Three representative RL algorithms are considered: Policy Gradient (PG) [37], Q-learning

²<https://huggingface.co/Qwen/Qwen2.5-7B>

³<https://huggingface.co/Qwen/Qwen2.5-14B>

TABLE IV: Comparison and ablation settings of the proposed framework

Name	Knowledge Enhancement			Base LLM		RL Algorithm			Dynamic Action			LoRA Fine-tuning
	Static	Dynamic	ToT	7B	14B	QL	PPO	PG	Sem. Rep.	Mask	LLM-Prior	
Proposed-1	✗	✓	✓	✓	✗	✗	✗	✓	–	–	–	✓
Proposed-2	✓	✗	✗	✓	✗	✗	✗	✓	–	–	–	✓
Proposed-3	✗	✓	✗	✓	✗	✗	✗	✓	–	–	–	✓
Proposed-4	✗	✗	✓	✓	✗	✗	✗	✓	–	–	–	✓
Proposed-5	✗	✓	✓	✗	✓	✗	✗	✓	–	–	–	✓
Proposed-6	✗	✓	✓	✓	✗	✓	✗	✗	–	–	–	✓
Proposed-7	✗	✓	✓	✓	✗	✗	✗	✓	–	–	–	✗
Proposed-8	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓	✓	✗
Proposed-9	✗	✗	✗	–	–	✗	✓	✗	✓	✓	✗	✗

(QL) [38], and Proximal Policy Optimization (PPO) [39]. PG and QL are used to compare policy- and value-based optimization, respectively, while PPO serves as the common optimizer in the dynamic-action experiments.

Proposed-1 is the main configuration, combining dynamic knowledge augmentation, ToT reasoning, a 7B LLM, PG optimization, and LoRA-based alignment. Proposed-2–Proposed-4 examine different knowledge-enhancement and ToT settings, Proposed-5 evaluates LLM scale, Proposed-6 replaces PG with QL, and Proposed-7 removes LoRA-based alignment. Proposed-8 and Proposed-9 evaluate the dynamic-action mechanism under PPO: Proposed-8 incorporates semantic action representation, feasibility masking, and the LLM prior, whereas Proposed-9 removes only the LLM prior by setting $\beta = 0$. A local variant, Proposed-8 (NoMask), disables only feasibility masking and is used specifically in the masking and candidate-set-size experiments; therefore, it is not listed separately in Table IV.

E. Evaluation Metrics

The evaluation considers playbook quality, operational effectiveness, and computational performance. Additional metrics are introduced to quantify the validity of LLM-generated atoms and the robustness of the dynamic action space.

1) *Strategy-atom validity*. To quantify hallucinated or operationally invalid outputs, the LLM is instructed to generate 200 strategy atoms, which are evaluated using the validation and corrective-generation procedure described above. The following metrics are reported:

- **First-pass rate**: the proportion of generated atoms that satisfy all validation requirements on the first attempt;
- **Final acceptance rate**: the proportion of atoms that are valid after the complete validation and corrective-generation procedure;
- **Retry recovery rate**: the proportion of initially invalid atoms corrected after receiving validation feedback;
- **Pre-validation invalid rate**: the proportion of raw LLM-generated atoms rejected during the initial validation;
- **Post-validation invalid rate**: the proportion of atoms that remain invalid after corrective generation.

2) *Effectiveness and robustness of orchestration*. Defense coverage rate measures the proportion of required defensive categories covered by the generated playbook. Playbook completion rate is the proportion of episodes in which the required

defensive coverage and effectiveness criteria are met. Operational failure rate measures the proportion of episodes that fail to achieve the orchestration objective because of invalid execution, unmet dependencies, insufficient threat coverage, or other operational failures. These metrics are further evaluated across Seen Normal, Seen High, Unseen Normal, and Unseen High conditions to examine robustness under both threat-distribution and execution-environment changes.

3) *Playbook generation duration*. Playbook generation duration measures the decision time required by the RL orchestration stage to generate a complete playbook. It includes policy-network inference, action-set processing, feasibility masking when enabled, and action sampling. The preceding LLM-prior scoring time is excluded so that the computational cost of the RL decision mechanism can be evaluated independently.

4) *Textual playbook quality*. To evaluate the semantic and structural quality of the generated playbooks, five reference playbooks are constructed for representative attack scenarios. GPT-4 is used to assist the initial organization and refinement, while the final reference playbooks are reviewed and validated by security experts before evaluation. BLEU [40] is used to measure lexical precision and n-gram consistency between generated playbooks and reference workflows, while ROUGE [41] evaluates recall-oriented structural similarity and sequence overlap. Specifically, BLEU-1, BLEU-2, and BLEU-4 assess unigram- and phrase-level consistency at different n-gram orders, whereas ROUGE-N and ROUGE-L measure n-gram recall and longest-common-subsequence similarity, respectively. GPT-4 further provides complementary assessments of completeness, interpretability, and scenario relevance on a five-point scale. The GPT-based score is used only as a complementary qualitative assessment, not as ground-truth supervision. The expert-validated references and operational execution metrics provide the primary basis for evaluating the reliability of the generated security playbooks.

