Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

LLM-ES: A Multi-Agent Hybrid Framework Integrating Large Language Models and Expert Systems for Autonomous Cyber Defense

Zhiqiang Qu
National University of Defense Technology
Jun He
hejun17c@nudt.edu.cn
Information Support Force Engineering University
Zhitao Long
Information Support Force Engineering University
Bo Wu
Information Support Force Engineering University
Xinmei Wang
National University of Defense Technology
Xiaohua Yang
Beijing Space Environment Institute
Yuting Zhao
Information Support Force Engineering University

Article

Keywords: Autonomous Cyber Defense(ACD), Large Language Models (LLMs), Expert Systems, Multi-Agent Systems, Hybrid LLM-Expert System

Posted Date: November 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11386230/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

LLM-ES: A Multi-Agent Hybrid Framework Integrating Large Language Models and Expert Systems for Autonomous Cyber Defense

Zhiqiang Qu[a, b], Jun He[b, *], Zhitao Long[b], Bo Wu[b], Xinmei Wang[a, b], Xiaohua Yang[c], Yuting Zhao[b]

[a] National University of Defense Technology, Changsha 410073, China;

[b] Information Support Force Engineering University, Wuhan 430035, China;

[c] Beijing Space Environment Institute, Beijing 100080, China;

E-mail: quzhiqiang@nudt.edu.cn(Z. Qu); hejun17c@nudt.edu.cn(J. He);

longzhitao@alumni.nudt.edu.cn(Z. Long); cherry_wb@nudt.edu.cn(B. Wu);

wangxinmei22@nudt.edu.cn(X. Wang); yxhua3@163.com(X. Yang); xty_zyt@163.com(Y. Zhao)

* Correspondence: hejun17c@nudt.edu.cn

Abstract

In recent years, Autonomous Cyber Defense (ACD) methods driven by large language models (LLMs) have demonstrated promising performance in decision interpretability and environmental transferability, yet suffer from low decision efficiency and unstable decision-making. To mitigate these limitations, this work proposes a multi-agent hybrid LLM–expert system framework for ACD. The framework seamlessly integrates the determinism and efficiency of expert systems with the deep reasoning capabilities of LLMs: the expert system makes deterministic decisions based on experiential knowledge, whereas the LLM is responsible for complex decision tasks demanding deep exploration and long-chain reasoning. We further strengthen the LLM via chain-of-thought (CoT) guided prompting, historical memory, retrieval-augmented generation (RAG), output alignment and feedback learning. Experimental results demonstrate that incorporating the expert system improves both decision efficiency and defense

performance. Evaluated on the CybORG CC4 environment, our approach achieves an average reward of −98 over ten independent rounds, representing an approximate 25.5% improvement over the best-performing reinforcement learning (RL) baseline. Compared with the standalone LLM method, the average reward increases by roughly 96.2%, while the average per-step runtime is reduced by approximately 14.6%. To the best of our knowledge, this is the first work to systematically integrate LLMs and expert systems for ACD in the CybORG CC4 environment.

Keywords. Autonomous Cyber Defense(ACD), Large Language Models (LLMs), Expert Systems, Multi-Agent Systems, Hybrid LLM-Expert System

1. Introduction

As cyberattacks continue to escalate in number, speed, and complexity, modern Security Operations Center (SOC) is facing increasing pressure. Traditional processes relying on rules and manual handling not only easily lead to alert fatigue, response delays, and high false positive rates, but also result in over-reliance on security analysts [1]. Meanwhile, cyberattacks are increasingly exhibiting automated and intelligent characteristics, rendering traditional defense models that rely solely on rules and human intervention unsustainable. ACD aims to leverage artificial intelligence agents to automatically perform security tasks such as monitoring, detection, and response, and is therefore regarded as an important approach to addressing the above challenges [2].

At present, the mainstream technical route in ACD research is defense decision-making methods based on RL. Such methods enable defense agents to learn adaptive policies through

repeated interactions with adversaries and dynamic environments, and have achieved remarkable results in simulated environments [2–7]. However, defense agents based on deep reinforcement learning (DRL) still face three limitations in real-world applications: first, insufficient policy interpretability, making it difficult to explain the specific reasons behind decisions; second, policy models require pre-training, resulting in high training costs; third, limited policy transferability, as changes in attack strategies or network topologies often necessitate retraining. Therefore, how to reduce training costs and improve the interpretability and transferability of defense policies has become a key issue that urgently to be addressed in ACD.

The rapid development of LLMs has brought new opportunities to address the above issues. LLMs possess strong capabilities in semantic understanding, commonsense reasoning, and knowledge generalization, and have demonstrated outstanding potential in cybersecurity tasks such as threat intelligence extraction, vulnerability detection, and incident response [8,9]. In the field of ACD, Castro et al. [10]were the first to integrate LLM agents with RL agents into the CybORG CC 4 multi-agent environment [11], validating the feasibility of LLMs as autonomous cyber defenders; however, their experimental results showed that the rewards of LLM-driven methods were significantly lower than those of reinforcement learning methods. Gao et al. [12] proposed an end-to-end LLM agent for in-context autonomous incident response; Farooq [13] integrated RL, RAG, and LLMs into a unified cyber defense framework, achieving few-shot adaptation based on RL trajectories. These works indicate that LLMs can provide natural language explanations for defense decisions and adapt to different network environments without additional training, thereby compensating for the inherent shortcomings of RL methods. Meanwhile, we also note that Mohammadi et al. [14] quantitatively compared LLM agents with the most successful RL agents on the CAGE-CC2 public challenge [15], finding that the difference in

mean rewards between the two was only 0.04, with highly overlapping 95% confidence intervals; Jacques [16] systematically evaluated LLM agents against a PPO RL baseline in CybORG++, and found that an LLM with few-shot learning combined with manually crafted rule prompts could surpass the RL baseline over long time horizons, but exhibited higher latency, greater runtime variance, and unstable transferability to different opponents and topologies.

Therefore, a key issue is how LLM-based methods can be improved and enhanced in a complex scenario such as ACD. To this end, we conducted a series of experiments in the CybORG CC 4 multi-agent environment [11]. We designed all blue agents to be LLM-driven and, through prompt engineering, incorporated the agents' scenarios, roles, tasks, and output examples into the prompts. Unfortunately, our experimental results were almost consistent with those of Castro et al. The LLM outputs were unstable and often exceeded the legal action range; defense decisions had high latency and low reward, lagging far behind reinforcement learning methods.

To enhance the correctness and usability of LLM responses, we made improvements in three aspects. First, we constructed chain-of-thought (CoT) -guided prompts that incorporated historical memory and reflection mechanisms. Second, we integrated a RAG mechanism to augment the LLM. Prior to generation, relevant domain knowledge is retrieved from an external knowledge base and injected into the prompt context, grounding the generation in external evidence and substantially improving factual accuracy and controllability. Third, we aligned the response outputs to standardize their format and content. Because different LLMs exhibit distinct output conventions, this alignment prevents valid actions from being rejected due to formatting issues or the omission of non-critical information such as subnet names.

