In-Context Autonomous Network Incident Response: An End-to-End Large Language Model Agent Approach

Yiran Gao[1], Kim Hammar[2], Tao Li[1]*

[1]Department of Systems Engineering, City University of Hong Kong, Hong Kong SAR, China
[2]Department of Electrical and Electronic Engineering, University of Melbourne, Australia
gaoyiran525@gmail.com, kim.hammar@unimelb.edu.au, li.tao@cityu.edu.hk

Abstract

Rapidly evolving cyberattacks demand incident response systems that can autonomously learn and adapt to changing threats. Prior work has extensively explored the reinforcement learning approach, which involves learning response strategies through extensive simulation of the incident. While this approach can be effective, it requires handcrafted modeling of the simulator and suppresses useful semantics from raw system logs and alerts. To address these limitations, we propose to leverage large language models' (LLM) pre-trained security knowledge and in-context learning to create an end-to-end agentic solution for incident response planning. Specifically, our agent integrates four functionalities, perception, reasoning, planning, and action, into one lightweight LLM (14b model). Through fine-tuning and chain-of-thought reasoning, our LLM agent is capable of processing system logs and inferring the underlying network state (perception), updating its conjecture of attack models (reasoning), simulating consequences under different response strategies (planning), and generating an effective response (action). By comparing LLM-simulated outcomes with actual observations, the LLM agent repeatedly refines its attack model and corresponding response, thereby demonstrating in-context adaptation. Our agentic approach is free of modeling and can run on commodity hardware. When evaluated on incident logs reported in the literature, our agent achieves recovery up to 23% faster than those of frontier LLMs.

Code — https://github.com/TaoLi-NYU/llmagent4incidense-response-aaai26summer

Introduction

Network incident response is a decision-making process in the post-attack stage aimed at containing, mitigating, and recovering from cyberattacks. Today's response practice relies heavily on manual operations, which can be slow and labor-intensive and fall short in the face of a continuously evolving security landscape. A recent study reports that more than 60% of surveyed organizations take more than 100 days to recover from incidents (IBM Security 2025).

To address the limitations of manual responses, substantial research efforts have been dedicated to automated responses, in which response planning and execution are delegated to an artificial intelligence (AI) agent with minimal human intervention (Li and Zhu 2025a). Prior work has extensively explored reinforcement learning (RL) approaches, where the incident response is formulated as a Markov decision process (MDP) or a game between the attack and the defense (Li, Zhao, and Zhu 2022). While RL agents have demonstrated success in simulations (Lohn et al. 2023), their practical implementations are hindered by the stringent requirement for structured network environment modeling, which compresses semantics in system logs and security alerts into succinct numeric data. Such a practice still requires manual labor and suppresses useful semantics.

Motivated by the emerging abilities of large language models (LLMs), the idea of LLM agent for autonomous defense has gained increasing momentum (Zhang et al. 2025; Li et al. 2025). While this emerging direction largely falls within academic research at the moment, industry stakeholders have attempted to commercialize it, as exemplified by IBM's LLM-based incident investigation service (Hussey 2025). Compared with RL agents, LLM agents excel at handling textual data from system logs and alerts and at leveraging their built-in security knowledge when planning response actions (Mohammadi et al. 2025; Castro et al. 2025), thereby sparing the manual labor of structured modeling.

However, unlike RL agents that are specifically tailored to long-horizon incident response tasks, most LLM-based methods proposed in the literature rely on prompt engineering of general-purpose LLMs. Consequently, the LLM agents are plagued by 1) **hallucinations**: generating response actions that appear plausible but are inappropriate, and 2) **context loss** in long-term planning: LLMs losing track of prior context as new findings overload history information, leading to incoherent response strategies (Lin et al. 2025).

This work aims to mitigate the above shortcomings by distilling RL-based planning principles in partially observed MDP (POMDP) into the LLM agentic workflows, facilitating an end-to-end incident response that directly maps logs and alerts to sequences of response actions. Our proposed agentic approach integrates four functionalities into a single lightweight LLM with 14 billion parameters, deployable on commodity hardware. Inspired by the online lookahead rollout methods in POMDP (Li, Lei, and Zhu 2023; Li et al. 2024), the four functionalities include 1) **perception**: pro-

*Corresponding author

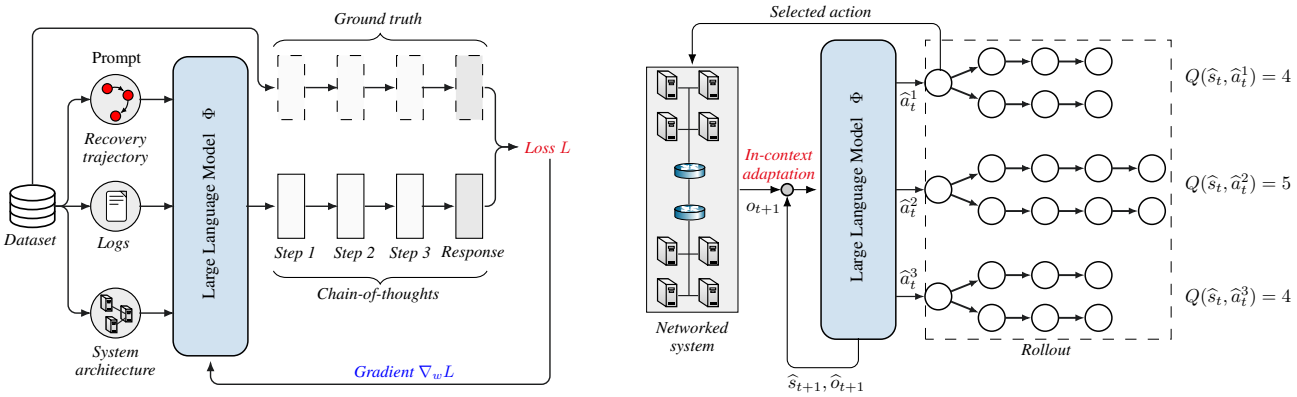

Figure 1: Overview of the two stages of our approach. In the first stage [cf. **a**)], an LLM is fine-tuned offline using a dataset of incident logs, each paired with corresponding response plans and chain-of-thought reasoning traces. In the second stage [cf. **b**)], the fine-tuned LLM processes system logs and threat intelligence online to generate N candidate response actions. A planning agent then evaluates these candidates through rollout and in-context adaptation, after which it selects the most effective action.