V. EXPERIMENTAL RESULTS ANALYSIS

The experiments are organized around the four research questions introduced above. Subsections V-A and V-C address **RQ1** by evaluating the relevance, compactness, and operational validity of generated strategy atoms. Subsection V-D addresses **RQ2** by examining semantic action representation, feasibility masking, and LLM guidance under varying candidate-set sizes, and further addresses **RQ3** by evaluating

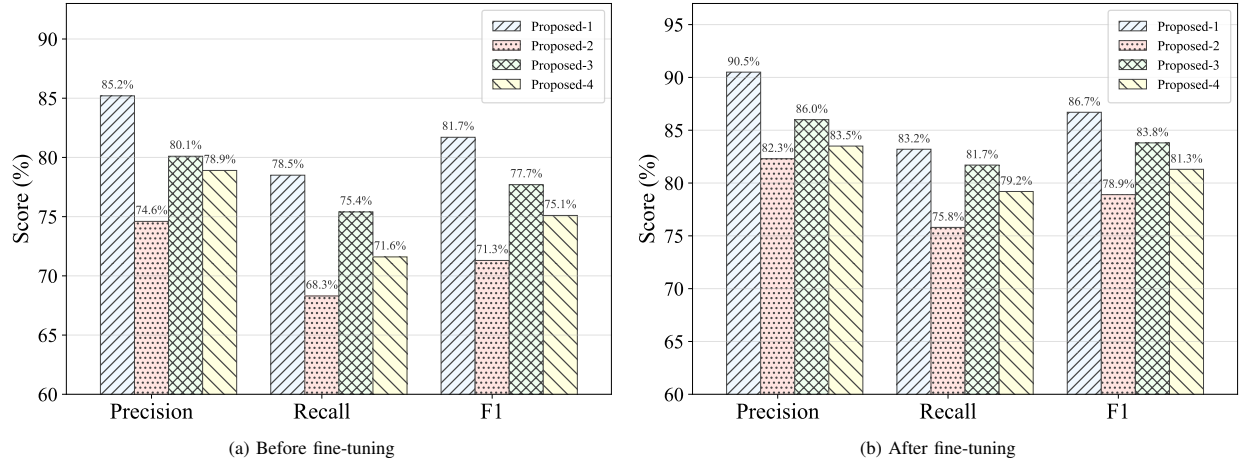


Fig. 4: Playbook Atom Score.

robustness under Seen/Unseen threats and Normal/High operational loads. Subsection V-E addresses **RQ4** by studying how different RL-trajectory feedback strategies affect the LLM action prior. Complementary experiments in Subsections V-B and V-F analyze the effects of different RL algorithms and LLM configurations on orchestration performance.

A. Effect of Knowledge Enhancement on Playbook Generation

We evaluate how different knowledge-enhancement mechanisms affect playbook generation in terms of precision, recall, F1-score, and the average number of generated strategy atoms.

From Fig. 4(a), before alignment fine-tuning, Proposed-1 achieves the best overall performance, with a precision of 85.2%, a recall of 78.5%, and an F1-score of 81.7%, while generating an average of 14.7 strategy atoms. In contrast, Proposed-2, which relies on a static knowledge base without dynamic augmentation or ToT reasoning, produces more atoms but achieves lower precision and recall, indicating that retrieving more candidate actions does not necessarily improve playbook quality. Proposed-3 and Proposed-4 further reveal the individual effects of dynamic knowledge augmentation and ToT reasoning. Both outperform Proposed-2 but remain below Proposed-1, suggesting that dynamic augmentation and structured reasoning provide complementary benefits in improving candidate relevance and reducing redundant actions.

Fig. 4(b) shows that alignment fine-tuning further improves all variants. The F1-score of Proposed-1 increases from 81.7% to 86.7%, while the average number of generated atoms decreases from 14.7 to 12.6. Similar trends are observed for Proposed-2–Proposed-4. These results indicate that alignment fine-tuning improves the selectivity of strategy-atom generation, increasing semantic relevance while suppressing redundant or low-utility actions.

B. Performance Analysis Under Different RL Algorithms

This subsection compares different RL algorithms from three perspectives: training convergence, reward stability after convergence, and the operational failure rate of generated playbooks in the experimental testbed.

Fig. 5 presents the reward trajectories and reward-density distributions. As shown in Fig. 5(a), all methods exhibit an overall upward trend during training, indicating progressive policy improvement through environment interaction. Their convergence behaviors, however, differ substantially. Proposed-1 and Proposed-6 converge more steadily than the other variants and reach final reward values above 70 around the 50th iteration, suggesting that both PG- and QL-based optimization can provide effective policy improvement when combined with dynamic knowledge augmentation and LLM guidance. Proposed-7 exhibits larger oscillations and converges to a slightly lower reward level, indicating that removing LLM–RL alignment weakens training stability. Baseline-1 performs worst, with stronger fluctuations and a final reward plateau of approximately 60–62, reflecting its limited adaptability under dynamic orchestration conditions. The reward-density distributions in Fig. 5(b) support these observations. Proposed-1 exhibits the most concentrated and right-shifted distribution, reflecting both higher final rewards and lower variance after convergence. Proposed-6 shows a similar but slightly broader distribution, whereas Proposed-7 and Baseline-1 exhibit broader and more left-shifted distributions, indicating less stable convergence and lower-quality final policies.

The practical execution results in Fig. 6 show a consistent trend. Proposed-1 achieves the lowest operational failure rate, while Proposed-6 performs slightly worse but still clearly outperforms the baselines. Proposed-7 again underperforms Proposed-1, indicating that alignment fine-tuning contributes not only to training stability but also to the execution reliability of generated orchestration policies. Overall, the policy-gradient configuration adopted by Proposed-1 provides the best balance among convergence stability, final reward quality, and operational reliability.

The following experiments further examine the practical reliability and adaptability of the proposed framework. They focus on three complementary aspects: the operational validity of LLM-generated strategy atoms, the ability to handle varying discrete candidate-action sets, and the robustness of LLM-guided decision-making under threat-distribution shifts and changing execution conditions.