LLM-based decision-making methods are inefficient, which stems not only from the inherent mechanism of autoregressive generation but also from the additional overhead of

prompt construction, RAG, and model inference. This raises a key question: should every decision be delegated to an LLM? While LLMs are known to excel at complex semantic understanding and long-horizon reasoning, decisions such as "if a root privilege alert is detected on a host, a Restore operation should be immediately performed on it to completely eliminate the threat" can be made entirely on the basis of experiential knowledge, without invoking an LLM. Therefore, we attempt to integrate expert systems with LLM-based methods, combining the determinism and efficiency of expert systems with the complex reasoning strengths of LLMs. When an alert is detected and a decision can be made based on experiential knowledge, it is delegated to the expert system. When there are no alerts or when there are many alerts, the strengths of the LLM are leveraged, allowing it to perform long-horizon tasks requiring in-depth data analysis and reasoning based on network state, historical information, and knowledge retrieval. Experimental results show that integrating expert systems with LLMs not only improves the average efficiency of defense decision-making but also substantially enhances defense effectiveness.

In summary, to tackle the challenges of LLM-based ACD approaches, including output instability, low decision efficiency and limited defense performance, this work seamlessly integrates the deterministic decision-making capacity of expert systems with the deep reasoning ability of LLMs. We present a multi-agent hybrid LLM–expert system framework. The main contributions of this work are outlined as follows:

(1) We propose a multi-agent hybrid LLM–expert system decision-making framework. Within this framework, expert systems and LLMs adopt separated responsibilities and cooperate to generate joint decisions. Evaluated in the multi-agent CybORG CC4 environment, our approach outperforms state-of-the-art (SOTA) RL and standalone LLM baselines. Specifically, it

achieves an approximate 25.5% improvement in average reward compared with the best RL baseline.

(2) We design five enhancement mechanisms for LLM decision-making: CoT-guided prompt engineering, RAG, historical memory, output alignment, and feedback learning. The effectiveness of each mechanism is verified through ablation studies.

(3) We conduct experiments on CybORG CC4 to verify the feasibility and effectiveness of the proposed method under diverse LLM backbones and varied adversary configurations, demonstrating the generalization capacity of our hybrid framework.

The remainder of this paper is organized as follows: Section 2 introduces related work, Section 3 details the design of the hybrid LLM-expert system method, Section 4 presents the experimental design and result analysis, and Section 5 provides the discussion.

2. Related Work

The core goal of ACD research is to build agents that can respond to security incidents quickly and accurately. Since direct trial-and-error on real networks carries unacceptable risks, such agents are typically trained in simulation environments known as "ACD training gym." Typical environments include CyberBattleSim [4], NASimEmu [17], CybORG [18,19], and CybORG++ [20]. These environments reconstruct network topologies at different levels of abstraction and simulate adversarial processes among red, blue, and green agents. In terms of decision-making methods, DRL [21], MARL [3], and HMARL [7] have been widely used to learn defense policies and have achieved many results. However, their policies are implicitly encoded in network parameters, and they face problems such as high training costs, insufficient generalization, and poor interpretability; when network structures, attack strategies, or attack intensities change, retraining is often required. In addition, the trustworthiness of policies and human-machine

collaboration issues also hinder their deployment in real military and critical infrastructure networks [22].

Therefore, LLM-based methods have been applied to the field of cyber defense due to their better interpretability and generality. Xu et al. [9] conducted a systematic review of 185 papers from top conferences and journals, summarizing prompt engineering, fine-tuning, and external augmentation as the three main techniques for adapting LLMs to security tasks. Among them, prompt engineering is adopted by almost all works, while the most advanced applications increasingly exhibit a hybrid system form of "LLM as the core reasoning engine + external knowledge/tools." Castro et al. [10] developed a scalable LLM adapter, first integrated LLM and RL defense agents in the CybORG CC 4, and designed inter-agent communication protocols. Their experiments found that LLM agents possess human-like security reasoning and interpretable decision-making capabilities, but their rewards and response speed are still inferior to those of RL agents. Mohammadi et al. [14] were the first to quantitatively compare LLM agents with the most successful RL agents on the CAGE-2 public challenge, verifying that high-complexity prompts significantly improve LLM action selection accuracy. L2M-AID [23] uses LLMs as a semantic bridge to transform unstructured observation data into contextual state representations, driving MAPPO multi-agent collaborative defense. Gao et al. [12] integrated four capabilities—perception, reasoning, planning, and action—into a lightweight LLM agent, achieving autonomous incident response planning through in-context learning. Jacques [16] conducted systematic experiments in CybORG++ on the PPO baseline, multiple LLM models, and multiple prompt strategies, finding that the best combination remains "few-shot + rules + memory", whose performance exceeds the PPO RL baseline, but LLM agents are highly sensitive to prompt engineering and have response latency several times that of RL methods.

In addition, LLM-based methods are also commonly used to enhance reinforcement learning methods. Jiang et al. [24] used LLMs to guide skill discovery and intrinsic reward design in hierarchical multi-agent reinforcement learning, achieving faster convergence than a flat MARL baseline in the CAGE 4 environment. CyberOps-Bots [25] constructed a hierarchical architecture of "upper-layer LLM agents + lower-layer RL agents," balancing adaptability and execution reliability. SecLoop [26] embedded LLMs into the full lifecycle of security policy generation, orchestration, response, and feedback, and proposed a security-aware group relative policy optimization algorithm to iteratively refine policies. Mukherjee et al. [27] further used LLMs to assist in designing DRL reward functions to generate diverse defense strategies. Ahmed et al. [28] adopted a hierarchical LLM critic to enhance RL autonomous agents, using prompt engineering to encode organizational policies and RAG to facilitate inter-LLM communication.

To address the hallucination problem of LLMs, RAG semantically retrieves relevant information from a knowledge base through an external retrieval module and fuses it with the generation input, thereby enhancing the factual accuracy, contextual completeness, and output controllability of generated content, and demonstrating outstanding performance in knowledge-intensive tasks [29]. Farooq [13] proposed a RAG-based few-shot learning method to build a general-purpose LLM-driven defense agent; CyberRAG [30] adopts an agentic RAG design to achieve integrated output of attack classification, explanation generation, and structured reporting; Tellache et al. [31] combined hybrid retrieval (CTI vector database similarity retrieval + standardized queries on external platforms) with LLM response generation to automatically generate executable mitigation strategies; Cadet et al. [32] verified that security incident analysis based on query filtering and RAG achieves a good balance between high recall and low cost; ReVul-CoT [33] combines RAG with chain-of-thought prompting for vulnerability assessment.

CTI-Thinker [34] unifies entity recognition, entity alignment, relation extraction, and ATT&CK semantic reasoning, and constructs a GraphRAG reasoning engine to support tactical-level inference; Lin et al. [29] proposed a "four-layer, one-module" construction method for a RAG-based cybersecurity large language model, ensuring the usability, trustworthiness, and security of the model through metadata-filtered retrieval, dynamic permission management, and multi-round evaluation feedback mechanisms.