cessing the log and security alert data and inferring the underlying network recovery state; 2) **reasoning**: combining the built-in security knowledge and conjecture of attack tactics to forecast future alerts and recovery state, which corresponds to a "world model" of the network environment; 3) **planning**: carrying out a lookahead tree-search by simulating different action sequences; and 4) **action**: translating the high-level response strategies to security commands.

Our **core contribution** lies in the interplay between the agent's offline fine-tuned reasoning capability and online planning to address the hallucinations and context loss, as illustrated in Fig. 1. The internal world model's simulated recovery trajectories will be scrutinized in the planning stage, where hallucinated actions are filtered out. Meanwhile, the planned responses will be terminated if the world model's predicted alerts deviate from the actual observations and replanned after the LLM calibrates the world model through in-context learning, ensuring self-consistency in long-horizon response planning. We evaluate our agent on real-world incidence log data and, our agent generates effective response plan 23% faster than the frontier LLMs and prior works.

Related Work

Towards automating security response and more broadly, autonomous cyber defense, prior and contemporary efforts have been focusing on decision/game-theoretic (Hammar and Stadler 2024b; Manshaei et al. 2013; Li, Pan, and Zhu 2024), RL-based (Lohn et al. 2023; Ge, Li, and Zhu 2023), and most recently, LLM-based approaches (Zhang et al. 2025; Li et al. 2025). Compared with the other two, LLMs can directly generate response strategies by taking in log data without mathematical modeling or extensive pretraining in simulations, thanks to their text processing, semantic understanding, and pre-trained knowledge base.

Recent efforts on the LLM-based methods can be categorized into two major classes: prompt-based LLM orchestration and LLM-RL hybrid agentic approaches. The first class breaks down the entire incident response into several subtasks and develops detailed prompts for LLMs or independent LLM sessions when tackling each task (Mohammadi et al. 2025; Lin et al. 2025; Li and Zhu 2025b). Featuring end-to-end operations, such an approach requires substantial effort in designing complex prompts to reduce hallucinations and maintain prior context, preserving coherence in lengthy interactions (Lin et al. 2025).

The hybrid approach alleviates these limitations by combining RL and LLM agents, where RL agents supervise the LLM's generation (Yan, Zhang, and Huang 2024), LLM agents augment RL agents through knowledge sharing and human interactions (Loevenich et al. 2024), and two agents communicate with each other (Castro et al. 2025). Despite the different nature of agentic interactions, these works require additional RL training in simulated environments.

Our work aims to develop an RL-inspired prompting for creating LLM agents capable of handling the entire response cycle, which is less explored in the literature (Hammar, Alpcan, and Lupu 2026; Ren et al. 2025). Combining the advantages of RL and LLM, our proposed LLM agent adopts an RL-type lookahead (rollout) planning procedure (Bertsekas 2024) to address hallucinations and context loss based on the predictive analytics from LLM's processing of raw log and alert data.

Incident Response Planning as Partially Observable Markov Decision Process

Incident response involves restoring a network system to a secure, operational state after cyberattacks. Response planning involves analyzing attack patterns, securing forensic evidence, containing the attacker, restoring critical services, and hardening the system to prevent recurrence. Incident response, as a post-attack security mechanism, focuses on restoring service quality in a minimal *response and recovery time*, during which response actions are planned and deployed. A key challenge in achieving a timely and effective

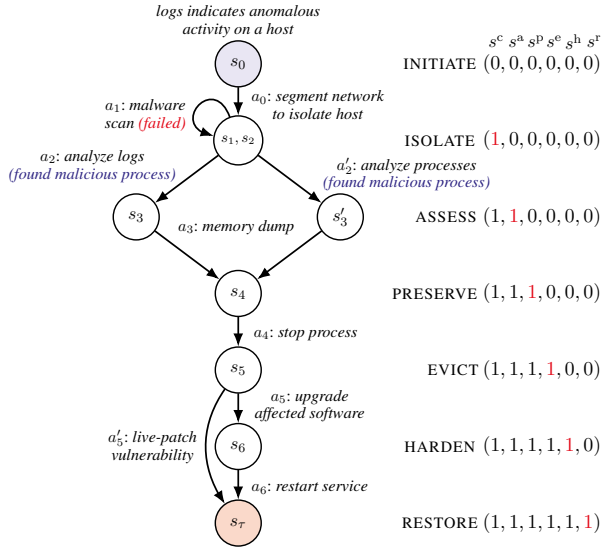

Figure 2: Two example evolutions of the recovery state s_t. The first recovery trajectory involves the actions $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ and the second trajectory involves the actions $a_0, a_1, a_2', a_3, a_4, a_5'$.

response is the defender's incomplete information about the attack's scope, pattern, and severity due to limited and partial indicators of compromise, such as log files and alerts. To facilitate our later discussion, we present a partially observable Markov decision process (POMDP) to capture the nature of incomplete information in incident response; Fig. 2 presents a visualization of response processes.