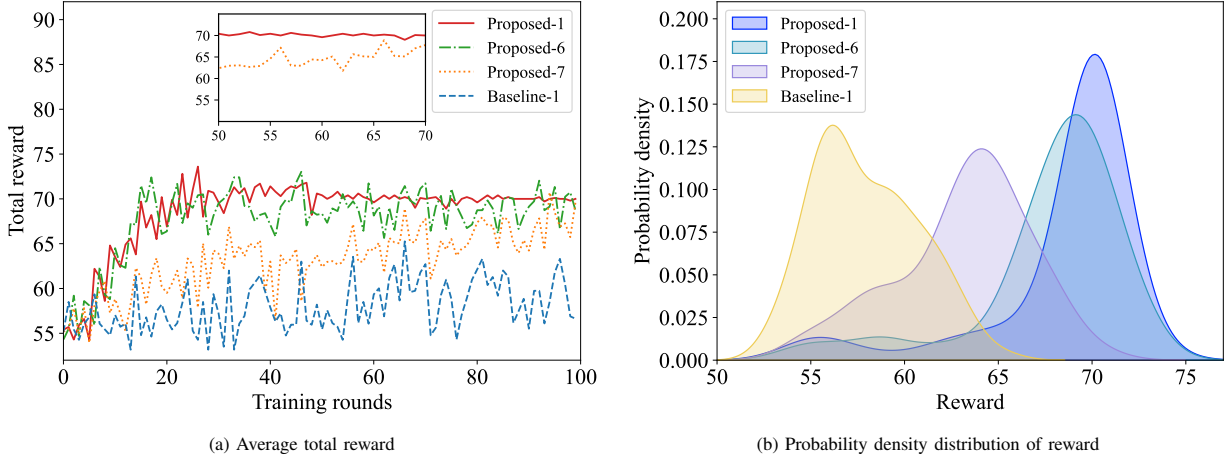


Fig. 5: Reward value and probability density distribution.

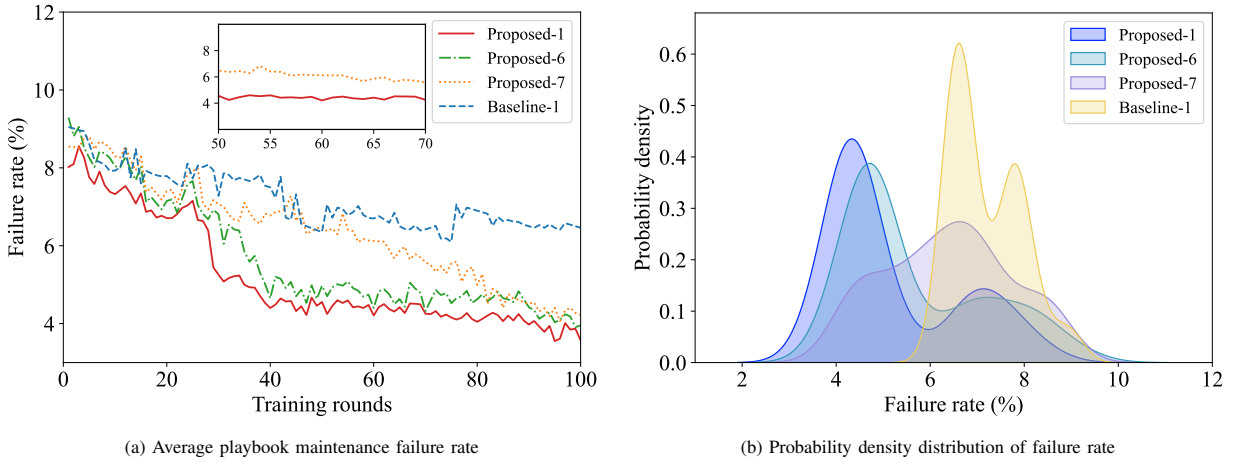


Fig. 6: Playbook maintenance failure rate and probability density distribution.

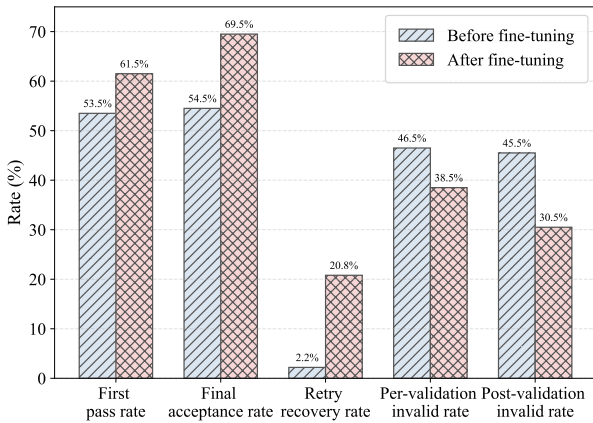


Fig. 7: Validity of LLM-generated strategy atoms.

C. Reliability of LLM-Generated Strategy Atoms

This subsection evaluates whether LLM-generated strategy atoms can reliably satisfy the operational requirements of the orchestration pipeline and examines the effectiveness of validation and corrective generation. We use five validity-

oriented metrics: first-pass rate, final acceptance rate, retry recovery rate, pre-validation invalid rate, and post-validation invalid rate. A total of 200 strategy atoms are generated under the same task instructions and subjected to the validation procedure described in Section IV. Atoms that fail the initial validation are returned to the LLM along with validation feedback for a single corrective generation attempt.

As shown in Fig. 7, LLM fine-tuning increases the first-pass rate from 53.5% to 61.5% and the final acceptance rate from 54.5% to 69.5%. The retry recovery rate increases from 2.2% to 20.8%, while the pre-validation invalid rate decreases from 46.5% to 38.5% and the post-validation invalid rate decreases from 45.5% to 30.5%. These results indicate that the fine-tuned LLM generates more valid strategy atoms initially and responds more effectively to explicit validation feedback. Nevertheless, 30.5% of the generated atoms remain invalid after corrective generation, showing that LLM adaptation alone cannot guarantee operational validity. The validation thus remains necessary to prevent unregistered, policy-incompatible, or otherwise infeasible actions from entering $\mathcal{N}^*$.