These studies indicate that LLMs have the potential to complement or surpass reinforcement learning (RL) for cyber defense tasks, offering interpretability and generalization advantages over conventional RL-based approaches. Nevertheless, LLMs still suffer from inherent drawbacks including hallucinations, output instability, and decision latency, which require further mitigation. Expert systems, in contrast, provide deterministic outputs, low latency and high interpretability for cyber defense decision-making. This contrast motivates our work to seamlessly integrate expert systems and LLMs for autonomous cyber defense.

3. Methods

3.1 Problem Statement

We consider a simulated network environment consisting of multiple interconnected subnets, where each subnet contains a set of servers and hosts. Three categories of agents operate within this environment. Red agents represent network adversaries, which automatically execute a series of attack and penetration operations to compromise the confidentiality and availability of network assets. One blue agent is assigned to each subnet. Blue agents can perform defensive actions including *Analyze*, *Remove*, *Restore*, *DeployDecoy*, *AllowTrafficZone*, and *BlockTrafficZone* to mitigate threats and maintain normal network operation. Green agents

simulate legitimate internal users that conduct local works and network access; they may inadvertently trigger phishing events or false positive alerts.

A penalty is applied to the defending blue agents when green agents fail their local works or red agents successfully compromise hosts. The cumulative penalty across all blue agents corresponds to the negative reward in this environment.

This work targets a decision framework for blue agents driven by large language models (LLMs). The observations collected by blue agents are transformed into textual prompts fed to the LLM. Given the real-time security state, the LLM assists blue agents in selecting effective defensive actions, with the overall objective of minimizing the cumulative penalty of all blue defenders.

Formally, at each time step t, each blue agent receives a partial observation $o_t \epsilon O$ from the environment and selects a defensive action $a_t \epsilon A$. The environment returns an immediate reward r_t . The objective is to learn a decision policy $\pi(o_t) \rightarrow a_t$ that minimizes the expected total cumulative reward $E[\Sigma_t r_t]$ over the episode.

3.2 Overall Framework

The core objective of the proposed hybrid LLM–expert system approach is to seamlessly combine the deterministic, high-efficiency decision-making capacity of expert systems with the deep reasoning capability of LLMs, so that the strengths of both paradigms can be fully exploited. The framework comprises four modules: situation awareness, LLM module, expert system module, and hybrid decision-making module.

The situation awareness module collects state and historical information from the environment and agents, and performs real-time conversion of numerical observation vectors into semantic descriptions interpretable by the LLM. The expert system module matches

predefined defensive responses against threat alerts raised by hosts or subnets; these response rules are constructed from domain expert knowledge. The LLM module generates appropriate defensive actions for each agent based on the prevailing security situation. It improves the reliability and quality of outputs via a suite of enhancement techniques: CoT-guided prompt engineering, RAG, historical memory, output alignment, and feedback learning. The hybrid decision-making module is responsible for organically fusing the decisions of the expert system and the LLM, ultimately outputting an final action for each agent. The overall structure of the framework is shown in Figure 1.

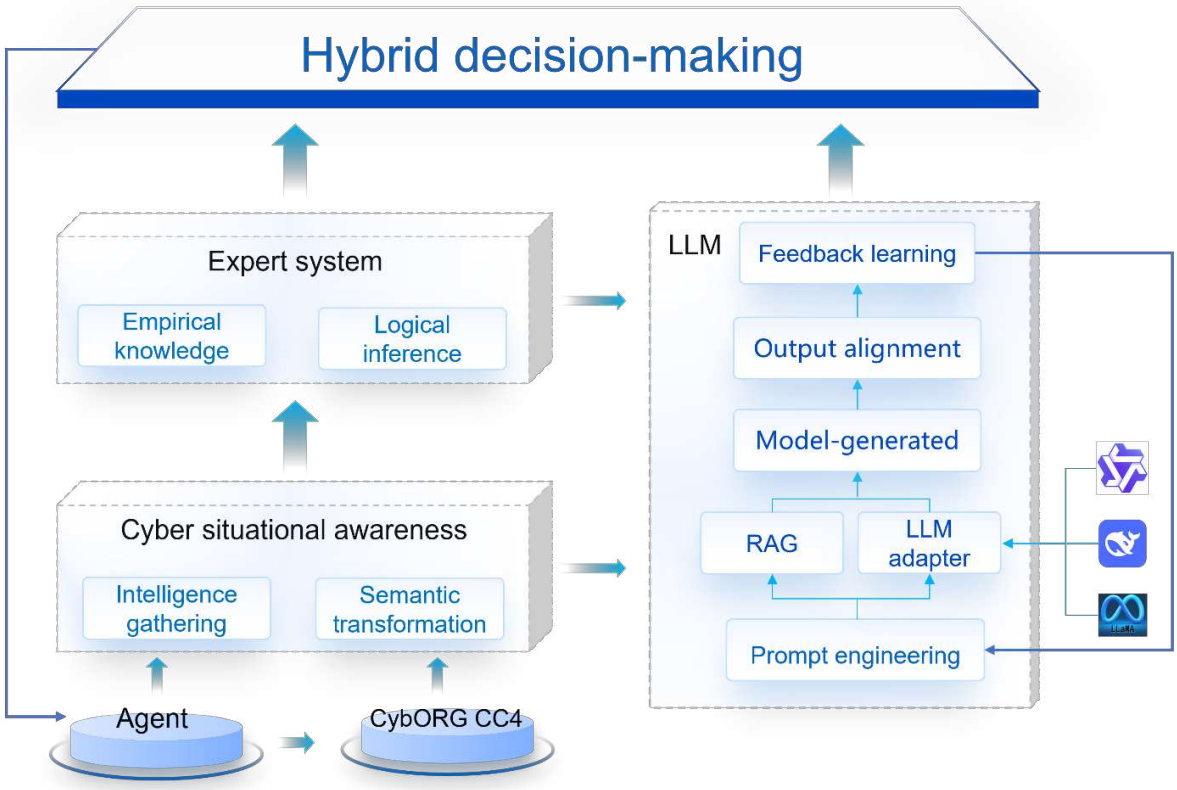


Figure 1. Hybrid LLM-Expert System Decision-Making Framework.

3.3 Decision-Making Mechanism of LLMs and Expert Systems

Hybrid decision-making is the core component of the proposed framework. To harness the complementary strengths of expert systems and LLMs, this mechanism defines the

conditions for invoking expert rules, the scope of LLM reasoning, and the sequential fusion workflow combining both outputs. Simple and well-defined threat scenarios are delegated to the expert system. This reduces the LLM's reasoning load, improves decision efficiency and stabilizes outputs, while complex scenarios requiring deep situational analysis, potential threat mining and long-chain reasoning are handled by the LLM.