Recovery State Throughout the discussion, we adopt a discrete time index $t \in \mathbb{N}$, with $t = 0$ the start of response phase. We first introduce the *recovery state* $s_t = (s_t^{\mathrm{c}}, s_t^{\mathrm{a}}, s_t^{\mathrm{p}}, s_t^{\mathrm{e}}, s_t^{\mathrm{h}}, s_t^{\mathrm{r}})$ defined as a six-dimensional Boolean vector with each entry representing the progress in a specific stage of the response:

1) **Containment**: s_t^{c} indicates whether the attack has been isolated and prevented from spreading.
2) **Assessment**: s_t^{a} indicates if the scope and severity of the attack have been identified.
3) **Preservation**: s_t^{p} indicates if the forensic evidence related to the incidence has been preserved.
4) **Eviction**: s_t^{e} indicates whether the attacker's access has been revoked and malicious processes terminated.
5) **Hardening**: s_t^{h} indicates if the system has been hardened to prevent future recurrence.
6) **Restoration**: s_t^{r} indicates whether services and user access have been restored

Partial Observation Due to the network's complex nature and advanced attack methods, the defender is uncertain about the actual recovery state, e.g., if the attack's scope is correctly identified, properly contained, and successfully removed. It relies on system logs and security alerts from an intrusion detection device to infer the Boolean vector. We denote by o_t these textual data related to the defender's partial information, see Fig. 3 for an example alert.

State Transition Based on its understanding to the network environment, the defense agent plans a sequence of recovery actions $a_t, a_{t+1,\ldots}$ to be executed, and each action a_t will influence the next recovery sate s_{t+1}, which is captured by the transition kernel $P_\theta(s_{t+1}|s_t, a_t)$, where θ encapsulate the attack's influence on the future recovery state. In practice, θ usually corresponds to attack tactics, techniques, and procedures (TTP) as recorded in the MITER ATT&CK knowledge base (Strom et al. 2018). Since the partial observation also depends on the attack TTP, following the convention in POMDP studies, we incorporate the observation kernel into the state transition: $(s_{t+1}, o_{t+1}) \sim P_\theta(\cdot|s_t, a_t)$.

Recovery Time Since the timely response is the primary concern of our study, we associate each pair of state and recovery action with a time cost $c(s_t, a_t)$, representing the required time (in minutes) to execute the action in a certain state. For instance, the time to isolate a compromised host may be a few seconds, whereas the time to perform a system scan and configuration updates of affected systems can be around half an hour. The total recovery time J is given by the cumulative time to reach the terminal state $s_T = (1,1,1,1,1,1)$, i.e., $J(s_0) \triangleq \sum_{t=0}^{\tau-1} c(s_t, a_t)$, $s_T \sim P_\theta(\cdot|s_{\tau-1}, a_{\tau-1})$ with probability 1.

In summary, we aim to find a response policy generated by the LLM agent, denoted by $\Phi(\cdot)$, such that the total recovery time is minimized:

$$\min\ J(s_0) = \sum_{t=0}^{\tau-1} c(s_t, a_t) \tag{1}$$
$$\text{s.t. } a_t = \Phi(o_{0:t-1}, a_{0:t-1}); (s_t, o_t) \sim P_\theta(\cdot|s_{t-1}, a_{t-1}).$$

LLM Agent for In-Context Adaptive Response

Perception The LLM agent maintains an estimate of the underlying recovery state, denoted by $\hat{s}_t$, based on past observations and actions: $\hat{s}_t \sim \Phi(o_{0:t-1}, a_{0:t-1})$. For simplicity, we denote by $h_{t-1} \triangleq \{o_{0:t-1}, a_{0:t-1}\}$ the history information. To steer the LLM agent toward effective estimation, we fine-tune it using supervised learning on a dataset of $50,000$ instruction-answer pairs $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^K$, where each instruction $\mathbf{x}^i$ consists of information related to an incident and an instruction to assess the current recovery state; see Fig. 3 for an incidence example. The associated answer $\mathbf{y}^i$ describes the ground-truth, which is tokenized into $\mathbf{y}^i = (y_1^i, \ldots, y_{\ell_i}^i)$, following the world-level tokenization with ℓ_i being the token length. We pair each data point with a sequence of chain-of-thought (COT) reasoning steps to explain the answers (Wei et al. 2022).

Given the training dataset $\mathcal{D}$, the fine-tuning proceeds by iteratively sampling a batch of instruction-answer pairs $(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^B, \mathbf{y}^B)$ and updating the LLM's parameters via gradient descent on the cross-entropy loss

$$L(w) = -\frac{1}{B} \sum_{i=1}^{B} \sum_{k=1}^{\ell_i} \log \Phi_w(y_k^i | \mathbf{x}^i, y_{1:k-1}^i), \tag{2}$$

where w denotes the trainable parameters in Low-Rank Adaptation (LoRA), a widely used parameter-efficient fine-tuning technique (Hu et al. 2022).

Incident Example from CTU-Malware-2014

System description: Two subnetworks (A and B) are connected via a switch that is also connected to the Internet. All servers run WINDOWS XP SP2. Their IPs are...
Snort alert logs:

```
[120:3:2](http_inspect) NO CONTENT-LENGTH..
[1:31033:6]MALWARE Win.Trojan.Cryptodefence
[Classification:A Network Trojan Detected]
[Priority 1]
{TCP}147.32.84.165:1057->222.88.205.195:443
```

Incident description: Server 147.32.84.165 is infected with the WIN.TROJAN.CRYPTODEFENSE ransomware. Alerts show the server is making outbound command and control (C2) connections to 222.88.205.195. ...
Response actions:
1. Disconnect the Ethernet cable of the infected server at 147.32.84.165 to sever its network connection. Concurrently, configure a rule on the main switch/firewall to block all outbound traffic to the C2 server 222.88.205.195.
2. Analyze the central switch to scan all network traffic from both subnetworks A and B for any other hosts attempting to make connections to the malicious IP 222.88.205.195.
3. Before altering the infected server, create a complete bit-for-bit forensic image of its hard drive.
4. Wipe the hard drive of 147.32.84.165. If other infected machines were discovered, they must also be taken offline and wiped.
5. Upgrade all servers from WINDOWS XP SP2 to a modern operating system that receives security patches.
6. Restore the server's data from a trusted backup. Once the server is rebuilt with a modern operating system, reconnect it to the network and closely monitor for any anomalous activity.