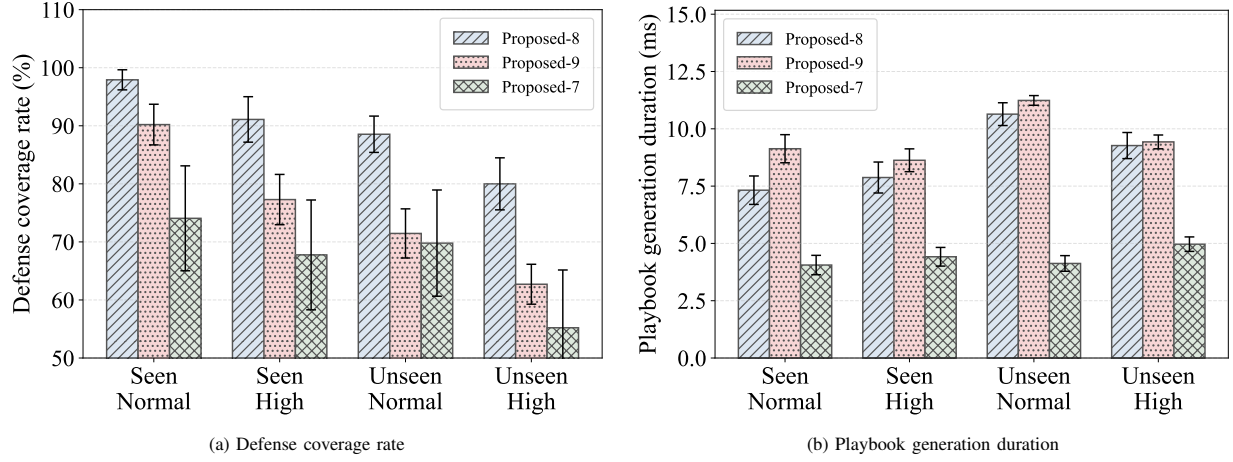


Fig. 8: Performance under different threat distributions and operational loads.

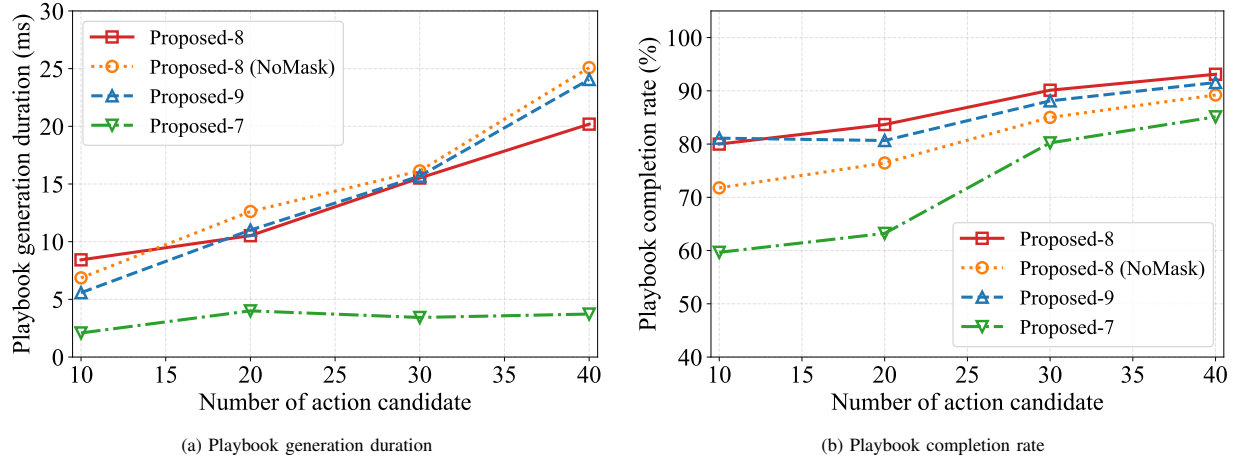


Fig. 9: Performance with different candidate-set sizes.

D. Performance Under Varying Action Spaces

This subsection evaluates the dynamic-action mechanism from two complementary perspectives: robustness under varying threat and operational conditions, and scalability as the candidate set size increases. PPO is adopted as the common optimizer for these experiments to evaluate the effects of feasibility masking and LLM guidance under a consistent RL setting. Proposed-8, Proposed-8 (NoMask), and Proposed-9 are used as controlled configurations to isolate the contributions of feasibility masking and the LLM-prior.

1) *Generalization under threat and load variations.* Fig. 8(a) compares defense coverage under Seen Normal, Seen High, Unseen Normal, and Unseen High conditions. Proposed-8 achieves the highest coverage, decreasing from 97.9% under Seen Normal to 80.0% under Unseen High. Proposed-9, which removes the LLM prior while retaining the same PPO optimizer and feasibility mask, decreases from 90.1% to 62.7%. Accordingly, the performance gap increases from 7.8 to 17.3 percentage points as the threat and execution conditions become more challenging. Because both configurations retain feasibility masking, this widening gap primarily reflects the growing contribution of LLM guidance when the RL policy

encounters threat characteristics that fall outside the training distribution. Fig. 8(b) reports the corresponding playbook generation duration. The RL decision stage remains within the millisecond range across all settings, with the maximum duration below 11.3 ms. Proposed-8 requires 7.3 ms under Seen Normal and 10.6 ms under Unseen Normal, compared with 9.1 ms and 11.2 ms for Proposed-9, respectively. These results indicate that incorporating the LLM prior does not introduce substantial additional overhead into the RL decision stage once the prior scores are available.