In this work, simple decision scenarios correspond to four types of high-risk alerts on hosts and servers. We construct a threat alert–preliminary action rule table derived from domain expert knowledge, as shown in Table 1:

(a) When a root-privilege alert is detected on a host, this indicates that the adversary has obtained maximum control over that host. A *Restore* action will be recommended to revert the host to a trusted state.

(b) When a user-privilege alert is detected on a host, this signifies low-privilege malicious activity. A *Remove* action will be recommended to eliminate malicious programs and suspicious connections.

(c) When a host is observed accessing a decoy, an *Analyze* action will be recommended to inspect the host for malicious programs or connections.

(d) When a green agent suffers a local work failure on a host, an *Analyze* action will be recommended to check the host for malicious programs or connections.

Table 1. Threat Alert–preliminary Action rule Table.

High-risk alert	Emergency action
Root access	Restore
User access	Remove
Access to the decoys	Analyze
Local work failed	Analyze

The rule table adopts the design principles of *minimal completeness and conservative safety*. It predefines defensive actions only for high-confidence high-risk alerts and remains extensible. The expert system performs rule matching using this table.

A core branching logic controls the workflow: if exactly one expert rule is triggered by a single alert, the expert-generated defensive action is executed directly, bypassing the LLM. When multiple rules trigger simultaneously or no rule matches the current situation, any preliminary candidate actions from the expert system are passed to the LLM for secondary reasoning and refinement. This role separation allows the two components to cooperate for multi-agent cyber defense. The full hybrid decision-making workflow is provided in Algorithms 1 and Algorithms 2.

Algorithm 1. Hybrid LLM-expert system Decision-Making Algorithm.

```
Input: O, A, N
if IN_PROGRESS == True then
    Return 'Sleep'
elif alert == True then
    es_actions = ES_AG(O, A)
    if len(es_actions) == 1 then
        Return es_actions
    else
        available_actions = es_actions
        best_action = LLM_AG(O, available_actions, N)
        if best_action == 'Sleep'
            Return RandomChoice (es_actions)
        else
            Return best_action
else
    best_action = LLM_AG(O, A, N)
    Return best_action
```

As illustrated in Algorithm 1, the decision workflow first checks whether the agent has an ongoing executing action in the current time step. If so, the algorithm directly returns the Sleep action and waits for the ongoing action to finish. When an alert is triggered, the expert system is invoked to generate preliminary defensive actions following predefined rules. If exactly one

preliminary defensive action is produced, this action is directly adopted as the final defensive action without LLM reasoning. When the expert system yields multiple preliminary candidates (e.g., concurrent alerts of different types), these candidate actions are passed to the LLM for selection. Leveraging real-time situational data, historical records and external knowledge, the LLM selects an appropriate defensive action from the candidate pool under alert conditions or from the full action space when no alert exists.

Notably, a feedback correction loop is embedded within the LLM decision procedure. If the LLM produces an invalid action, the model refines its output iteratively via this loop (Algorithm 2). We limit the maximum number of retries to 3 for efficiency. If no valid output is obtained after three retries, the algorithm returns Sleep when no expert candidate actions exist; otherwise, it randomly selects and returns one action from multiple expert candidate actions.

Algorithm 2. LLM_AG() Sub-algorithm.

```
Input: O, A, N
n = 0
while (True) do
    n++
    best_action = LLM.response(O, A)
    if best_action in available_actions then
        Return best_action
    elif n<N then
        add_prompt()
    else
        Return 'Sleep'
```

Across both decision modes, the LLM relies on four auxiliary mechanisms to guarantee output validity and reliability:

(a) CoT-guided prompts, which instruct the model to "first analyze adversarial intent, then assess asset value, and finally select an appropriate action from available options." This improves reasoning transparency and interpretability.

(b) RAG enhancement, which retrieves relevant information from a vector-based knowledge base for the current security situation. It injects domain knowledge to mitigate hallucinations and compensate for limited built-in domain expertise of LLMs.

(c) Historical memory, which stores recent alert records, selected actions and execution outcomes into the context window. This avoids repeated decisions and decision oscillation.

(d) Output alignment, including format normalization, semantic alignment and legality validation, to reduce the rate of invalid actions.

3.4 Situational Awareness and Semantic Transformation

ACD simulation environments such as CybORG are originally developed for RL, where observations are encoded as numerical vectors. Directly feeding these raw vectors into an LLM cannot yield reliable reasoning outputs. This section addresses how to build a conversion interface between the LLM and the ACD environment without modifying the native RL interface. The interface enables the LLM to interpret machine-readable environment observations and alert vectors, while producing structured actions that can be executed by the simulation environment.

To this end, this work builds upon the approach proposed by Castro et al. [10] and constructs a scalable Observation Adapter to convert environment observations into natural language. Unlike the original design, our adapter converts numerical observation vectors into a structured situation template instead of unstructured free text, guaranteeing consistent input format for the LLM. The template contains the agent identifier, task phase, historical actions and execution outcomes, lists of detected suspicious processes and connections, and parsed intrusion threat alerts.

For alert information, we introduce targeted enhancements compared with the original CybORG observation vectors and the method of Castro et al. [10], We add alert fields for decoy access and green-agent local work failures, together with parsed information derived from file analysis to identify root-privilege and user-privilege intrusion alerts.

3.5 Chain-of-Thought-Guided Prompt Engineering

The core problem to be addressed in this section is how to guide a general-purpose LLM through prompt engineering, without fine-tuning model parameters, to reason according to the rules of the environment and output reasonable, interpretable defense strategies.

Given the characteristics of the CybORG scenario, this work adopts a CoT-guided prompt design method and introduces a historical memory mechanism and a reflection mechanism, designing a six-element formatted prompt consisting of "scenario–goal–situation–history–example–reflection." Drawing on the method of Castro et al. [10], we also design system prompts and user prompts, shown in Figure 2 . The system prompt remains consistent across all episodes and describes overall information such as the role, task, action definitions, output examples, and key requirements. The user prompt contains the formatted security situation information for the current step, historical memory information, the available action space, and error response information, all of which may change at each step. The specific design includes:

(a) **Role setting**: The LLM is assigned the role of "a cybersecurity expert defending the network". This role prompting elicits the model's security-domain knowledge and relevant decision heuristics.

2) **Goals and constraints**: The defensive objective and scope of responsibility are clearly defined. Concise definitions of valid actions together with response examples are provided to prevent misjudgment arising from ambiguous action names.

3) **Situation injection**: Structured information such as phase information, situation observations, and alert information (intrusion alerts and suspicious alerts) is injected in natural language.

4) **Few-shot examples**: Illustrations of correct decisions are appended alongside the action definitions.

5) **CoT-guidance**: The prompt explicitly requires the model to "first analyze and forecast threats from the security situation, then prioritize threats according to asset importance, and finally select an appropriate defensive action from the available action set."