Figure 3: An incidence example from (García et al. 2014).

Reasoning We aim to enable the LLM agent to predict future observations based on its understanding of the underlying state and attack assessment, which are continuously calibrated as new observations emerge. Specifically, the LLM agent first generates observation (alert) prediction $\hat{o}_t$ based on past observations $o_{o:t-1}$ and state estimate $\hat{s}_t$, $\hat{o}_t \sim \Phi(h_{t-1}, \hat{s}_t)$; See Fig. 4 and 5 for the generation prompts. By combining the state estimate and future observation prediction, the agent essentially creates an internal "world model" of the network environment and the attack, which helps simulate the possible consequences of different response actions. Given a sequence of actions to be evaluated through simulation starting from time t: $\{a_t, a_{t+1}, \ldots\}$, the LLM agent can simulate a *recovery trajectory* as below. For $\tau = t, t+1, \ldots$,

$$\hat{s}_{\tau+1} \sim \Phi(h_t, \hat{o}_{t:\tau}, a_{t:\tau}), \quad \hat{o}_{\tau+1} \sim \Phi(h_t, \hat{o}_{t:\tau}, a_{t:\tau}, \hat{s}_{\tau+1}),$$

which essentially creates a world model $P_\Phi(\hat{s}_{t+1}|\hat{s}_t, a_t)$. The fine-tuning of the observation (alert) generation follows the same practice as in the perception part, where the answers become the alert classification and priority as shown in Fig. 5. Note that the attack tactic is required in the fine-tuning when predicting alerts. As discussed in the following paragraph, the agent needs to consider possible tactics during the planning stage.

State Generation Prompt Template

###System description: ...
###logs: ...
###Incident description:...
###MITER ATT&CK tactics being used: Initial Access, Execution, Persistence.
###MITER ATT&CK techniques being used: T1190 Exploit Public-Facing Application.
Previous State: "is_attack_contained" : true; "is_knowledge_sufficient": false, "are_forensics_preserved": false, "is_eradicated": false, "is_hardened": false, "is_recovered": false.
###Previous recovery actions:
Action: Acquire full disk and memory images of REDIS01, collect Redis RDB/AOF files, NAT/firewall logs, and export system logs to write-protected storage.
Explanation: Preserves evidence and provides the data needed for thorough eradication.
###Instruction: You have been given information about a security incident, the state of recovery from the incident, and a recovery action. Your task is to predict what the next state of the recovery will be after applying the recovery action. For example, if the given recovery action effectively contains the attack and "is_attack_contained" is "false" in the current state, then the next state should have "is_attack_contained" set to "true". It is also possible that multiple state properties change values from false to true. It is also possible that the state remains the same, i.e., no property changes. It is important that the state only changes if the action is effective in achieving one of the recovery goals: containment, information gathering, preserving evidence, eradication, hardening, or recovery. A state variable can only change from "false" to "true", it cannot be changed from "true" to "false".
Response: <think>

Figure 4: The prompt for generating recovery states. The instruction presents an example of recovery state transition under attack tactics/techniques and recovery actions.

Planning Utilizing the LLM-based cyber world model, we propose an online conjectural lookahead planning method, inspired by an RL paradigm: Monte-Carlo tree search under misspecification in POMDP (Hammar et al. 2025; Hammar and Li 2025). The intuition behind such an RL algorithm is to first predict future consequences under diverse actions using the world model, which may be inaccurate (mispecified with respect to the true model), and then execute the best-performing one (under misspecification). The gap between newly emerged alerts and the predicted outcomes from the world model will provide context for LLM to reflect on its early conjectures and refine its action generation.

Specifically, the LLM agent starts by conjecturing attack tactics by screening system logs and incident information, then selecting a tactic from a set of plausible tactics, denoted by $\hat{\theta} \in \Theta$. At each time of the response, we prompt the agent to 1) infer the underlying recovery state $\hat{s}_t$; 2) generate N candidate actions $\mathcal{A}_t = \{\hat{a}_t^1, \hat{a}_t^2, \ldots, \hat{a}_t^N\}$,

Alert Generation Prompt Template

###System description: ...
###logs: ...
###Incident description:...
###MITER ATT&CK tactics being used: Initial Access, Execution, Persistence.
###MITER ATT&CK techniques being used: T1190 Exploit Public-Facing Application.
Instruction: Generate fields produced by an intrusion detection system (e.g., Snort) during a cyberattack by an attacker following this MITRE ATT&CK tactic: Impact. Frame your answers as "[Classification: alert type] [Priority: level]."
Response: <think>

Figure 5: The prompt for generating alerts. The LLM generations are structured into pairs of alert-type classifications, e.g., Bad Traffic and Network Trojans, and priority levels.