2) *Effect of candidate-set size and feasibility masking.* The number of candidate actions is varied among 10, 20, 30, and 40 to evaluate scalability under varying discrete action sets. All three configurations use the same semantic action representation and PPO optimizer: Proposed-8 (NoMask) removes feasibility masking, while Proposed-9 removes the LLM prior from Proposed-8. As shown in Fig. 9(a), generation duration increases gradually with the number of candidate actions. With 40 actions, Proposed-8 requires 20.2 ms, compared with 25.1 ms for Proposed-8 (NoMask) and 24.1 ms for Proposed-9. This result indicates that filtering infeasible actions becomes increasingly beneficial as the candidate set expands,

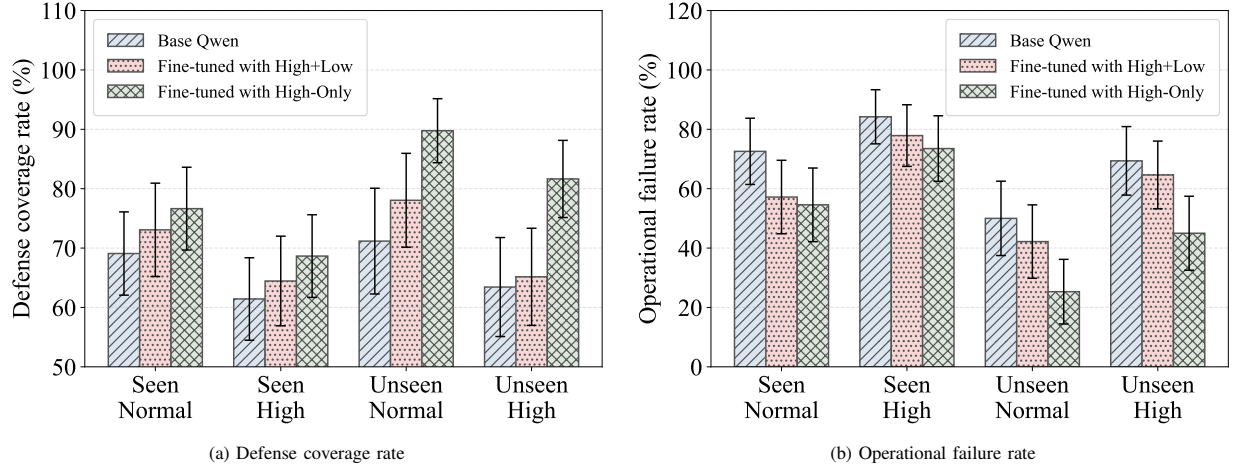


Fig. 10: Effect of different RL-trajectory alignment strategies on LLM guidance.

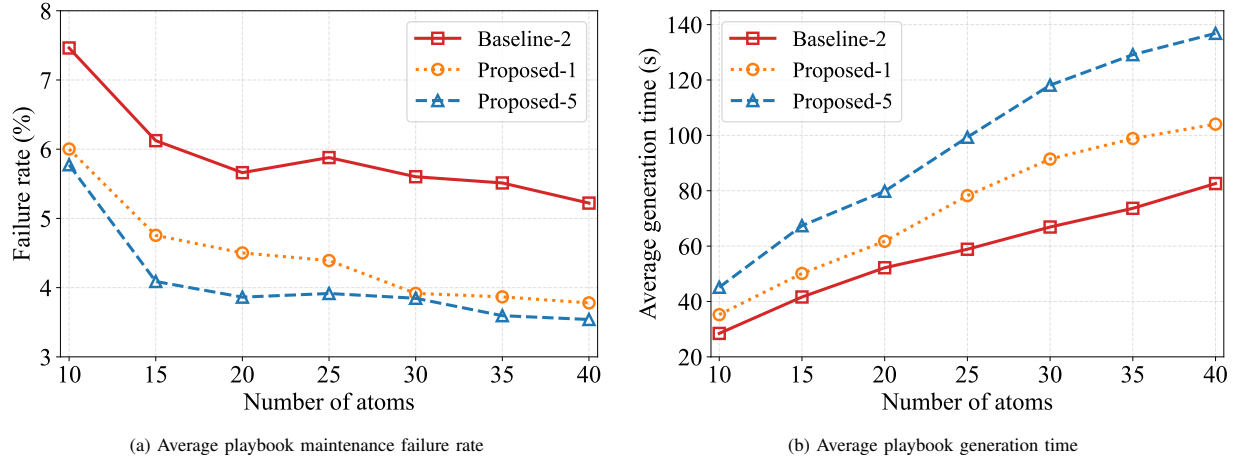


Fig. 11: Maintenance failure rate and playbook generation time with different atomic numbers.

while the shared state-action scoring mechanism accommodates different candidate-set sizes without resizing the policy network. Fig. 9(b) shows a similar advantage in playbook completion. Proposed-8 achieves the highest completion rate with 20, 30, and 40 candidate actions, reaching 93.2% with 40 actions. Its advantage over Proposed-8 (NoMask) confirms the contribution of feasibility masking, while the widening gap over Proposed-9 indicates that LLM guidance becomes more beneficial as the candidate set grows.

E. Effect of RL-Trajectory Alignment on LLM Guidance

This experiment examines whether RL trajectory feedback improves the LLM action prior rather than merely reinforcing exploitation patterns already learned by the RL policy. To isolate this effect, the trained RL policy is fixed throughout the experiment, and only the LLM prior is varied. Base uses the original Qwen2.5-7B without trajectory-based adaptation; High-Only fine-tunes Qwen2.5-7B with LoRA using positive supervision derived from high-return trajectories; and High+Low additionally incorporates corrective samples constructed from low-return trajectories. The resulting LLMs are used as alternative action priors under the same RL policy.