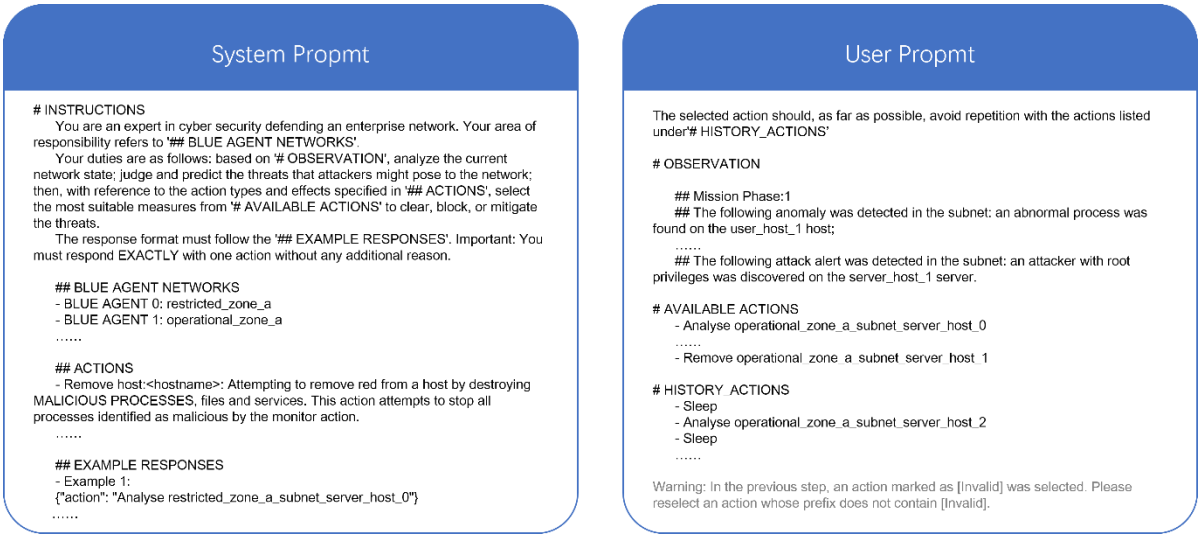


Figure 2. Prompt Example.

To further boost the LLM's reasoning performance, we incorporate a historical memory mechanism and a prompt-based reflection mechanism.

Historical memory mechanism: Records of alerts and selected actions from the most recent K timesteps are embedded into the prompt context. On one hand, this supports long-chain reasoning for the LLM; on the other hand, it mitigates repeated actions and decision oscillation.

Reflection mechanism: Before generating the final output, the LLM performs self-checks across three dimensions: whether the response format matches the provided examples,

whether the candidate action is reasonable, and whether it duplicates recent historical actions. This step serves as a final safeguard to guarantee output quality.

In terms of LLM invocation, this work locally deploys open-source LLMs including the Qwen, DeepSeek, and LLaMA by Ollama.

3.6 RAG and Knowledge Base Design

Our experimental results show that general-purpose LLMs underperform expectations for tasks within the CybORG CC4 scenario. Inspired by methods such as [13,29,35], this work introduces RAG and a domain knowledge base to augment the LLM and supply supplementary domain knowledge for reasoning. In implementation, retrieval queries are constructed from the current security situation. The top-k retrieved passages, alongside existing contextual information, are incorporated into the prompt, which is fed into the LLM to produce the final defensive decision, realizing domain knowledge injection.

The knowledge base is built following lightweight and efficient design principles. The collected domain materials include CybORG CC4 documentation, MITRE ATT&CK and D3FEND frameworks, and CWE vulnerability records. We use the open-source BAAI/bge-m3 embedding model to create vector representations for document chunks, which are indexed and stored in ChromaDB. Each chunk is capped at 600 characters with an 80-character sliding overlap.

3.7 Output Alignment and Feedback Learning

Our experiments show that raw LLM outputs lack stability. Even under identical prompts, different LLM families exhibit distinct output styles, and the same model may produce inconsistent output formats. To accommodate various LLMs and stabilize outputs, we apply four-stage post-processing to LLM responses.

First, format alignment. We adopt adaptive regular-expression matching to extract candidate defensive actions from model responses, avoiding invalid-action rejection caused by redundant text or format irregularities.

Second, semantic alignment. Although example outputs are provided within the prompt, the LLM sometimes shortens action names. For instance, it may abbreviate "Remove restricted_zone_a_subnet_server_host_0" to "Remove server_host_0". In such cases, our system supplements missing subnet identifiers using situational information to recover the full valid action name.

Third, validity checking. The parsed candidate action is cross-checked against the available action space; only valid actions are retained.

Fourth, historical consistency checking. The candidate action is compared against recent historical actions to prevent repeated decisions.

Even after these processing steps, the LLM may still generate invalid actions. To address this issue, we introduce a feedback correction mechanism: when the LLM produces an invalid action, failure information is appended to the prompt for the next round of generation. With this iterative correction loop, the LLM refines its outputs repeatedly.

4. Experiments

To evaluate the effectiveness and feasibility of our proposed hybrid LLM-expert system method in ACD scenarios, we conducted experiments in the CybORG CC4 environment [11]. All experiments run on a single workstation equipped with an NVIDIA RTX 3090 GPU. The primary software used in our experiments included: Ubuntu 22.04, Python 3.10.18, CUDA 12.1, CybORG v4.0, Gymnasium 0.28.1, Ray 2.10.0, PyTorch 2.2.0, Modelscope 1.39.1, OpenAI 2.38.0, and Ollama 0.24.0.

This experiment seeks to answer the following research questions:

(a) Method effectiveness: Does the hybrid LLM–expert system approach outperform existing baselines in multi-agent cyber adversarial environments?

(b) Benefits of hybrid decision-making: Does hybrid decision-making combining an expert system and an LLM yield lower decision latency and a higher valid defense action rate compared with pure-LLM and pure expert-system baselines?

(c) Module contributions: What performance contributions come from the proposed components — CoT-guided prompt engineering, RAG, historical memory mechanism, output alignment, and feedback correction?

(d) Adaptability and robustness: Can the proposed method adapt to different LLMs and red agents?

4.1 Effectiveness Validation Experiment

To validate the effectiveness of the defense strategies generated by our proposed LLM-ES method, we selected the current best RL-based method [6] and LLM-based method [10] for comparison. For ease of presentation, we refer to the method of Qu et al. as HC-MARL, the method of Castro et al. as Castro-LLM, and our proposed method as LLM-ES. In the experiments, each episode was set to 500 steps, and the red, blue, and green agents each selected one action per step. The red agent used the default FiniteStateRedAgent, and the green agent adopted the default policy. The LLM used in the Castro-LLM method was GPT-4o-mini, while our LLM-ES method used a locally deployed open-source qwen2.5:3b, with the temperature set to 0.3. Each method was run independently for 10 rounds, and the average reward and standard deviation over the 10 rounds were used as the main evaluation metrics. The experimental results are shown in Table 2.

Table 2. Rewards of Different Methods

Methods	HC-MARL	Castro-LLM	LLM-ES
Rewards ($mean \pm std$)	-131.6 ± 26.83	-2547.2 ± 498.8	-98 ± 27.61

From the data in Table 2, it can be seen that our proposed LLM-ES method achieved a 10-round average reward of -98. Compared with the HC-MARL method, the current best RL-based method, an improvement of about 25.5%, and compared with the Castro-LLM method, an improvement of about 96.2%. The experimental results show that our proposed LLM-ES method surpasses mainstream reinforcement learning methods and current LLM-based methods in defense performance.