Attack Tactics Calibration Prompt Template

###System description: ...
###logs: ...
###Incident description:...
###MITER ATT&CK tactics being used: Initial Access, Execution, Persistence.
###PREDICTED ALERTS CHARACTERISTICS: "[Classification: alert type] [Priority: level]."
###OBSERVED ALERTS CHARACTERISTICS: "[Classification: alert type] [Priority: level]."
Instruction: Your task is to reassess the previously chosen MITRE ATT&CK tactic label(s), using the evidence in the above fields and comparing the predicted and observed alert characteristics. You must propose revised tactic candidates STRICTLY from the provided candidate tactics set, which includes the common tactics for such an incidence.
Response: <think>

Figure 6: The prompt for attack tactics conjecture calibration using GPT-5.2.

and 3) for each action, predict the next recovery state and alert, $(\hat{s}_{t+1}, \hat{o}_{t+1}) \sim \Phi(h_t, \hat{a}_t^k)$, $k =\in \{1, \ldots, N\} \triangleq [N]$, leveraging perception and reasoning functionalities. Given the generated, tentative actions, the agent needs to identify the best performer by 4) simulating M recovery trajectories starting from $(\hat{s}_{t+1}, \hat{a}_{t+1}^k)$, which is defined as a sequence of state-action pair generated by the LLM until the terminal sate $\hat{s}_T = (1, 1, 1, 1, 1, 1)$: $q^i \triangleq \{(\hat{s}_{t+1}, \hat{a}_{t+1}^k), (\hat{s}_{t+2}^i, \hat{a}_{t+1}^{k,i}), \ldots, \hat{s}_T\}$, $i \in \{1, \ldots, M\} \triangleq [M]$. 5) The estimated cumulated cost associated with $\hat{a}_{t+1}^k$ is given by the sample average, denoted by the Q function, based on which the agent selects the cost minimizer:

$$Q(\hat{s}_{t+1}, \hat{a}_{t+1}^k) = \frac{1}{M} \sum_{i \in [M]} \sum_{(\hat{s}, \hat{a}) \in q^i} c(\hat{s}, \hat{a}), \tag{3a}$$

$$a_{t+1} \in \arg\min_{a \in \mathcal{A}_t} Q(\hat{s}_{t+1}, a). \tag{3b}$$

Action Generation Prompt Template

###MITER ATT&CK tactics being used: Initial Access, Execution, Persistence.
###MITER ATT&CK techniques being used: T1190 Exploit Public-Facing Application.
Previous State:...
###Previous recovery actions:
Action: Acquire full disk and memory images of REDIS01, collect Redis RDB/AOF files, NAT/firewall logs.
Explanation: Preserves evidence and provides the data needed for thorough eradication.
###Instruction: You are a security operator ... . The goal when selecting the recovery action is to change the state so that one of the state-properties that is currently "false" becomes "true". The ideal recovery action sequence is: 1. contain the attack 2. gather information 3. preserve evidence 4. eradicate the attacker 5. harden the system 6. recover operational services. When selecting the recovery action, make sure that it is concrete and actionable and minimizes unnecessary service disruptions. Vague or unnecessary actions will not change the state and should be avoided. Return a JSON object with two properties: "Action" and "Explanation", both of which should be strings. The property "Action" should be a string that concisely describes the concrete recovery action. The property "Explanation" should be a string that concisely explains why you selected the recovery action and motivates why the action is needed.
Response: <think>

Figure 7: The prompt for generation recovery actions. The LLM is instructed to follow an ideal recovery trajectory and explain its motivations.

The executed action incurs actual alerts $o_{t+1} \sim P_\theta(s_{t+1}, a_{t+1})$, which enables the agent to 6) compare the actual and predicted alerts and calibrate the conjectured attack tactics if there exists a significant discrepancy between the two. We rely on a frontier LLM, GPT-5.2, to digest the newly emerged alert information and recommend a more likely alternative: $\hat{\theta}_{t+1} \leftarrow \text{GPT}(\hat{o}_{t+1}, o_{t+1}, a_{t+1})$, see Fig. 6 for the calibration prompt. Note that such a calibration only requires API access to the frontier models without local deployment, which remains commodity hardware-friendly. Moreover, such calibration can also be performed by the proposed LLM agent itself after retrieving relevant information from external threat intelligence, which is one of the future extensions. Finally, the agent repeats the same planning procedure under the calibrated conjecture from $t + 1$. Algo. 1 summarizes the planning cycle.

Action During the planning stage, the LLM agent needs to generate candidate actions based on its perception of the current recovery status. Towards this end, we fine-tune the LLM to generate effective response actions with reduced hallucinations and consistent with previous actions. Similar to state- and alert-generation fine-tuning, we present the LLM with pairs of instructions and answers, adding the previous estimated recovery state, previous recovery actions, and conjectured attack tactics to the instruction. Fig. 7 presents the

Algorithm 1: Online Conjectural Lookahead Planning with In-Context Adaptation

Input: LLM Φ, system logs, action batch N, sample trajectory batch M.
Output: Response actions $\pi = \{a_1, \ldots, a_T\}$.
Initialization $\hat{s}_0 \leftarrow (0,0,0,0,0,0), \hat{\theta}_0, \pi \leftarrow \{\}, t \leftarrow 0$.
while $\hat{s}_t \neq (1,1,1,1,1,1)$ **do**
  Sample $\{\hat{a}_t^1, \hat{a}_t^2, \ldots, \hat{a}_t^N\} = \mathcal{A}_t$ from $\Phi(\cdot|h_t)$.
  for $i \in [M]$ **do**
    $Q(\hat{s}_t, \hat{a}_t^i) = \frac{1}{M} \sum_{k=1}^{M}$ RECOVERY-TO-GO$(\hat{s}_t, \hat{a}_t^i)$
  end for
  Execute $a_t = \arg\min_{a \in \mathcal{A}_t} Q(\hat{s}_t, \hat{a}_t^i)$.
  Simulate $(\hat{s}_{t+1}, \hat{o}_{t+1}) \sim \Phi(h_t, a_t)$, and receive the alert o_{t+1}.
  Context adaptation $\hat{\theta}_{t+1} \leftarrow$ GPT$(\hat{o}_{t+1}, o_{t+1}, a_{t+1})$.
  $h_{t+1} \leftarrow h_t \cup \{o_{t+1}, a_t\}, t \leftarrow t+1$.
end while
procedure RECOVERY-TO-GO(s,a)
Simulate $s' \sim \Phi(s, a)$.
if $s' = (1,1,1,1,1,1)$ **then**
  return c(s,a)
end if
Sample $a' \sim \Phi(\cdot|s')$.
return $c(s,a) +$ RECOVERY-TO-GO(s', a').
end procedure

prompt template. For actions that lead to longer recovery paths and for predicted alerts that are inconsistent with actual ones, we regard them as likely instances of hallucinations and context-rot generation and will filter them out.