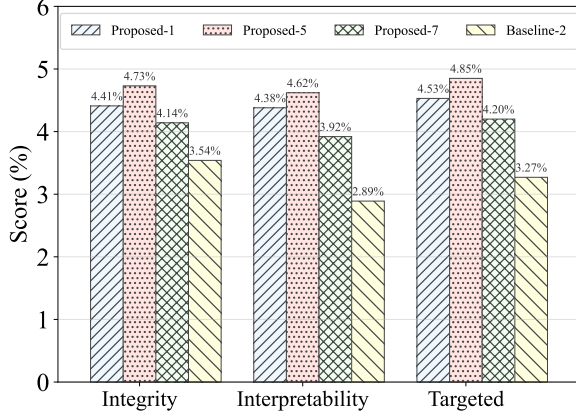
Fig. 10(a) shows that High-Only provides the strongest defense coverage. Under Seen Normal and Unseen Normal, High-Only achieves coverage rates of 76.7% and 89.8%, respectively, whereas High+Low achieves 73.1% and 78.0%. This result indicates that incorporating corrective samples derived from low-return trajectories does not provide complementary benefits and can instead weaken the learned LLM prior. The operational failure rates in Fig. 10(b) show the same trend. Under Unseen Normal, High-Only reduces the failure rate from 50.1% with the Base LLM to 25.3%. Under Unseen High, the failure rate decreases from 69.8% to 45.2%, whereas High+Low remains at 64.7%. These results indicate that high-return trajectories provide more consistent supervision for transferring effective RL action preferences to the LLM, whereas low-return corrective samples can introduce conflicting signals. Accordingly, High-Only trajectory alignment is adopted as the default configuration.

F. Performance Analysis Under Different LLM Configurations

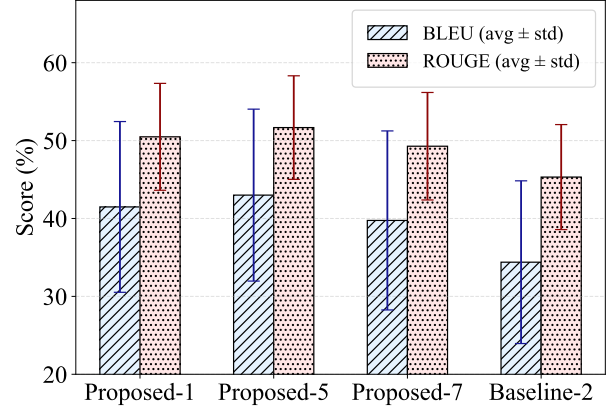
This subsection examines how different LLM configurations affect playbook reliability, generation efficiency, and output quality. The analysis considers the effect of semantic-retrieval

TABLE V: Quality Evaluation Results of Security Playbooks

Methods	BLEU (%)			ROUGE (%)			GPT-4 Score		
	-1	-2	-4	-1	-2	-L	A	I	T
Proposed-1	55.49	40.23	28.73	59.82	43.53	48.10	4.41	4.38	4.53
Proposed-5	56.82	42.37	29.81	60.73	44.97	49.31	4.73	4.62	4.85
Proposed-7	54.17	39.03	26.05	58.68	42.32	46.83	4.14	3.92	4.20
Baseline-2	47.28	34.19	21.70	54.31	38.09	43.55	3.54	2.89	3.27



(a) Playbook GPT-4 score



(b) BLEU-ROUGE Performance Comparison

Fig. 12: Quality score of security playbooks.

size on operational failure rate and generation duration, followed by reference-based and GPT-4-based evaluation of the resulting playbooks.

1) *Impact of Retrieval Size on Cybersecurity Playbook Generation.* To examine the effect of retrieval size on playbook reliability and generation efficiency, the Top- K value used in semantic retrieval is varied. Fig. 11 reports the resulting operational failure rate and playbook generation time.

As shown in Fig. 11(a), the average operational failure rate decreases as more strategy atoms are retrieved, indicating that a richer retrieved set can improve orchestration reliability. Proposed-5 achieves the lowest failure rate across all settings and stabilizes at approximately 3.5% when more than 25 atoms are retrieved, whereas Baseline-2 remains above 6%, reflecting its lower robustness in complex orchestration scenarios. Fig. 11(b) shows the corresponding increase in generation time. Baseline-2 is the fastest, requiring approximately 50–80 s because it relies on LLM generation without RL-based optimization. Proposed-5 incurs the highest latency, reaching approximately 130 s due to its larger LLM backbone. These results indicate that increasing model scale can improve reliability but also introduces additional generation overhead, while the 7B-based Proposed-1 provides a more balanced tradeoff between reliability and efficiency.

2) *Playbook Quality Evaluation.* GPT-4 scores are assigned on a five-point scale, where 1 represents the lowest quality and 5 the highest. Table V shows that Proposed-5 achieves the best overall BLEU, ROUGE, and GPT-4-based results, indicating stronger lexical consistency, structural similarity, and scenario relevance. Proposed-1 performs closely behind while retaining better generation efficiency. By contrast, Proposed-7 exhibits

clear degradation after removing alignment fine-tuning, while Baseline-2 performs worst across the reported quality metrics.

Fig. 12(a) further shows that Proposed-5 achieves the highest GPT-4-based scores in completeness, interpretability, and scenario relevance, whereas Baseline-2 receives the lowest ratings. Fig. 12(b) reports the averaged BLEU and ROUGE results with error bars and confirms the same overall ranking. These results suggest that increasing LLM scale can further improve playbook quality, while knowledge augmentation and RL-based policy alignment remain important for producing reliable and context-aware security playbooks.

VI. CONCLUSION

This paper develops an LLM-guided RL framework for adaptive cybersecurity orchestration beyond predefined playbooks and fixed action spaces. By integrating strategy-atom augmentation and validation, scenario-dependent action-space handling, and RL-informed LLM alignment, the framework improves both decision flexibility and operational reliability. Experiments show that knowledge enhancement improves strategy-atom quality, feasibility masking and LLM guidance provide stronger robustness under varying action spaces and unseen threats, and high-return trajectories offer more effective supervision for LLM alignment than mixed high- and low-return feedback. The modular design also allows the knowledge base, action templates, LLM backbone, and RL optimizer to be adapted to different security environments, supporting broader deployment across heterogeneous orchestration scenarios. Future work will investigate multi-agent coordination and continual adaptation to evolving threats.