In the above experiments, a very lightweight model, qwen2.5:3b, was used. To investigate whether model parameter size affects performance, we selected four models from the qwen2.5 series with different parameter sizes—0.5b, 1.5b, 3b, and 7b—for further experiments. The basic settings remained consistent with those of the previous experiments, except that two additional comparison metrics were introduced: the average runtime per step and the number of invalid actions generated by the LLM. Here, invalid actions refer to actions generated by the LLM that cannot be corrected into correct actions based on response information and that fall outside the available action space. The results are shown in Table 3.

Table 3. Impact of LLMs with Different Parameter Sizes on Results

LLM	qwen2.5:0.5b	qwen2.5:1.5b	qwen2.5:3b	qwen2.5:7b
Rewards ($mean \pm std$)	-197.1 ± 34.32	-192 ± 32.95	-98 ± 27.61	-86.7 ± 21.67
Run time (s)/step	6.81 ± 2.15	2.31 ± 0.33	2.63 ± 0.26	4.87 ± 0.5
Invalid action count	1168.9 ± 217.83	213.9 ± 31.32	3.5 ± 2.38	5.1 ± 3.78

The experimental results show that the average reward tends to increase with the number of model parameters, with qwen2.5:7b achieving the best performance. In terms of average runtime per step, qwen2.5:0.5b is significantly slower than the other models. Among the remaining models, the average runtime per step also increases with parameter size. Taking qwen2.5:3b as the baseline, the per-step operational efficiency of qwen2.5:7b decreases by

approximately 85.2%. Regarding the number of invalid actions, qwen2.5:0.5b generates significantly more invalid actions than the other models, whereas qwen2.5:3b and qwen2.5:7b perform relatively well. Therefore, balancing decision efficiency and effectiveness, qwen2.5:3b offers the best overall trade-off. In subsequent experiments, unless otherwise specified, the LLM used is qwen2.5:3b.

In addition, to verify the effectiveness of the hybrid decision-making of the LLM-ES method in terms of decision efficiency and decision effectiveness, we designed three groups of comparative experiments. The first group is pure LLM-driven decision-making (only using the LLM for decisions, without expert system participation). The second group is pure expert system decision-making (only making decisions according to predefined rules, without LLM participation). The third group is hybrid decision-making of LLM and ES. The results are shown in Table 4.

Table 4. Comparison of Experimental Results of Different Methods

Methods	LLM-only	ES-only	LLM-ES
Rewards ($mean \pm std$)	-3720.9 ± 1923.48	-252.2 ± 60.76	-98 ± 27.61
Run time (s)/step	3.08 ± 0.13	0.043 ± 0.004	2.63 ± 0.26
Invalid action count	2.9 ± 1.92	-	3.5 ± 2.38

The experimental results demonstrate that the hybrid LLM-ES method proposed in this study significantly outperforms the pure LLM method. In terms of average reward, it achieves improvements of 97.4% over the pure LLM method and 61.1% over the pure expert system method. In terms of decision efficiency, the average runtime per step is reduced from 3.08 s/step to 2.63 s/step compared with the pure LLM method, corresponding to a 14.6% improvement. Meanwhile, the results indicate that integrating the expert system increases the number of invalid actions generated by the LLM. This is because, once the expert system is incorporated, the available action range defined in this study is narrowed from the original action space to the

candidate action set provided by the expert system when it offers multiple emergency actions. Consequently, any LLM-generated action beyond the candidate set is regarded as invalid.

4.2 Ablation Experiment

To validate the design of the proposed method and analyze the contribution of each module to the reward, we used the full framework as the baseline and designed five groups of ablation experiments, in which modules such as the RAG, short-term historical memory, output alignment, and feedback learning were disabled. The evaluation metrics include average reward, average runtime per step, and number of invalid actions. The results are shown in Table 5.

Table 5. Ablation Experiment Results

Methods	Rewards (mean±std)	Run time (s)	Invalid action count
LLM-ES	-98 ± 27.61	2.63 ± 0.26	3.5 ± 2.38
No RAG	-108 ± 37.08	1.44 ± 0.16	5.6 ± 4.15
No History	-230.9 ± 144.08	2.79 ± 0.17	0.8 ± 1.53
No Alignment	-3595.1 ± 803.99	1.99 ± 0.25	926.5 ± 121.01
No Feedback Learning	-114.1 ± 33.19	2.54 ± 0.29	17.4 ± 14.83

As can be seen from the data in Table 4, when the RAG is disabled, the average reward drops from -98 to -108, a decrease of 10.2%, and the average number of invalid actions also increases slightly; however, the average runtime per step decreases from 2.63s to 1.44s, a decrease of about 45.2%. The experimental results show that while RAG improves the average reward, it significantly increases the average runtime per step. When short-term historical memory is disabled, the average reward drops from -98 to -230.9, a decrease of about 135.6%, and the average runtime per step increases from 2.63 s to 2.79 s. The experimental results show that introducing short-term memory can improve the effectiveness of the LLM to a certain extent. When output alignment is disabled, the average reward drops to -3595.1, and the average number of invalid actions rises to 926.5. The experimental results show that aligning the format and semantics of LLM outputs is necessary in this task. When feedback learning is disabled, the average reward drops from -98 to -114.1, a decrease of about 16.4%, and the average runtime per

step decreases from 2.63 s to 2.54 s; however, it is also noted that the average number of invalid actions increases significantly. The experimental results show that the feedback learning mechanism can improve the correctness and effectiveness of the LLM.

4.3 Algorithm Adaptability and Robustness Validation Experiments

Although the hybrid LLM-expert system method proposed in this work has achieved good results in experiments, how about its applicability and generality? First, is it applicable to other LLMs? Second, does the method remain effective after changing opponents? To this end, we designed two categories of comparative experiments. The first category mainly considers the LLM factor, selecting different series of LLMs, and examining their decision effectiveness and decision efficiency. The second category mainly considers the opponent factor, selecting different opponent strategies (i.e., red agent strategies), to examine the generalization ability of the method to different opponents.

In the first category of experiments, we selected four LLMs, namely Qwen2.5:3b, Qwen2.5:7b, LLaMA3.1:8b, and DeepSeek-R1:7b. The experimental results are shown in Table 6.