Experiment

Perception & Reasoning: LoRA Fine-Tuning

Experiment Setup We fine-tune a DeepSeek-14B (Qwen-compatible) language model using LoRA-based supervised fine-tuning on `CSLE-IncidentResponse-V1` (`states_examples.json`) (Hammar and Stadler 2024a). We take the first $50,000$ instruction–answer pairs from the dataset and train LoRA adapters with the hyperparameters detailed in Table 1.

Parameter(s)	Value(s)
LoRA rank, scaling, dropout	64,128,0.05
learning rate	0.00095
per_device_batch_size	2
gradient_accumulation_steps	16
temperature	0.6
action generation batch	3
recovery trajectory batch	3

Table 1: A summary of hyperparameters

Evaluation Setup and Metrics The evaluation data are randomly sampled, excluding the training data, and we collect a total of $17,600$ pairs for the testing dataset. We use two metrics to measure the prediction performance. 1) Exact-match accuracy: We treat both the prediction and the ground-truth label as JSON text and count a prediction as correct only if it matches the label *exactly* on all entries. This metric is intentionally strict and captures whether the model can produce a fully correct, structured output in a single shot. The exact accuracy is 0.98. 2) Multi-label F1: since recovery-state prediction consists of multiple Boolean entries, it is naturally a multi-label binary classification problem. We therefore report two averaged F1 scores: (i) class-agnostic average F1 (caa-F1), which aggregates True Positive (TP)/ False Positive (FP)/ False Negative (FN) across all entries before computing F1, and (ii) class-specific average F1 (csa-F1), which computes F1 per Boolean entry and then averages across entries. In addition, we report Per-entry F1 to diagnose which entries are harder to predict or exhibit systematic bias. Table 2 summarizes the F1 scores, which suggests that our model can accurately estimate the underlying state.

	caa	csa	s^c	s^a	s^p	s^e	s^h	s^r
F1	0.9902	0.9822	0.9975	0.9964	0.9970	0.9952	0.9541	0.9533

Table 2: A summary of F1 scores under class-agnostic average (caa) and class-specific average (csa).

Tactics(% over all data points)	**F1**
Normal Activity (15.59%)	0.5711
Initial Access, Execution, Collection, Exfiltration (6.92%)	0.8579
Access, Execution, Credential Access, Exfiltration (1.71%)	0.8599
Impact (1.55%)	0.8758
Access, Execution, Command and Control, Exfiltration (1.52%)	0.7424

Table 3: Most frequent tactics and corresponding F1 scores.

Alert Prediction Evaluation We further evaluate the model's ability to predict intrusion-detection system (IDS) alert fields (`classification` & `priority`) using the prompt shown in Fig. 5. We adopt a *unique-pair precision/recall* metric detailed below.

Given one evaluation sample with instruction prompt $\mathbf{x}$, we obtain the model's prediction $\hat{\mathbf{y}} \sim \Phi(\cdot|\mathbf{x})$ to be compared with the ground-truth $\mathbf{y}$. Since $\hat{\mathbf{y}}$ contains the full prompt followed by the generated continuation, we remove the prompt prefix to keep only the generated part. We then extract all alert tuples of the form `[Classification: ...][Priority: ...]` from both $\hat{\mathbf{y}}$ and $\mathbf{y}$ using a regular expression, and remove duplicates to obtain the *unique* prediction set $\widehat{\mathcal{P}}$ and label set $\mathcal{P}$. For each sample, we compute the overlap $|\widehat{\mathcal{P}} \cap \mathcal{P}|$ and define $\text{Precision} \triangleq |\widehat{\mathcal{P}} \cap \mathcal{P}|/|\widehat{\mathcal{P}}|$, $\text{Recall} \triangleq |\widehat{\mathcal{P}} \cap \mathcal{P}|/|\mathcal{P}|$.

We report the F1 score for alert generation in Table 3 across the top 5 most frequent tactics, from which we find that our model is better suited to handle generation under attack than in normal activity, since the corresponding alerts are false alarms and do not exhibit any pattern.

Dataset	Systems	Attacks	logs
CTU-Malware-2014	WINDOWS XP SP2	Malwares and ransomwares	SNORT alerts
CIC-IDS-2017	WINDOWS, LINUX	Denial-of-Service, web attacks, SQL injections, and etc.	SNORT alerts
AIT-IDS-V2-2022	WINDOWS, LINUX	Multi-stage attacks from reconnaissance to escalation	WAZUH alerts
CSLE-IDS-2024	LINUX	Software exploits, e.g., CVE-2015-1427	SNORT alerts

Table 4: The four evaluation datasets, from (García et al. 2014), (Sharafaldin, Habibi Lashkari, and Ghorbani 2018), (Landauer, Skopik, and Wurzenberger 2024), and (Hammar and Stadler 2024b), respectively, include diverse attacks, logs, and systems.