REFERENCES

- [1] X. Dong, Y. Lai, X. Zhang, and X. Xu, "Counteracting new attacks in CPS: A few-shot class-incremental adaptation strategy for intrusion detection system," *IEEE Trans. Netw. Serv. Manag.*, vol. 22, no. 3, pp. 2473–2488, 2025.
- [2] T. Wang, Q. Liu, H. Shen, and G. Bai, "IDNet: Instance-adaptive dynamic network with adversarial training for intrusion detection," *Comput. Netw.*, p. 112336, 2026.
- [3] A. K. Yadav, A. Braeken, and M. Liyanage, "A provably secure lightweight three-factor 5G-AKA authentication protocol relying on an extendable output function," *IEEE Trans. Netw. Serv. Manag.*, vol. 23, pp. 2396–2414, 2026.
- [4] S. Tariq, M. Baruwat Chhetri, S. Nepal, and C. Paris, "Alert fatigue in security operations centres: Research challenges and opportunities," *ACM Comput. Surv.*, vol. 57, no. 9, pp. 1–38, 2025.
- [5] V. G. Bilali, E. Magklaris, D. Kosyvas, L. Karagiannidis, E. Ouzounoglou, and A. Amditis, "A SOAR platform for standardizing, automating operational processes and a monitoring service facilitating auditing procedures among IoT trustworthy environments," in *Proc. Int. Conf. Availability, Reliab. Secur.*, 2024, pp. 1–8.
- [6] M.-D. Nguyen, W. Mallouli, A. R. Cavalli, and E. Montes de Oca, "AI4SOAR: A security intelligence tool for automated incident response," in *Proc. Int. Conf. Availability, Reliab. Secur.*, 2024, pp. 1–8.
- [7] D. Giarimpampa, R. Meier, T. F. Bissyandé, V. Lenders, and J. Klein, "Exploring the role of artificial intelligence in enhancing security operations: A systematic review," *ACM Comput. Surv.*, vol. 58, no. 3, pp. 1–38, 2025.
- [8] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3779–3795, 2023.
- [9] Z. Hu, M. Zhu, and P. Liu, "Adaptive cyber defense against multi-stage attacks using learning-based POMDP," *ACM Trans. Priv. Secur.*, vol. 24, no. 1, pp. 1–25, 2020.
- [10] M. He, X. Wang, P. Wei, L. Yang, Y. Teng, and R. Lyu, "Reinforcement learning meets network intrusion detection: A transferable and adaptable framework for anomaly behavior identification," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 2, pp. 2477–2492, 2024.
- [11] A. Applebaum, C. Dennler, P. Dwyer, M. Moskowitz, H. Nguyen, N. Nichols, N. Park, P. Rachwalski, F. Rau, A. Webster *et al.*, "Bridging automated to autonomous cyber defense: Foundational analysis of tabular Q-learning," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2022, pp. 149–159.
- [12] H. Shen, X. Li, X. Wang, Y. Dai, T. Wang, and G. Bai, "LLM-augmented contrastive learning for misinformation detection in social networks," *IEEE Trans. Comput. Soc. Syst.*, vol. 13, no. 1, pp. 999–1014, 2026.
- [13] H. Shen, Q. Liu, F. Li, T. Wang, Y. Dai, and G. Bai, "Split-federated BERT with adversarial training for edge intrusion detection," *IEEE Internet Things J.*, vol. 13, no. 12, pp. 27382–27394, 2026.
- [14] F. Li, H. Shen, J. Mai, T. Wang, Y. Dai, and X. Miao, "Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection," *Peer-to-Peer Netw. Appl.*, vol. 17, no. 1, pp. 227–245, 2024.
- [15] H. Shen, X. Li, Y. Wang, T. Wang, and G. Bai, "Collaborative path penetration in 5G-IoT networks: A multi-agent deep reinforcement learning approach," *Peer-to-Peer Netw. Appl.*, vol. 18, no. 3, p. 113, 2025.
- [16] H. Shen, Y. Tian, T. Wang, and G. Bai, "Slicing-based task offloading in space-air-ground integrated vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4009–4024, 2024.
- [17] 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 Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 6, pp. 9737–9757, 2025.
- [18] M. Pternea, P. Singh, A. Chakraborty, Y. Oruganti, M. Milletari, S. Bapat, and K. Jiang, "The RL/LLM taxonomy tree: Reviewing synergies between reinforcement learning and large language models," *J. Artif. Intell. Res.*, vol. 80, pp. 1525–1573, 2024.
- [19] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Trans. Inf. Syst.*, vol. 43, no. 2, pp. 1–55, 2025.
- [20] A. Shaked, Y. Cherdantseva, P. Burnap, and P. Maynard, "Operations-informed incident response playbooks," *Comput. Secur.*, vol. 134, p. 103454, 2023.
- [21] C. Islam, M. A. Babar, and S. Nepal, "Design and generation of a set of declarative APIs for security orchestration," *IEEE Trans. Serv. Comput.*, vol. 17, no. 1, pp. 127–141, 2023.
- [22] R. Friel and A. Sanyal, "ChainPoll: A high efficacy method for LLM hallucination detection," *arXiv preprint arXiv:2310.18344*, 2023.
- [23] A. Martino, M. Iannelli, and C. Truong, "Knowledge injection to counter large language model (LLM) hallucination," in *Proc. Eur. Semantic Web Conf.*, 2023, pp. 182–185.
- [24] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.
- [25] S. Zhou, J. Liu, Y. Lu, J. Yang, D. Hou, Y. Zhang, and S. Hu, "April: Towards scalable and transferable autonomous penetration testing in large action space via action embedding," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 3, pp. 2443–2459, 2025.
- [26] F. Terranova, A. Lahmadi, and I. Chrisment, "Scalable and generalizable rl agents for attack path discovery via continuous invariant spaces," in *Proc. Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, 2025, pp. 440–457.
- [27] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.
- [28] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3779–3795, 2021.
- [29] Y. Cao, K. Liu, Y. Lin, L. Wang, and Y. Xia, "Deep-reinforcement-learning-based self-evolving moving target defense approach against unknown attacks," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33027–33039, 2024.
- [30] J. M. B. Murcia, A. M. Zarca, and A. Skármeta, "BASTION: Beyond automated service and security orchestration for next-generation networks," *Comput. Netw.*, p. 111352, 2025.
- [31] L. Belcastro, C. Carlucci, C. Cosentino, P. Liò, and F. Marozzo, "Enhancing network security using knowledge graphs and large language models for explainable threat detection," *Future Gener. Comput. Syst.*, vol. 176, p. 108160, 2026.
- [32] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu, "A survey on interpretable deep reinforcement learning," *Mach. Learn.*, vol. 113, no. 8, pp. 5847–5890, 2024.
- [33] A. Khediri, H. Slimi, A. Yahiaoui, M. Derdour, H. Bendjenna, and C. E. Ghenai, "Enhancing machine learning model interpretability in intrusion detection systems through shap explanations and LLM-generated descriptions," in *Proc. Int. Conf. Pattern Anal. Intell. Syst. (PAIS)*, 2024, pp. 1–6.
- [34] J. Lin, J. Zhao, L. Liu, Y. Deng, Y. Zhao, L. Huang, X. Lin, W. Zhou, and H. Li, "RL-LLM-DT: An automatic decision tree generation method based on RL evaluation and LLM enhancement," *arXiv preprint arXiv:2412.11417*, 2024.
- [35] Y. Yao, Z. Liu, Z. Cen, J. Zhu, W. Yu, T. Zhang, and D. Zhao, "Constraint-conditioned policy optimization for versatile safe reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023, pp. 12555–12668.
- [36] S. Liu, B. Sabir, S. I. Jang, Y. Kansal, Y. Gao, K. Moore, A. Abuadbaa, and S. Nepal, "From solitary directives to interactive encouragement! LLM secure code generation by natural language prompting," *arXiv preprint arXiv:2410.14321*, 2024.
- [37] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 1999, pp. 1057–1063.
- [38] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
- [39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [40] E. Reiter, "A structured review of the validity of BLEU," *Comput. Linguist.*, vol. 44, no. 3, pp. 393–401, 2018.
- [41] C.-Y. Lin, "Looking for a few good metrics: ROUGE and its evaluation," in *Proc. NTCIR Workshop*, 2004, pp. 1–8.