Table 6. Experimental Results Comparison of Different LLM Series

Methods	Rewards (mean±std)	Run time (s)	Invalid action count
Qwen2.5:3b	-98 ± 27.61	2.63 ± 0.26	3.5 ± 2.38
Qwen2.5:7b	-86.7 ± 21.67	4.87 ± 0.5	5.1 ± 3.78
LLaMA3.1:8b	-139.4 ± 43.87	7.79 ± 0.58	13.9 ± 7.57
DeepSeek-R1:7b	-142.4 ± 43.58	13.28 ± 2.93	18.1 ± 5.11

The experimental results demonstrate that the method proposed in this work maintains good effectiveness and compatibility with multiple models. At the same time, there are certain differences in the performance of different series of models. In terms of average rewards, LLaMA3.1:8b and DeepSeek-R1:7b are lower than Qwen2.5:3b, but are roughly on par with RL-based methods. In terms of average runtime per step, taking the runtime of Qwen2.5:3b as the

baseline, Qwen2.5:7b is 1.85 times, LLaMA3.1:8b is 2.96 times, and DeepSeek-R1:7b is 5.05 times. In terms of the number of invalid actions, LLaMA3.1:8b and DeepSeek-R1:7b are also significantly higher than Qwen2.5:3b. We therefore analyzed their responses and found that, despite prompting the LLMs to output only the final result without reasoning, a substantial proportion of responses from the two models still contained a "think" process, partly explaining their longer average runtime and greater number of invalid actions.

To evaluate the generalization ability of the proposed method against different opponents, the second set of experiments employed five red agents—FiniteState, AggressiveFSM, StealthyFSM, ImpactFSM, and DegradeServiceFSM—together with the Qwen2.5:3b LLM. For ease of comparison, we adopted an RL-based method [7] and an LLM-based method [10] as baselines. To our knowledge, these two works are among the few that comparatively and comprehensively examine the impact of different red agents on defense in the CybORG CC4 environment. The experimental results are shown in Table 7.

Table 7. Comparison of Experimental Results Across Different Types of Red Agents

Red agent	Ours(LLM-ES)	Singh(MARL)	Castro(LLM)
FiniteState(default)	-98 ± 27.61	-179.8 ± 92.98	2475.5
AggressiveFSM	-132.3 ± 34.81	-227.93 ± 87.98	2884.5
StealthyFSM	-97.1 ± 41.87	-165.8 ± 53.45	1611.5
ImpactFSM	-161.4 ± 38.86	-247.15 ± 71.53	2994
DegradeServiceFSM	-154.1 ± 47.92	–	2770.5

The data in the last two columns of Table 7 are both sourced from the original papers. Since Singh et al. [7] did not perform experiments on the DegradeServiceFSM agent, the corresponding entry is unavailable. The experimental results indicate that, irrespective of the method employed, different types of red agents exert a considerable influence on the average reward; however, the proposed method consistently outperforms existing methods in all cases. In scenarios with multiple attacking agents, the proposed method maintains a relatively favorable

defense reward level, demonstrating its strong generalization capability across different opponents.

5. Discussion

5.1 Advantages of the Proposed Method

We proposed a novel multi-agent hybrid LLM–expert system approach that integrates the determinism and efficiency of expert systems with the deep reasoning capabilities of LLMs. Experimental results demonstrate that our method substantially outperforms state-of-the-art RL-based baselines in terms of average reward, achieving an approximate 25.5% improvement. Incorporating the expert system not only boosts defense effectiveness but also yields considerable gains in decision efficiency.

Unlike conventional RL-based approaches, where learned policies are typically black-box, the decision logic and trigger conditions of our hybrid framework are fully traceable. This property delivers auditability and improved decision interpretability. In addition, the method requires no retraining when deployed to new scenarios; defense strategies can be updated by adjusting prompts or expert-system rules, which greatly enhances environmental transferability.

The feedback learning mechanism designed in this work enables the LLM to accommodate erroneous outputs. Error causes are encapsulated into prompts and fed back to the LLM, offering a preliminary paradigm for continual learning within LLM-based autonomous cyber defense systems.

The expert knowledge and rules adopted in this framework can be flexibly modified and iteratively updated, affording good flexibility for adapting to changing cyber scenarios.

5.2 Challenges and Limitations

While our hybrid LLM-expert system decision framework achieves promising results on CybORG CC4, the study has several inherent limitations.

First, the predefined strategies encoded in the expert system remain relatively coarse-grained. Second, CybORG CC4 was originally built for reinforcement learning and outputs numerical observation vectors. In this study, only selected fields from these vectors were converted into natural language descriptions. Such transformation may produce inaccurate textual descriptions and discard implicit information embedded in raw state vectors, creating a semantic gap between RL simulation environments and LLM-based decision agents. Third, although multiple augmentation techniques are adopted to strengthen the LLM component, hallucinations and output instability cannot be completely eliminated by our current design. Fourth, average per-step runtime is used only as a proxy metric for decision latency. This measurement introduces bias and can only reflect relative speed differences among compared methods.

5.3 Future Work

Several promising directions remain for further exploration. More sophisticated, fine-grained expert strategies can be designed to enrich the rule base. New approaches to bridge the semantic gap between numerical simulation observations and natural language inputs for LLMs deserve dedicated investigation. Advanced retrieval techniques, including hybrid retrieval and reranking, can be incorporated into the RAG module to upgrade the knowledge base. Future research will also explore more effective mitigation strategies for LLM hallucinations and unstable outputs.

Author contributions: Zhiqiang Qu: Conceptualization, Data curation, Formal analysis, Investigation, Resources, Methodology, Software, Validation, Visualization, Writing – Original Draft, Writing – Review & Editing.

Jun He: Conceptualization, Investigation, Methodology, Resources, Formal analysis, Funding acquisition, Supervision, Writing – Original Draft, Writing – Review & Editing.

Zhitao Long: Data curation, Software, Investigation, Validation, Visualization, Writing – Original Draft, Writing – Review & Editing.

Bo Wu: Resources, Formal analysis, Funding acquisition, Project administration, Supervision, Writing – Original Draft, Writing – Review & Editing.

Xinmei Wang: Software, Project administration, Validation, Writing – Original Draft, Writing – Review & Editing.

Xiaohua Yang: Formal analysis, Software, Writing – Original Draft, Writing – Review & Editing.

Yuting Zhao: Data curation, Visualization, Writing – Original Draft, Writing – Review & Editing.

Z.Q. conceived the research framework, developed the core codebase, carried out the experiments, and drafted the original manuscript. J.H. acquired funding, supervised the whole project, and revised the manuscript. Z.L. assisted with data curation, software implementation, experimental validation and visualization. B.W. managed the project, helped obtain funding and provided research supervision. X.W. supported code development, experimental validation and project coordination. X.Y. contributed to formal analysis and software debugging. Y.Z. aided in data curation and result visualization. All authors reviewed and commented on the manuscript.

Competing interests: The author(s) declare no competing interests.

Data Availability Statement: The CybORG CC4 environment used in this work is an open-source framework available at https://github.com/cage-challenge/cage-challenge-4.