Calibration When a set of candidate tactics Θ is provided externally (e.g., from a frontier model such as GPT-5.2), we first inspect these tactics to ensure they are consistent with the logs. For each tactic $\hat{\theta} \in \Theta$, we prompt the LLM to produce a set of unique [Classification]–[Priority] pairs and compare them against the labeled log text. We compute a unique-pair precision (and recall) score and require the precision to exceed a threshold τ_{AP} (e.g., $\tau_{\text{AP}} = 0.6$) to proceed with multi-step planning. If no tactic's generation passes the threshold, we trigger the *calibration* procedure rather than running a full multi-step lookahead. Table 1 presents the configuration of the action generation batch N and trajectory batch M.

Evaluation and Baselines

Our experiment aggregates four evaluation datasets summarized in Table 4, which include a variety of attacks, alert logs, and system configurations. The recovery actions in these evaluation datasets are used to assess the effectiveness of the actions we generate. We rely on GPT-5.2 for such assessment, and measure the recovery time in discrete time units, assigning a time cost of 1 to all actions, and for those superfluous, less effective steps assessed by GPT-5.2, an additional cost of 1 is assigned as a kind of penalty. For actions generated that do not lead to a recovery terminal state, the time cost is set to 20.

We compare our LLM agent with three frontier models: DEEPSEEK-R1 (Guo et. al. 2025), GEMINI 2.5 PRO (Comanici et. al. 2025), and OPENAI O3 (OpenAI et. al. 2024). We remind the reader that our model, with 14b parameters, is significantly lightweight and of the same size as the baseline from (Hammar, Alpcan, and Lupu 2026).

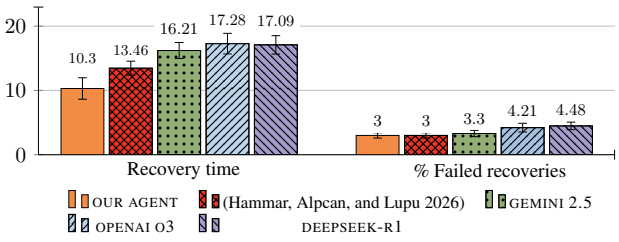

Figure 8: Evaluation results (↓ better): comparison between our method and frontier LLMs. Bar colors relate to different methods; bar groups indicate performance metrics; numbers and error bars indicate the mean and the standard deviation from 5 evaluations with different random seeds.

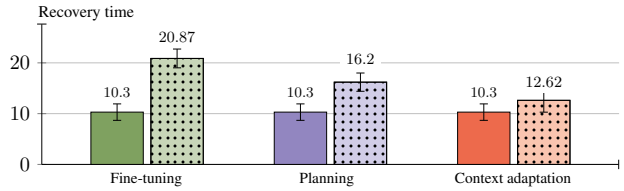

Figure 9: Ablation-study results for the recovery time metric (↓ better). Bar groups correspond to a specific step of our method; filled bars show performance with each step, and dotted bars show performance with the step removed; numbers and error bars indicate the mean and standard deviation across 5 evaluations with different random seeds.

Figure 8 summarizes the major evaluation results. While they all share similar failure rates, our model leads to the shortest recovery time than others, as shown in Fig. 8, whereas those frontier models, even though equipped with the latest security knowledge base, do not create an effective response plan, as they are not tailored to the security task.

To further evaluate the importance of each functionality, we conduct ablation studies removing the fine-tuning (i.e., Perception & Reasoning functions), planning, and the in-context adaptation mechanism. Fig. 9 compares the response performance before and after removing these key functionalities, which demonstrates that fine-tuning and planning play a significant role in the entire workflow, whereas the context adaptation, while still improving the performance, is not as instrumental as the other two. We speculate that the modest improvement is due to the fact that the test data points are mostly short sequences of recovery actions, typically 5 actions. Another extension is to evaluate our model on longer response processes, where the challenging longer-context situation can highlight the impact of our in-context adaptation scheme.

We finally remark on the scalability of the proposed LLM agent, which is the main limitation of our approach. The major computational expense is due to the Monte Carlo tree search and scales in $O(MN)$ time. All implementations are operated on Google Cloud with one A100 GPU. We observe that processing one incident takes, on average, 20 minutes to generate a five-action response plan. When deploying our agent on more complex network systems against more sophisticated tactics, the agent needs larger search trees, which soon render generation time disappointing. One of the most pressing extensions is to develop cost-efficient simulation or parallel computing methods for prompt incident response.

Conclusion

We present an in-context adaptive response planning method based on an LLM agent, resulting in an end-to-end incident response workflow without the structured modeling and simplification used in prior work on RL-based incident response. We model response planning as a partially observed Markov decision process and integrate Perception, Reasoning, Planning, and Action functionalities into a single lightweight LLM model inspired by the Monte-Carlo tree search method. We evaluate our LLM agent on diverse incidence datasets against frontier LLM models, and our LLM agent achieves 23% shorter recovery time than baselines.

The most pressing extension is to address the scalability issue in our LLM-based simulation, which can be time-consuming when deploying the agent to complex network environments. Another extension is to improve the current evaluation procedures by introducing more realistic time costs, comprehensive response action assessment, and long action sequences in the log data.

References

Bertsekas, D. P. 2024. Model Predictive Control and Reinforcement Learning: A Unified Framework Based on Dynamic Programming. *IFAC-PapersOnLine*, 58(18): 363–383. 8th IFAC Conference on Nonlinear Model Predictive Control NMPC 2024.

Castro, S. R.; Campbell, R.; Lau, N.; Villalobos, O.; Duan, J.; and Cardenas, A. A. 2025. Large Language Models are Autonomous Cyber Defenders. In *IEEE Conference on Artifical Intelligence Workshop on Adaptive Cyber Defense*.

Comanici et. al. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *arXiv:2507.06261*.

García, S.; Grill, M.; Stiborek, J.; and Zunino, A. 2014. An empirical comparison of botnet detection methods. *Computers & Security*, 45: 100–123.