Hang Shen (Member, IEEE) received the Ph.D. degree with honors in Computer Science from the Nanjing University of Science and Technology, Nanjing, China, in 2015. He worked as a Full-Time Postdoctoral Fellow with the Broadband Communications Research (BBCR) Lab, Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada, from 2018 to 2019. He is an Associate Professor with the Department of Computer Science and Technology at Nanjing Tech University, Nanjing, China. His research in-

terests involve vehicular networks, cybersecurity, and agentic AI. He serves as an Associate Editor for *J. Inf. Process.*, *Front. Blockchain*, and IEEE ACCESS, and has served as a Guest Editor for *Peer-to-Peer Netw. Appl.*. He serves/served as a TPC member of IEEE GLOBECOM, IEEE SVCC, IEEE ISPA, IEEE HPCCE, IEEE CCECE, PST, and ICA3PP. He is a member of the IEEE Computer Society, Communication Society, and Vehicular Technology Society, an Executive Committee Member of the ACM Nanjing Chapter, a CCF Senior Member, and a CAAI member.



Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in Computer Science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he worked as a Research Scientist at the German National Research Center for Information Technology, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been working at Nanjing Tech University,

Nanjing, China, as a Professor in Computer Science. From October to December 2010, he was a Visiting Professor with the Electrical and Computer Engineering Department at the University of Waterloo, Waterloo, ON, Canada. His research interests include wireless networking, cybersecurity, and agentic AI. He has authored and co-authored more than 90 peer-reviewed papers in international journals and conferences. He is a CCF Distinguished Member.



Kerui Li received the B.Eng. degree in Computer Science from Jiamusi University, Jiamusi, China, in 2025. He is now pursuing his M.Eng. degree in Computer Science at Nanjing Tech University, Nanjing, China. His research interests include knowledge graphs and large-language-model-based agents for cybersecurity orchestration.



Xin Liang received the B.Eng. and M.Eng. degrees in Computer Science from Nanjing Tech University, Nanjing, China, in 2022 and 2025, respectively. She is currently an engineer at BMW (Nanjing) Information Technology Co., Ltd. Her research interests include pretrained language models and deep reinforcement learning for cybersecurity orchestration.



Tianjing Wang (Member, IEEE) holds a B.Sc. in Mathematics at Nanjing Normal University in 2000, an M.Sc. in Mathematics at Nanjing University in 2002, and a Ph.D. in Signal and Information System at the Nanjing University of Posts and Telecommunications in 2009. From 2011 to 2013, she was a Full-Time Postdoctoral Fellow with the School of Electronic Science and Engineering, Nanjing University of Posts and Telecommunications. From 2013 to 2014, she was a Visiting Scholar with the Electrical and Computer Engineering Department at the State

University of New York at Stony Brook. She is an Associate Professor with the Communication Engineering Department at Nanjing Tech University, Nanjing, China. Her research interests include cybersecurity and V2X communication networks with agentic AI. She has published papers in prestigious international journals, including *Elsevier Comput. Netw.*, *Elsevier J. Syst. Archit.*, IEEE TMC, IEEE TITS, IEEE TNSM, IEEE TCSS, IEEE TBC, IEEE IoT-J, and IEEE TCE.