Statement: During the preparation of this manuscript, the authors used DeepSeek V4 for linguistic polishing. After using this tool, the authors have reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. Srinivas, S. *et al.* AI-augmented SOC: A survey of LLMs and agents for security automation. *JCP* **5**, 95 (2025).
2. Vyas, S., Hannay, J., Bolton, A. & Burnap, P. P. Automated Cyber Defence: A Review. Preprint at https://doi.org/10.48550/arXiv.2303.04926 (2023).
3. Kiely, M. *et al.* Exploring the efficacy of multi-agent reinforcement learning for autonomous cyber defence: A CAGE challenge 4 perspective. *AAAI* **39**, 28907–28913 (2025).
4. Kunz, T., Fisher, C., Novara-Gsell, J. L., Nguyen, C. & Li, L. A Multiagent CyberBattleSim for RL Cyber Operation Agents. Preprint at https://doi.org/10.48550/arXiv.2304.11052 (2023).
5. Li, Y. *et al.* ACDZero: Graph-embedding-based tree search for mastering automated cyber defense. Preprint at https://doi.org/10.48550/arXiv.2601.02196 (2026).
6. Qu, Z., He, J., Wu, B., Long, Z. & Xia, T. HC-MARL: A general hierarchical cascaded multi-agent collaborative architecture for cyber wargaming applications. *Electronics* **15**, 3369 (2026).
7. Singh, A. V. *et al.* Hierarchical multi-agent reinforcement learning for cyber network defense. (2025).

8. Hassanin, M. & Moustafa, N. A Comprehensive Overview of Large Language Models (LLMs) for Cyber Defences: Opportunities and Directions. Preprint at https://doi.org/10.48550/arXiv.2405.14487 (2024).
9. Xu, H. *et al.* Large language models for cyber security: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.* 3769676 (2025) doi:10.1145/3769676.
10. Castro, S. R. *et al.* Large Language Models are Autonomous Cyber Defenders. Preprint at https://doi.org/10.48550/arXiv.2505.04843 (2025).
11. TTCP CAGE Working Group. TTCP CAGE challenge 4. (2023).
12. Gao, Y., Hammar, K. & Li, T. In-context autonomous network incident response: An end-to-end large language model agent approach. Preprint at https://doi.org/10.48550/arXiv.2602.13156 (2026).
13. Farooq, M. O. Towards generic defensive cyber agents: Leveraging large language models and RAG-based few-shot learning. *Computer Networks* **285**, 112349 (2026).
14. Mohammadi, H., Davis, J. J. & Kiely, M. Leveraging large language models for autonomous cyber defense: Insights from CAGE-2 simulations. *IEEE Intell. Syst.* **40**, 29–36 (2025).
15. Maxwell Standen, S. H., David Bowman & Kiely, M. CAGE challenge 2. (2022).
16. Jacques, T. A systematic evaluation of large language models for autonomous cyber defense. *Appl Intell* **56**, 367 (2026).
17. Janisch, J., Pevný, T. & Lisý, V. NASimEmu: Network Attack Simulator & Emulator for Training Agents Generalizing to Novel Scenarios. Preprint at https://doi.org/10.48550/arXiv.2305.17246 (2023).
18. Maxwell Standen, S. H., David Bowman. CAGE challenge 1. (2021).

19. Standen, M. *et al.* CybORG: A Gym for the Development of Autonomous Cyber Agents. Preprint at https://doi.org/10.48550/arXiv.2108.09118 (2021).
20. Emerson, H., Bates, L., Hicks, C. & Mavroudis, V. CybORG++: An Enhanced Gym for the Development of Autonomous Cyber Agents. Preprint at https://doi.org/10.48550/arXiv.2410.16324 (2024).
21. Palmer, G., Parry, C., Harrold, D. J. B. & Willis, C. Deep Reinforcement Learning for Autonomous Cyber Defence: A Survey. Preprint at https://doi.org/10.48550/arXiv.2310.07745 (2024).
22. Loevenich, J. F. *et al.* Training autonomous cyber defense agents: Challenges & opportunities in military networks. in *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)* 158–163 (IEEE, Washington, DC, USA, 2024). doi:10.1109/MILCOM61039.2024.10773923.
23. Xu, T. *et al.* L2M-AID: Autonomous cyber-physical defense by fusing semantic reasoning of large language models with multi-agent reinforcement learning (preprint). Preprint at https://doi.org/10.48550/arXiv.2510.07363 (2025).
24. Jiang, G., Imani, M., Bastian, N. D. & Lan, T. Agentic AI for cyber defense: LLM-guided hierarchical multi-agent reinforcement learning. in *MILCOM 2025 - 2025 IEEE Military Communications Conference (MILCOM)* 1518–1523 (IEEE, Los Angeles, CA, USA, 2025). doi:10.1109/MILCOM64451.2025.11310253.
25. Peng, Y. *et al.* Enhancing cloud network resilience via a robust LLM-empowered multi-agent reinforcement learning framework. Preprint at https://doi.org/10.48550/arXiv.2601.07122 (2026).

26. Cao, X. *et al.* Advancing LLM-Based Security Automation With Customized Group Relative Policy Optimization for Zero-Touch Networks. *IEEE J. Sel. Areas Commun.* **44**, 2669–2686 (2026).
27. Mukherjee, S., Chatterjee, S., Purvine, E., Fujimoto, T. & Emerson, T. Large language model-based reward design for deep reinforcement learning-driven autonomous cyber defense. Preprint at https://doi.org/10.48550/arXiv.2511.16483 (2025).
28. Ahmed, H. *et al.* Autonomous multi-agent cyber defense: A novel approach using reinforcement learning with hierarchical LLM critics. in *2025 Cyber Awareness and Research Symposium (CARS)* 1–5 (IEEE, Grand Forks, ND, USA, 2025). doi:10.1109/CARS67163.2025.11337724.
29. Lin, W. Q., Liu, X. H. & Zhang, Y. C. The construction method of cybersecurity large model based on RAG. *Command Control & Simulation* **48**, 146–153 (2026).
30. Blefari, F., Cosentino, C., Pironti, F. A., Furfaro, A. & Marozzo, F. CyberRAG: An agentic RAG cyber attack classification and reporting tool. *Future Generation Computer Systems* **176**, 108186 (2026).
31. Tellache, A., Korba, A. A., Mokhtari, A., Moldovan, H. & Ghamri-Doudane, Y. Advancing autonomous incident response: Leveraging LLMs and cyber threat intelligence. Preprint at https://doi.org/10.48550/arXiv.2508.10677 (2025).
32. Cadet, X. *et al.* Retrieval-augmented LLMs for security incident analysis. Preprint at https://doi.org/10.48550/arXiv.2603.18196 (2026).
33. Chen, Z., Chen, X., Li, Z., Xue, J. & Gao, C. ReVul-CoT: Towards effective software vulnerability assessment with retrieval-augmented generation and chain-of-thought prompting. *Journal of Systems and Software* **237**, 112855 (2026).

686 34. Yang, X. *et al.* CTI-thinker: An LLM-driven system for CTI knowledge graph construction
687 and attack reasoning. *Cybersecurity* **9**, 106 (2026).
688 35. Li, Z. *et al.* PenExpert: A multi-agent hybrid LLM–expert system framework for
689 autonomous penetration testing. *Expert Systems with Applications* **331**, 133284 (2026).
690