Ge, Y.; Li, T.; and Zhu, Q. 2023. Scenario-Agnostic Zero-Trust Defense with Explainable Threshold Policy: A Meta-Learning Approach. In *IEEE Conference on Computer Communications*, 1–6.

Guo et. al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081): 633–638.

Hammar, K.; Alpcan, T.; and Lupu, E. C. 2026. Incident Response Planning Using a Lightweight Large Language Model with Reduced Hallucination. In *33rd Annual Network and Distributed System Security Symposium, NDSS 2026, San Diego, California, USA, February 23-27, 2026*.

Hammar, K.; and Li, T. 2025. Online incident response planning under model misspecification through Bayesian learning and belief quantization. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, 40–51.

Hammar, K.; Li, T.; Stadler, R.; Zhu, Q.; and Hammar, K. 2025. Adaptive Security Response Strategies Through Conjectural Online Learning. *IEEE Transactions on Information Forensics and Security*, 20: 4055–4070.

Hammar, K.; and Stadler, R. 2024a. The CSLE-IDS-2024 dataset. [Online] Avaiable at https://doi.org/10.5281/zenodo.10706475.

Hammar, K.; and Stadler, R. 2024b. Intrusion Tolerance for Networked Systems through Two-Level Feedback Control. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 338–352.

Hu, E. J.; yelong shen; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Hussey, S. 2025. Resolve incidents faster with IBM Instana Intelligent Incident Investigation powered by agentic AI. [Online] Available at https://www.ibm.com/new/announcements/resolve-incidents-faster-with-ibm-instana-intelligent-incident-investigation-powered-by-agentic-ai.

IBM Security. 2025. Cost of a Data Breach Report 2025. Technical report, IBM and Ponemon Institute.

Landauer, M.; Skopik, F.; and Wurzenberger, M. 2024. Introducing a New Alert Data Set for Multi-Step Attack Analysis. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, 41–53.

Li, T.; Hammar, K.; Stadler, R.; and Zhu, Q. 2024. Conjectural Online Learning with First-order Beliefs in Asymmetric Information Stochastic Games. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, 6780–6785.

Li, T.; Lei, H.; and Zhu, Q. 2023. Self-Adaptive Driving in Nonstationary Environments through Conjectural Online Lookahead Adaptation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 7205–7211.

Li, T.; Pan, Y.; and Zhu, Q. 2024. Decision-Dominant Strategic Defense Against Lateral Movement for 5G Zero-Trust Multi-Domain Networks. In *Network Security Empowered by Artificial Intelligence*, 25–76. Cham: Springer Nature.

Li, T.; Yang, Y.-T.; Pan, Y.; and Zhu, Q. 2025. From Texts to Shields: Convergence of Large Language Models and Cybersecurity. *arXiv:2505.00841*.

Li, T.; Zhao, Y.; and Zhu, Q. 2022. The role of information structures in game-theoretic multi-agent learning. *Annual Reviews in Control*, 53: 296–314.

Li, T.; and Zhu, Q. 2025a. Agentic AI for Cyber Resilience: A New Security Paradigm and Its System-Theoretic Foundations. *arXiv:2512.22883*.

Li, T.; and Zhu, Q. 2025b. Symbiotic Game and Foundation Models for Cyber Deception Operations in Strategic Cyber Warfare. In *Foundations of Cyber Deception*, 25–62. Springer Cham.

Lin, X.; Zhang, J.; Deng, G.; Liu, T.; Zhang, T.; Chen, R.; and Guo, Q. 2025. IRCOPILOT: Automated Incident Response with Large Language Models. *arXiv:2505.20945*.

Loevenich, J. F.; Adler, E.; Mercier, R.; Velazquez, A.; and Lopes, R. R. F. 2024. Design of an Autonomous Cyber Defence Agent using Hybrid AI models. *2024 International Conference on Military Communication and Information Systems (ICMCIS)*, 00: 1–10.

Lohn, A.; Knack, A.; Burke, A.; and Jackson, K. 2023. Autonomous Cyber Defense. Technical report, Center for Security and Emerging Technology.

Manshaei, M. H.; Zhu, Q.; Alpcan, T.; Bacşar, T.; and Hubaux, J.-P. 2013. Game theory meets network security and privacy. *ACM Comput. Surv.*, 45(3).

Mohammadi, H.; Davis, J. J.; Kiely, M.; and Mohammadi, H. 2025. Leveraging Large Language Models for Autonomous Cyber Defense: Insights from CAGE-2 Simulations. *IEEE Intelligent Systems*, 40(4): 29–36.

OpenAI et. al. 2024. GPT-4 Technical Report. *arXiv:2303.08774*.

Ren, Y.; Wang, J.; Zhao, Z.; Wen, H.; Li, H.; and Zhu, H. 2025. Automated tactics planning for cyber attack and defense based on large language model agents. *Neural Networks*, 191: 107842.

Sharafaldin, I.; Habibi Lashkari, A.; and Ghorbani, A. A. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*, 108–116.

Strom, B. E.; Applebaum, A.; Miller, D. P.; Nickels, K. C.; Pennington, A. G.; and Thomas, C. B. 2018. MITRE ATT&CK: Design and philosophy. Technical report, The MITRE Corporation.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, 24824–24837.

Yan, Y.; Zhang, Y.; and Huang, K. 2024. Depending on yourself when you should: Mentoring LLM with RL agents to become the master in cybersecurity games. *arXiv*.

Zhang, J.; Bu, H.; Wen, H.; Liu, Y.; Fei, H.; Xi, R.; Li, L.; Yang, Y.; Zhu, H.; and Meng, D. 2025. When LLMs meet cybersecurity: a systematic literature review. *Cybersecurity*, 8(1): 55.